BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka dari penelitian yang dilakukan sebelumnya dalam mendukung penelitian yang sedang dilakukan. Berikut ini adalah penelitian yang telah dilakuan sebelumnya terkait dengan penelitian yang akan dilakukan oleh penulis:

Tabel 2.1 Tinjauan Pustaka

1.	Judul	Web Scraping for Hospitality Research: Overview, Opportunities, and Implications.
	Penulis	Saram Han dan Christopher K. Anderson
	Tanggal/tahun	November 2020
	Permasalahan	Penelitian dari pasar perhotelan sering mengalami kegagalan dikarenakan kompleksitas dari pasar perhotelan dikarenakan sulitnya mendapatkan data online dalam skala besar.
	Solusi	Diterapkan sebuah metode yang secara otomatis mengumpulkan data online menggunakan selenium.
	Hasil penelitian	Penelitian menghasilkan sebuah system yang secara otomatis mengambil data online perhotelan diberbagai platform, situs ulasan, dan situs perhotela itu sendiri.
2.	Judul	Analisis Sentiment Terhadap Vaksin Covid-19 di Indonesia pada Twitter Menggunakan Metode Lexicon Based

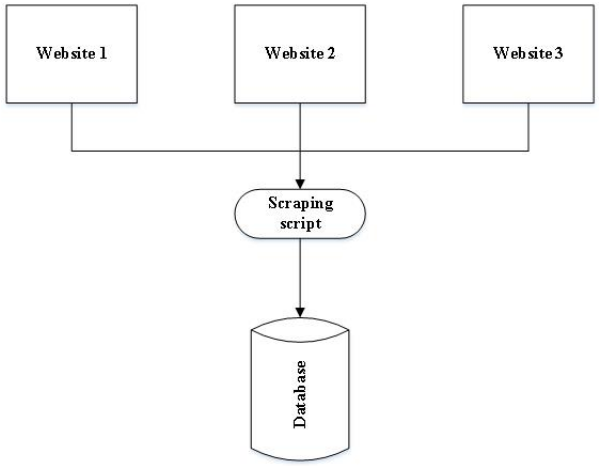
	Penulis	Putri Amira Sumitro, Rasiban, Dadang Iskandar Mulyana, dan Wahyu Saputro
	Tanggal/tahun	Oktober 2021
	Permasalahan	Seiring dengan kebijakan pemerintah RI wajib vaksin covid-19, perlu dilakukan analisis untuk melihat tingkat kepuasan terhadap kebijakan tersebut.
	Solusi	Dibuatkan sebuah web scrapping untuk mengambil data online, menggunakan ntlk.lib untuk pemrosesan data, dan merapkan metode lexicon based untuk pengkategorian data.
	Hasil penelitian	Penelitian menghasilkan pengelompokan opini para pengguna twitter menjadi beberapa kategori, yaitu: positif, sedikit positf, netral, negatif, dan sedikit negative.
3.	Judul	Perancangan Aplikasi Otomisasi Menggunakan Bahasa Pemograman Python Pada Aktivitas Monitoring Pemakaian Data Harian Kartu Internet Of Things.
	Penulis	Jaka Naufal Semendawai, Indah Febiola, Bima Pamungkas, dan Muhammad Deka Ruliansyah.
	Tanggal/tahun	15 November 2021
	Permasalahan	Aktivitas monitoring masih dilakukan secara manual memakan waktu yang lebih lama, biaya yang lebih besar, dan faktor human error yang dapat mengakibat laporan mingguan atau bulan menjadi tidak akurat.
	Solusi	Dibuatkan aplikasi web crawler yang secara otomatis melakukan aktivitas monitoring.

