



Yulian Findawati, ST., M.MT
Mochamad Affan Rosid, S.Kom., M.Kom



Buku Ajar Text Mining

Yulian Findawati, Mochamad Affan Rosid

ISBN 978-623-6833-19-3 (PDF)



9 786236 833193



UMSIDA Press
Universitas Muhammadiyah Sidoarjo
Jl. Mojopahle No. 46B
Sidoarjo, Jawa Timur

BUKU AJAR TEXT MINING

Oleh

Yulian Findawati, S.T., M.MT.

Muhammad Alfian Rosid, S.Kom., M.Kom.



Diterbitkan oleh
UMSIDA PRESS

**BUKU AJAR
TEXT MINING**

Penulis :

Yulian Findawati, S.T., M.MT.

Muhammad Alfian Rosid, S.Kom., M.Kom.

ISBN :

978-623-6833-19-3

Editor :

Rohman Dijaya, S.Kom., M.Kom.

Design Sampul dan Tata Letak :

Mochammad Nashrulloh, S.Pd.

Amy Yoga Prajati, S.Kom.

Penerbit :

UMSIDA Press

Anggota IKAPI No. 218/Anggota Luar Biasa/JTI/2019

Anggota APPTI No. 002 018 1 09 2017

Redaksi :

Universitas Muhammadiyah Sidoarjo

Jl. Mojopahit No 666B

Sidoarjo, Jawa TImur

Cetakan pertama, November 2020

© Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dengan suatu apapun
tanpa ijin tertulis dari penerbit.

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa, sehingga Usulan Buku Text Mining ini dapat disusun dengan baik dan selesai pada waktu yang telah ditentukan oleh Universitas Muhammadiyah Sidoarjo khususnya Direktorat Akademik yang bersedia untuk mengeluarkan dana dalam penulisan buku ajar ini. Penulisan buku ajar ini ditulis dalam 8 BAB secara garis besar, berdasarkan pengalaman penulis didunia kerja serta diambil dari hasil penelitian baik dari hasil penelitian mandiri maupun dari pendanaan Kemenristekdikti yang penulis dapatkan. Tak lupa kami juga mengucapkan terima kasih kepada:

1. Dr. Hidayatulloh, M.Si selaku Rektor Universitas Muhammadiyah Sidoarjo yang telah memberikan dan memfasilitasi dalam penulisan buku ajar ini.
2. Direktorat Akademik Universitas Muhammadiyah Sidoarjo yang telah memfasilitasi dan mengkoordinasi dalam penulisan buku ajar ini.
3. Dr. Hindarto, S.Kom, MT. selaku Dekan Fakultas Teknik, Universitas Muhammadiyah Sidoarjo yang telah memberikan dukungan untuk mengikuti penulisan buku ajar ini.
4. Dosen-dosen Teknik Informatika Universitas Muhammadiyah Sidoarjo yang telah memberikan dukungan untuk mengikuti penulisan buku ajar ini.
5. Para narasumber yang penulis tidak dapat sebutkan satu persatu yang telah banyak membantu, atas pengetahuan dan keterampilan yang diberikan dalam penyusunan penulisan buku ajar ini.
6. Aslab Teknik Informatika dan mahasiswa Teknik Informatika yang penulis tidak dapat sebutkan satu persatu yang telah banyak membantu, atas pengetahuan dan keterampilan yang diberikan dalam penyusunan penulisan buku ajar ini

Akhir kata, kritik dan saran sangat diharapkan untuk penyempurnaan buku ajar ini. Harapan kami semoga buku ajar ini dapat digunakan sebagai tambahan informasi dan bermanfaat bagi aktivitas pembelajaran mata kuliah Text Mining di Program Studi Informatika , Fakultas Teknik, Universitas Muhammadiyah Sidoarjo dan pembaca lainnya.

Penulis

DAFTAR ISI

1. BAB I. Definisi dasar Text Mining.....	1
2. BAB II. Preprocessing dalam text Mining.....	2
3. BAB III. Pembobotan Kata.....	4
4. BAB IV Information Retrieval.....	5
5. BAB V. Information Extraction.....	20
6. BAB VI. Summarization.....	23
7. BAB VIII. Document Classification.....	45
8. VII.Tools Text Mining dan Studi Kasus.....	55

CAPAIAN PEMBELAJARAN

Fakultas	:	Teknik
Program Studi	:	Informatika
Mata Kuliah (MK)	:	Text Mining
Kode MK	:	TI00419
SKS	:	3 (tiga)
Semester	:	VI I(Tujuh)
Mata Kuliah	:	BASIS DATA 2
Prasyarat		
StandarKompetensi (SK)	:	mahasiswa diharapkan mampu memahami dan menerapkan konsep umum Text Mining meliputi Information Retrieval, Information Extraction, Summarization, Document Clustering, Document Classification serta menciptakan aplikasi Text Mining untuk menyelesaikan permasalahan dalam bidang Informatika.

NO	Pokok Bahasan Dan TIU	Sub Pokok Bahasan dan TIK
BAB I	Mahasiswa/i dapat menjelaskan Mampu memahami konsep dasar Text Mining	Mahasiswa mampu memahami dan menjelaskan Pengenalan Text Mining dan implementasinya
BAB II	<p>Preprocessing dalam text Mining</p> <p>Mahasiswa/ Mampu memahami konsep preprocessing dalam Text Mining</p>	<ol style="list-style-type: none"> 1. Mahasiswa mampu memahami dan menjelaskan Tahap-tahap dalam Preprocessing 2. Mahasiswa mampu memahami dan menjelaskan Tahap-tahap dalam Case folding 3. Mahasiswa mampu memahami dan menjelaskan Tahap-tahap dalam Filtering 4. Mahasiswa mampu memahami dan menjelaskan Tahap-tahap dalam Stemming

BAB III	<p>Pembobotan Kata</p> <p>Mahasiswa/i mampu memahami dan menjelaskan konsep pembobotan kata pada Text Mining</p>	<ol style="list-style-type: none"> 1. Mahasiswa mampu memahami dan menjelaskan Pembobotan Kata 2. Mahasiswa mampu memahami dan menjelaskan TF IDF
BAB IV	<p>Information Retrieval</p> <p>Mahasiswa/i Mampu memahami dan menjelaskan konsep Information Retrieval.</p>	<ol style="list-style-type: none"> 1. Mahasiswa mampu memahami dan menjelaskan Pengertian information retrieval 2. Mahasiswa mampu memahami dan menjelaskan Boolean Retrieval 3. Mahasiswa mampu memahami dan menjelaskan Cosine Similarity
BAB V	<p>Information Extraction</p> <p>Mahasiswa/i Mampu memahami dan</p>	<ol style="list-style-type: none"> 1. Mahasiswa mampu memahami dan menjelaskan Pengertian Information Extraction

	menjelaskan konsep Information Extraction.	2. Mahasiswa mampu memahami dan menjelaskan NER
BAB VI	<p>Summarization</p> <p>Mahasiswa Mampu memahami dan menjelaskan konsep Summarization.</p>	<ol style="list-style-type: none"> 1. Mahasiswa mampu memahami dan menjelaskan Pengertian summarization 2. Mahasiswa mampu memahami dan menjelaskan Peringkasan menggunakan TF-IDF
BAB VIII	<p>Document Classification</p> <p>Mahasiswa Mampu memahami dan menjelaskan serta mengimplementasikan konsep algoritma Document Classification</p>	<ol style="list-style-type: none"> 1. Mahasiswa Mampu memahami dan menjelaskan serta mengimplementasikan Pengertian Document Classification 2. Mahasiswa Mampu memahami dan menjelaskan serta mengimplementasikan Contoh penggunaan Document Classification

		3. Mahasiswa Mampu memahami dan menjelaskan serta mengimplementasikan klasifikasi dokumen dengan naïve bayes
BAB IX	<p>Tools Text Mining</p> <p>Mahasiswa mampu memahami dan menggunakan tools Text Mining</p>	<p>Mahasiswa mampu menggunakan R studio maupun Rapid Miner untuk mengimplementasikan Text Mining</p>

BAB I.DEFINISI TEXT MINING

I.1 Definisi Text Mining

Teks Mining dapat didefinisikan secara luas sebagai proses intensif pengetahuan di mana pengguna berinteraksi dengan kumpulan dokumen dari waktu ke waktu dengan menggunakan seperangkat alat analisis. Penambangan teks berusaha untuk mengekstrak informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi pola yang menarik. Dalam kasus penambangan teks, bagaimanapun, sumber data adalah kumpulan dokumen, dan pola yang menarik ditemukan bukan di antara catatan database yang diformalkan tetapi dalam data tekstual yang tidak terstruktur dalam Koleksi dokumen. Oleh karena itu penambangan teks dan sistem penambangan data menunjukkan banyak kesamaan arsitektur tingkat tinggi. Contohnya, kedua jenis sistem ini bergantung pada rutinitas preprocessing, algoritma penemuan pola, dan elemen lapisan presentasi seperti alat visualisasi untuk meningkatkan penelusuran set jawaban.

Text mining adalah salah satu bidang khusus dalam data mining yang memiliki definisi menambang data berupa teks dimana sumber data biasanya didapatkan dari dokumen dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen (Mooney, 2006). Text mining dapat menganalisa dokumen, mengelompokkan dokumen berdasarkan kata-kata yang terkandung di dalamnya, serta menentukan kesamaan di antara dokumen untuk mengetahui bagaimana mereka berhubungan dengan variabel lainnya (Statsoft, 2015). Penerapan yang paling umum dilakukan text mining saat ini misalnya penyaringan spam, analisa sentimen, mengukur preferensi pelanggan, meringkas dokumen, pengelompokan topik penelitian, dan banyak lainnya.

Text Mining sendiri memiliki beberapa tipe antara lain (Abbot, 2013) :

1. Search and Information Retrieval Menyimpan dan menemukan kembali dokumen teks, termasuk mesin pencari dan kata kunci pencarian.
2. Document Clustering Pengelompokan dan pengkategorian istilah, potongan, paragraf, atau dokumen menggunakan metode mining .
3. Document Classification Pengelompokan dan pengkategorian istilah, potongan, paragraf, atau dokumen menggunakan metode document classification.
4. Web Mining Data dan text mining pada internet yang fokus pada skala dan antar hubungan pada website.
5. Information Extraction Identifikasi dan ekstraksi fakta yang relevan.
6. Natural Language Processing Pemrosesan bahasa tingkat rendah yang biasanya digunakan untuk bahasa komputasi.
7. Concept Extraction Pengelompokan kata dan frase dalam grup yang sama Tahap Preprocessing Text

BAB II. PREPROCESSING TEXT MINING

Text Preprocessing merupakan tahapan dari proses awal terhadap teks untuk mempersiapkan teks menjadi data yang akan diolah lebih lanjut. Suatu teks tidak dapat diproses langsung oleh algoritma pencarian, oleh karena itu dibutuhkan preprocessing text untuk mengubah teks menjadi data numeric . Proses ini terdiri dari beberapa tahap pembersihan dokumen seperti berikut:

a) Tokenizing

Tokenizing merupakan proses penguraian deskripsi yang semula berupa kalimat menjadi kata . Contoh proses tokenizing pada sebuah kalimat dapat dilihat pada Tabel

Tabel 2.1. Proses Tokenizing

Text Input	Hasil Token
Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...	go until jurong point crazy available ... wat

Semua kata yang menyusun kalimat pada kolom “Text Input” dipotong berdasarkan kata yang menyusunnya seperti terlihat di kolom “Hasil Token” pada Tabel 1.

b) Filtering

Filtering adalah tahap mengambil kata penting dari hasil proses token. Bisa menggunakan algoritma stop list atau word list . Filtering dapat juga diartikan sebagai proses mengambil kata – kata penting dari hasil proses token atau penghapusan stopwords. Stopwords merupakan kosa kata yang bukan merupakan ciri (kata unik) dari suatu dokumen . Untuk contoh tahap filtering terlihat pada Tabel 2. Kolom “Hasil Token” pada Tabel 3 adalah kata – kata yang berasal dari proses tokenizing

sedangkan kata yang berada pada kolom “Hasil Filtering” adalah hasil setelah proses filtering yaitu menghilangkan kata yang tidak penting seperti kata penghubung “go”, “in”, dan “there”.

Tabel 2.2 Proses Filtering

Hasil Token	Hasil Filtering
go	until
until	jurong
jurong	point
point	crazy
crazy	available
available	bugis
in	...
bugis	wat
...	
Wat	

c) Stemming

Stemming merupakan tahap untuk mencari root kata dari hasil filtering. Stemming adalah proses pemetaan dan penguraian berbagai bentuk (variants) dari suatu kata menjadi bentuk kata dasarnya (stem). Contoh t kalimat hasil proses filtering yaitu “Nah don’t think goes usf, lives here”, setelah adanya proses stemming kata “goes” berubah menjadi “go”.

d) Tagging

Tagging merupakan tahap untuk mencari bentuk awal/root dari tiap kata lampau atau hasil dari proses stemming Terdapat beberapa kata lampau yang dikembalikan ke bentuk awal, misalkan pada data pesan dengan kata “won” dirubah ke bentuk awal menjadi “win”.

e) Tahap Analyzing

Analyzing merupakan tahap penentuan seberapa jauh keterhubungan antar suatu kata atau term terhadap suatu

dokumen atau kalimat dengan menghitung nilai/bobot
keterhubungan

BAB III. PEMBOBOTAN KATA

Algoritma TF/IDF digunakan dalam proses perhitungan bobot (TF/IDF) terminologi kata. Algoritma ini digunakan untuk menghitung bobot setiap kata yang paling umum digunakan pada information retrieval. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat [12]. Persamaan yang digunakan untuk menghitung bobot (W) masing – masing dokumen terhadap kata kunci adalah [12] :

$$W_{a,t} = tf_{dt} * IDF$$

Dimana

W = bobot dokumen ke – n

d = dokumen

t = kata kunci

tf = terms frequency (jumlah kemunculan kata)

IDF = Inverse Document Frequency

Nilai tf diperoleh dari

$$tf_d = \frac{\text{Jumlah munculnya kata } t \text{ dalam dokumen}}{\text{Total jumlah seluruh kata dalam dokumen}}$$

Nilai IDF didapatkan

$$IDF = \log_2 \frac{d}{df}$$

dimana : D = total dokumen, dalam hal ini total kalimat yang ada
. df = jumlah dokumen yang mengandung kata kunci.

BAB IV. INFORMATION RETRIEVAL

Information Retrieval tetap menjadi salah satu masalah yang paling menantang di Text Mining ataupun pemrosesan Bahasa Alami. Ratusan juta orang terlibat dalam pengambilan informasi setiap hari saat menggunakan mesin pencari web atau mencari email. Pencarian database tradisional menjadi usang karena sebagian besar waktu pengguna tidak jelas apa yang dia cari. Pencarian kami seperti 'restoran Italia yang bagus di sekitar saya' atau 'universitas sains di India' atau 'penerbangan ke London besok' membutuhkan pemahaman yang mendalam tentang bahasa manusia (melakukan pencarian semantik sebagai pengganti pencarian tradisional.)

Pengambilan informasi bekerja pada skala yang berbeda. Dalam penelusuran web, sistem harus mencari miliaran dokumen yang disimpan di jutaan komputer yang memberikan jawaban atas pertanyaan tidak lengkap dan ambigu yang diajukan oleh pengguna agar tidak tertipu oleh penyedia situs yang memanipulasi konten situs dalam upaya untuk meningkatkan peringkat mesin telusur mereka. Ekstrem lainnya adalah pengambilan informasi pribadiseperti Mac's Spotlight, program email yang menyediakan pencarian serta klasifikasi email (spam, promosi, pembaruan, dll. dalam kasus Gmail), asisten pribadi chatbots dll. Di antaranya adalah ruang pencarian khusus perusahaan, institusional, dan domain, di mana pengambilan mungkin untuk dokumen internal perusahaan, database paten, atau artikel penelitian. Dalam hal ini, dokumen biasanya akan disimpan pada sistem file terpusat dan satu atau beberapa mesin khusus akan menyediakan pencarian atas koleksi tersebut.

Indeks Terbalik

Salah satu teknik pencarian informasi yang paling populer adalah penggunaan indeks terbalik. Diberikan satu set dokumen, kata kunci dan atribut lainnya (misalnya peringkat relevansi) diberikan ke setiap dokumen. Indeks terbalik adalah daftar kata kunci dan tautan ke dokumen yang sesuai. Seringkali beberapa langkah pra-pemrosesan diambil sebelum membuat indeks terbalik.

Tokenizing

Diberikan urutan karakter di dalam dokumen (kalimat frase), tokenisasi adalah tugas memotongnya menjadi beberapa bagian, yang disebut token, dan mungkin pada saat yang sama membuang karakter tertentu seperti tanda baca. Misalnya - 'Friends, Romans, Countrymen, pinjamkan telinga Anda.' tugas itu mungkin terdengar sepele. Keluarannya adalah Friends Romans Countrymen meminjamkan telinga Anda. Anda memotong spasi dan menghilangkan tanda baca. Tetapi ada banyak kasus rumit hanya untuk bahasa Inggris, apalagi bahasa sulit lainnya.

Misalnya, apa yang Anda lakukan jika ada penggunaan apostrof seperti "O'Reilly adalah jurnalis terkemuka Amerika selama tahun 70-an dan 80-an dan tidak terlibat dalam banyak kontroversi." Bagaimana Anda menandai O'Reilly atau America's? Atau tidak? tidak, bukan, bukan atau bukan?

Dalam bahasa Inggris, tanda hubung digunakan untuk berbagai tujuan mulai dari memisahkan vokal dalam kata-kata (pendidikan bersama) hingga menggabungkan kata benda sebagai nama (HewlettPackard) hingga perangkat penyalinan untuk menunjukkan pengelompokan kata (hold-him-back-and-drag-him manuver menjauh). Mudah untuk merasakan bahwa contoh pertama harus

dianggap sebagai satu tanda (dan memang lebih sering ditulis hanya sebagai pendidikan bersama), yang terakhir harus dipisahkan menjadi kata-kata, dan bahwa kasus tengah tidak jelas. Dengan demikian, menangani tanda hubung secara otomatis bisa jadi rumit.

Membagi spasi juga terkadang sulit. Hal ini paling sering terjadi dengan nama (San Francisco, Los Angeles) tetapi juga dengan kata-kata yang terkadang ditulis sebagai satu kata dan terkadang dipisahkan spasi (seperti spasi vs. spasi putih).

StopWords

Kadang-kadang, beberapa kata yang sangat umum yang tampaknya tidak berguna dalam membantu memilih dokumen yang sesuai dengan kebutuhan pengguna dikecualikan dari kosakata sepenuhnya. Kata-kata ini disebut kata-kata berhenti. Strategi umum untuk menentukan daftar berhenti adalah mengurutkan istilah berdasarkan frekuensi pengumpulan (total berapa kali setiap istilah muncul dalam kumpulan dokumen), dan kemudian mengambil istilah yang paling sering. Namun menjatuhkan kata-kata berhenti tidak selalu berguna. Frase query 'President of the United States' yang mengandung dua stopwords 'of' dan 'the' lebih berguna daripada query pencarian 'President United States'. Beberapa kutipan umum ('menjadi atau tidak menjadi'), frasa, puisi, lirik lagu (Let It Be) mengandung banyak stopwords dan mengecualikannya akan membuat pencarian informasi lebih sulit. Tren umum dalam sistem IR dari waktu ke waktu adalah dari penggunaan standar daftar berhenti yang cukup besar (200-300 istilah) hingga daftar berhenti yang sangat kecil (7-12 istilah) hingga tidak ada daftar berhenti sama sekali. Mesin pencari web umumnya tidak menggunakan daftar berhenti. Pada akhirnya penggunaan daftar kata berhenti bergantung pada tugas IR dan jenis dokumen.

Normalisasi

Setelah memecah dokumen (dan juga kueri kami) menjadi token, kasus mudahnya adalah jika token dalam kueri hanya cocok dengan token dalam daftar token dokumen. Namun, ada banyak kasus ketika dua urutan karakter tidak persis sama tetapi Anda ingin terjadi kecocokan. Misalnya, jika Anda mencari USA, Anda mungkin berharap untuk juga mencocokkan dokumen yang berisi USA

Normalizing juga sangat subjektif dalam kasus bahasa Inggris. Misalnya kami ingin menormalkan semua AS, AS, dan AS ke satu token, tetapi tidak untuk CAT & cat atau WHO & who. Kami juga ingin menormalkan jendela dan jendela menjadi satu token tetapi kueri penelusuran untuk 'OS Windows' tidak akan mengembalikan hasil terkait jendela.

Stemming dan lemmatization

Untuk alasan gramatikal, dokumen akan menggunakan berbagai bentuk kata, seperti mengatur, mengatur, dan mengatur. Selain itu, ada rumpun kata-kata yang berhubungan secara turunan dengan arti yang mirip, seperti demokrasi, demokrasi, dan demokratisasi. Dalam banyak situasi, tampaknya akan berguna untuk mencari salah satu dari kata-kata ini untuk mengembalikan dokumen yang berisi kata lain dalam kumpulan.

Stemming biasanya mengacu pada proses heuristik kasar yang memotong akhir kata dengan harapan mencapai tujuan ini dengan benar hampir sepanjang waktu. Lemmatisasi biasanya mengacu pada melakukan sesuatu dengan benar dengan menggunakan kosa kata dan analisis morfologi kata, biasanya bertujuan untuk menghilangkan akhiran infleksional saja dan mengembalikan bentuk dasar atau kamus dari sebuah kata, yang dikenal sebagai lemma. Jika

dihadapkan dengan token 'saw', stemming mungkin mengembalikan s, sedangkan lemmatization akan mencoba untuk mengembalikan baik lihat atau lihat tergantung pada apakah penggunaan token itu sebagai kata kerja atau kata benda. Oleh karena itu, lemmatisasi jauh lebih umum di NLP saat ini.

Meskipun sangat membantu untuk beberapa kueri, itu sama-sama merugikan kinerja untuk orang lain. Sebagai contoh dari apa yang bisa salah, membendung atau melemahkan kata-kata berikut: 'operasikan operasi operasi operasi operasi operasional' mungkin mengarah ke kata yang sama 'mengoperasikan'. Namun kami akan kehilangan ketepatan yang cukup besar pada kueri seperti berikut ini: operasional dan penelitian, operasi dan sistem, dll.

Seperti yang kita temui pada pengecualian di atas, tidak ada pendekatan 'satu ukuran cocok untuk semua' untuk pra-proses teks dokumen. Namun, teknik di atas berguna sebelum membuat indeks terbalik untuk pengambilan informasi.

Contoh-contoh Information Retrieval (IR)

1. Searching Text melalui Web Search Engine

Keyword dimasukkan oleh user untuk pencarian informasi yang diinginkan pada Search Engine, yang mana informasi yang didapatkan mengandung relevansi/keterkaitan dengan yang diharapkan.

2. Information retrieval di Perpustakaan

Perpustakaan adalah salah satu institusi pertama yang mengadopsi sistem IR untuk mendapatkan informasi. Pada umumnya, sistem yang digunakan di perpustakaan pada awalnya dikembangkan oleh institusi akademis dan kemudian oleh produsen komersil. Pada generasi pertama, sistem pada dasarnya terdiri dari suatu otomatisasi dari teknologi sebelumnya (seperti kartu katalog) dan memungkinkan pencarian berdasar judul dan nama pengarang. Pada

generasi kedua , kemampuan pencarian ditambahkan dengan pencarian berdasarkan pokok utama, dengan kata kunci, dan tambahan lagi fasilitas kueri kompleks. Pada generasi ketiga, yang sekarang ini yang sedang menyebar, fokusnya adalah meningkatkan antarmuka grafis, format elektronik, fitur *hypertext*, dan sistem arsitektur terbuka.

3. CBIR(Content Based Image Retrieval) Technology

Retrieval berdasarkan kategori konten dan warna. Dimana *user* mendeskripsikan *image* apa yang akan dicari dengan cara memilih kategori misalnya jenis *image*, Negara, tahun pembuatan dsb.

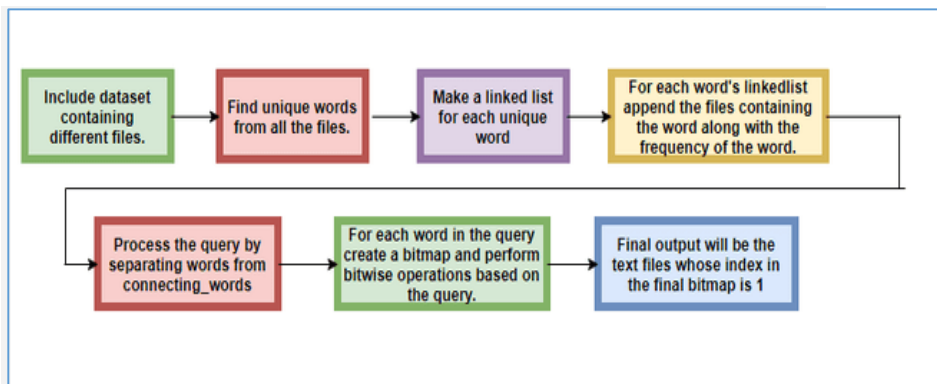
Pengambilan Informasi menggunakan Boolean Query dengan Python

Pengambilan informasi adalah proses penggalian informasi yang berguna dari data tidak terstruktur yang memenuhi kebutuhan informasi dari kumpulan data yang besar. Ini tetap menjadi salah satu tugas NLP yang paling menantang, karena banyaknya data tidak terstruktur yang digunakan untuk pemrosesan. Jutaan orang mengambil informasi dalam satu bentuk atau lainnya. Misalnya, ketika kita mencari frase “countries in asia”, dua kata utama, countries dan asia muncul dan kita perlu memastikan bahwa mesin menyertakan semantik dari frase tersebut saat mengambil informasi dari dokumen.

Pengambilan informasi bekerja pada skala yang berbeda. Ini bisa berupa pencarian web, di mana informasi yang relevan dipilih dari jutaan dokumen atau bisa dalam bentuk pencarian informasi pribadi, seperti yang diamati dalam kasus pemfilteran email sederhana. Dengan adanya kumpulan dokumen, pengambilan informasi membantu dalam menyaring dokumen yang paling penting berdasarkan kata kunci yang ditentukan dalam kueri yang disediakan oleh pengguna. Pada artikel ini, kami akan menggunakan kueri boolean untuk mengambil dokumen terpenting dari semua dokumen

dalam kumpulan data. Dokumen yang diekstrak akan memenuhi permintaan pengguna dengan mengambil informasi berdasarkan sifat semantik kueri.

Masih bingung??Jangan khawatir, mari kita lihat contohnya.Pertimbangkan bahwa kita memiliki kumpulan dokumen ini: india.txt, narendra_modi.txt,, rahul_gandhi.txt, apple.txt, australia.txt, cricket.txt, football.txt, volleyball.txt. Dokumen-dokumen ini berisi informasi mengenai nama masing-masing dokumen. Sekarang ketika pengguna memasukkan "bjp dan india atau kongres dan india" sebagai kueri, kami mengambil narendra_modi.txt dan rahul_gandhi.txt sebagai dokumen keluaran. Pada dasarnya kita mendapatkan kumpulan dokumen tersebut dari kumpulan data, yang memenuhi permintaan input. **Gambaran umum dari pendekatan yang digunakan dapat dilihat pada gambar 1.**



Gambar 3. 1. Tahap Information Retrieval

Linkedlist digunakan dalam pendekatan ini karena ia menempati lebih sedikit ruang karena hanya menghubungkan dan memproses file-file yang berisi kata, daripada memproses semua file dari dataset. Jika diagram alirnya tidak terlalu jelas, jangan khawatir beberapa istilah teknis akan dibahas di bagian pengkodean. **Kode Python untuk**

mengimplementasikan Information Retrieval menggunakan Boolean Query

Langkah-1 Mengimpor perpustakaan yang diperlukan

```
import nltk
dari nltk.corpus import stopwords
dari nltk.stem import WordNetLemmatizer, PorterStemmer
dari nltk.tokenize import sent_tokenize, word_tokenize
import glob
import import ulang
os
import numpy sebagai np
import sys
Stopwords = set (stopwords.words ('bahasa Inggris'))
```

Seperti yang saya sebutkan di artikel saya sebelumnya, NLTK adalah perpustakaan paling penting untuk NLP dengan Python. NLTK berisi paket untuk lemmatizing dan tokenizing words, yang merupakan langkah pra-pemrosesan penting saat menangani data teks.

Langkah-2 Menemukan kumpulan kata-kata unik dari semua dokumen kumpulan data

```
all_words = []
dict_global = {}
file_folder = 'data / *'idx = 1
files_with_index = {}
untuk file di glob.glob (file_folder):
    print (file)
    fname = file
    file = open (file, "r")
    text = file.read ()
    text = remove_special_characters (text)
    text = re .sub (re.compile ('\ d'), '', text)
    kalimat = sent_tokenize (text)
    words = word_tokenize (text)
    words = [kata demi kata dalam kata jika len
(words)> 1]
    words = [word .lower () untuk kata dalam kata-
kata]
```

```

    kata = [kata demi kata dalam kata jika kata tidak
dalam stopwords]
    dict_global.update
(menemukan_all_unique_words_and_freq (kata))
    files_with_index [idx] = os.path.basename (fname)
    idx = idx + 1

unique_words_all = set (dict_global.keys ())

```

Kode ini membantu dalam menemukan dokumen penting dari daftar dokumen. Variabel `file_folder` adalah jalur ke kumpulan data, yang berisi file dengan informasi tentang berbagai topik. Di sini kami mengakses semua file satu per satu dan melakukan pra-proses informasi di setiap file menggunakan langkah-langkah yang ditentukan dalam artikel saya sebelumnya. Artikel saya sebelumnya bisa diakses dari [sini](#) . Setelah pra-pemrosesan, kami memperbarui variabel `dict_global` setiap kali dengan menambahkan semua kata unik yang ditemukan di dokumen. Variabel `files_with_index` menyimpan indeks dari setiap file. Ini pada dasarnya menyimpan (indeks, nama file) sebagai pasangan (kunci, nilai). Akhirnya kami menemukan semua kata unik dengan `set (dict_global.keys ())` yang memberikan kumpulan kata-kata unik dan menyimpannya di `unique_words_all`. Beberapa fungsi yang digunakan dalam kode di atas dijelaskan di bagian selanjutnya. File data disimpan dalam folder `data`.

Langkah-3 Menerapkan fungsi pembantu

```

def find_all_unique_words_and_freq (words):
    words_unique = []
    word_freq = {}
    untuk kata dalam kata:
        jika kata tidak ada dalam words_unique:
            words_unique.append (word)
    for word in words_unique:
        word_freq [word] = words.count (word)
    return word_freq
def find_freq_of_word_in_doc
(kata, kata):

```

```

    freq = words.count (kata)

def remove_special_characters (teks):
    regex = re.compile ('[^ a-zA-Z0-9 \ s]')
    text_returned = re.sub (regex , '', teks)
    mengembalikan text_returned

```

Fungsi `find_all_unique_words_and_freq` menemukan semua kata unik bersama dengan frekuensinya. Untuk menghapus semua karakter khusus, kami menggunakan fungsi `remove_special_characters`.

Langkah-4 Mendefinisikan daftar tertaut

```

class Node:
    def __init__ (self, docId, freq = None):
        self.freq = freq
        self.doc = docId
        self.nextval = Tidak ada

kelas SlinkedList:
    def __init__ (self, head = None):
        self.head = head

```

Kelas `Node` bertindak sebagai node untuk setiap kata, dengan menyimpan `docId` dan frekuensi kata tersebut di masing-masing `docId`. Dokumen berikutnya yang berisi kata tersebut ditautkan ke `Node` saat ini menggunakan variabel `nextval`. Kelas `SlinkedList` membuat penunjuk kepala untuk setiap kata unik dalam kumpulan data.

Langkah-5 Membuat linkedlist untuk setiap kata dan menyimpan semua node (berisi nama file dan frekuensi kata masing-masing) dalam linkedlist.

```

linked_list_data = {}
untuk kata di unique_words_all:
    linked_list_data [kata] = SLinkedList ()
    linked_list_data [kata] .head = Node (1,
Node)word_freq_in_doc = {}
idx = luntuk file di glob.glob (file_folder):
    file = open (file, "r")
    text = file.read ()
    text = remove_special_characters (text)
    text = re.sub (re.compile ('\ d'), ' ', teks)
    kalimat = sent_tokenize (teks)
    kata = kata_tokenize (teks)
    kata = [kata demi kata dalam kata jika len
(kata)> 1]
    kata = [kata.lower () untuk kata dalam kata]
    kata = [kata untuk kata dalam kata-kata jika kata
tidak ada di Stopwords]
    word_freq_in_doc = find_all_unique_words_and_freq
(kata-kata)
    untuk kata di word_freq_in_doc.keys ():
        linked_list = linked_list_data [kata] .head
        sementara linked_list.nextval bukan Tidak
ada:
            linked_list = linked_list.nextval
            linked_list.nextval = Node (idx,
word_freq_in_doc [kata])
            idx = idx + 1

```

Kita akan mulai dengan menginisialisasi daftar tertaut baru untuk setiap kata unik. Node pertama dari setiap daftar tertaut berisi 1 sebagai docId default yang dapat diabaikan, yang dapat diabaikan untuk tugas pemrosesan nanti. Setelah linkedlist diinisialisasi, setiap file dalam kumpulan data dibaca kata demi kata dan semua kata unik dalam file disimpan dalam word_freq_in_doc. Kemudian kata-kata dapat diakses satu per satu dari kamus word_freq_in_doc dan daftar kata terkait menambahkan simpul baru (berisi file dan frekuensi kata itu dalam file). Mari kita lihat contoh yang ditentukan di bawah ini. Daftar tautan kata apel saat apel ada di file nomor 1,2,5 dan 7:

Apple-> 1-> 2-> 5-> 7

Langkah -6 Pemrosesan kueri dan pembuatan keluaran

```
query = input ('Masukkan kueri Anda:')
query = word_tokenize (kueri)
connect_words = []
cnt = 1
different_words = []
untuk kata dalam kueri:
    if word.lower () != "and" and word.lower () !=
"or" and word.lower () != "not":
        different_words.append (word.lower ())
    else :
        connect_words.append (word.lower ())
cetak (
connect_words )
total_files = len
(files_with_index)
zeroes_and_ones = []
zeroes_and_ones_of_all_words = []
untuk kata dalam (different_words):
    jika word.lower () di unique_words_all:
        zeroes_and_ones = [0] * TOTAL_FILES
        linkedlist = linked_list_data [kata] .head
        cetak (kata)
        sementara linkedlist.nextval tidak ada :
            zeroes and ones [linkedlist.nextval.doc -
1] = 1
            linkedlist = linkedlist.nextval
        zeroes_and_ones_of_all_words.append
(zeroes_and_ones)
    else:
        print (kata, "tidak ditemukan")
        sys.exit ()
cetak
(nol_dan_ones_of_all_words)
untuk kata dalam
connect_words : word_list1 =
zeroes_and_ones_of_all_words [0]
word_list2 = zeroes_and_ones_of_all_words [1]
if word == "and":
    bitwise_op = [w1 & w2 for (w1, w2) in zip
(word_list1, word_list2)]
    zeroes_and_list )
    zeroes_and_ones_of_all_words.remove
(word_list2)
    zeroes_and_ones_of_all_words.insert (0,
bitwise_op);
    kata elif == "atau":
        bitwise_op = [w1 | w2 untuk (w1, w2) di zip
(word_list1, word_list2)]
```

```

        zeroes_and_ones_of_all_words.remove
(word_list1)
        zeroes_and_ones_of_all_words.remove
(word_list2)
        zeroes_and_ones_of_all_words.insert (0,
bitwise_op);
        kata elif == "tidak"
        bitwise_op = [bukan w1 untuk w1 di
word_list2]
        bitwise_op = [int (b == True) untuk b di
bitwise_op]
        zeroes_and_ones_of_all_words.remove
(word_list2)
        zeroes_and_ones_of_all_words.remove
(word_list1)
        bitwise_op = [w1 & w2 untuk (w1 & w2) dalam
zip (word_list1,
bitwise_op)]zeroes_and_ones_of_all_words.insert (0,
bitwise_op);

files = []
print (zeroes_and_ones_of_all_words)
lis = zeroes_and_ones_of_all_words [0]
cnt = 1
untuk indeks dalam lis:
    if index == 1:
        files.append (files_with_index [cnt])
        cnt = cnt + 1

print (files)

```

Kami mulai meminta pengguna untuk memasukkan kueri. Kueri harus berupa kueri boolean dalam bentuk: kata1 kata_hubung kata2 kata_hubung kata3dan seterusnya. Perhatikan di sini, `connecting_word` mengacu pada **dan** , **atau** dan **bukan** .

Misalnya: Apel dan buah-buahan dan india atau mangga.

Output dari query ini akan mengambil semua dokumen yang berisi ketiga kata (apel, buah dan india) atau hanya berisi kata mangga.

Kode ini memisahkan semua kata_hubung dari kata lain. Di sini (apel, buah, india dan mangga) akan disimpan dalam variabel different_words dan (dan, dan, atau) akan disimpan dalam variabel connect_words. Untuk tujuan operasi boolean, kami membuat bitmap untuk setiap kata selain kata_koneksi dalam kueri. Bitmap ini menyimpan 1 dalam indeks file jika file tersebut berisi kata, 0 jika tidak. Setelah bitmap dibuat, operasi bitwise dapat dilakukan satu per satu dengan memproses bitmap berdasarkan connect_word yang disediakan di antara dua bitmap. Setelah pemrosesan selesai, akhirnya kami mengeluarkan file-file itu di mana indeks pada bitmap menunjukkan 1.

Misalnya (contoh dokumen): india.txt, narendra_modi.txt`` rahul_gandhi.txt, apple.txt, australia.txt, cricket.txt, football.txt, volleyball.txt. Saat kita memasukkan: "**bjp dan india atau kongres dan india**" sebagai kueri, kita mendapatkan bitmap sebagai [0,1,1,0,0,0,0]. Dari bitmap kita mengamati bahwa, 1 diamati dalam indeks dokumen keluaran. Karenanya kami mendapatkan dokumen keluaran berikut: narendra_modi.txt dan rahul_gandhi.txt.

Kode lengkap dapat dibuat dengan menggabungkan kode individu yang disediakan dalam artikel ini dalam urutan berikut: Langkah 1-> Langkah 3 -> Langkah 4-> Langkah 2-> Langkah 5-> Langkah 6.

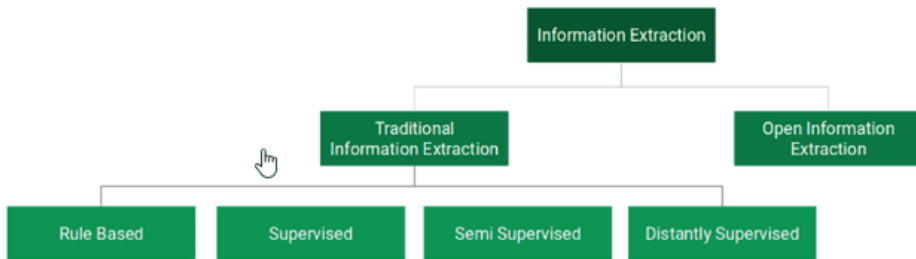
.

BAB V. INFORMATION EXTRACTION

Pengantar Ekstraksi Informasi

Information Extraction (IE) adalah roda penggerak penting dalam bidang Natural Language Processing (NLP) dan linguistik. Ini banyak digunakan untuk tugas-tugas seperti Sistem Penjawab Pertanyaan, Terjemahan Mesin, Ekstraksi Entitas, Ekstraksi Peristiwa, Penautan Entitas Bernama, Resolusi Coreference, Ekstraksi Relasi, dll. Dalam ekstraksi informasi, ada konsep penting yaitu triplet

Triplet mewakili beberapa entitas dan hubungan di antara mereka. Misalnya, (Obama, born, Hawaii) is a triple in which 'Obama' and 'Hawaii' are the related entities, and the relation between them is 'born'. Pendekatan Ekstraksi Informasi secara luas menjadi dua cabang seperti yang ditunjukkan di bawah ini:



Gambar 5.1 . Pendekatan Ekstraksi Informasi

Dalam Ekstraksi Informasi Tradisional, hubungan yang akan diekstraksi telah ditentukan sebelumnya. Pada artikel ini, kita akan membahas metode berbasis aturan saja.

Dalam Ekstraksi Informasi Terbuka, hubungan tidak ditentukan sebelumnya. Sistem ini bebas untuk mengekstrak hubungan apa pun yang ditemuinya saat melalui data teks.

Hubungan Semantik: Dapatkan Pengetahuan Terstruktur dari Teks Tidak Terstruktur

Lihat potongan teks di bawah ini:

"Food tutorials are infinitely better when directed by wes Anderson. Bruce lee's biopic, 'little dragon', to be directed by shekhar Kapoor. Stallone directed his first short film Vic"

Dalam kalimat pertama, memiliki dua entitas ("Tutorial Makanan" dan "Wes Anderson"). Entitas-entitas ini terkait dengan istilah "directed". Oleh karena itu, (Wes Anderson, directed, food tutorial) adalah triple. Demikian pula, dapat mengekstrak hubungan dari kalimat lain juga:

"Food tutorials are infinitely better when directed by wes Anderson. Bruce lee's biopic, 'little dragon', to be directed by shekhar Kapoor. Stallone directed his first short film Vic"

Berbagai Pendekatan untuk Ekstraksi Informasi

Di dunia nyata, ukuran data sangat besar dan ekstraksi informasi terstruktur secara manual tidak layak. Oleh karena itu, mengotomatisasi ekstraksi informasi ini menjadi penting. Ada beberapa pendekatan untuk melakukan ekstraksi informasi secara otomatis. Mari kita pahami satu per satu:

1. Pendekatan berbasis aturan: seperangkat aturan untuk sintaks dan properti gramatikal lainnya dari bahasa alami dan kemudian menggunakan aturan ini untuk mengekstrak informasi dari teks
2. Supervised: Katakanlah memiliki kalimat S. Ia memiliki dua entitas E1 dan E2. Sekarang, model pembelajaran mesin yang diawasi harus mendeteksi apakah ada hubungan (R) antara E1 dan E2. Jadi, dalam pendekatan yang diawasi, tugas ekstraksi relasi berubah menjadi tugas deteksi relasi. Satu-satunya kelemahan dari pendekatan ini adalah bahwa dibutuhkan banyak data berlabel untuk melatih model
3. Semi-Supervised: Jika tidak memiliki cukup data berlabel, dapat menggunakan satu set contoh benih (tiga kali lipat) untuk merumuskan pola presisi tinggi yang dapat digunakan untuk mengekstrak lebih banyak hubungan dari teks

BAB V. SUMMARIZATION

Apa perlunya peringkasan teks?

Didorong oleh inovasi teknologi modern, data hingga abad ini sama dengan minyak pada abad sebelumnya. Saat ini, dunia kita diterjunkan oleh pengumpulan dan penyebaran data dalam jumlah besar. Faktanya, International Data Corporation (IDC) memproyeksikan bahwa jumlah total data digital yang beredar setiap tahun di seluruh dunia akan tumbuh dari 4,4 zettabyte pada tahun 2013 menjadi 180 zettabyte pada tahun 2025. Itu data yang sangat banyak.

Dengan banyaknya jumlah data yang beredar di ruang digital, ada kebutuhan untuk mengembangkan algoritma pembelajaran mesin yang secara otomatis dapat mempersingkat teks yang lebih panjang dan menyampaikan ringkasan akurat yang dapat dengan lancar menyampaikan pesan yang dimaksud.

Selain itu, menerapkan peringkasan teks mengurangi waktu membaca, mempercepat proses penelitian informasi, dan meningkatkan jumlah informasi yang dapat muat di suatu area.

Peringkasan teks adalah teknik untuk menghasilkan ringkasan teks tebal yang ringkas dan tepat sambil berfokus pada bagian yang menyampaikan informasi bermanfaat, dan tanpa kehilangan makna keseluruhan. Peringkasan teks otomatis bertujuan untuk mengubah dokumen yang panjang menjadi versi yang dipersingkat, sesuatu yang mungkin sulit dan mahal untuk dilakukan jika dilakukan secara manual. Algoritma pembelajaran mesin dapat dilatih untuk memahami dokumen dan mengidentifikasi bagian yang menyampaikan fakta dan informasi penting sebelum menghasilkan teks ringkasan yang diperlukan. Misalnya, gambar di bawah ini adalah artikel berita yang telah dimasukkan ke dalam algoritme pembelajaran mesin untuk menghasilkan ringkasan.

Jenis utama peringkasan teks

Secara umum, ada dua pendekatan untuk meringkas teks di NLP: ekstraksi dan abstraksi.

Peringkasan berbasis ekstraksi

Dalam peringkasan berbasis ekstraksi, sekumpulan kata yang mewakili poin terpenting ditarik dari sepotong teks dan digabungkan untuk membuat ringkasan. Anggap saja sebagai penyorot — yang memilih informasi utama dari teks sumber.

Text Sumber : Peter and Elizabeth took a taxi to attend the night party in the city. While in the party, Elizabeth collapsed and was rushed to the hospital

Summary : Peter and Elizabeth attend party, Elizabeth rushed hospital

Seperti yang kita lihat di atas, ringkasan yang diekstrak terdiri dari kata-kata yang dicetak tebal, meskipun hasilnya mungkin tidak akurat secara tata bahasa.

Peringkasan berbasis abstraksi

Dalam peringkasan berbasis abstraksi, teknik pembelajaran mendalam lanjutan diterapkan untuk memparafrasekan dan mempersingkat dokumen asli, seperti yang dilakukan manusia. Anggap saja sebagai pena — yang menghasilkan kalimat baru yang mungkin bukan bagian dari dokumen sumber.

Karena algoritma pembelajaran mesin abstraktif dapat menghasilkan frasa dan kalimat baru yang mewakili informasi paling penting dari teks sumber, algoritme tersebut dapat membantu mengatasi ketidakakuratan tata bahasa dari teknik ekstraksi. Berikut ini contohnya:

Text Sumber : Peter and Elizabeth took a taxi to attend the night party in the city. While in the party, Elizabeth collapsed and was rushed to the hospital

Summary : Elizabeth was hospitalized after attending a party with peter

Meskipun abstraksi bekerja lebih baik pada peringkasan teks, mengembangkan algoritmanya membutuhkan teknik pembelajaran mendalam yang rumit dan pemodelan bahasa yang canggih.

Untuk menghasilkan keluaran yang masuk akal, pendekatan peringkasan berbasis abstraksi harus mengatasi berbagai macam masalah NLP, seperti generasi bahasa alami, representasi semantik, dan permutasi inferensi.

Dengan demikian, pendekatan peringkasan teks ekstraktif masih sangat populer. Dalam artikel ini, kami akan berfokus pada metode berbasis ekstraksi.

Bagaimana melakukan peringkasan teks

Mari gunakan paragraf pendek untuk menggambarkan bagaimana peringkasan teks ekstraktif dapat dilakukan.

Ini paragrafnya:

“Peter dan Elizabeth naik taksi untuk menghadiri pesta malam di kota. Saat di pesta, Elizabeth pingsan dan dilarikan ke rumah sakit. Karena dia didiagnosis mengalami cedera otak, dokter menyuruh Peter untuk tetap di sampingnya sampai dia sembuh. Oleh karena itu, Peter tinggal bersamanya di rumah sakit selama 3 hari tanpa pergi.” Berikut adalah langkah-langkah yang harus diikuti untuk meringkas paragraf di atas, sambil berusaha mempertahankan makna yang dimaksudkan, sebanyak mungkin.

Langkah 1: Ubah paragraf menjadi kalimat

Pertama, mari kita pisahkan paragraf menjadi kalimat yang sesuai. Cara terbaik untuk melakukan konversi adalah dengan mengekstrak kalimat setiap kali titik muncul.

1. Peter dan Elizabeth naik taksi untuk menghadiri pesta malam di kota
2. Saat di pesta, Elizabeth pingsan dan dilarikan ke rumah sakit
3. Sejak dia didiagnosis dengan cedera otak, dokter menyuruh Peter untuk tetap di sampingnya sampai dia sembuh
4. Oleh karena itu, Peter tinggal bersamanya di rumah sakit selama 3 hari tanpa pergi

Langkah 2: Pemrosesan teks

Selanjutnya, mari kita lakukan pemrosesan teks dengan menghapus kata-kata stop (kata-kata yang sangat umum dengan sedikit arti seperti “dan” dan “the”), angka, tanda baca, dan karakter khusus lainnya dari kalimat. Melakukan penyaringan membantu dalam menghilangkan informasi yang berlebihan dan tidak penting yang mungkin tidak memberikan nilai tambah apapun pada makna teks. Berikut hasil dari pengolahan teks tersebut:

1. Peter Elizabeth naik taksi menghadiri pesta malam kota
2. Pihak Elizabeth runtuh rumah sakit terburu-buru
3. Mendiagnosis cedera otak Dokter menyuruh Peter tetap di samping sembuh
4. Peter tinggal beberapa hari di rumah sakit tanpa pergi

Langkah 3: Tokenisasi

Tokenisasi kalimat dilakukan untuk mendapatkan semua kata yang ada dalam kalimat. Berikut adalah daftar kata-katanya:

['peter', 'elizabeth', 'took', 'taxi', 'attend', 'night', 'party', 'city', 'party', 'elizabeth', 'collapse', 'rush', 'hospital', 'diagnose', 'brain', 'injury', 'doctor', 'told', 'peter', 'stay', 'besides', 'get', 'well', 'peter', 'stayed', 'hospital', 'days', 'without', 'leaving']

Langkah 4: Evaluasi frekuensi kemunculan berbobot dari kata-kata tersebut

Setelah itu, mari kita hitung frekuensi kemunculan berbobot dari semua kata. Untuk mencapai hal ini, bagi frekuensi kemunculan setiap kata dengan frekuensi kata yang paling sering muncul dalam paragraf, yaitu "Peter" yang muncul tiga kali.

Berikut adalah tabel yang memberikan frekuensi kemunculan tertimbang dari masing-masing kata.

Tabel 5.1 Tabel Frekuensi kemunculan kata

Kata	Frekuensi	Frekuensi Tertimbang
peter	3	1
elizabeth	2	0.67
mengambil	1	0.33
taksi	1	0.33
menghadiri	1	0.33
malam	1	0.33
pesta	2	0.67
kota	1	0.33
jatuh	1	0.33
buru-buru	1	0.33
Rumah Sakit	2	0.67
mendiagnosis	1	0.33

otak	1	0.33
cedera	1	0.33
dokter	1	0.33
diberitahu	1	0.33
tinggal	2	0.67
selain	1	0.33
Dapatkan	1	0.33
baik	1	0.33
hari	1	0.33
tanpa	1	0.33
pergi	1	0.33

Langkah 5: Gantikan kata-kata dengan frekuensi tertimbangya

Mari kita gantikan setiap kata yang ditemukan dalam kalimat asli dengan frekuensi tertimbangya. Kemudian, kami akan menghitung jumlahnya. Karena frekuensi bobot dari kata-kata yang tidak penting, seperti kata-kata berhenti dan karakter khusus, yang telah dihapus selama tahap pemrosesan, adalah nol, maka tidak perlu menambahkannya.

Tabel 5.2 Perhitungan frekuensi berbobot

No	Tambahkan frekuensi berbobot	Jumlah	
1	Peter dan Elizabeth naik taksi untuk menghadiri pesta malam di kota	$1 + 0,67 + 0,33 + 0,33 + 0,33 + 0,33 + 0,67 + 0,33$	3.99
2	Saat di pesta, Elizabeth pingsan dan dilarikan ke rumah sakit	$0,67 + 0,67 + 0,33 + 0,33 + 0,67$	2.67

3	Karena dia didiagnosis mengalami cedera otak, dokter menyuruh Peter untuk tetap di sampingnya sampai dia sembuh.	$0,33 + 0,33 + 0,33 + 0,33 + 1 + 0,33 + 0,33 + 0,33 + 0,33 + 0,33$	3.97
4	Oleh karena itu, Peter tinggal bersamanya di rumah sakit selama 3 hari tanpa pergi	$1 + 0,67 + 0,67 + 0,33 + 0,33 + 0,33$	3.33

Dari penjumlahan frekuensi tertimbang kata-kata tersebut, kita dapat menyimpulkan bahwa kalimat pertama memiliki bobot paling besar dalam paragraf. Oleh karena itu, ini dapat memberikan ringkasan perwakilan terbaik tentang apa isi paragraf tersebut.