	Hasil penelitian	Penelitian menghasilkan sebuah aplikasi web crawler yang secara otomatis melakukan pengunduhan, upload, dan memonitoring penggunaan bandwith harian.
4.	Judul	Analisa Judul Skripsi untuk Menentukan Peminatan Mahasiswa Menggunakan Vector Space Model dan Metode K-Nearest Neighbor
	Penulis	Dewi Marini Umi Atmaja dan Rila Mandala
	Tanggal/tahun	2021
	Permasalahan	Sulitnya menentukan klasifikasi topik dan permintaan judul tugas akhir mahasiswa yang hanya dikelompokkan berdasarkan perkiraan isi konten yang akan diteliti mahasiswa sehingga kesesuaian Antara judul, topik, dan permintaan yang dipilih oleh mahasiswa tidak sesuai.
	Solusi	Diterapkan metode vector space model dan K-Neareast Neighbor untuk mengukur tingkat kemiripan antar dokumen.
	Hasil penelitian	Penelitian menghasilkan analisis perbandingan dari 2 metode dimana algoritma KNN mempunyai akurasi lebih besar, namun kurang efektif untuk data yang memiliki banyak kelas.
5.	Judul	Implementasi Vector Space Model Pada Sistem Pencarian Mesin Karaoke
	Penulis	Anna dan Ade Hendini.
	Tanggal/tahun	2018

	Permasalahan	Semakin banyaknya bisnis karaoke diperlukan sistem pencarian lagu berdasarkan beberapa kategori judul, band, tahun, genre, album demi kenyamanan konsumen.
	Solusi	Dibuatkan sebuah sistem yang menenrapkan metode vector space model.
	Hasil penelitian	Sistem yang dapat menampilkan lagu berdasarkan input dari pengguna dan dikelompokan berdasarkan beberapa kategori.

2.2 *Web Scraping*

Web scraping merupakan kegiatan yang dilakukan untuk mengambil data tertentu secara semi-terstruktur dari sebuah halaman *website*. Halaman tersebut umumnya dibangun menggunakan bahasa *markup* seperti HTML atau XML, proses akan menganalisis dokumen sebelum memulai mengambil data. Dengan melakukan *web scraping*, pengumpulan data menjadi lebih cepat, dan apabila data dikumpulkan dalam jumlah besar tidak perlu melakukan secara manual. *Web scraping* dapat dilakukan dengan menggunakan Scrapper, BeautifulSoup, Scraper API, dan lain – lain (Yoel Julianto, Djoni Haryadi Setiabudi, Silvia Rostianingsih 2022). Proses web *scraping* dadpat dilihat pada Gambar 2.1



Gambar 2.1 Gambaran *web scrapping*

2.2.1 BeautifulSoup

BeautifulSoup merupakan *library* bawaan dari Python untuk *parsing* HTML dan XML . BeautifulSoup bekerja dengan *parser* bawaan Python atau *parser* lain lxml atau html5lib untuk mempermudah anda dalam mengambil data dari suatu situs web. Pada saat ini Beautiful soup telah sampai pada versi yang ke 4. Kelebihan *library* Beautifulsoup adalah bekerja lebih baik untuk data berukuran besar, mudah dipelajari bagi pemula, dan menggunakan deteksi *encoding* otomatis. Satu-satunya kelemahan yang dimiliki *library* ini adalah lebih lambat jika dibandingkan dengan *library* lxml pada Python.

2.2.2 Selenium

Selenium merupakan alat *auto testing* yang digunakan untuk mengotomatisasi tes aplikasi web yang dilakukan pada browser. Pengujian selenium dapat dilakukan menggunakan browser apapun, dan script tes dapat ditulis dalam banyak bahasa pemrograman seperti Java, C#, Python, Rubym PHP, dan lain sebagainya. Auto testing tools dikhususkan untuk otomatisasi testing pada web apps, sehingga pengujian pada aplikasi desktop dan seluler tidak akan bisa

dilakukan. Selenium bersifat *open source*, dimana *tools* ini dapat diunduh dan digunakan secara gratis. Karena sifatnya yang terbuka, pengguna dapat membagi, memperluas, dan memodifikasi kode yang tersedia.