Selanjutnya, jika kalimat pertama digabungkan dengan kalimat ketiga, yang merupakan kalimat paling berbobot kedua dalam paragraf, ringkasan yang lebih baik dapat dibuat.

Contoh di atas hanya memberikan ilustrasi dasar tentang cara melakukan peringkasan teks berbasis ekstraksi dalam pembelajaran mesin. Sekarang, mari kita lihat bagaimana kita dapat menerapkan konsep di atas dalam membuat generator ringkasan dunia nyata.

Ringkasan teks dari artikel Wikipedia

Mari kita mengotori tangan kita dengan membuat ringkasan teks yang dapat mempersingkat informasi yang ditemukan dalam artikel web yang panjang. Agar semuanya tetap sederhana, selain dari toolkit NLTK Python, kami tidak akan menggunakan pustaka pembelajaran mesin lainnya.

Berikut adalah cetak biru kode dari summarizer:


```

# Creating a dictionary for the word frequency
table
frequency_table = _create_dictionary_table(article)

# Tokenizing the sentences
sentences = sent_tokenize(article)

# Algorithm for scoring a sentence by its words
sentence_scores =
_calculate_sentence_scores(sentences,
frequency_table)

# Getting the threshold
threshold =
_calculate_average_score(sentence_scores)

# Producing the summary
article_summary = _get_article_summary(sentences,
sentence_scores, 1.5 * threshold)

print(article_summary)

```

Berikut langkah-langkah untuk membuat ringkasan teks sederhana dengan Python.

Langkah 1: Mempersiapkan data

Dalam contoh ini, kita ingin merangkum informasi yang ditemukan di [ini](#) artikel Wikipedia, yang hanya memberikan gambaran peristiwa penting selama abad ke-20. Untuk memungkinkan kami mengambil teks artikel, kami akan menggunakan [perpustakaan BeautifulSoup](#). Berikut adalah kode untuk mengorek konten artikel:

```

import bs4 as BeautifulSoup
import urllib.request

```

```

# Fetching the content from the URL
fetched_data =
urllib.request.urlopen('https://en.wikipedia.org/wi
ki/20th_century')

article_read = fetched_data.read()

# Parsing the URL content and storing in a variable
article_parsed =
BeautifulSoup.BeautifulSoup(article_read, 'html.pars
er')

# Returning <p> tags
paragraphs = article_parsed.find_all('p')

article_content = ''

# Looping through the paragraphs and adding them to
the variable
for p in paragraphs:
    article_content += p.text

```

Pada kode di atas, kita mulai dengan mengimpor pustaka penting untuk mengambil data dari halaman web. The BeautifulSoup perpustakaan digunakan untuk parsing halaman sedangkan urllib perpustakaan digunakan untuk menghubungkan ke halaman dan mengambil HTML.

BeautifulSoup mengonversi teks yang masuk menjadi karakter Unicode dan teks keluar menjadi karakter UTF-8, sehingga Anda tidak perlu repot mengelola penyandiaksaraan rangkaian karakter yang berbeda saat mengambil teks dari web.

Kami akan menggunakan `urlopen` fungsi dari `urllib.request` utilitas untuk membuka halaman web. Kemudian, kita akan menggunakan `read` fungsi tersebut untuk membaca objek data salinan. Untuk memarsing data, kita akan memanggil `BeautifulSoup` objek tersebut dan meneruskan dua parameter padanya; yaitu, `article_readdan.html.parser`.

The `find_all` Fungsi digunakan untuk mengembalikan semua `<p>` elemen hadir dalam HTML. Selain itu, menggunakan `.text` memungkinkan kita untuk memilih hanya teks yang ditemukan di dalam `<p>` elemen.

Langkah 2: Memproses data

Untuk memastikan data tekstual yang dihapus sebebaskan mungkin dari noise, kami akan melakukan beberapa pembersihan teks dasar. Untuk membantu kami melakukan pemrosesan, kami akan mengimpor daftar stopwords dari pustaka `nlTK` .

Kami juga akan mengimpor `PorterStemmer` , yang merupakan algoritme untuk mereduksi kata ke dalam bentuk akarnya. Misalnya, membersihkan , dibersihkan , dan bersih dapat dikurangi dengan akar bersih .

Selanjutnya, kita akan membuat tabel kamus yang memiliki frekuensi kemunculan setiap kata dalam teks. Kami akan mengulang teks dan kata yang sesuai untuk menghilangkan kata-kata berhenti.

Kemudian, kami akan memeriksa apakah kata-kata tersebut ada di `frequency_table`. Jika kata tersebut sebelumnya tersedia dalam kamus, nilainya akan diperbarui dengan 1. Jika tidak, jika kata tersebut dikenali untuk pertama kalinya, nilainya disetel ke 1.

Misalnya, tabel frekuensi akan terlihat seperti berikut:

Tabel 5.3 Frekuensi Kata

Kata	Frekuensi
abad	7
dunia	4
Amerika Serikat	3
komputer	1

Ini kodenya: `from nltk.corpus import stopwords`

```
from nltk.stem import PorterStemmer
def _create_dictionary_table(text_string) -> dict:
    # Removing stop words
    stop_words = set(stopwords.words("english"))
    words = word_tokenize(text_string)

    # Reducing words to their root form
    stem = PorterStemmer()

    # Creating dictionary for the word frequency
    table
    frequency_table = dict()
    for wd in words:
        wd = stem.stem(wd)
        if wd in stop_words:
            continue
        if wd in frequency_table:
```

```
        frequency_table[wd] += 1
    else:
        frequency_table[wd] = 1
    return frequency_table
```

Langkah 3: Tokenisasi artikel menjadi kalimat

Untuk membagi `article_content` menjadi satu set kalimat, kita akan menggunakan metode **bawaan** dari pustaka **nlk**.

```
from nltk.tokenize import word_tokenize,
sent_tokenize
sentences = sent_tokenize(article)
```

Langkah 4: Menemukan frekuensi tertimbang dari kalimat

Untuk mengevaluasi skor setiap kalimat dalam teks, kami akan menganalisis frekuensi kemunculan setiap istilah. Dalam kasus ini, kita akan menilai setiap kalimat dengan kata-katanya; yaitu menambahkan frekuensi setiap kata penting yang ditemukan dalam kalimat.

Perhatikan kode berikut:

```
def _calculate_sentence_scores(sentences,
frequency_table) -> dict:
```

```

# Algorithm for scoring a sentence by its words
sentence_weight = dict()

for sentence in sentences:
    sentence_wordcount =
(len(word_tokenize(sentence)))
    sentence_wordcount_without_stop_words = 0
    for word_weight in frequency_table:
        if word_weight in sentence.lower():

sentence_wordcount_without_stop_words += 1
        if sentence[:7] in sentence_weight:
            sentence_weight[sentence[:7]]
+= frequency_table[word_weight]
        else:
            sentence_weight[sentence[:7]] =
frequency_table[word_weight]

    sentence_weight[sentence[:7]] =
sentence_weight[sentence[:7]] /
sentence_wordcount_without_stop_words

return sentence_weight

```

Yang penting, untuk memastikan kalimat panjang tidak memiliki skor tinggi yang tidak perlu dibandingkan kalimat pendek, kami membagi setiap skor kalimat dengan jumlah kata yang ditemukan dalam kalimat itu.

Selain itu, untuk mengoptimalkan memori kamus, kami menambahkan **kalimat [: 7] secara** sewenang-wenang, yang mengacu pada 7 karakter pertama di setiap kalimat. Namun, untuk dokumen yang lebih panjang, di mana Anda kemungkinan besar akan menemukan kalimat dengan kalimat yang sama terlebih dahulu `n_chars`, lebih baik menggunakan fungsi hash atau fungsi indeks cerdas untuk memperhitungkan kasus tepi tersebut dan menghindari benturan.

Langkah 5: Menghitung ambang kalimat

Untuk lebih menyesuaikan jenis kalimat yang memenuhi syarat untuk diringas, kami akan membuat skor rata-rata untuk kalimat tersebut. Dengan threshold ini, kita bisa menghindari pemilihan kalimat dengan skor yang lebih rendah dari skor rata-rata.

Ini kodenya:

```
def _calculate_average_score(sentence_weight) ->
int:

    # Calculating the average score for the
sentences

    sum_values = 0
    for entry in sentence_weight:
        sum_values += sentence_weight[entry]
```

```
# Getting sentence average value from source
text

average_score = (sum_values /
len(sentence_weight))

return average_score
```

Langkah 6: Mendapatkan ringkasan

Terakhir, karena kita memiliki semua parameter yang diperlukan, sekarang kita dapat membuat ringkasan untuk artikel tersebut. Ini kodenya:

```
def _get_article_summary(sentences,
sentence_weight, threshold):
    sentence_counter = 0
    article_summary = ''

    for sentence in sentences:
        if sentence[:7] in sentence_weight and
sentence_weight[sentence[:7]] >= (threshold):
            article_summary += " " + sentence
            sentence_counter += 1

    return article_summary
```


Membungkus

Berikut adalah gambar yang menampilkan alur kerja untuk membuat generator ringkasan.



Gambar 5.2 Alur kerja dasar untuk membuat algoritme peringkasan

Berikut ini seluruh kode untuk ringkasan teks ekstraktif sederhana dalam pembelajaran mesin:

```
#importing libraries
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize,
sent_tokenize
import bs4 as BeautifulSoup
import urllib.request

#fetching the content from the URL
```

```

fetched_data =
urllib.request.urlopen('https://en.wikipedia.org/wiki/20th_century')

article_read = fetched_data.read()

#parsing the URL content and storing in a variable
article_parsed =
BeautifulSoup.BeautifulSoup(article_read, 'html.parser')

#returning <p> tags
paragraphs = article_parsed.find_all('p')
article_content = ''

#looping through the paragraphs and adding them to
the variable
for p in paragraphs:
    article_content += p.text

def _create_dictionary_table(text_string) -> dict:

    #removing stop words
    stop_words = set(stopwords.words("english"))
    words = word_tokenize(text_string)

    #reducing words to their root form
    stem = PorterStemmer()

```

```

    #creating dictionary for the word frequency
table
    frequency_table = dict()
    for wd in words:
        wd = stem.stem(wd)
        if wd in stop_words:
            continue
        if wd in frequency_table:
            frequency_table[wd] += 1
        else:
            frequency_table[wd] = 1
    return frequency_table

def _calculate_sentence_scores(sentences,
frequency_table) -> dict:

    #algorithm for scoring a sentence by its words
    sentence_weight = dict()

    for sentence in sentences:
        sentence_wordcount =
(len(word_tokenize(sentence)))
        sentence_wordcount without stop words = 0
        for word_weight in frequency_table:
            if word_weight in sentence.lower():

```

```

sentence_wordcount_without_stop_words += 1
        if sentence[:7] in sentence_weight:

sentence_weight[sentence[:7]] +=
frequency_table[word_weight]
        else:

sentence_weight[sentence[:7]] =
frequency_table[word_weight]

        sentence_weight[sentence[:7]] =
sentence_weight[sentence[:7]] /
sentence_wordcount_without_stop_words
        return sentence_weight

def _calculate_average_score(sentence_weight)
-> int:

    #calculating the average score for the
sentences
    sum_values = 0
    for entry in sentence_weight:
        sum_values += sentence_weight[entry]

```

```
        #getting sentence average value from
source text
        average_score = (sum_values /
len(sentence_weight))

        return average_score

def _get_article_summary(sentences,
sentence_weight, threshold):
    sentence_counter = 0
    article_summary = ''

    for sentence in sentences:
        if sentence[:7] in sentence_weight and
sentence_weight[sentence[:7]] >= (threshold):
            article_summary += " " + sentence
            sentence_counter += 1

    return article_summary

def _run_article_summary(article):

    #creating a dictionary for the word
frequency table
    frequency_table =
    _create_dictionary_table(article)
```

```
#tokenizing the sentences
sentences = sent_tokenize(article)
#algorithm for scoring a sentence by its
words
    sentence_scores =
    _calculate_sentence_scores(sentences,
    frequency_table)

    #getting the threshold
    threshold =
    _calculate_average_score(sentence_scores)

#producing the summary
    article_summary =
    _get_article_summary(sentences,
    sentence_scores, 1.5 * threshold)

    return article_summary

if __name__ == '__main__':
    summary_results =
    _run_article_summary(article_content)
    print(summary_results)
```

Dalam kasus ini, kami menerapkan ambang 1,5x dari skor rata-rata. Ini adalah nilai [hyperparameter](#) yang memberikan hasil yang baik bagi kami setelah beberapa percobaan. Tentu saja, Anda dapat menyesuaikan nilai sesuai dengan preferensi Anda dan meningkatkan hasil peringkasan. Berikut adalah gambar versi ringkasan artikel Wikipedia.

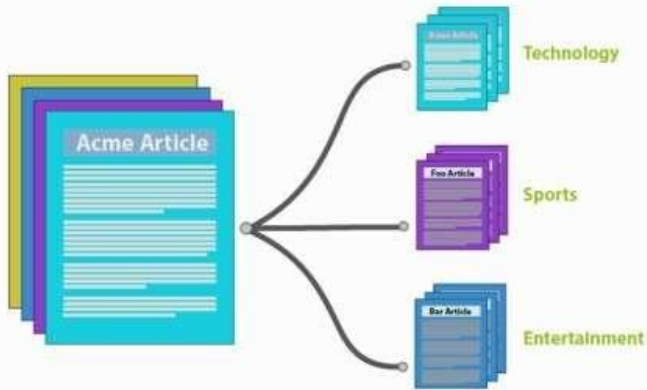
It is distinct from the century known as the 1980s which began on January 1, 1980 and ended on December 31, 1999. Terms like ideology, world war, genocide, and nuclear war entered common usage. Humans explored space for the first time, taking their first footsteps on the Moon. However, these same wars resulted in the destruction of the imperial system. The victorious Bolsheviks then established the Soviet Union, the world's first communist state. In total, World War II left some 68 million people dead. During the century, the social taboo of sexism fell. Communications and information technology, transportation technology, and medical advances had radically altered daily lives. Since the US was in a dominant position, a major part of the process was Americanization. Terrorism, dictatorship, and the spread of nuclear weapons were pressing global issues. Millions were infected with HIV, the virus which causes AIDS. This includes deaths caused by wars, genocide, politicide and mass murders. Later in the 20th century, the development of computers led to the establishment of a theory of computation

Seperti yang Anda lihat, menjalankan kode meringkas artikel Wikipedia yang panjang dan memberikan gambaran sederhana tentang kejadian utama di abad ke-20.

Meskipun demikian, generator ringkasan dapat ditingkatkan untuk membuatnya lebih baik dalam menghasilkan ringkasan teks tebal yang ringkas dan tepat.

BAB VII. DOCUMENT CLASSIFICATION

Klasifikasi dokumen adalah contoh Machine Learning (ML) dalam bentuk Natural Language Processing (NLP). Dengan mengklasifikasikan teks, kami bertujuan untuk menetapkan satu atau lebih kelas atau kategori ke dokumen, sehingga lebih mudah untuk dikelola dan diurutkan. Ini sangat berguna untuk penerbit, situs berita, blog, atau siapa pun yang berurusan dengan banyak konten. Secara umum, ada dua kelas teknik ML: Supervised dan Unsupervised. Dalam metode yang diawasi, model dibuat berdasarkan pada set pelatihan. Kategori sudah ditentukan sebelumnya dan dokumen dalam set data pelatihan ditandai secara manual dengan satu atau beberapa label kategori. Klasifikasi kemudian dilatih tentang dataset yang artinya dapat memprediksi kategori dokumen baru sejak saat itu. Bergantung pada algoritma klasifikasi atau strategi yang digunakan, classifier mungkin juga memberikan ukuran kepercayaan untuk menunjukkan seberapa yakin label klasifikasi itu benar.



Gambar 7.1 Ilustrasi Klasifikasi Dokumen sederhana

Kita dapat menggunakan kata-kata di dalam dokumen sebagai "fitur" untuk membantu kami memprediksi klasifikasi dokumen. Sebagai contoh, kita dapat memiliki tiga dokumen yang sangat pendek dan sepele dalam rangkaian pelatihan kita seperti yang ditunjukkan di bawah ini

Tabel 7.1 klasifikasi 3 dokumen

Reference Document Class 1	Reference Document Class 2	Reference Document Class 3
Some tigers live in the zoo	Green is a color	Go to New York city

Untuk mengklasifikasikan dokumen-dokumen ini, kami akan mulai dengan mengambil semua kata dalam tiga dokumen dalam set pelatihan kami dan membuat tabel atau vektor dari kata-kata ini.

(some,tigers,live,in,the,zoo,green,is,a,color,go,to,new,york,city)
classkelas

Kemudian untuk setiap dokumen pelatihan, kami akan membuat vektor dengan menetapkan 1 jika ada kata dalam dokumen pelatihan dan 0 jika tidak, menandai dokumen dengan kelas yang sesuai

Kemudian untuk setiap dokumen pelatihan, kami akan membuat vektor dengan menetapkan 1 jika ada kata dalam dokumen pelatihan dan 0 jika tidak, menandai dokumen dengan kelas yang sesuai sebagai berikut:

Tabel 7.2 Penandaan Dokumen

	some	tigers	live	in	the	zoo	green	is	a	color	go	to	new	york	city	
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	class 1
0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	class 2
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	class 3

Ketika dokumen baru tanpa tanda tiba untuk klasifikasi dan berisi kata-kata "Oranye adalah warna" kami akan membuat vektor kata untuk itu dengan menandai kata-kata yang ada dalam vektor klasifikasi kami.

Tabel 7.3 Penandaak kata

some	tigers	live	in	the	zoo	green	is	a	color	go	to	new	york	city	
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	unknown class

Jika kita membandingkan vektor ini untuk dokumen kelas yang tidak diketahui, dengan vektor yang mewakili tiga kelas dokumen kita, kita dapat melihat bahwa vektor ini sangat mirip dengan vektor untuk dokumen kelas 2, seperti yang ditunjukkan di bawah ini:

< 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0 > Unknown class

< 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 > class 1 (6 matching terms)

< 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0 > class 2 (14 matching terms - winner!!)

< 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1 > class 3 (7 matching terms)

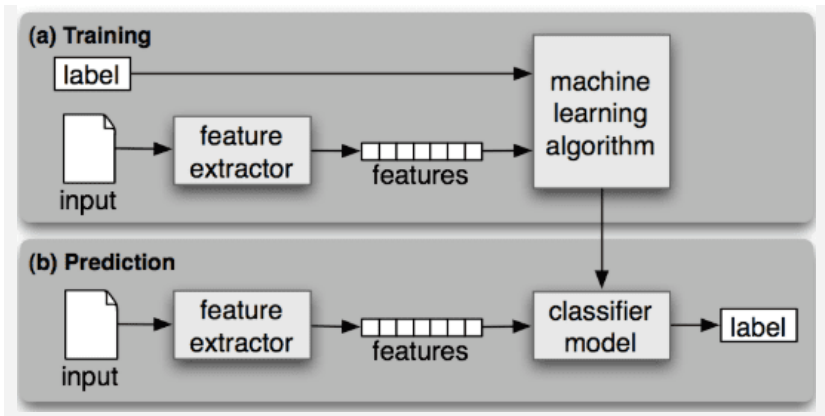
Maka dimungkinkan untuk memberi label pada dokumen baru sebagai dokumen kelas 2 dengan tingkat kepercayaan yang memadai. Ini adalah contoh yang sangat sederhana namun umum dari metode Pemrosesan Bahasa Alami secara statistik.

Jenis Klasifikasi dan Teknik Dokumen

1. Klasifikasi Dokumen yang Diawasi
2. Klasifikasi Dokumen Tidak Dibimbing

(i) Klasifikasi Dokumen yang Diawasi:

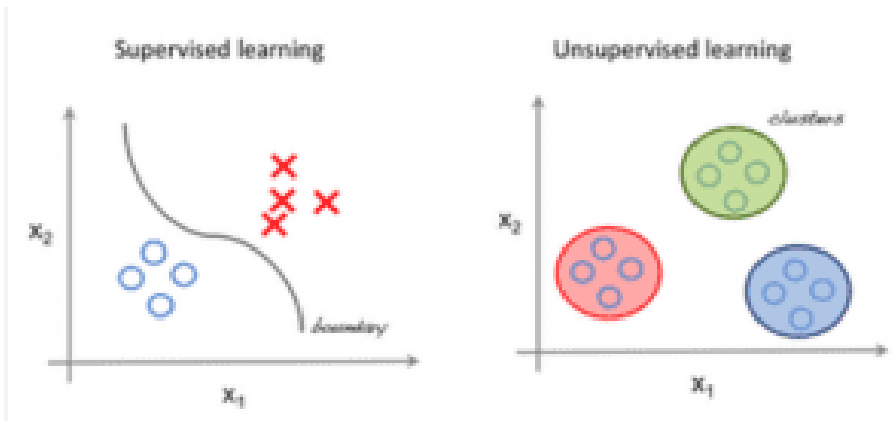
Dalam klasifikasi terbimbing, mekanisme eksternal (seperti umpan balik manusia) memberikan informasi yang benar tentang klasifikasi dokumen.



Gambar 7.2 Klasifikasi Supervised

(ii) Klasifikasi Dokumen yang Tidak Diawasi:

Dalam klasifikasi dokumen tanpa pengawasan, disebut juga clustering dokumen, dimana klasifikasi harus dilakukan seluruhnya tanpa mengacu pada informasi eksternal. Pengelompokan dokumen melibatkan penggunaan deskriptor dan ekstraksi deskriptor. Deskriptor adalah kumpulan kata yang mendeskripsikan konten di dalam cluster. Pengelompokan dokumen umumnya dianggap sebagai proses terpusat. Contoh pengelompokan dokumen termasuk pengelompokan dokumen web untuk pengguna pencarian.



Gambar 7.3 Klasifikasi yang Diawasi vs Tidak Diawasi

Aplikasi Pengelompokan Dokumen

Penerapan document clustering dapat dikategorikan menjadi dua jenis yaitu online dan offline. Aplikasi online biasanya terkendala oleh masalah efisiensi jika dibandingkan dengan aplikasi offline. Pengelompokan teks dapat digunakan untuk berbagai tugas, seperti mengelompokkan dokumen serupa (berita, tweet, dll.) Dan analisis umpan balik pelanggan / karyawan, menemukan subjek implisit yang bermakna di semua dokumen.

Algoritma

Secara umum, ada dua algoritma umum.

(i) Yang pertama adalah algoritma berbasis hierarki, yang mencakup tautan tunggal, tautan lengkap, rata-rata grup, dan metode Ward. Dengan menggabungkan atau membagi, dokumen dapat dikelompokkan menjadi struktur hierarki, yang cocok untuk

penelusuran. Namun, algoritma seperti itu biasanya mengalami masalah efisiensi.

(ii) Algoritma lain dikembangkan menggunakan algoritma K-means dan variannya. Umumnya, algoritme hierarkis menghasilkan informasi yang lebih mendalam untuk analisis terperinci, sementara algoritme yang didasarkan pada varian algoritme K-Means lebih efisien dan memberikan informasi yang cukup untuk sebagian besar tujuan. Algoritma ini selanjutnya dapat diklasifikasikan sebagai algoritma pengelompokan keras atau lunak.

Pengelompokan keras menghitung tugas berat - setiap dokumen adalah anggota dari satu kluster. Penetapan algoritme pengelompokan lunak bersifat lunak - penugasan dokumen adalah distribusi ke semua kluster. Dalam tugas lunak, dokumen memiliki keanggotaan pecahan di beberapa cluster. Metode pengurangan dimensi dapat dianggap sebagai subtype pengelompokan lunak; untuk dokumen, ini termasuk pengindeksan semantik laten (dekomposisi nilai singular terpotong pada histogram istilah) dan model topik

Dalam praktiknya, pengelompokan dokumen sering kali mengambil langkah-langkah berikut:

1. Tokenisasi

Tokenisasi adalah proses penguraian data teks menjadi unit yang lebih kecil (token) seperti kata dan frasa.

2. Stemming dan Lemmatization

Token yang berbeda mungkin menjalankan informasi serupa (misalnya tokenisasi dan tokenisasi). Dan Anda dapat menghindari menghitung informasi serupa berulang kali dengan mengurangi semua token ke bentuk dasarnya menggunakan berbagai kamus stemming dan lemmatization.

3. Menghapus Kata Berhenti dan Tanda Baca

Beberapa token kurang penting dari yang lain. Misalnya, kata-kata umum seperti "the" mungkin tidak terlalu membantu untuk mengungkapkan karakteristik esensial dari sebuah teks. Jadi biasanya ada baiknya untuk menghilangkan kata-kata henti dan tanda baca sebelum melakukan analisis lebih lanjut.

4. Menghitung frekuensi istilah atau tf-idf

Setelah melakukan pra-pemrosesan data teks, Anda dapat melanjutkan untuk menghasilkan fitur. Untuk pengelompokan dokumen, salah satu cara paling umum untuk menghasilkan fitur untuk dokumen adalah menghitung frekuensi jangka dari semua tokennya. Meski tidak sempurna, frekuensi ini biasanya dapat memberikan beberapa petunjuk tentang topik dokumen.

5. Pengelompokan

Anda kemudian dapat mengelompokkan dokumen yang berbeda berdasarkan fitur yang telah dibuat.

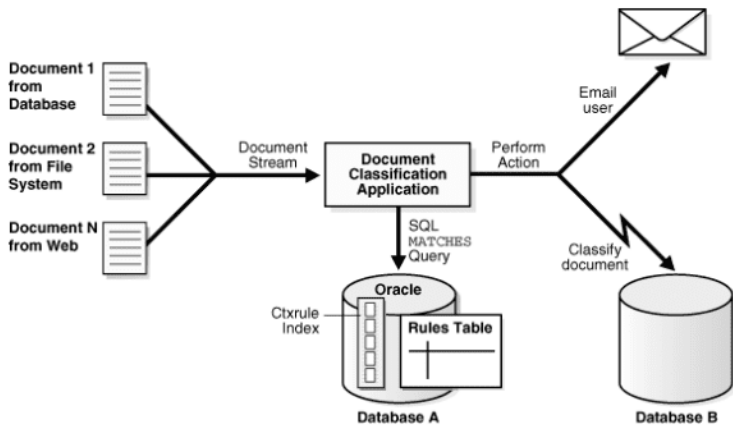
6. Evaluasi dan Visualisasi

Akhirnya, model pengelompokan dapat dinilai dengan berbagai metrik. Dan terkadang membantu untuk memvisualisasikan hasil dengan memplot cluster ke dalam ruang dimensi rendah (dua).

Teknik Klasifikasi Dokumen Otomatis Termasuk:

- Expectation maximization (EM)
- [Naive Bayes classifier](#)
- Instantaneously trained neural networks
- Latent semantic indexing

- Support vector machines (SVM)
- Artificial neural network
- K-nearest neighbour algorithms
- Decision trees such as ID3 or C4.5
- Concept Mining
- Rough set-based classifier
- Soft set-based classifier
- Multiple-instance learning
- Natural language processing approaches



Gambar 7.4 Gambaran Umum Aplikasi Klasifikasi Dokumen

BAB VIII.TOOLS TEXT MINING DENGAN STUDI KASUS

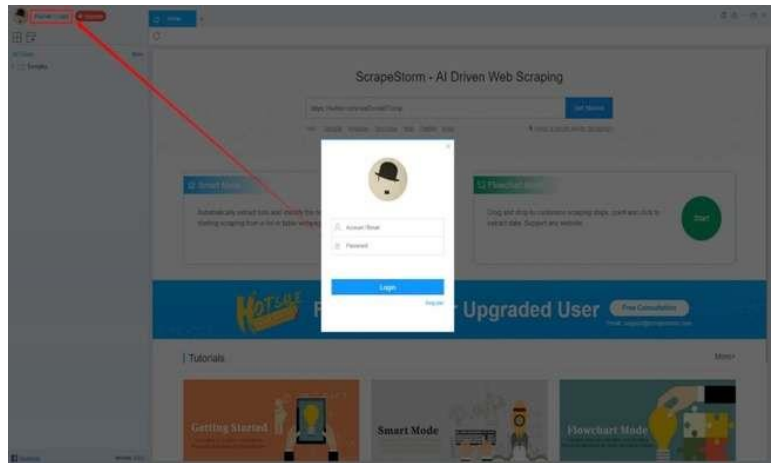
Studi Kasus : Analisis Teks tentang Komentar Layanan Lazada vs Shopee menggunakan Tools Orange

A. Scapping Menggunakan ScrapeStorm

ScrapeStorm (www.scrapestorm.com) adalah alat pengikis web visual yang didukung AI, yang dapat digunakan untuk mengekstrak data dari hampir semua situs web tanpa menulis kode apa pun. Ini kuat dan sangat mudah digunakan. Untuk pengguna yang berpengalaman dan tidak berpengalaman, ini menyediakan dua mode gesekan yang berbeda (Mode Cerdas dan Mode Diagram Alir). ScrapeStorm mendukung sistem operasi Windows, Mac OS dan Linux. Anda dapat menyimpan data output dalam berbagai format termasuk Excel, HTML, Txt dan CSV. Selain itu, Anda dapat mengeksport data ke database dan situs web. Aplikasi ini hanya dapat dipakai jika terkoneksi dengan internet (Online). Berikut adalah langkah-langkah dalam meng*scraping* komentar youtube.

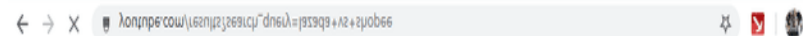
1. Download dan install scrapestorm, kemudian register dan log in.

1. Buka website scrapestorm (<https://www.scrapestorm.com/>), unduh dan install.
2. Tekan Register/ Login untuk daftar akun baru kemudian log in ke dalam scrapestorm.

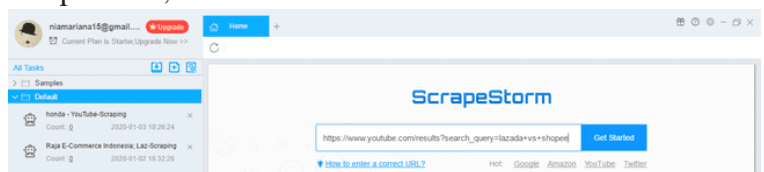


2. Buat sebuah *task*.

1. Copy URL dari video youtube yang dituju.



2. Masukkan kedalam kotak pencarian video dalam scrapestorm, kemudian tekan tombol **Get Started**.



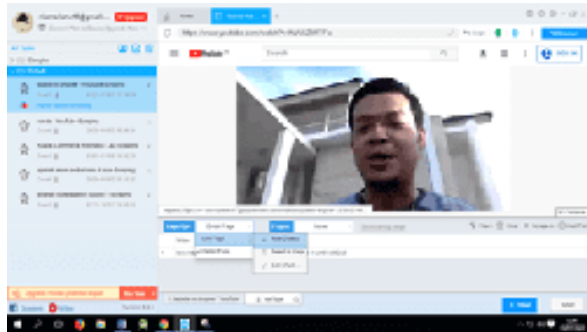
3. Setelah itu akan muncul tampilan dengan berbagai video dengan kata kunci yang telah dimasukkan di kotak pencarian. Di bagian bawah tampilan terdapat tabel berisi kolom yang terdapat dalam list video. Mulai dari judul, pembuat, waktu pembuatan, hingga durasi.
4. Kita dapat menghapus kolom yang tidak digunakan maupun mengubah nama kolom dengan mengklik kanan pada judul kolom yang diinginkan.



5. Setelah itu tekan kolom **Title_Link** kemudian pilih **scrape into** yang berada di samping opsi **Add Field**.



6. Kemudian akan muncul tampilan video teratas yang berada dalam list video. Scroll video ke bagian **Komentar**, kemudian pada bagian **Page Type** pilih opsi **List Page** → **Auto Detect**. Maka akan muncul kolom berisi baris komentar yang terdetek secara otomatis.

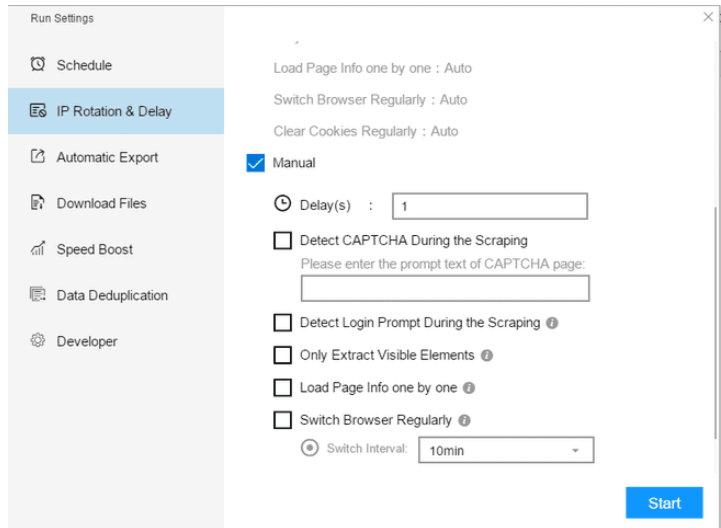


7. Setelah itu, tekan tombol **Start** yang berada di pojok kanan bawah di sebelah tombol save.

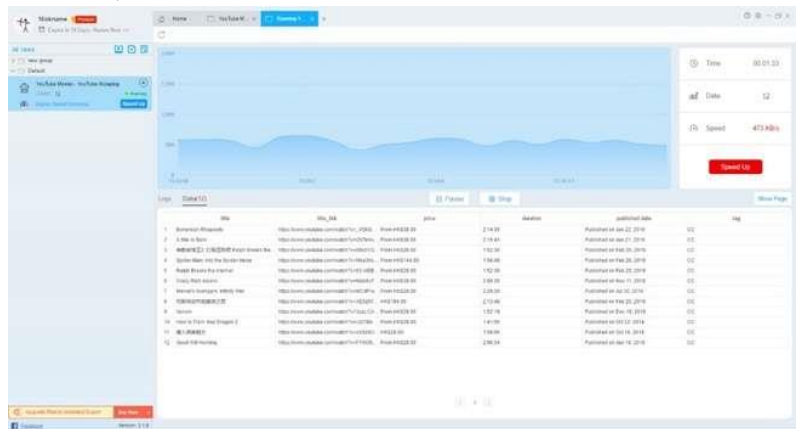


8. Kemudian kita akan masuk kedalam halaman setting sebelum proses scraping dijalankan. Dalam halaman ini terdapat beberapa pilihan yang perlu dipilih. Namun yang dirubah hanya pada pengaturan **IP Rotation & Delay**. Pada pengaturan ini kita pilih opsi **Manual** dengan mencentangkannya.

Setelah itu tekan tombol start.

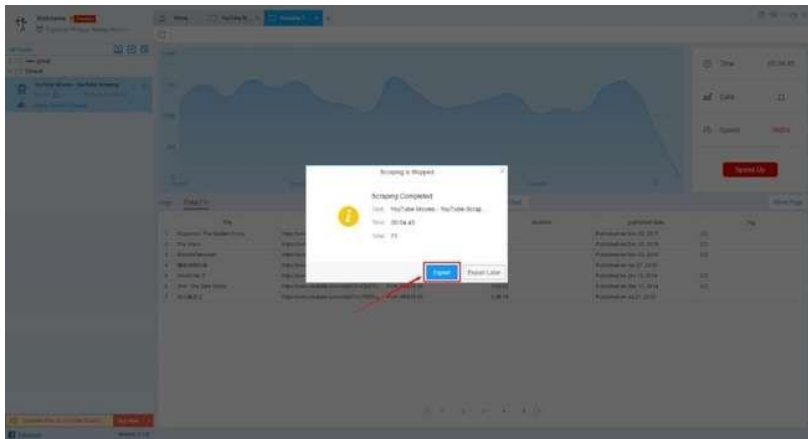


9. Setelah itu proses scraping data komentar sedang berlangsung. Jika data telah terscraping, maka akan muncul di kolom **Data**. Jika proses scraping/ data sudah cukup, tekan tombol **Stop**, maka proses akan berakhir, dan muncul

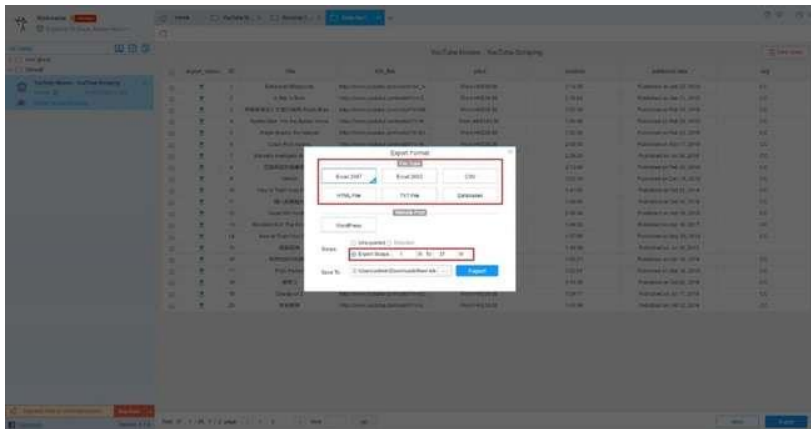


10. Tekan tombol **Export** untuk mengunduh data.

i

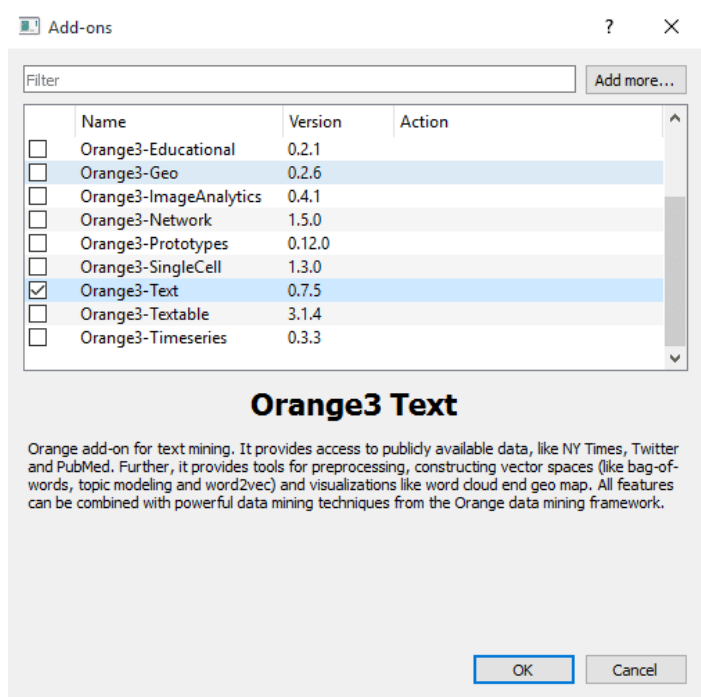


11. Pilih format data sesuai yang dibutuhkan.
Disarankan untuk memilih format **csv** agar mudah saat di sentimen analisis di orange.

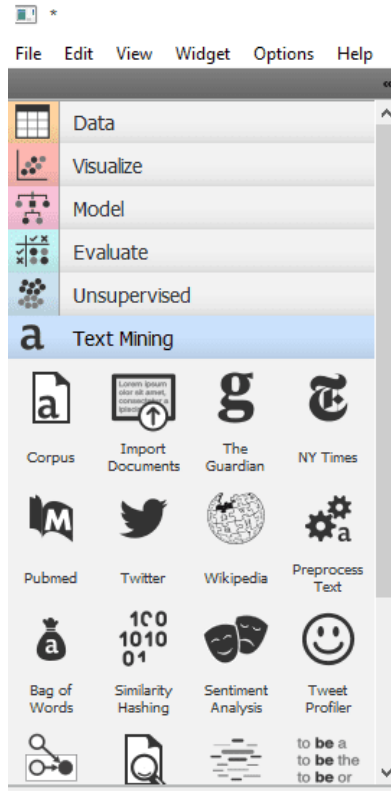


B. Analisa Teks Mining menggunakan Tools Orange

1. Hal pertama yang harus dilakukan adalah **menginstall add-on text** yang berada pada menu **Options >Add ons**. Kemudian dalam menu tersebut pilih widget **Orange3-Text**, kemudian tekan tombol **ok**. **Catatan: Pada saat menginstall widget, Laptop harus terkoneksi dengan WI-FI.**



2. Setelah terinstall, maka widget Text Mining akan muncul pada sisi kiri dashboard.

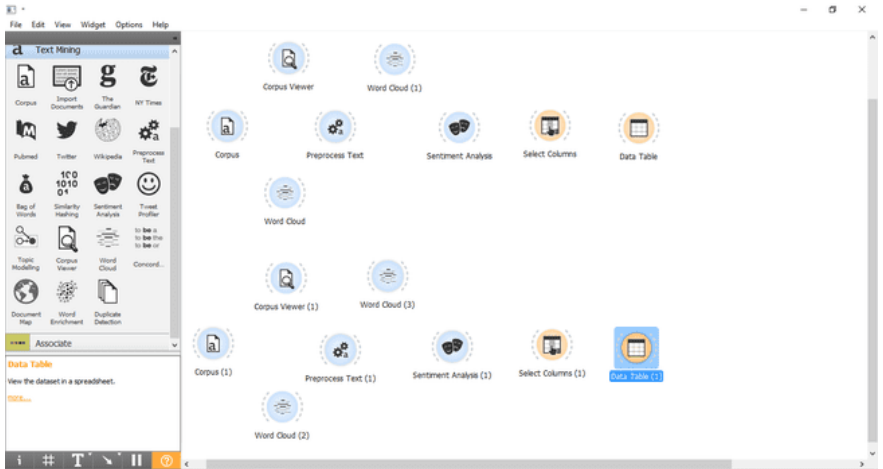


Select a widget to show its description.

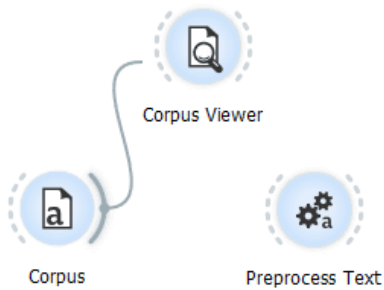
See [workflow examples](#), [YouTube tutorials](#), or open the [welcome screen](#).

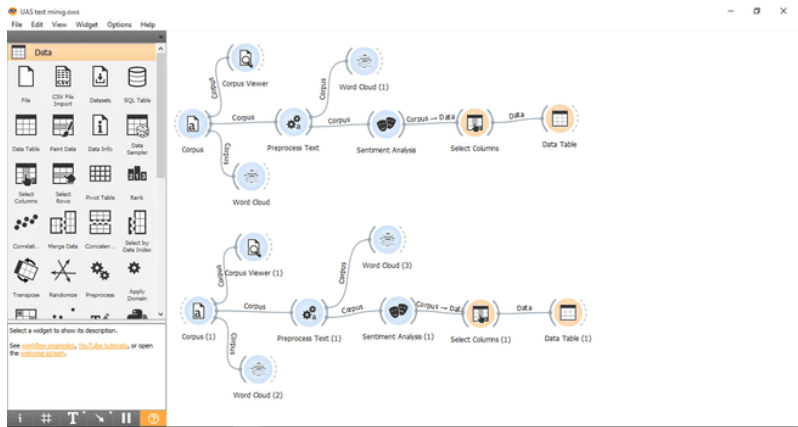


3. Kemudian klik Widget Corpus, Corpus Viewer, Preprocess Text, Word Cloud 2 kali, Sentimnt Analysis pada widget Text Mining, serta Widget Select Columns dan Data Table pada Widget Data. Lakukan sebanyak 2 kali utuk mengoah 2 data sehingga menjadi seperti gambar dibawah ini.



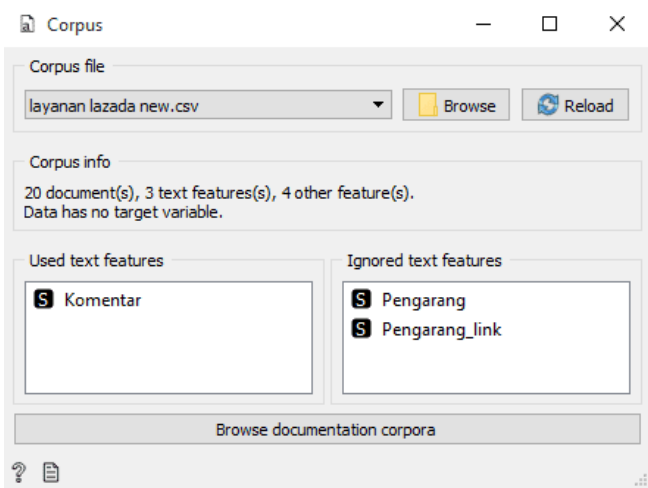
4. Koneksikan setiap widget dengan mengklik widget asal ke widget tujuan sehingga membentuk alur seperti berikut.



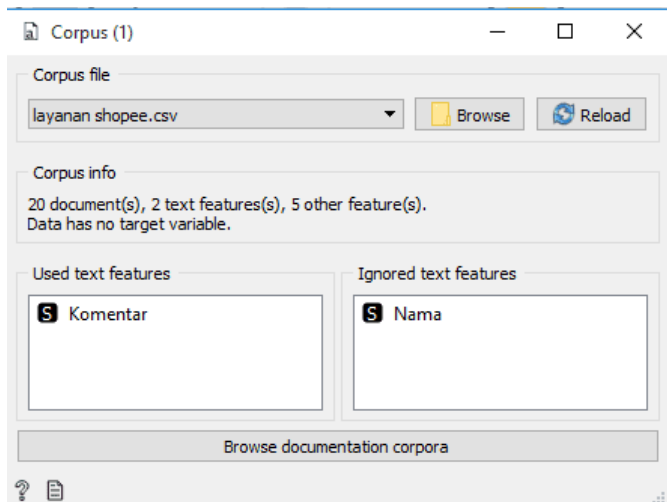


5. Masukkan dataset pada masing-masing corpus (dataset layanan lazada new pada corpus alur atas dan dataset layanan shopee pada alur bagian bawah) dengan menekan tombol **browse** pada sebelah kiri tombol reload untuk membuka dataset. Setelah itu tekan **reload** agar dataset dapat terdeteksi/set. Jika dataset terdeteksi maka akan muncul informasi tentang isi dari dataset seperti kolom yang berada dalam dataset, jumlah dokumen dalam dataset.

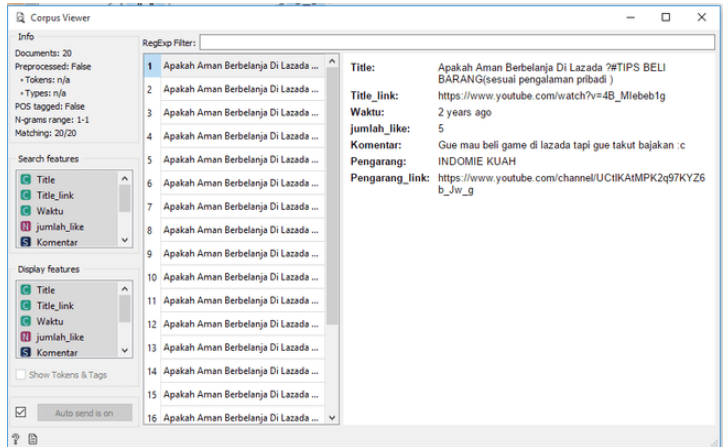
a) Dataset layanan lazada new



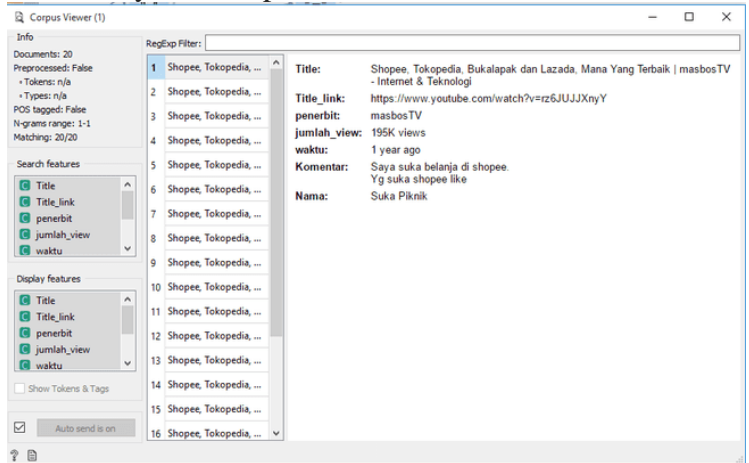
b) Dataset layanan shopee



6. Untuk melihat detail dari masing-masing dokumen pada dataset, klik widget **corpus viewer**.
 1. Dataset layanan lazada new



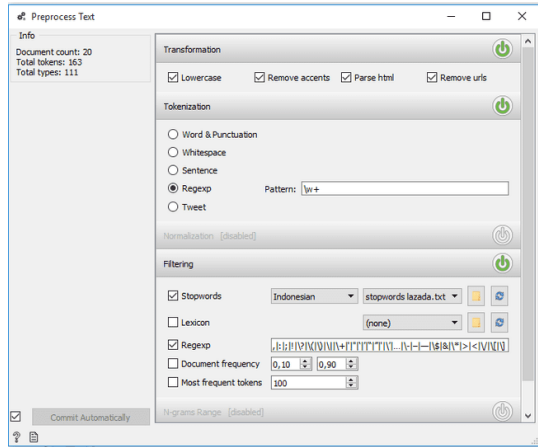
2. Dataset layanan shopee



3. Untuk melihat kata yang dominan pada dataset, klik widget wordcloud sebelum di preprocessing. Pada wordcloud ini kita juga dapat menentukan kata apa saja yang akan dibuang (stopword).

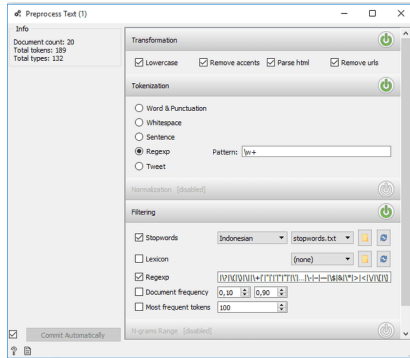
1. Dataset layanan lazada new

1. Text preprocessing layanan lazada new



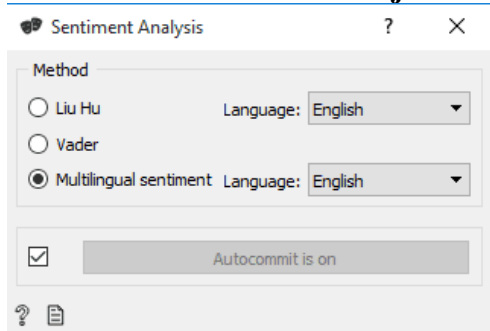
```
stopwords lazada - Notepad
File Edit Format View Help
ji
yang
nya
kita
emang
ya
bang
kalo
gue
sih
cod
bulan
tolong
dah
c
pengalamanguadilazada
vr
box
janganinstalllazada
jutaan
ngeplay
eh
gitu
dan
1jt
pikir2
kaki
ke
2tb
80
dr
3x
hp
cek
10
udah
<
```

2. Text preprocessing layanan shopee

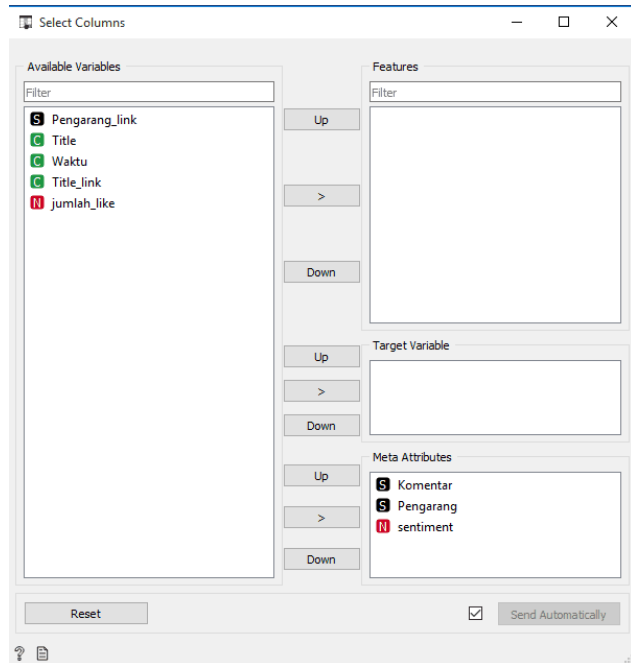


```
stopwords - Notepad
File Edit Format Vie
ji
yg
gue
ini
dan
klaw
gx
nya
ga
.
2
bli
4
atau
sih
jadi
jd
klo
mah
lah
lbih
tmpt
:)
gw
id
gua
klaw
mha
belix
...
suarax
...??
1
ane
hp
3
<
```

5. Setelah preprocessing, dataset akan di proses di widget **Sentiment Analysis** dimana widget ini memiliki 3 opsi metode analisis, antara lain ada Liu Hui, Vader, dan Multilingual Sentiment. Pilih **Multilingual Sentiment, dengan kondisi laptop harus terkoneksi internet saat menjalankannya.**



6. Kemudian setelah dari widget Sentiment, Pilih dahulu kolom yang akan dimunculkan dan dikelola. Kolom atribut yang akan diproses berada pada bagian **Meta Atributes di pojok kanan bawah. Untuk menambahkan atau mengurangi** kolom atribut, dapat dilakukan dengan mengklik dan tahan kolom yang ingin ditambahkan, letakkan pada kolom Meta Atributes.