2.3 Natural Language Toolkit

Natural Language Toolkit merupakan rangkaian perpustakaan dan program pengolahan bahasa simbolik dan statistic alami (NLP) yang ditulis dalam bahasa Pemrograman Python. NLTK dimaksudkan untuk mendukung penelitian dan pengajaran di NLP yang termasuk ilmu linguistic empiris, ilmu kognitif, kecerdasan buatan, pencarian informasi, dan pembelajaran mesin. NLTK telah berhasil digunakan sebagai alat belajar pribadi, dan sebagai alat perancah dan sistem riset bangunan (Jiawei Yao 2019). NLTK menyediakan librari untuk text processing mulai dari penghapusan *stopwords*, *tokenizing*, *stemming*, *lemmitazing*, dll. NLTK tersedia adalah salah satu open source tools yang bisa diakses secara gratis, dan tersedia baik untuk sistem operasi Windows, Mac OS X dan Linux.

2.3.1 *Translate*

Translate (terjemah) pada dasarnya (Andy Bayu Nugroho, 2007) adalah proses penyampaian makna atau maksud dari suatu wacana linguistik tertentu suatu bahasa ke dalam bahasa lain, lebih dari sekedar mentransfer kata atau struktur gramatikal. Arti kata atau himpunan kata dapat dipahami dengan baik karena perannya dalam keseluruhan linguistic ekspresi di mana mereka terjadi. Karena itu, arti sebuah kata tidak hanya ditentukan oleh objek atau ide yang dirujuk, tetapi juga diatur oleh penggunaan kata atau frasa dengan cara, konteks, dan efek tertentu.

2.3.2 *Tokenizinng*

Dalam penelitian (Utami, 2017) *Tokenizing* adalah proses memisahkan deretan kata di dalam kalimat, paragraf atau halaman menjadi token atau potongan kata tunggal atau *termmed* word yang berdiri sendiri. Di dalam *tokenizing* karakter dan symbol selain a-z dihilangkan, pemecahan kalimat dan kata dilakukan berdasarkan pada spasi di dalam kalimat tersebut. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua token ke bentuk huruf kecil (*lower case*).

2.3.3 Penghapusan *Stopwords*

Menurut (Kaur and Buttar, 2018) *stopword* yang dikenal sebagai *stoplist* merupakan sebuah metode otomatis yang dikembangkan untuk mengidentifikasi sebuah data dengan menghapus data dari *dataset* sebelumnya. Hal ini dilakukan untuk mengambil informasi yang mempunyai proporsi besar yang berguna bagi yang melakukan penelitian. *Stopwords* pertama kali diperkenalkan pada tahun 1958 oleh H.P. Luhn. *Stopwords* adalah kata-kata yang paling sering muncul dalam sebuah dokumen dan berisi sedikit informasi yang biasanya tidak diperlukan dalam penelitian. Misalnya, dalam bahasa inggris ada beberapa kata seperti di atas, setelah, lagi, terhadap, semua, pagi, sebuah, dan, menjadi, selama, dan lain-lain.

2.3.4 *Stemming*

Stemming adalah proses pencarian bentuk dasar suatu kalimat dengan cara menghilangkan imbuhannya. *Stemming* merupakan suatu proses yang terdapat

dalam sistem IR yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu. *Stemming* sangat penting dalam mendukung efektivitas pencarian informasi dalam bahasa Indonesia, penerjemahan dokumen, dan pencarian dokumen teks. Imbuhan bahasa Indonesia lebih kompleks dari pada bahasa inggris karena di dalam bahasa Indonesia terdapat awalan (prefiks), sisispan (infiks), akhiran (sufiks), konfiks (gabungan prefiks dan sufiks). Sehingga *stemming* bahasa Indonesia harus mampu menemukan akar kata sesuai dengan aturan baku bahasa Indonesia.