7. Untuk Melihat Hasil Sentimen dari masing masing komentar, dapat dilihat pada widget **Data Table**. **Jika nilai sentimen bernilai 0 maka komentar termasuk kategori netral, jika nilai sentimen bernilai negatif (-) maka komentar bernilai negatif, dan jika nilai sentimen bernilai positif (tidak ada tanda -) maka komentar bernilai positif.**
 - i. Dataset layanan lazada new

2.

Info

20 instances (no missing values)
No features
No target variable.
3 meta attributes (no missing values)

Variables

Show variable labels (if present)
 Visualize numeric values
 Color by instance classes

Selection

Select full rows

Restore Original Order

Send Automatically

	Komentar	Pengarang	sentiment
1	Gue mau beli g...	INDOMIE KUAH	-8.33333
2	bukan ketipu a...	Bothz Kun	-12.5
3	PengalamanGu...	Nicholas DesTr...	5.26316
4	beli nya di laz...	R.R AIGHifari	11.1111
5	LAZADA eman...	iHYDEZT	0
6	mau aman? ba...	Dwi Prasatya	14.2857
7	Bang ada lapto...	Santri Cyber Sa...	9.09091
8	Kaget aku kira s...	Nurul Airifah	-7.69231
9	Pengen beli ka...	how to write	-11.1111
10	cara bayar dite...	My Name Is	-20
11	lazada itu akan ...	antok Pratama	-0.892857
12	elash emang a...	Ardy Winanda	0
13	Kenapa saya ser...	Dito 'cs84	3.8961
14	amanlah gw aja...	Delphin raydafa	-10
15	nama gamernya...	dhiaul	0
16	Cara bayarnya s...	Dicky Achmad	-6.66667
17	sangat kecewa ...	Sisil Noe	0
18	sistem cod itu ...	Project Studio	0
19	sama bang lam...	Raihan Farandy	0
20	Makasih bgt at...	amy satwami	11.1111

2. Dataset layanan shopee

Info

20 instances (no missing values)
No features
No target variable.
3 meta attributes (no missing values)

Variables

Show variable labels (if present)
 Visualize numeric values
 Color by instance classes

Selection

Select full rows

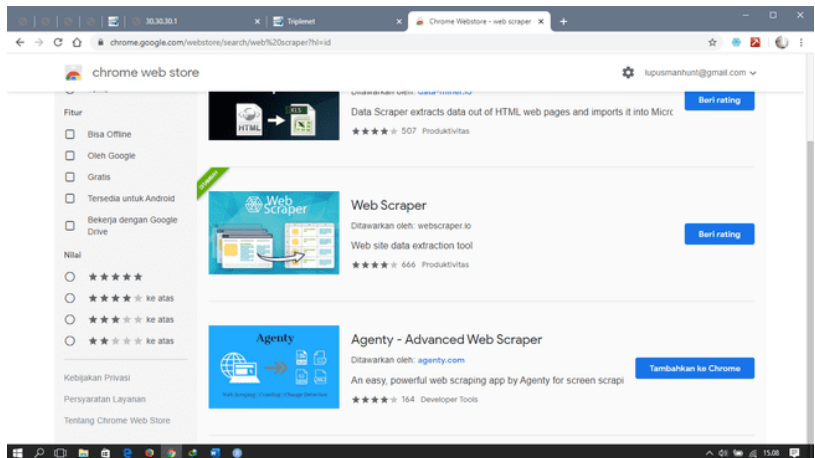
Restore Original Order

Send Automatically

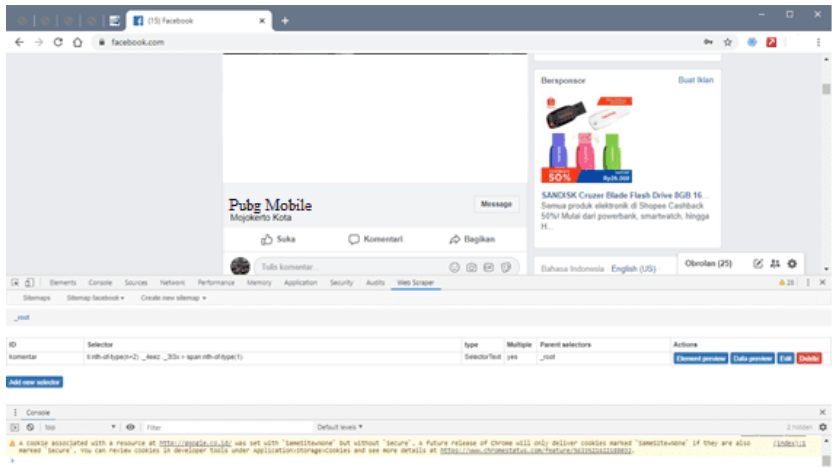
	Nama	Komentar	sentiment
1	Suka Piknik	Saya suka belan...	30
2	by aslan official	klaw aku mha b...	0
3	Arka axelsen	Lazada angkat j...	0
4	ARIFIN CHANN...	1.Kalau ane beli...	5.45455
5	Pepeng Jelek	Mana yang bag...	8.33333
6	Gadgetarian	KENAPA HARU...	4.7619
7	fathur rachman	Lazada la boy.....	-6.89655
8	Chairunnisa Dini	Shopee itu sud...	6.89655
9	khanyul RHECOX	Sy lbih pilih laz...	0
10	maria assumpta	Shopee ongkir...	0
11	Putra Pga	Kalau gw yg ho...	5.26316
12	Mecca Imut	Ada ongkir pun...	20.8333
13	eka novita soraya	Aku mah tokop...	14.2857
14	Rafi Putra	paling jelek buk...	0
15	Muhammad Sy...	Id buka lapak: ...	0
16	Aly Jaya	Tokped & shop...	-6.25
17	Indra Kusuma	Kelemahan- To...	-2.63158
18	Retzzamir Eran...	pengguna TOK...	16.6667
19	Adam JuLiano	Gw pilih lazada ...	-3.0303
20	bakul tape	Shopee lah ong...	-11.1111

Studi Kasus 2 : Analisa Sentiment Komentar Negatif dan Positif Di Facebook menggunakan Software Tools Program R

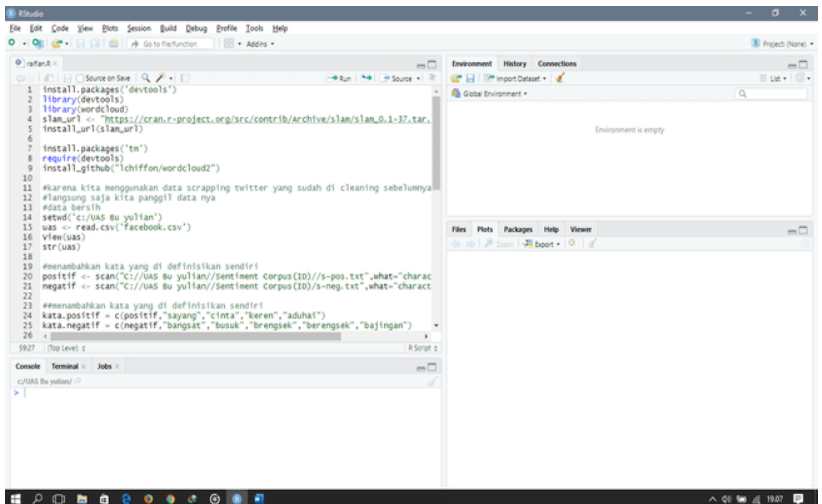
1. Langkah Pertama Yang harus disiapkan adalah data scraping yang berformat csv agar program R software bisa menganalisa . jika belum punya software scraping dapat mendownload lewatv extentions di google chrome seperti contoh dibawah ini



2. Cari data di facebook dengan membuka facebook dan temukan komentar positif dan negative tentang game .



3. Setelah memperoleh data tersebut maka akan di proses melalui program R



4. Kemudian install packagenya melalui scrib di bawah ini.
`install.packages('wordcloud')`
`install.packages('devtools')`
`library(devtools)`
`library(wordcloud)`

```

Console Terminal x Jobs x
c:/UAS Bu yulian/
Installing package into 'C:/Users/Flux/Documents/R/win-library/3.6'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/devtools_2.2.1.zip'
Content type 'application/zip' length 342853 bytes (334 KB)
downloaded 334 KB

package 'devtools' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\Flux\AppData\Local\Temp\Rtmpwufk3K\downloaded_packages
> library(devtools)
Loading required package: usethis
warning message:
package 'devtools' was built under R version 3.6.2
> library(wordcloud)
Loading required package: RColorBrewer
warning message:
package 'wordcloud' was built under R version 3.6.2

```

```

Slam_url <- "https://cran.r-
project.org/src/contrib/Archive/slam/slam_0.1-37.tar.gz"
install_url(slam_url)

```

```

Console Terminal x Jobs x
c:/UAS Bu yulian/
> slam_url <- "https://cran.r-project.org/src/contrib/Archive/slam/slam_0.1-37.tar.gz"
> install_url(slam_url)
Downloading package from url: https://cran.r-project.org/src/contrib/Archive/slam/slam_0.1-37.tar.gz
√ checking for file 'C:/Users/Flux/AppData/Local/Temp/Rtmpwufk3K/remotes38883eed5c9b\slam/DESCRIPTION' ...
- preparing 'slam':
√ checking DESCRIPTION meta-information ...
- cleaning src
- checking for LF line-endings in source and make files and shell scripts
- checking for empty or unneeded directories
- building 'slam_0.1-37.tar.gz'

Installing package into 'C:/Users/Flux/Documents/R/win-library/3.6'
(as 'lib' is unspecified)
ERROR: failed to lock directory 'C:/Users/Flux/Documents/R/win-library/3.6' for modifying
Try removing 'C:/Users/Flux/Documents/R/win-library/3.6/00LOCK-slam'
Error: Failed to install 'unknown package' from URL:
  (converted from warning) installation of package 'C:/Users/Flux/AppData/Local/Temp/Rtmpwufk3K/file388844a023b/slam_0.1-37.tar.gz' had non-zero exit status
> |

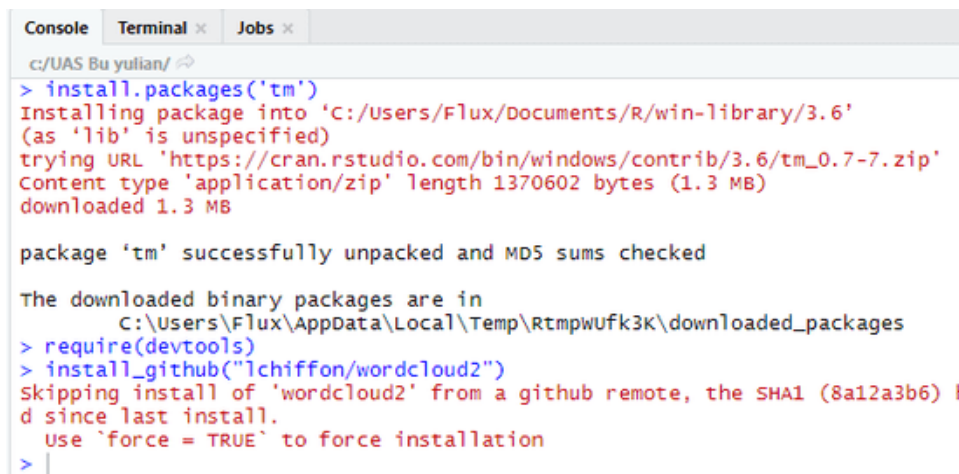
```

```

install.packages('tm')
require(devtools)

```

```
install_github("lchiffon/wordcloud2")
```



```
Console Terminal x Jobs x
c:/UAS Bu yulian/ ↵
> install.packages('tm')
Installing package into 'C:/Users/Flux/Documents/R/win-library/3.6'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/tm_0.7-7.zip'
Content type 'application/zip' length 1370602 bytes (1.3 MB)
downloaded 1.3 MB

package 'tm' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\Flux\AppData\Local\Temp\Rtmpwufk3K\downloaded_packages
> require(devtools)
> install_github("lchiffon/wordcloud2")
Skipping install of 'wordcloud2' from a github remote, the SHA1 (8a12a3b6)
d since last install.
  use `force = TRUE` to force installation
> |
```

5. Panggil data berformat csv seperti berikut ini.

```
setwd('c:/UAS Bu yulian')
uas<- read.csv('facebook.csv')
str(uas)
View(uas)
```

The screenshot shows RStudio with two panes. The top pane displays a data frame named 'komentar...' with 13 rows of text comments. The bottom pane is a terminal window showing R code execution:

```

c:/UAS Bu yulian/
> setwd('c:/UAS Bu yulian')
> uas <- read.csv('facebook.csv')
> view(uas)
> str(uas)
'data.frame':  24 obs. of  1 variable:
 $ komentar....: Factor w/ 24 levels "apa cmn gw yg berharap nemu kata2 dan gold bisa di
trade antar player ? :3;;;";...: 21 22 16 10 1 9 3 14 2 23 ...
>

```

6. Setelah bisa menambahkan sedikit definisi yang diinginkan bisa lakukan seperti cara dibawah ini:
- ```

positif <- scan("C://UAS Bu yulian//Sentiment Corpus(ID)//s-
pos.txt",what="character",comment.char=";")
negatif <- scan("C://UAS Bu yulian//Sentiment Corpus(ID)/s-
neg.txt",what="character",comment.char=";")
kata.positif = c(positif,"sayang","cinta","keren","aduhai")
kata.negatif = c(negatif,"bangsat","busuk","brengek","berengsek","bajingan")

```

```

Console Terminal x Jobs x
c:/UAS Bu yulian/
> positif <- scan("C://UAS Bu yulian//Sentiment Corpus(ID)//s-pos.txt",what="character",comment.char=";")
Read 1473 items
> negatif <- scan("C://UAS Bu yulian//Sentiment Corpus(ID)/s-neg.txt",what="character",comment.char=";")
Read 2960 items
> kata.positif = c(positif,"sayang","cinta","keren","aduhai")
> kata.negatif = c(negatif,"bangsat","busuk","brengekek","berengsek","bajingan")
> |

```

7. Kemudian dibawah ini merupakan fungsi agar penilaian atau pembobotan agar dapat menjalankan kata-kata seperti dibawah ini :

```

score.sentiment = function(kalimat2, kata.positif, kata.negatif,
.progress='none') {
require(plyr)
require(stringr)
scores = lapply(kalimat2, function(kalimat, kata.positif, kata.negatif)
{
kalimat = gsub('[:punct:]]', '', kalimat)
kalimat = gsub('[:cntrl:]]', '', kalimat)
kalimat = gsub('\\d+', '', kalimat)
kalimat = tolower(kalimat)

list.kata = str_split(kalimat, '\\s+')
kata2 = unlist(list.kata)
positif.matches = match(kata2, kata.positif)
negatif.matches = match(kata2, kata.negatif)
positif.matches = !is.na(positif.matches)
negatif.matches = !is.na(negatif.matches)
score = sum(positif.matches) - (sum(negatif.matches))
return(score)
}, kata.positif, kata.negatif, .progress=.progress)
scores.df = data.frame(score=scores, text=kalimat2)
return(scores.df)
}

```

```

Console Terminal Jobs
c:/UAS Bu yulian/
> score.sentiment = function(kalimat2, kata.positif, kata.negatif, .progress='none')
+ {
+ require(plyr)
+ require(stringr)
+ scores = lapply(kalimat2, function(kalimat, kata.positif, kata.negatif) {
+ kalimat = gsub('[[:punct:]]', '', kalimat)
+ kalimat = gsub('[[:cntrl:]]', '', kalimat)
+ kalimat = gsub('\\d+', '', kalimat)
+ kalimat = tolower(kalimat)
+
+ list.kata = str_split(kalimat, '\\s+')
+ kata2 = unlist(list.kata)
+ positif.matches = match(kata2, kata.positif)
+ negatif.matches = match(kata2, kata.negatif)
+ positif.matches = !is.na(positif.matches)
+ negatif.matches = !is.na(negatif.matches)
+ score = sum(positif.matches) - (sum(negatif.matches))
+ return(score)
+ }, kata.positif, kata.negatif, .progress=.progress)
+ scores.df = data.frame(score=scores, text=kalimat2)
+ return(scores.df)
+ }
> |

```

8. Langkah Selanjutnya bisa melakukan perhitungan score text seperti berikut dengan cara :

```

hasil = score.sentiment(uas$komentar.... , kata.positif,
kata.negatif)
head(hasil)

```

```

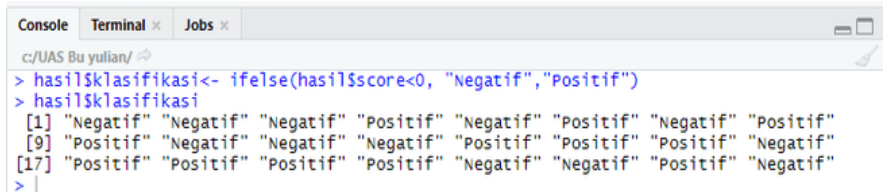
Console Terminal Jobs
c:/UAS Bu yulian/
> hasil = score.sentiment(uas$komentar...., kata.positif, kata.negatif)
> head(hasil)
 score
1 -2
2 -2
3 -1
4 0
5 -1
6 0

 text
1 rilis rilis rilis rilis; bosan main game lain; nunggu wodn r
ilis ghggghghghg;;
2 Semoga saja mekanisme dungeon/nest layaknya di PC version masih ada; biar teamwork part
y/raid juga jalan;;
3 Mudah2an gamenya g kaku. Malah le
bih manten DN M.;;;
4 ke 4 pihak dev lupa memberitahukan bahwa gameny
a gak jadi rilis;;;
5 apa cmn gw yg berharap nemu kata2 dan gold bisa di trade a
ntar player ? :3;;;
6 kapan
rilisnya badut?;;;
> |

```

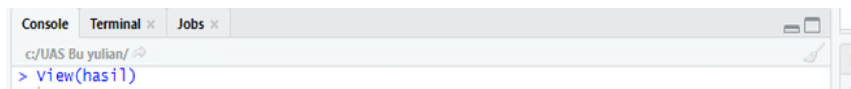
Kemudian jangan lupa melakukan labeling untuk nilainya yang kurang dari nol sebagai negative dan jika lebih dari nol maka bernilai positif ,sebagai contoh bisa melihat contoh nya seperti dibawah ini :

```
hasil$klasifikasi<- ifelse(hasil$score<0, "Negatif","Positif")
hasil$klasifikasi
```



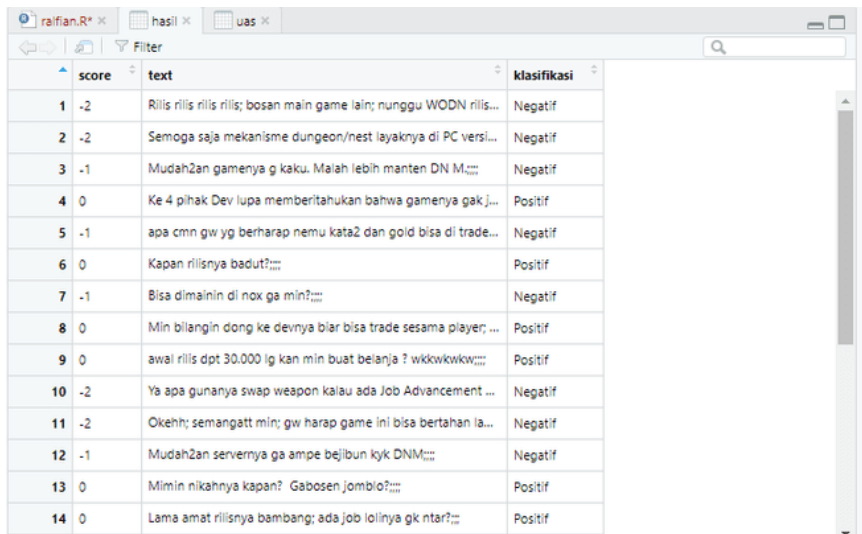
```
Console Terminal x Jobs x
c:/UAS Bu yulian/
> hasil$klasifikasi<- ifelse(hasil$score<0, "Negatif","Positif")
> hasil$klasifikasi
[1] "Negatif" "Negatif" "Negatif" "Negatif" "Negatif" "Negatif" "Negatif" "Positif"
[9] "Positif" "Negatif" "Negatif" "Negatif" "Positif" "Positif" "Positif" "Negatif"
[17] "Positif" "Positif" "Positif" "Positif" "Negatif" "Negatif" "Positif" "Negatif"
> |
```

View(hasil)



```
Console Terminal x Jobs x
c:/UAS Bu yulian/
> view(hasil)
```

Hasilnya



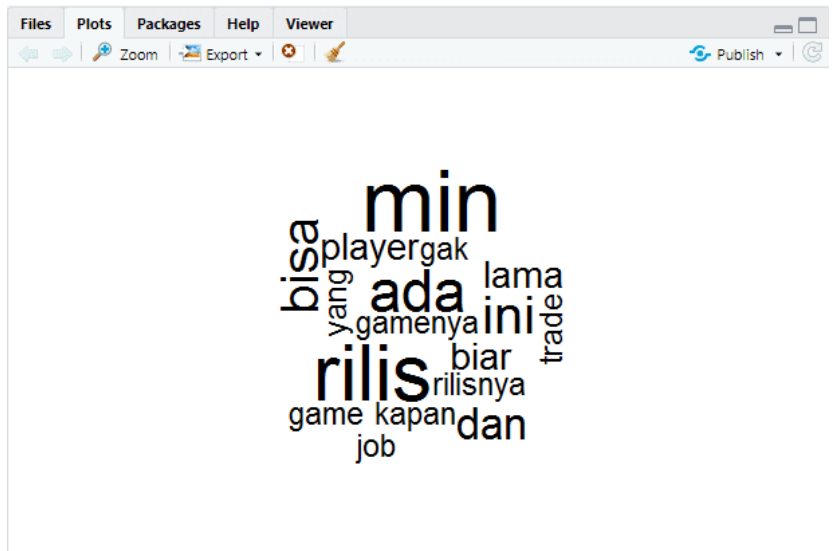
|    | score | text                                                                | klasifikasi |
|----|-------|---------------------------------------------------------------------|-------------|
| 1  | -2    | Rilis rilis rilis rilis; bosan main game lain; nunggu WODN rilis... | Negatif     |
| 2  | -2    | Semoga saja mekanisme dungeon/nest layaknya di PC versi...          | Negatif     |
| 3  | -1    | Mudah2an gamenya g kaku. Malah lebih manten DN M;...                | Negatif     |
| 4  | 0     | Ke 4 pihak Dev lupa memberitahukan bahwa gamenya gak j...           | Positif     |
| 5  | -1    | apa cmn gw yg berharap nemu kata2 dan gold bisa di trade...         | Negatif     |
| 6  | 0     | Kapan rilisnya badut?;...                                           | Positif     |
| 7  | -1    | Bisa dimainkan di nox ga min?;...                                   | Negatif     |
| 8  | 0     | Min bilangin dong ke devnya biar bisa trade sesama player; ...      | Positif     |
| 9  | 0     | awal rilis dpt 30.000 lg kan min buat belanja ? wkkwkwkw;...        | Positif     |
| 10 | -2    | Ya apa gunanya swap weapon kalau ada Job Advancement ...            | Negatif     |
| 11 | -2    | Okehh; semangatt min; gw harap game ini bisa bertahan la...         | Negatif     |
| 12 | -1    | Mudah2an servernya ga ampe bejibun kyk DNM;...                      | Negatif     |
| 13 | 0     | Mimin nikahnya kapan? Gabosen Jomblo?;...                           | Positif     |
| 14 | 0     | Lama amat rilisnya bambang; ada job lolinya gk ntar?;...            | Positif     |

wordcloud(uas\$komentar...)



```
Console Terminal x Jobs x
c:/UAS Bu yulian/
> wordcloud(uas$komentar....)
warning messages:
1: In tm_map.simpleCorpus(corpus, tm::removePunctuation) :
 transformation drops documents
2: In tm_map.simpleCorpus(corpus, function(x) tm::removeWords(x, tm::stopwords())) :
 transformation drops documents
> |
```

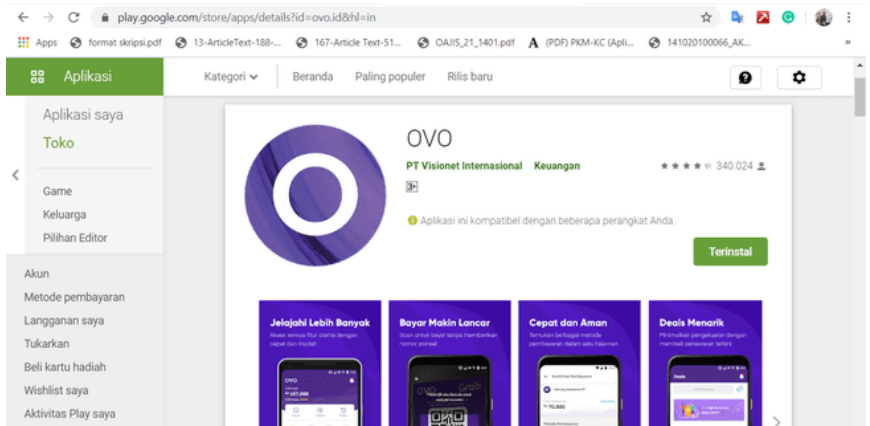
9. Berikut ini adalah Kata kata yang sering muncul



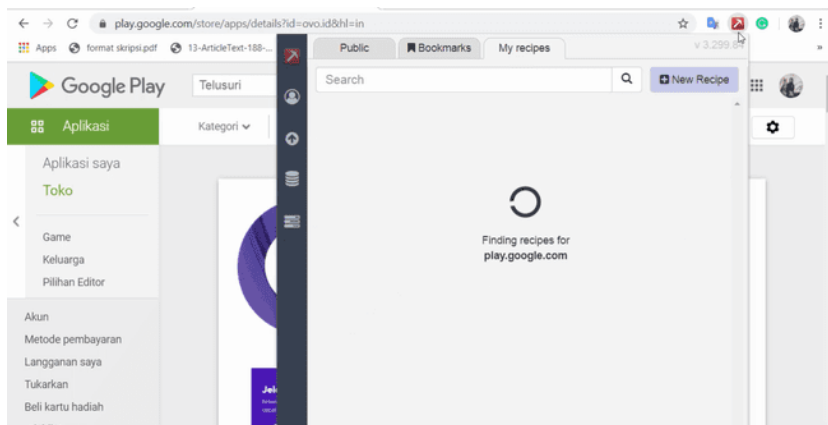
Studi Kasus 3: Analisa Teks Mining menggunakan Rapid Miner  
Step 1 : Scraping Data Ovo dan Dana

**Data ovo**

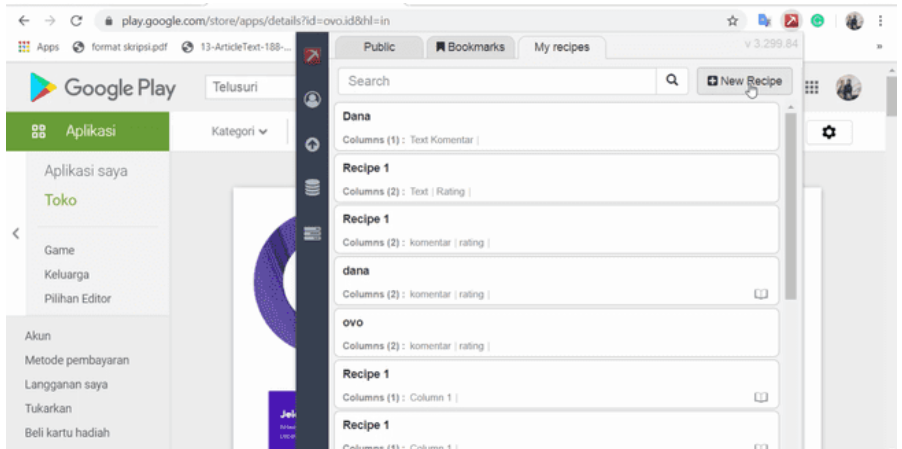
- Buka web yang ingin di scraping, pada contoh disini melakukan scraping dari ovo yang mengambil dari google play store.