2.4 SQLite

SQLite merupakan sebuah sistem manajemen basisdata relasional yang bersifat *ACID-compliant* dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa C. SQLite merupakan proyek yang bersifat *public domain* yang dikerjakan oleh D. Richard Hipp. Tidak seperti pada paradigma *client-server* umumnya, inti SQLite bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead*, *latency times*, dan secara keseluruhan lebih sederhana. Seluruh elemen basisdata (definisi data, tabel, indeks, dan data) disimpan sebagai sebuah file. Kesederhanaan dari sisi disain tersebut bisa diraih dengan cara mengunci keseluruhan file basis data pada saat sebuah transaksi dimulai.

Pustaka SQLite mengimplementasikan hampir seluruh elemen-elemen standar yang berlaku pada SQL-93, termasuk transaksi yang bersifat *atomic*, konsistensi basisdata, isolasi, dan durabilitas (dalam bahasa inggris lebih sering disebut *ACID*), *trigger*, dan kueri-kueri yang kompleks. Tidak ada pengecekan tipe sehingga data bisa dientrikan dalam bentuk string untuk sebuah kolom bertipe integer. Beberapa kalangan melihat hal ini sebagai sebuah inovasi yang menambah nilai guna dari sebuah basisdata, utamanya ketika digunakan dalam bahasa pemrograman berbasis script (*PHP*, *Perl*).

Beberapa proses ataupun *thread* dapat berjalan secara bersamaan dan mengakses basisdata yang sama tanpa mengalami masalah. Hal ini disebabkan karena akes baca data dilakukan secara paralel. Sementara itu akses tulis data hanya bisa dilakukan jika tidak ada proses tulis lain yang sedang dilakukan; jika tidak, proses tulis tersebut akan gagal dan mengembalikan kode kesalahan (atau bisa juga secara otomatis akan mencobanya kembali sampai sejumlah nilai waktu yang ditentukan habis). Hanya saja ketika sebuah tabel temporer dibuat, mekanisme penguncian pada proses multithread akan menyebabkan masalah.

2.5 *Term Frequency – Inverse Document Frequency* (TF-IDF)

Term Frequency-Inverse Document Frequency merupakan teknik dalam memberikan bobot hubungan suatu *term* terhadap sebuah dokumen. TF-IDF bekerja dengan memberikan bobot pada suatu *term* dalam sebuah dokumen (Anggiharto, A. 2013). Metode ini bekerja dengan menggabungkan dua konsep untuk perhitungan bobot, yaitu frekuensi kemunculan sebuah *term* di dalam sebuah dokumen (tf) dan inversi frekuensi dokumen (idf) yang mengandung kata

tersebut (Uden & V., M 2011). Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata tersebut di dalam dokumen tersebut. Jumlah frekuensi dokumen yang mengandung kata tersebut menunjukan seberapa umum kata tersebut. Semakin sedikit jumlah dokumen yang mengandung term yang dimaksud maka nilai pada idf akan semakin besar. Berikut persamaan untuk penghitungan *term frequency*: $tf = tf_{i,j}$

Dengan tf adalah *term frequency*, dan $tf_{i,j}$ adalah banyaknya kemunculan *term* t_i dalam dokumen d_j, *Term frequency* (tf) dihitung dengan menghitung banyaknya kemunculan *term* t_i dalam dokumen d_j.

Dengan idf_i adalah *inverse document frequency*, *N* adalah jumlah dokumen, dan df_i adalah banyaknya dokumen dalam koleksi dimana *term* ti muncul di dalamnya, maka perhitungan idf_i digunakan untuk mengetahui banyaknya *term* yang dicari (df_i) yang muncul dalam dokumen lain yang ada pada basisdata. Persamaan untuk perhitungan *inverse document frequency* dapat dilihat pada persamaan (1).