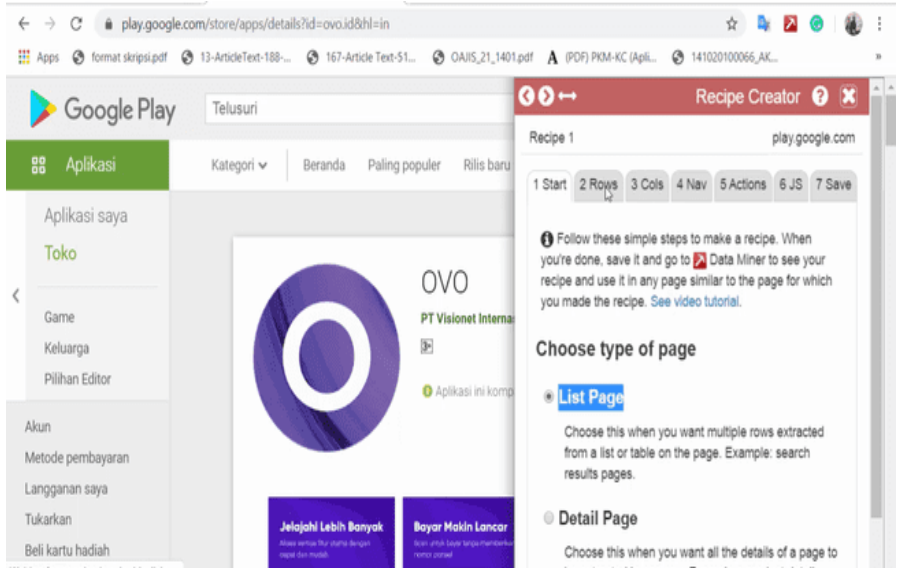
- Disini kelompok kami menggunakan tools Data Miner untuk scraping, lalu klik “data miner”.



- Klik New “New Recipe”.

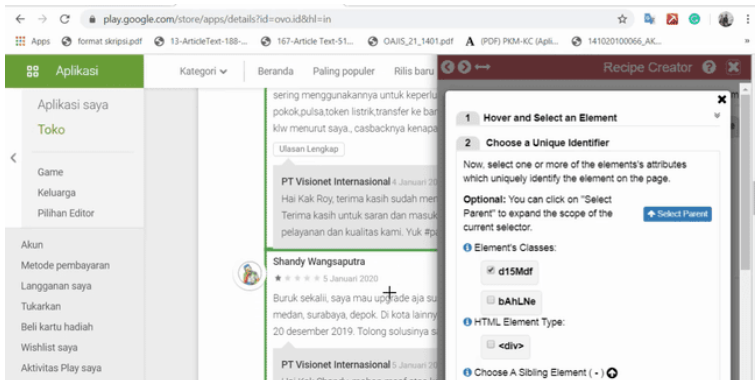


- Pada tahap 1 Start di data miner pilih opsi “List Pge”.

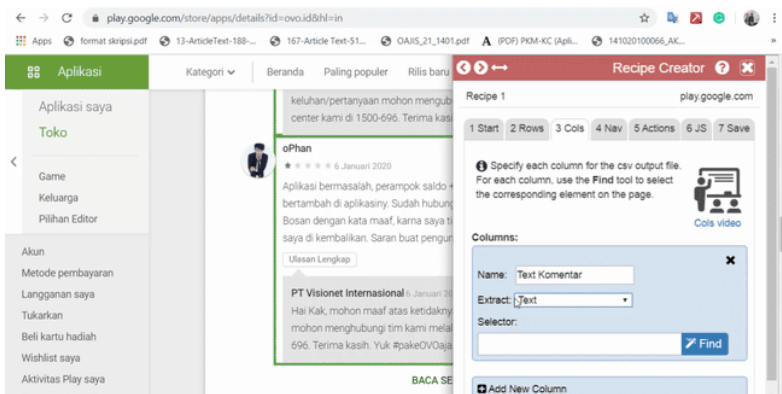


- Pada tahap 2 Rows di data miner klik “Find” lanjut pemilihan text dengan menekan Shift pada text yang

diperlukan dan ceklist pada element class kemudian pastikan semua text koentar sudah terblok hijau, lalu klik “Confrim” .

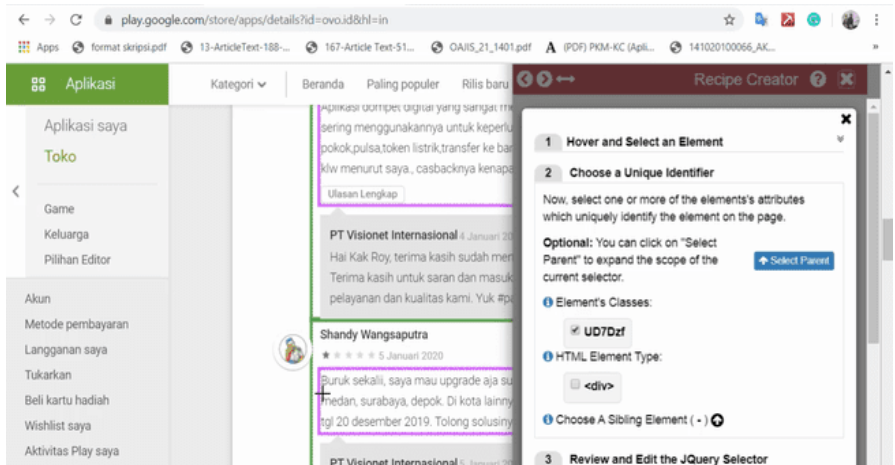


- Pada tahap 3 Cols di data miner pilih bagian “Columns” ke-1 lalu ganti name sesuai keperluan dan pilih extract dengan opsi text.

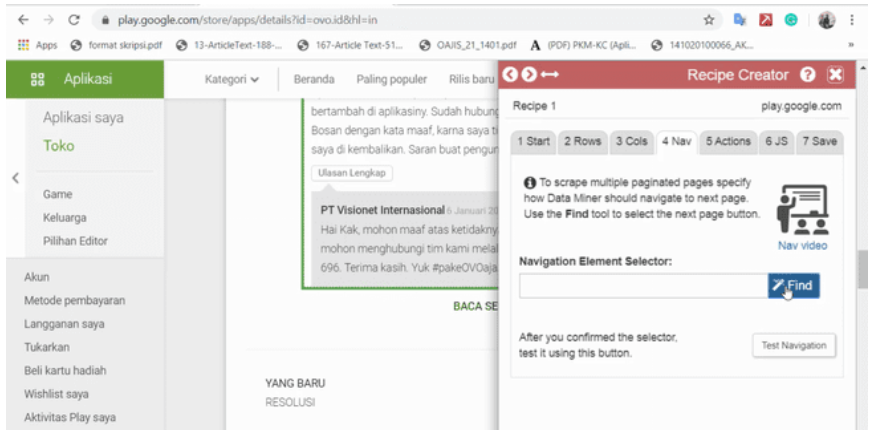


- Lalu pilih bagian text kometar yang akan digunakan dengan menekan Shift dan pilih elements classes lalu

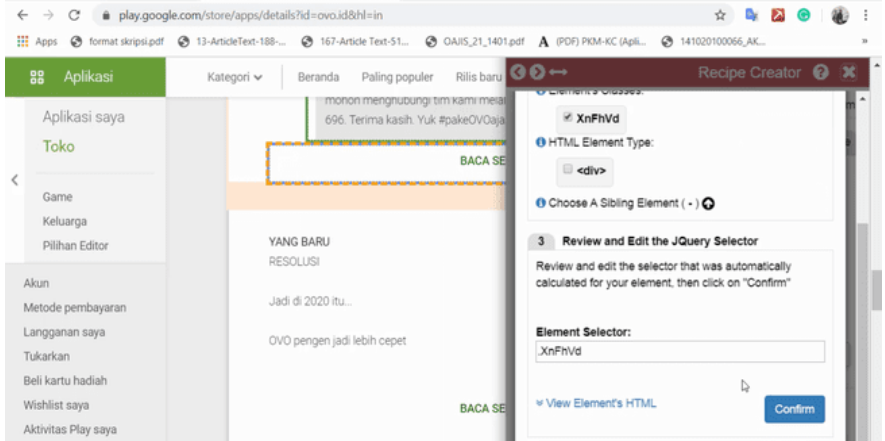
ceklis dibagian bawah elements classes lalu klik “Confirm”.



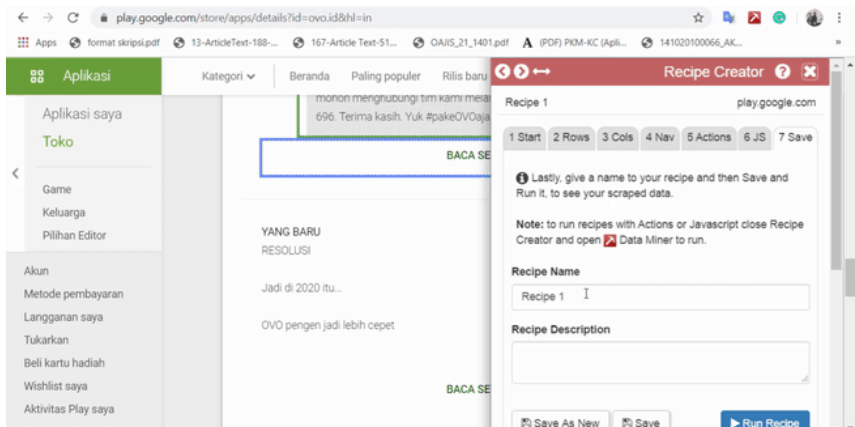
- Pada tahap 4 nav klik “find”



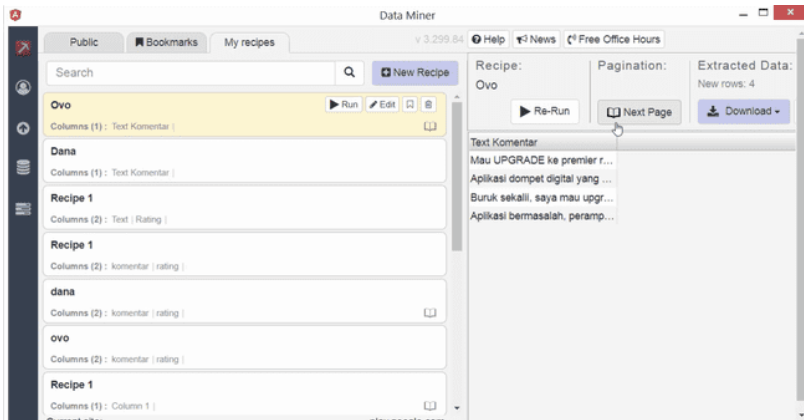
- Kemudian tekan shift pada tombol “Baca Selengkapnya” agar text komentar bisa terambil semua lalu ceklist pada bagian elements classes dan klik “Confirm”.



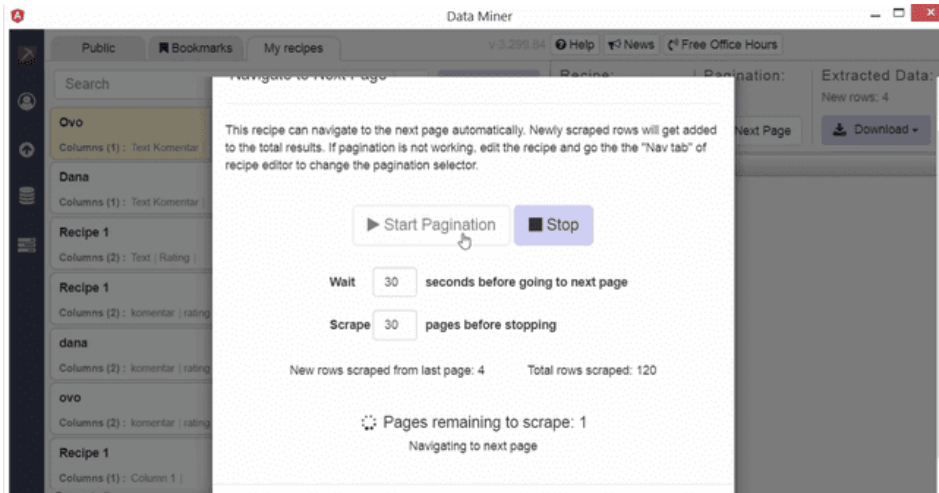
- Pada tahap 7 save ubah nama pada bagian Recipe Name sesuai keinginan dan klik “Run Recipe”.



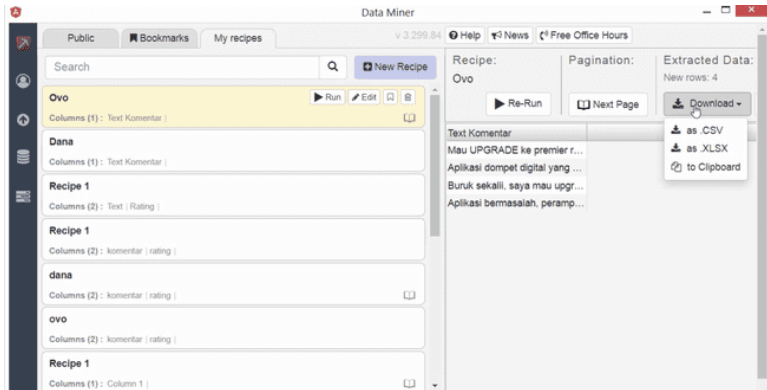
- Lalu pilih Next Page mengambil data yang lebih banyak lagi.



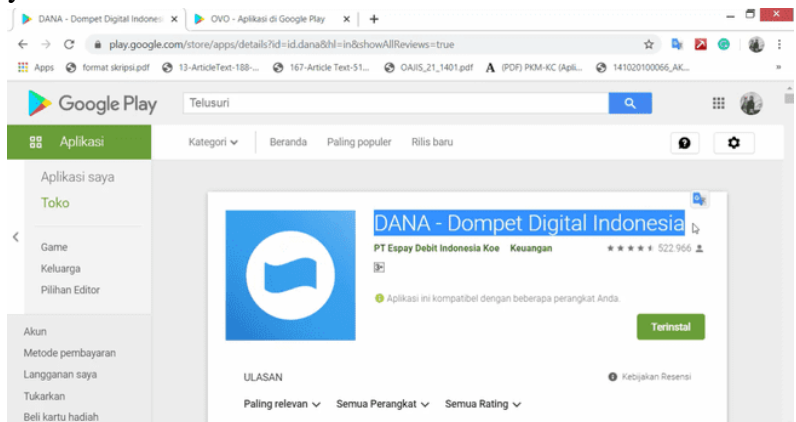
- Kemudian pilih wait lalu isi sesuai keinginan untuk data yang diambil dan isikan pada bagian scrape dan tunggu proses reaiming to scrape dengan klik strat pagination tunggu menjadi angka 1.



- Dan yang terakhir klik tombol “download” lalu pilih format untuk data yang disimpan dan data akan otomatis terdownload.

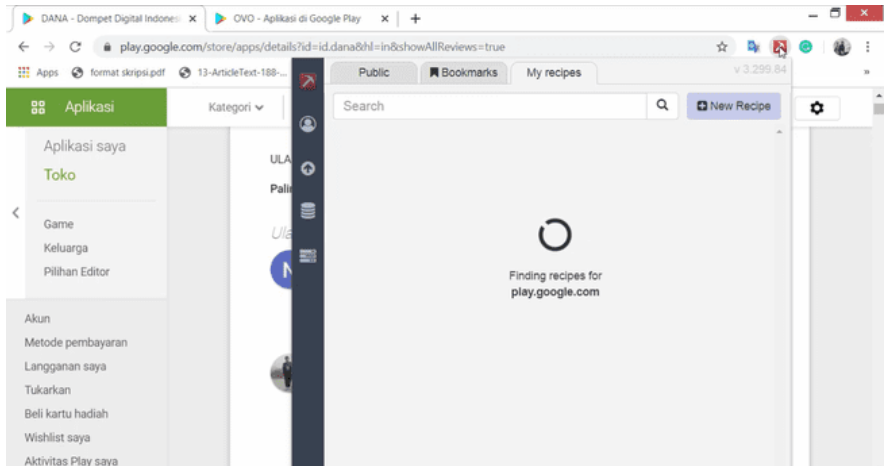


- Buka web yang ingin di scraping, pada contoh disini melakukan scraping dari dana yang mengambil dari google play store.

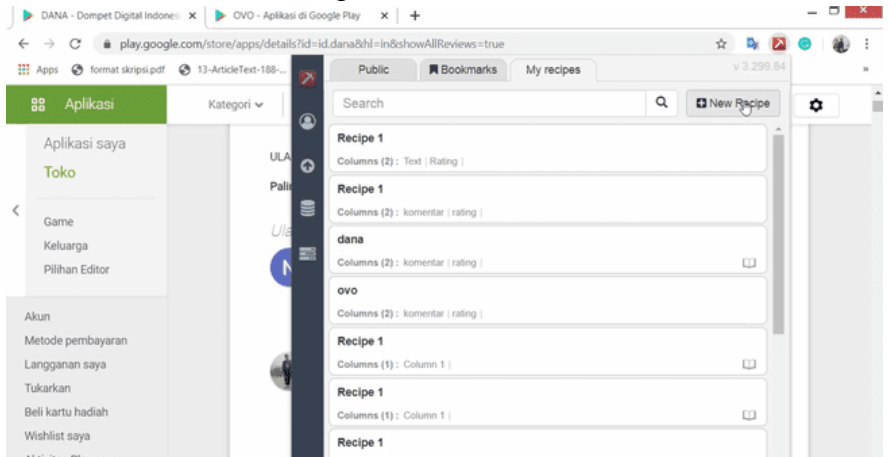


- Disini kelompok kami menggunakan tools Data Miner untuk scraping, lalu klik “data miner”.

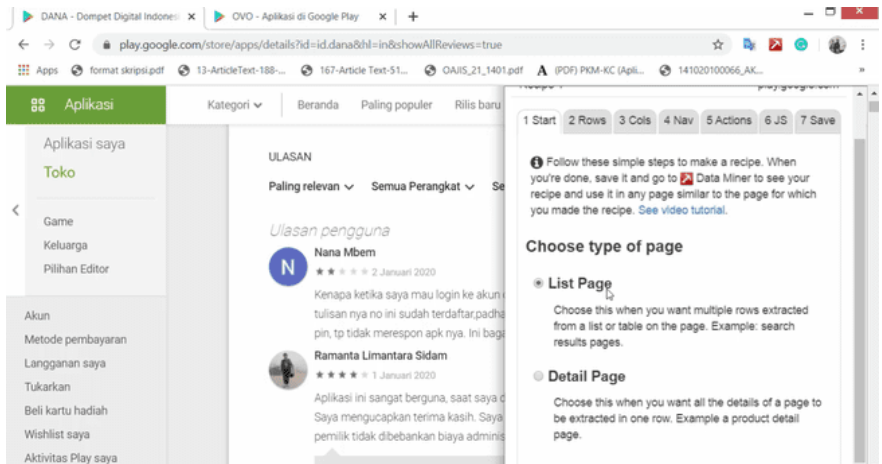




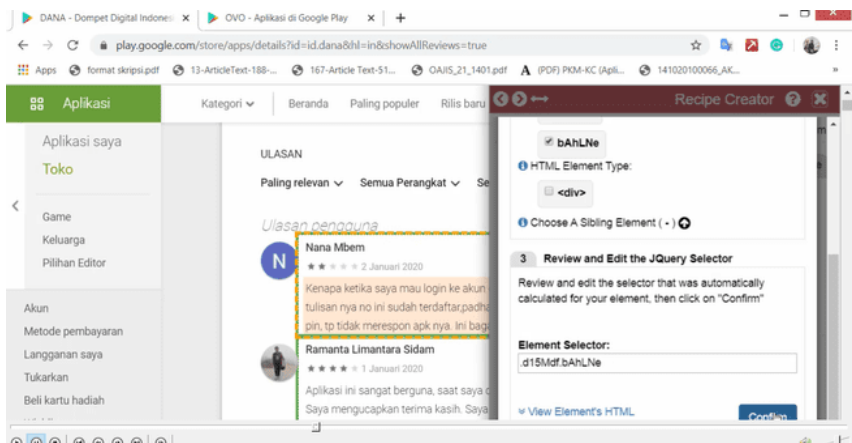
- Klik New “New Recipe”.



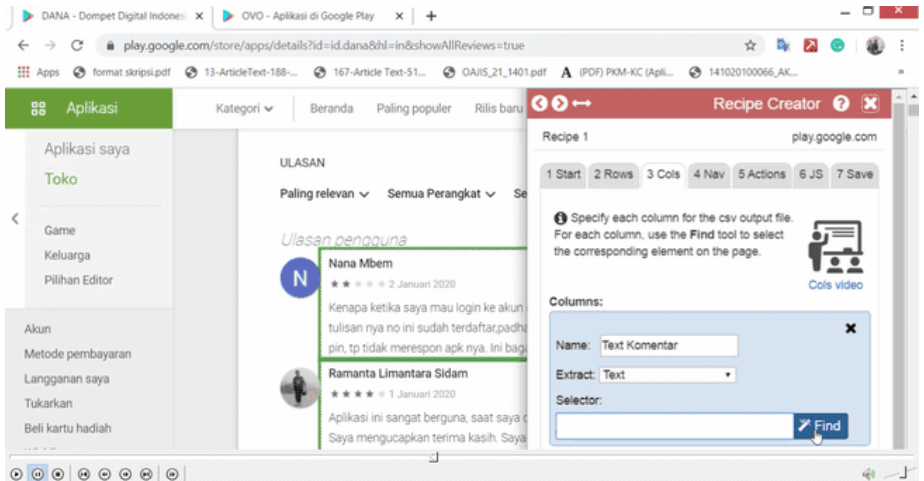
- Pada tahap 1 Start di data miner pilih opsi “List Pge”.



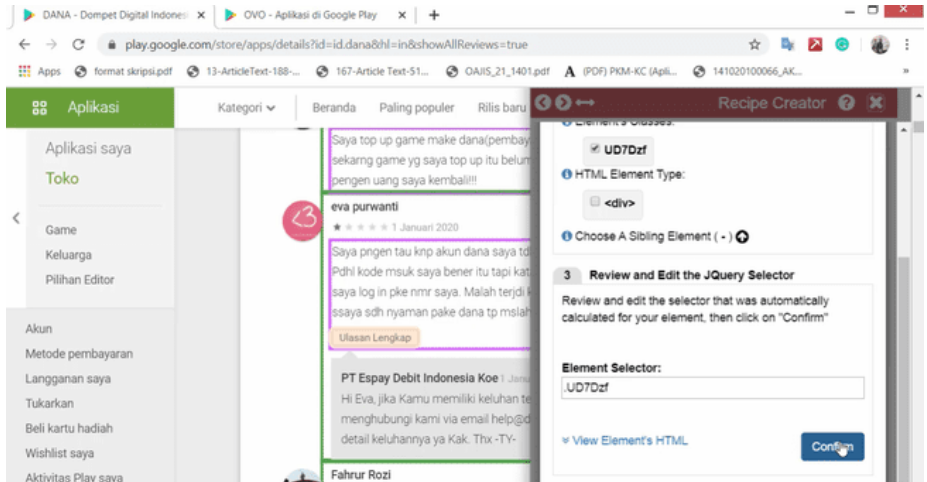
- Pada tahap 2 Rows di data miner klik “Find” lanjut pemilihan text dengan menekan Shift pada text yang diperlukan dan ceklist pada element class kemudian pastikan semua text koentar sudah terblok hijau, lalu klik “Confrim”.