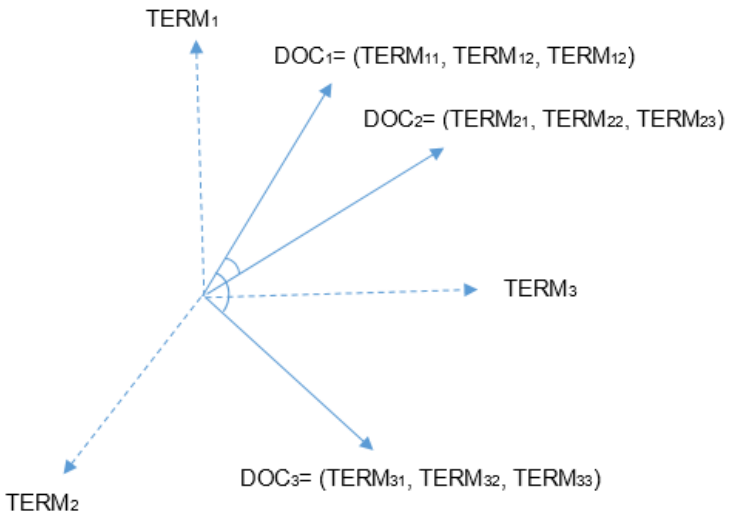
$$idf_i = \log(\frac{N}{df_i}) \qquad (1)$$

Perhitungan bobot dari *term* tertentu dalam sebuah dokumen menggunakan perkalian nilai tf dan idf menunjukkan bahwa deskripsi terbaik dari dokumen adalah term yang banyak muncul dalam dokumen tersebut dan sangat sedikit muncul pada dokumen yang lain. Perhitungan bobot *term frequency-inverse document frequency* dapat dilihat pada persamaan (2).

$$W_{i,j} = tf_{i,j} \log(\frac{N}{df_i}) \qquad (2)$$

2.6 *Vector Space Model* (VSM)

Vector Space Model (VSM) merupakan suatu metode yang digunakan untuk mengukur tingkat kedekatan atau kesamaan (*similarity*) *term* dengan cara pembobotan pada *term* (Amin, F 2012). Dokumen diasumsikan sebagai sebuah *vektor-vektor* yang memiliki jarak (*magnitude*) dan arah (*direction*). Dalam metode ini, sebuah *term* direpresentasikan dengan sebuah dimensi dari ruang *vektor*. *Term* yang digunakan umumnya berdasarkan kepada *term* yang ada pada *query* atau *keyword*. Relevansi sebuah dokumen ke sebuah *query* didasarkan pada *similaritas* di antara *vector* dokumen dan *vektor query*. Perhitungan kesamaan *antara vektor query* dengan *vektor* dokumen dilihat dari sudut yang paling kecil. Gambar 2.2 merupakan contoh dari model ruang *vektor* tiga dimensi untuk 2 dokumen di mana D adalah dokumen, Q adalah *query*, dan T adalah *term* yang menjadi dimensi dari VSM.



Gambar 2.2 *Vector Space Model*

Vector Space Model dan pembobotan TF-IDF digunakan untuk merepresentasikan nilai numerik dokumen sehingga kemudian dapat dihitung kedekatan antar dokumen. Semakin dekat dua *vektor* di dalam suatu VSM maka semakin mirip dua dokumen yang diwakili oleh *vektor* tersebut. Kemiripan antar dokumen dihitung menggunakan suatu fungsi ukuran kemiripan *(similarity measure*). Ukuran ini memungkinkan perankingan dokumen sesuai dengan kemiripan relevansinya terhadap *query*. Perhitungan untuk mengukur tingkat kemiripan antara dokumen dan *query* dapat dilihat pada persamaan (3).

$$Sim(d_j, q) = \frac{\sum_{i=1}^{t}(W_{ij} \times W_{iq})}{\sqrt{\sum_{i=1}^{t}(W_{ij})^2 \times \sum_{i=1}^{t}(W_{iq})^2}} \qquad (3)$$

Keterangan :

d_j : dokumen ke *j*

q : *query*

$\sum_{i=1}^{t} W_{ij}$: jumlah bobot kata *i* pada dokumen *j*

$\sum_{i=1}^{t} W_{iq}$: jumlah bobot kata *i* pada *query*