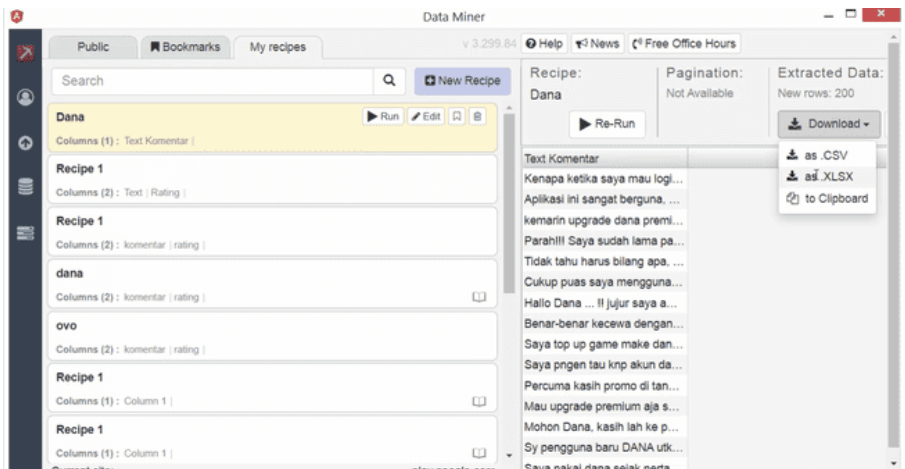
- Pada tahap 3 Cols di data miner pilih bagian “Columns” ke-1 lalu ganti name sesuai keperluan dan pilih extract dengan opsi text.



- Lalu pilih bagian text kometar yang akan digunakan dengan menekan Shift dan pilih elements classes lalu ceklist dibagian bawah elements classes lalu klik “Confrim” .

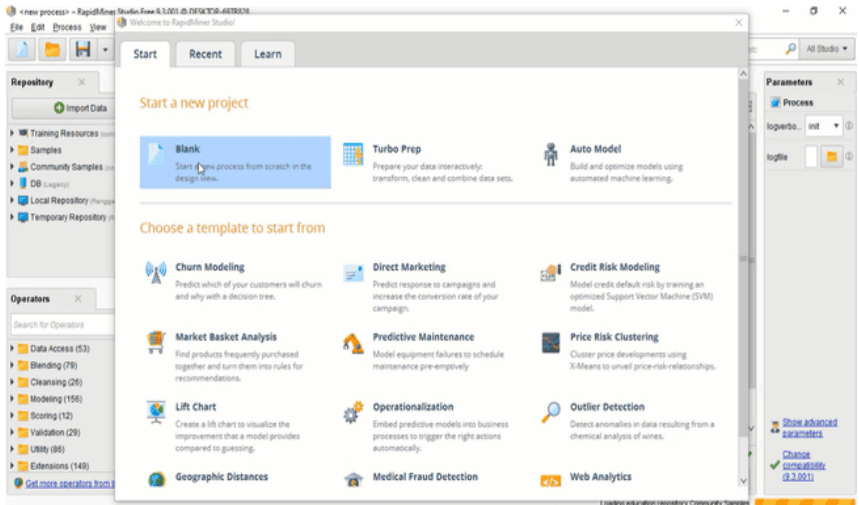


- Dan yang terakhir klik tombol “download” lalu pilih format untuk data yang disimpan dan data akan otomatis terdownload.

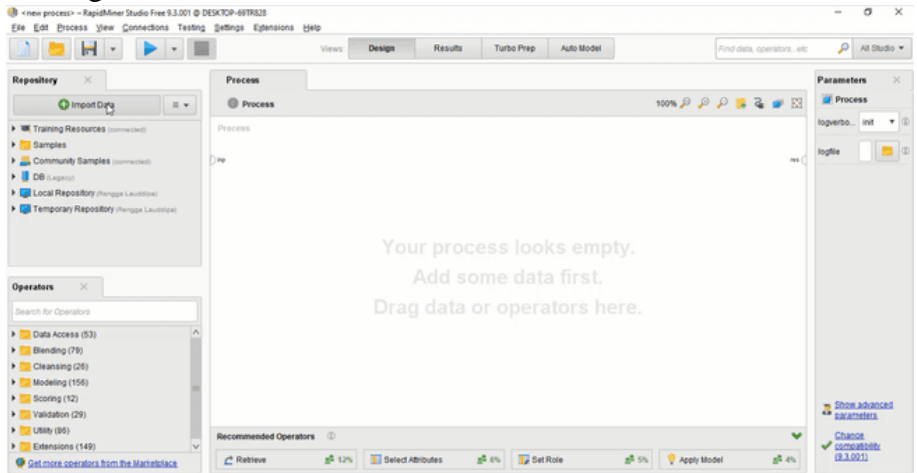


## Step 2 : (ANALISA SENTIMENT) KLASIFIKASI TEXT

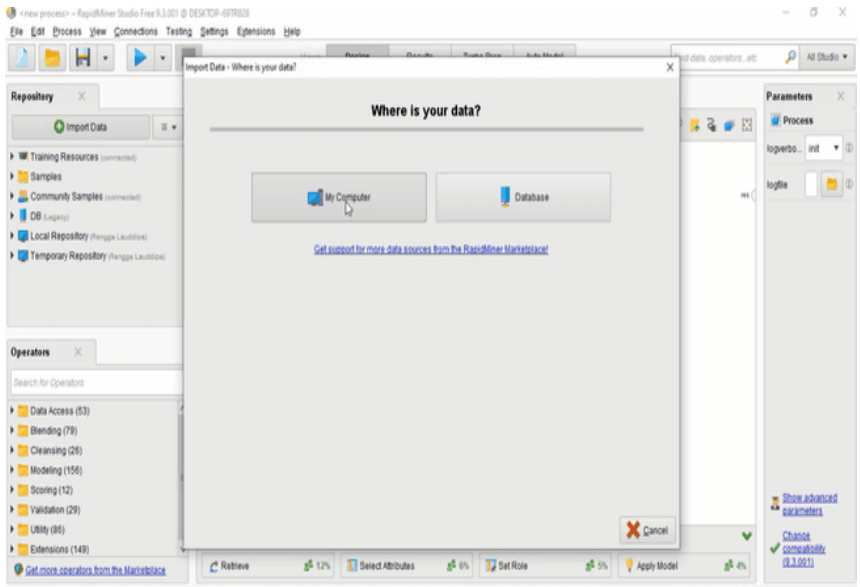
- Pertama membuka Rapidminer dan pilih “Blank” untuk start projek.



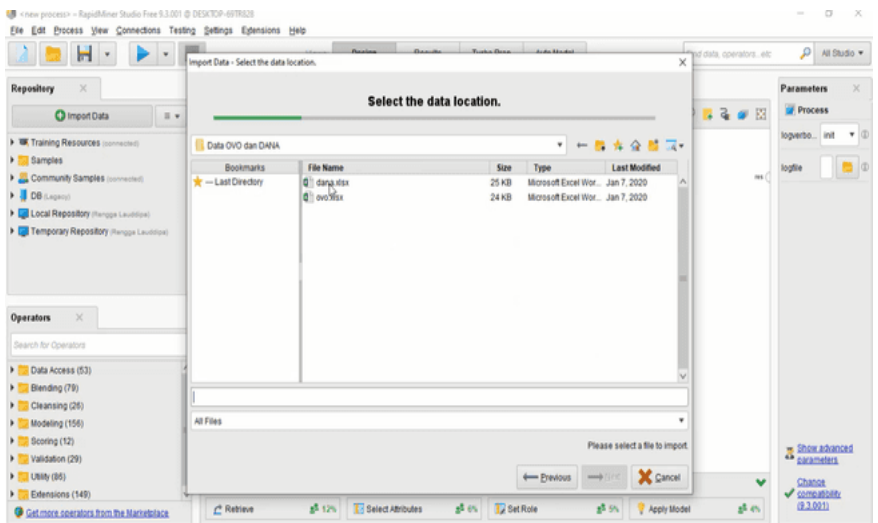
- Klik “import data” untuk mengambil data yang ingin digunakan.



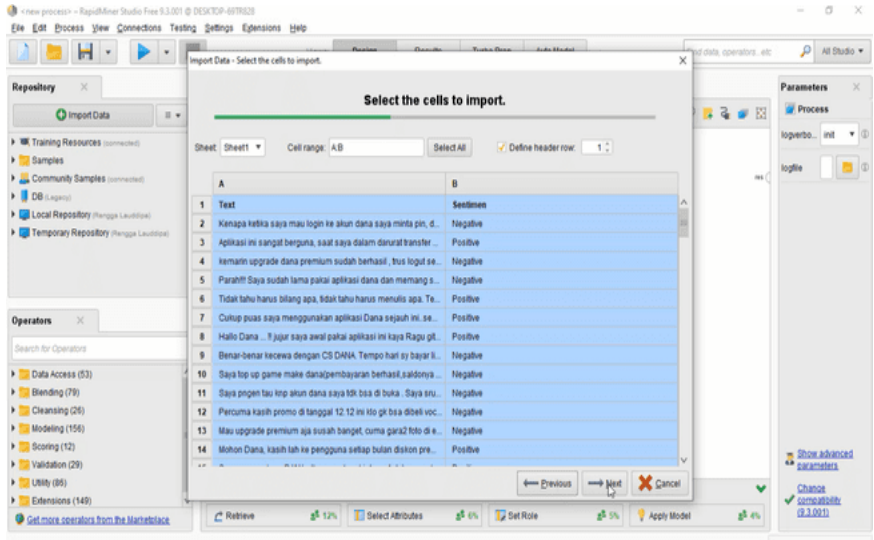
- Klik “My Computer”.



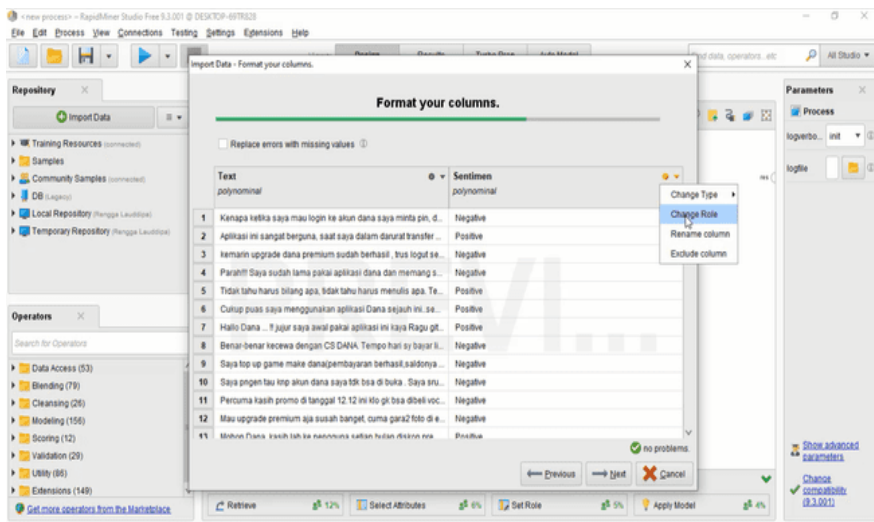
- Pilih file data DANA yang ingin digunakan lalu klik “Next”.



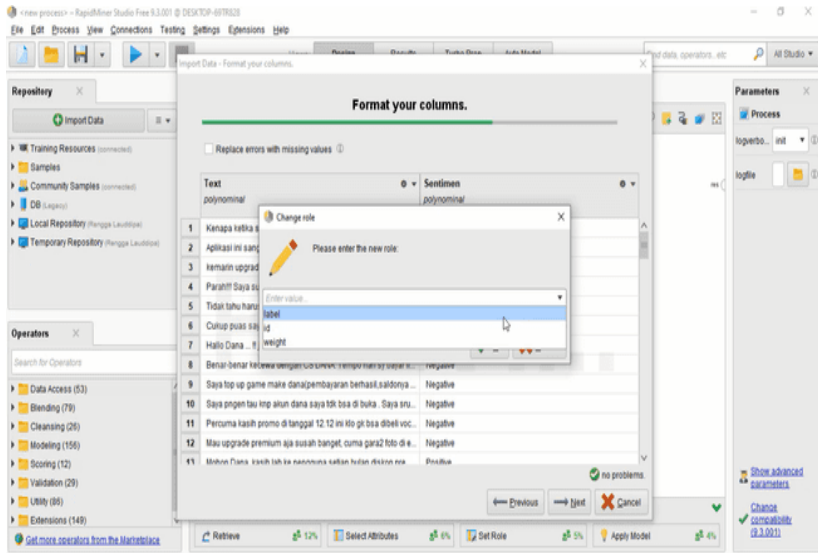
- Klik “next”.



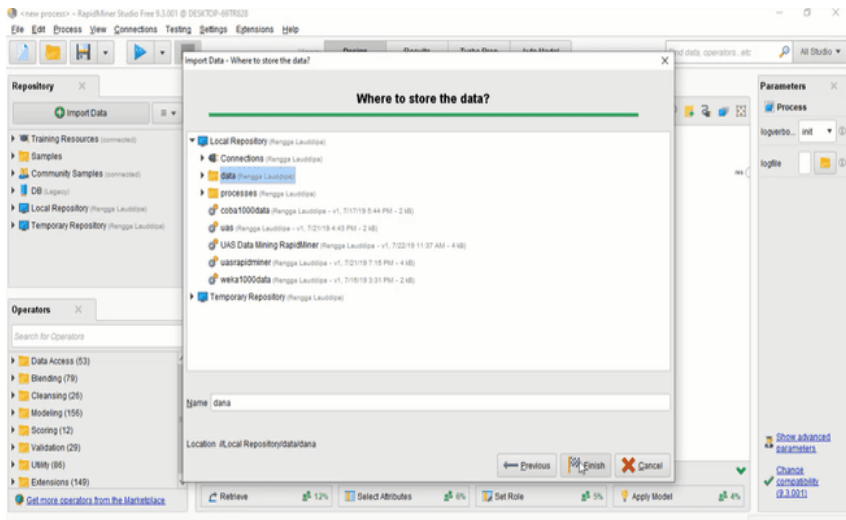
- Klik “Change Role” pada kolom sentimen.



- Lalu pilih “label”.



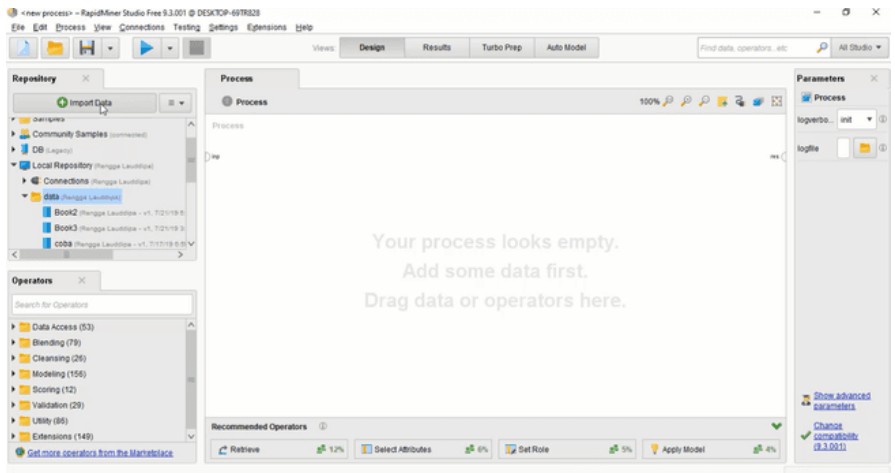
- Pilih tempat untuk penyimpanan data.



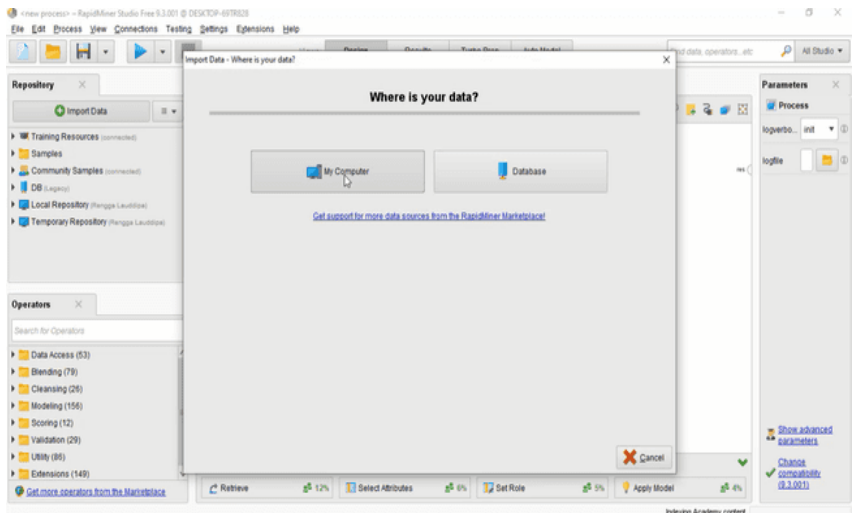
## Import Data Ovo



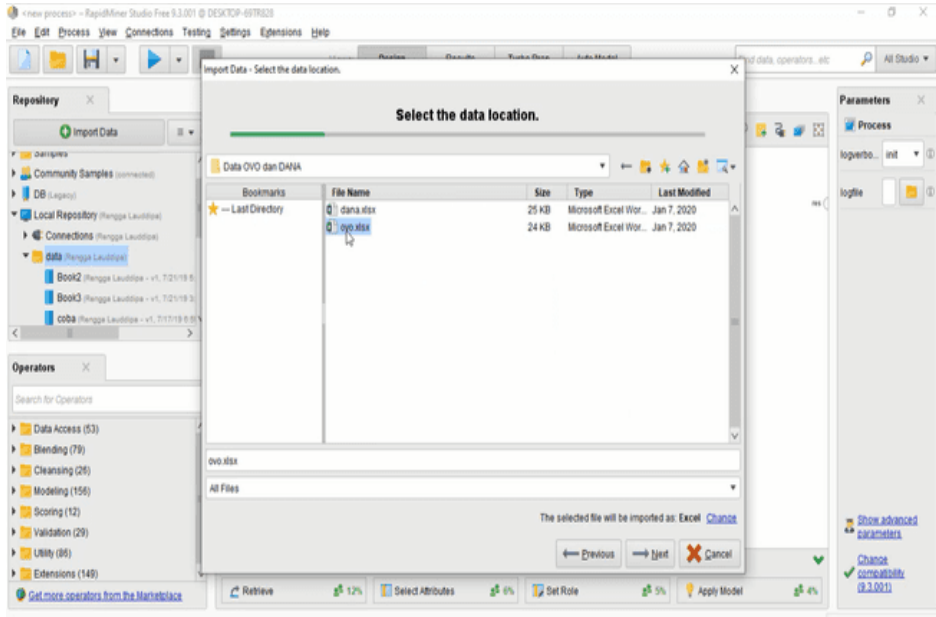
- Klik “import data” untuk mengambil data yang ingin digunakan.



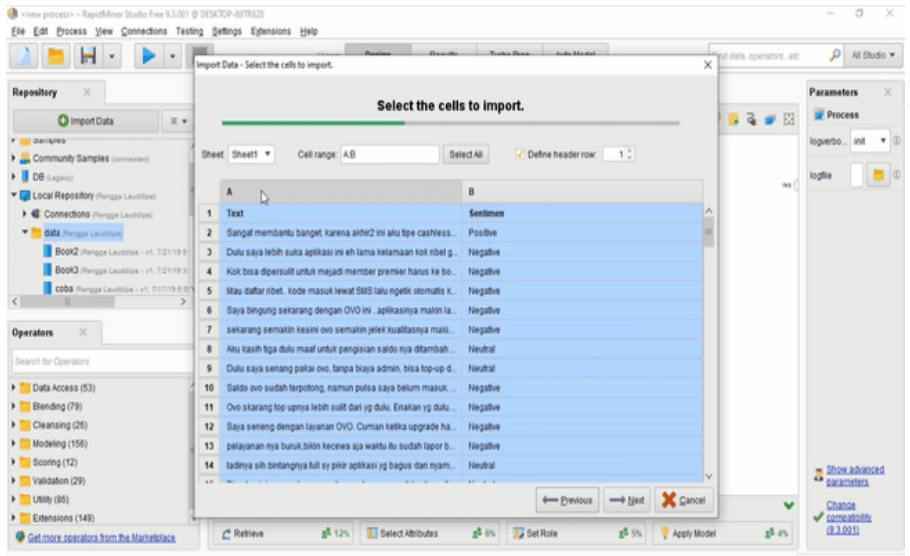
- Klik “My Computer”.



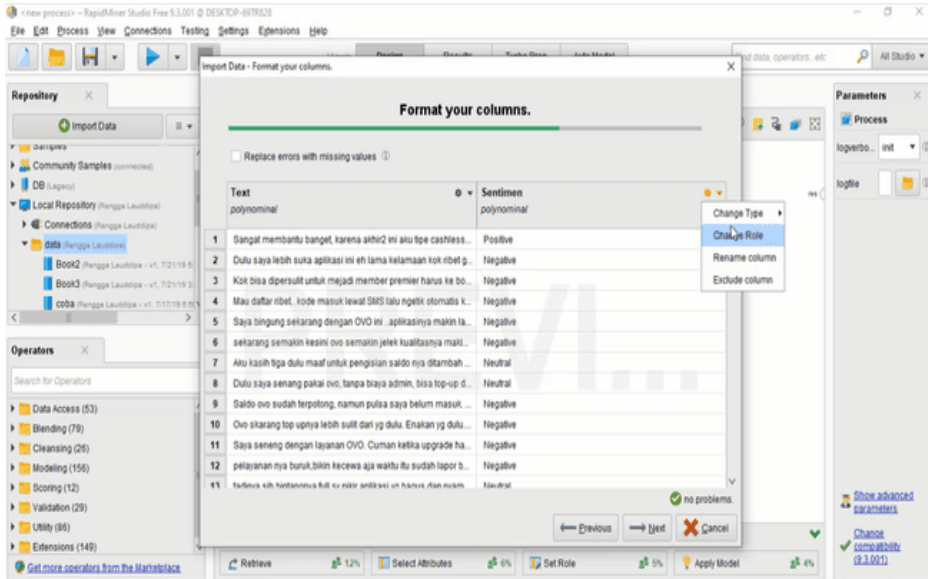
- Pilih file data OVO yang ingin digunakan lalu klik “Next”.



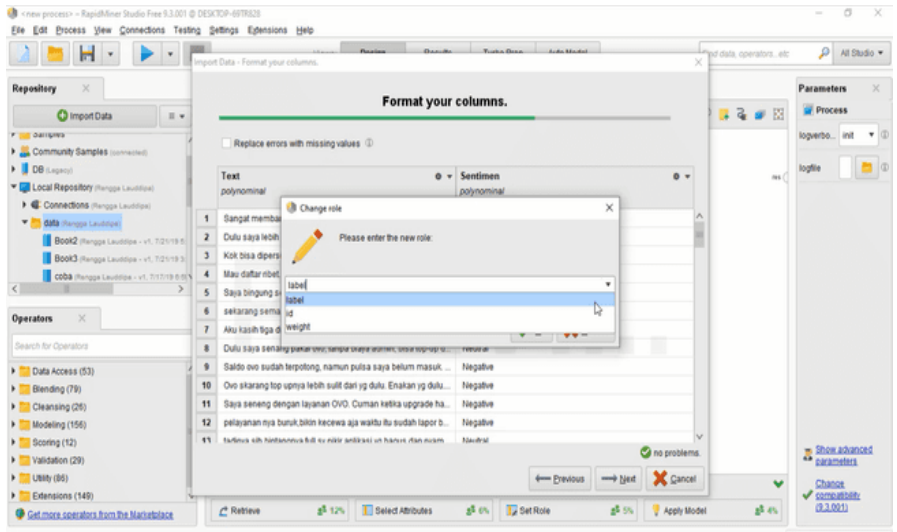
- Klik “next”.



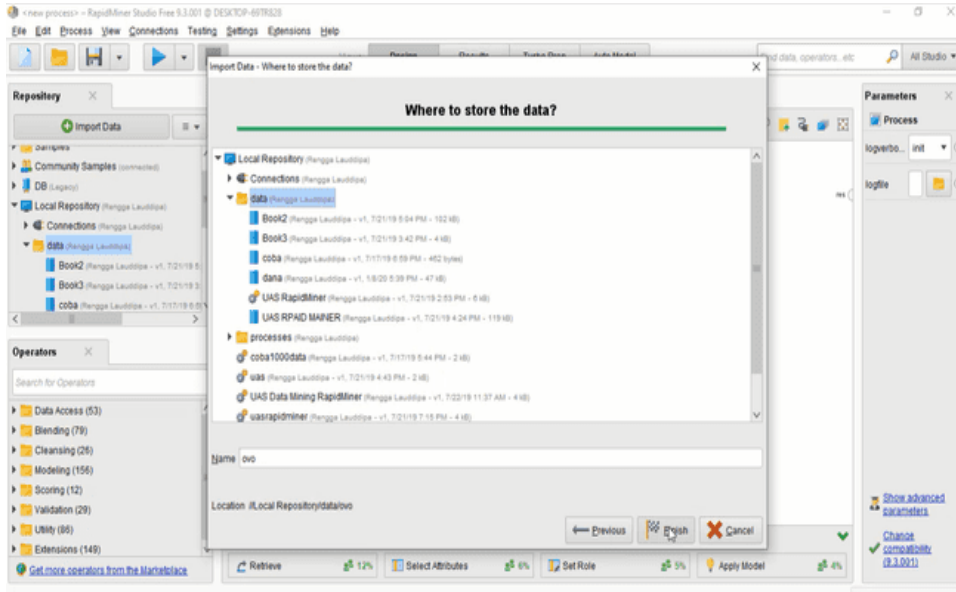
- Klik “Change Role” pada kolom sentimen.



- Lalu pilih “label”.

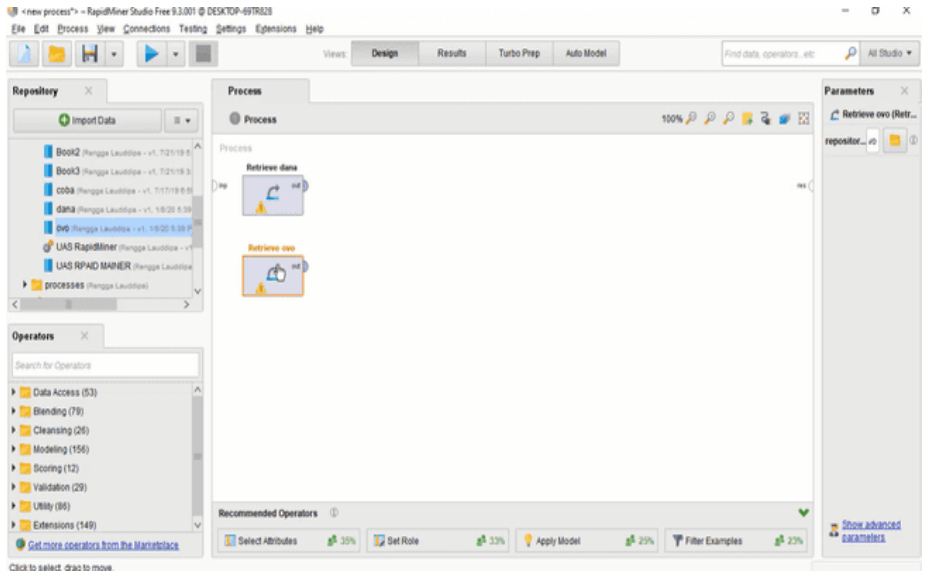


- Pilih tempat untuk penyimpanan data.

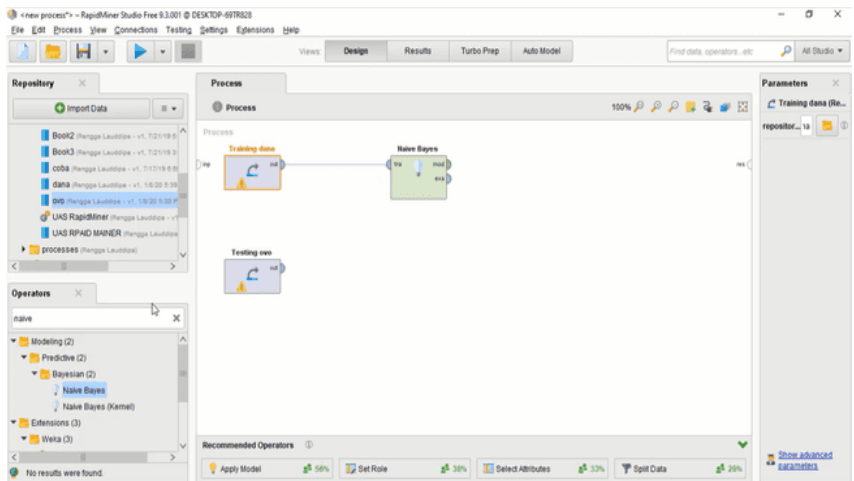


## Step Training Dana dan Testing Ovo

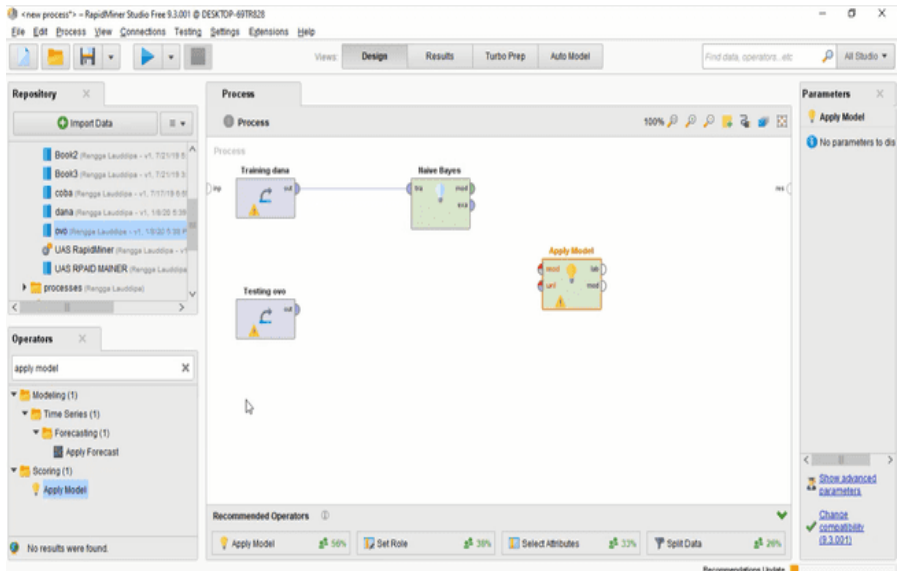
- Drak data dana da novo ke dalam process.



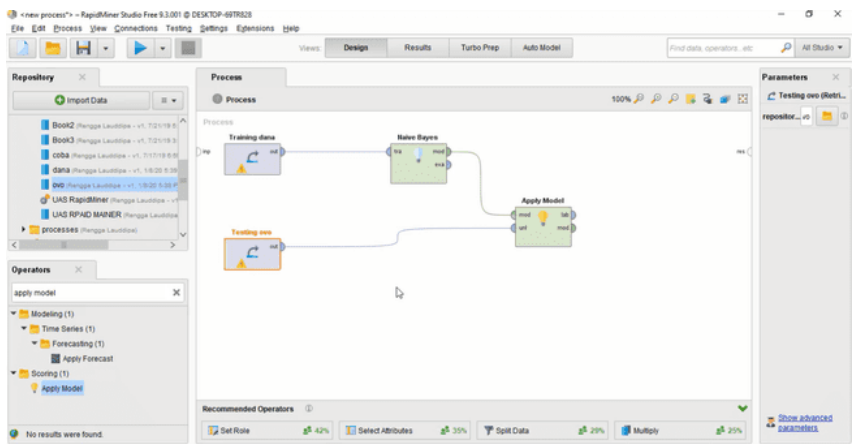
- Mencari metode Naïve Bayes di pencarian Operators, lalu drak pada process.



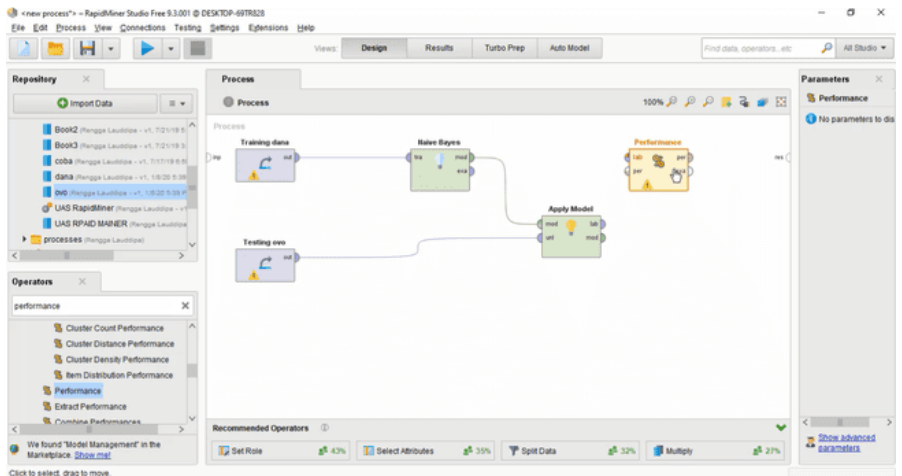
- Mencari Apply Model di pencarian Operators, lalu drak pada process.



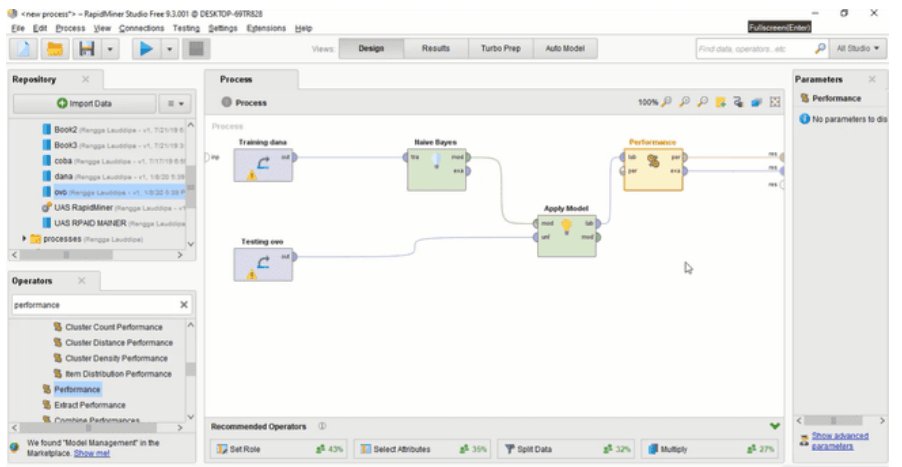
- Lalu sambungkan Training data ke naïve bayes dan apply model kemudian sambungkan juga testing ovo ke dalam apply model.



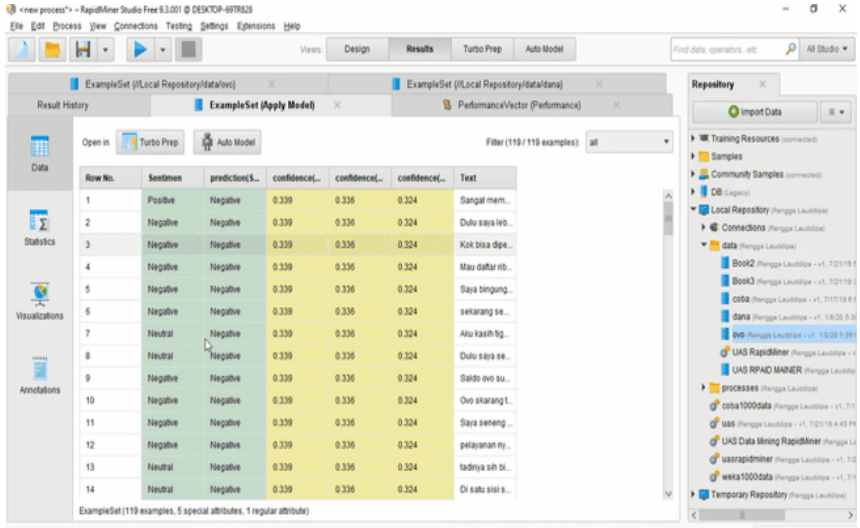
- Mencari Performance di pencarian Operators, lalu drak pada process.



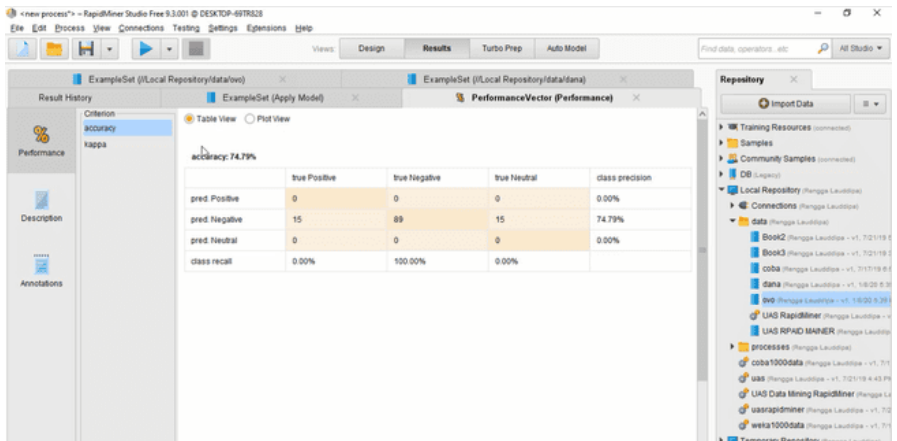
- Lalu semua process sambungkan pada performance



- Tampilan data yang berhasil di run.

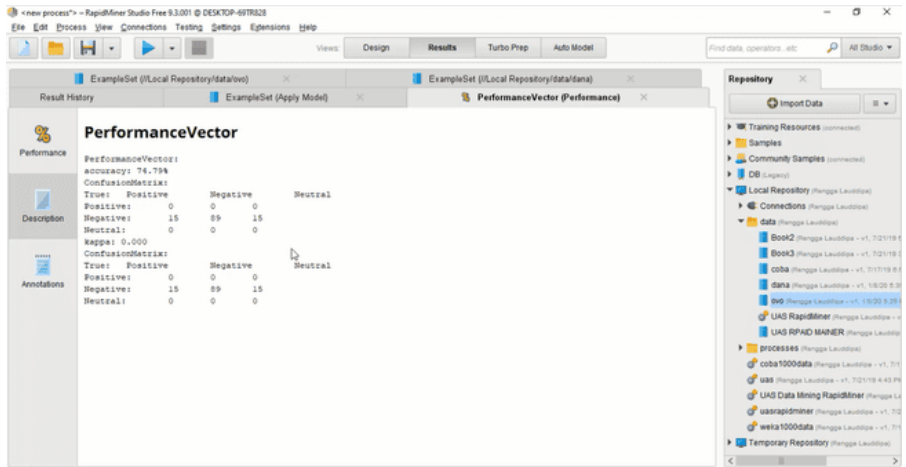


- Tampilan performance accuracy dengan nilai 74.79% dengan menggunakan metode naïve bayes.

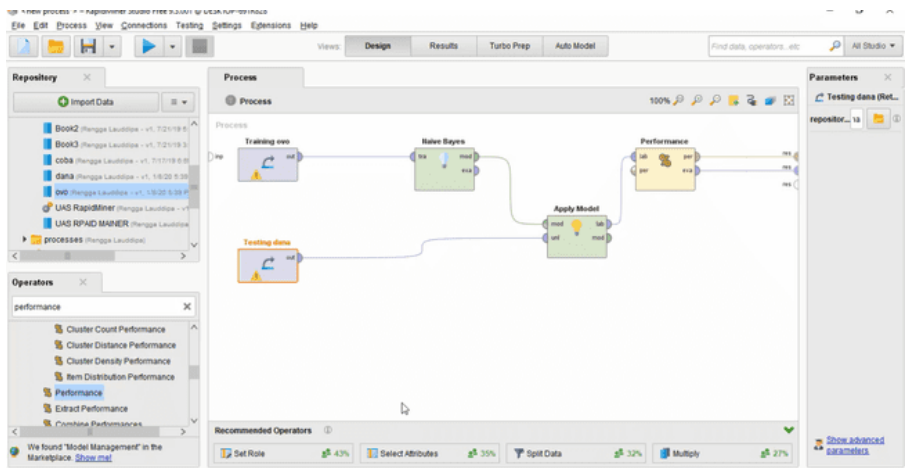


- Tampilan dari Performance Vector .





- Merubah data training dana menjadi ovo dan testing ovo menjadi dana lalu klik “Run”.



- Tampilan data yang berhasil di run.

ExampleSet (119 examples, 5 special attributes, 1 regular attribute)

| Row No. | Sentimen | predictedSentimen | confidence | confidence | confidence | Text             |
|---------|----------|-------------------|------------|------------|------------|------------------|
| 3       | Negative | Negative          | 0.327      | 0.345      | 0.327      | Kemarin upgr...  |
| 4       | Negative | Negative          | 0.327      | 0.345      | 0.327      | Parah!!! Saya... |
| 5       | Positive | Negative          | 0.327      | 0.345      | 0.327      | Tidak tahu ha... |
| 6       | Positive | Negative          | 0.327      | 0.345      | 0.327      | Cukup posi...    |
| 7       | Positive | Negative          | 0.327      | 0.345      | 0.327      | Halo Dana ...    |
| 8       | Negative | Negative          | 0.327      | 0.345      | 0.327      | Benar-benar ...  |
| 9       | Negative | Negative          | 0.327      | 0.345      | 0.327      | Saya top-up g... |
| 10      | Negative | Negative          | 0.327      | 0.345      | 0.327      | Saya pengen t... |
| 11      | Negative | Negative          | 0.327      | 0.345      | 0.327      | Peruma kas...    |
| 12      | Negative | Negative          | 0.327      | 0.345      | 0.327      | Mau upgrade ...  |
| 13      | Positive | Negative          | 0.327      | 0.345      | 0.327      | Mohon Dana...    |
| 14      | Positive | Negative          | 0.327      | 0.345      | 0.327      | Sy pengguna...   |
| 15      | Neutral  | Negative          | 0.327      | 0.345      | 0.327      | Saya pakai d...  |
| 16      | Negative | Negative          | 0.327      | 0.345      | 0.327      | Pelayanan To...  |

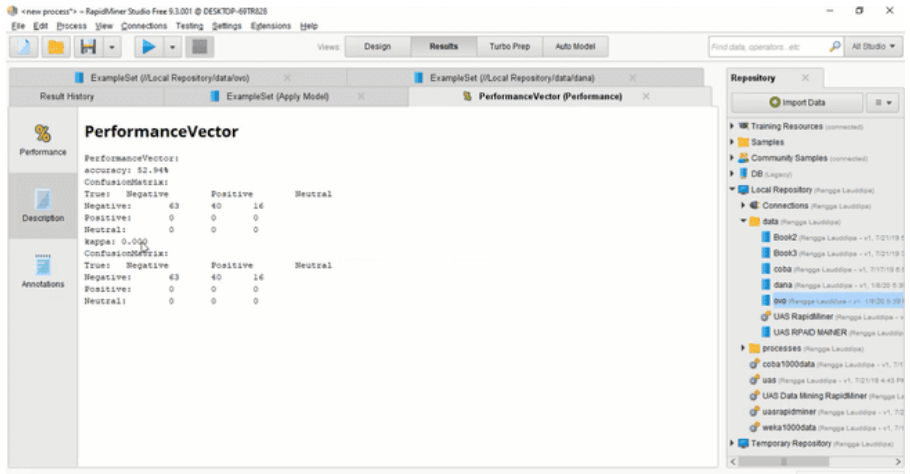
- Tampilan performance accuracy dengan nilai 52.94% dengan menggunakan metode naïve bayes.

PerformanceVector (Performance)

accuracy: 52.94%

|                | true Negative | true Positive | true Neutral | class precision |
|----------------|---------------|---------------|--------------|-----------------|
| pred. Negative | 63            | 40            | 16           | 52.94%          |
| pred. Positive | 0             | 0             | 0            | 0.00%           |
| pred. Neutral  | 0             | 0             | 0            | 0.00%           |
| class recall   | 100.00%       | 0.00%         | 0.00%        |                 |

- Tampilan PerformanceVector

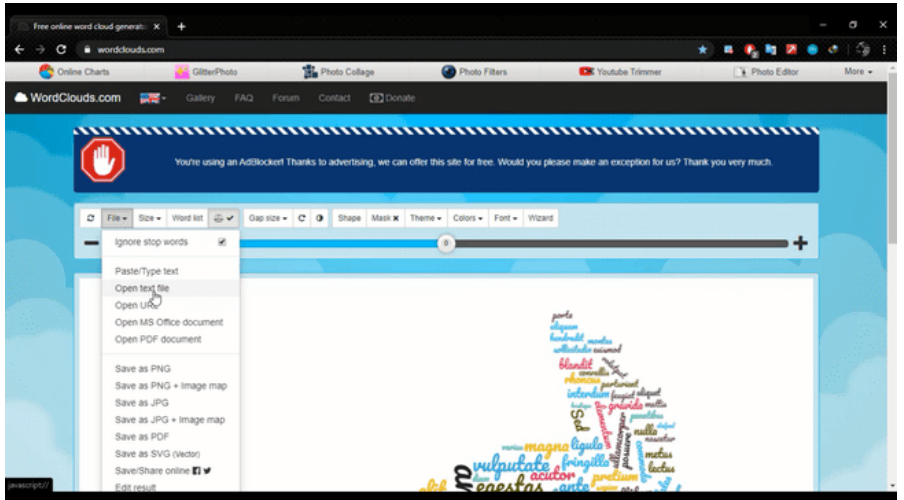


**Kesimpulan:** Jadi dengan menggunakan metode naïve bayes pada data training dana dan testing ovo menghasilkan nilai accuracy 74.79%, sedangkan menggunakan data training ovo dan testing dana menghasilkan nilai accuracy 52.94%.

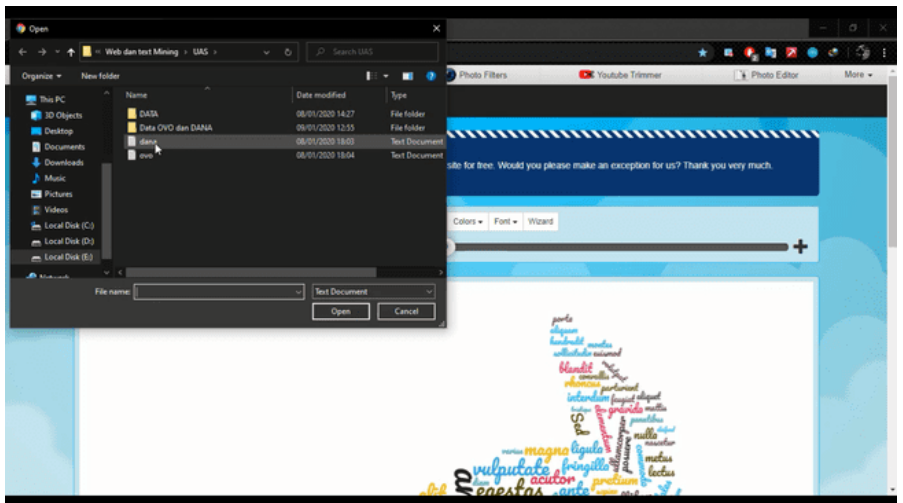
Step 3 : Wordcloud

### Step Wordcloud Data Dana

- Pada tahap wordcloud disini menggunakan web online, klik”File” pilih “Open text file”.



- Lalu pilih data pada lokasi file yang ingin di lakukan proses wordcloud.



- Berikut ini tampilan dari wordcloud data DANA.







Studi Kasus text summarization menggunakan Phyton

5 langkah implementasi Perquisites Python3, NLTK library python, editor teks atau IDE favorit Anda

1. Buat tabel frekuensi kata kami membuat kamus untuk tabel frekuensi kata dari teks. Untuk ini, kita hanya harus menggunakan kata-kata yang bukan bagian dari array stopWords.

```
1 def _create_frequency_table(text_string) -> dict:
2
3 stopWords = set(stopwords.words("english"))
4 words = word_tokenize(text_string)
5 ps = PorterStemmer()
6
7 freqTable = dict()
8 for word in words:
9 word = ps.stem(word)
10 if word in stopWords:
11 continue
12 if word in freqTable:
13 freqTable[word] += 1
14 else:
15 freqTable[word] = 1
16
17 return freqTable
```

create\_frequency\_table.py hosted with ❤ by GitHub [view raw](#)

kami menerapkan metode ini pada text\_string, yang bisa berupa artikel berita, halaman buku, atau email. Tabel frekuensi akan terlihat seperti gambar di bawah ini:



```
▼ { 📄
 "1990" : 1 📄
 "resili" : 2
 "stay" : 2
 "In" : 1
 "game" : 3
 "longer" : 2
 "" : 7
 "On" : 1
 "mountain" : 1
 "truth" : 1
 "never" : 2
```

fig: frequency table

Sekarang, kami membagi `text_string` menjadi satu set kalimat. Untuk ini, kita akan menggunakan metode inbuilt dari `nlTK`

`sent_tokenize(text_string)`

Berikut ini adalah contoh daftar kalimat dari teks yang diberikan.

<https://becominghuman.ai/text-summarization-in-5-steps-using-nltk-65b21e352b65>

## DAFTAR PUSTAKA

1. Abbot, D 2013. Introduction to Text Mining : Virtual Data Intensive Summer School. Abbot Analytics, Inc.
2. Mooney, R. J. 2006. CS 391L Machine Learning Text Categorization. University of Texas, Austin.
3. Statsoft. 2015. Text Mining Introductory Overview. Retrieved August 21, 2020, from Statsoft:  
<http://www.statsoft.com/textbook/text-mining>
4. <https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f>
5. <https://medium.com/@shubhamagarwal328/information-retrieval-in-natural-language-processing-part-1-96c9a62fdb5f>
6. <https://blog.floydhub.com/gentle-introduction-to-text-summarization-in-machine-learning/>
7. <https://medium.com/voice-tech-podcast/information-retrieval-using-boolean-query-in-python-e0ea9bf57f76>
8. <https://becominghuman.ai/text-summarization-in-5-steps-using-nltk-65b21e352b65>

## BIODATA PENULIS



Yulian Findawati, Lahir di Sidoarjo tanggal 25 Juli 1983. Setelah menyelesaikan SLTP, dan SLTA di Sidoarjo, melanjutkan Pendidikan ke ITTELKOM Bandung. Meraih gelar sarjana (ST) pada Prodi Informatika tahun 2007 dan meraih gelar Magister (M.MT) pada jurusan Manajemen Teknologi Informasi ITS. Aktifitas keseharian sebagai dosen tetap Universitas Muhammadiyah Sidoarjo. Ketua Prodi Teknik Informatika Universitas Muhammadiyah Sidoarjo Sejak 2010 sampai dengan 2018. Mata Kuliah yang di ampuh penulis yaitu Rekayasa Perangkat Lunak, Pemrograman Berorientasi Objek, Data Mining, Text Mining, Kecerdasan Buatan , Algoritma Struktur Data dan Teknik Kompilasi.



Mochamad Alfian Rosid, S.Kom., M.Kom Lahir di Sidoarjo, 25 April 1986. Aktifitas keseharian Sebagai Sekretaris Prodi Teknik Informatika Universitas Muhammadiyah Sidoarjo Sejak 2018 sampai dengan sekarang. Mata Kuliah yang di ampuh penulis yaitu Interaksi Manusia Komputer, Algoritma Struktur Data, Algoritma dan Pemrograman, Text Mining,