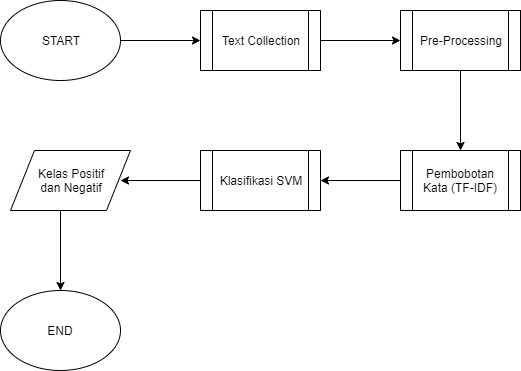
**3. METODE PENELITIAN**

Metode penelitian pada bab ini menjelaskan mengenai alur dan metode analisis sentimen terhadap topik Komisi Pemilihan Umum (KPU) yang lebih mendetail, juga perancangan terhadap aplikasi *interface* (antarmuka), yang berfungsi untuk menampilkan hasil dari analisis.

* + 1. **Alur Proses Analisis Sentimen**

Proses analisis sentimen pada penelitian ini memiliki berberapa tahapan yang harus dilakukan untuk mendapatkan hasil yang diinginkan. Berikut adalah tahapan atau metode yang digunakan dalam melakukan analisis sentimen pada penelitian ini.



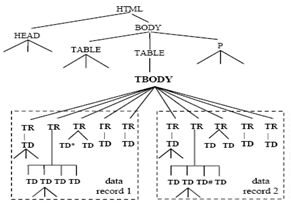
**Gambar 3.1** Alur Proses Analisis Sentimen

Gambar 3.1 menunjukan langkah – langkah yang harus di lakukan, berikut penjelasannya:

1. *Text* *Collection*, disebut juga sebagai ekstraksi data, adalah proses pengambilan data dengan menggunakan twitter API.
2. Pre-Proces, adalah proses setelah data diekstraksi, yaitu *cleansing* untuk penyaringan dan pembersihan data tweet dari kata dan karakter yang tidak perlu, *case folding* untuk mengubah seluruh tweet menjadi huruf kecil dan *stemming* untuk mengubah kalimat menjadi kata dasar.
3. Pembobotan kata, pembobotan kata menggunakan TF-IDF (*Term Frequency – Inverse Term Frequency*) yang melakukan perhitungan bobot setiap kata yang telah di ekstrak.
4. *Support* *Vector* *Machine*, pada proses ini dilakukan training data SVM dan *Testing* *data*.
5. Menampilkan *Output*, *output* hasil akhir akan ditampilkan.

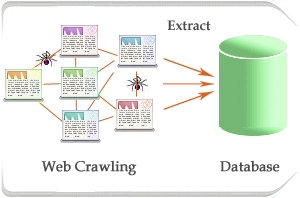
**3.2 Ekstraksi Tweet**

Ekstraksi tweet adalah pengambilan data dari *Twitter* yang akan dijadikan sebagai objek penelitian dengan kata pencarian “#kpuid” dan “#kpu” yang dilakukan dengan menggunakan *Twitter API* dan bahasa pemrograman Python. Ekstraksi data pada halaman web menggunakan algoritma *partial tree alignment* yang memetakan sebuah website seperti sebuah pohon dan mengambil datanya pada setiap elemen html. Gambaran dari algoritma partial tree alignment dapat dilihat pada gambar 3.2.



**Gambar 3.2** *Partial Tree Alignment*

Ekstraksi dilakukan dengan menggunakan web crawler untuk mengambil data secara otomatis. *Web* *crawler* atau yang dikenal juga dengan istilah web *spider* atau *web* *robot* adalah *program* yang bekerja dengan metode tertentu dan secara otomatis mengumpulkan semua informasi yang ada dalam suatu website [20]. Sebelum melakukan pengambilan data, dibutuhkan berberapa kode untuk mendapatkan hak akses ke *Twitter.* Untuk mendapatkan hak akses tersebut diperlukan berberapa tahapan dalam mendapatkan kode hak akses. Gambaran dari proses *web* *crawler* dapat dilihat pada gambar 3.3.



**Gambar 3.3** Proses Ekstraksi

**3.2.1 Membuat Aplikasi Twitter API**

Langkah pertama yang harus dilakukan adalah mendaftarkan akun twitter menjadi pengembang pada *website* <https://developer.twitter.com/en/apps> dan melakukan pendaftaran apps. Tahapan ini adalah tahapan paling dasar untuk mengambil data dari *Twitter.* Berikut adalah tahapannya:

* 1. Pertama yang harus dilakukan adalah login ke halaman <https://developer.twitter.com/en/apps> untuk melakukan langkah ini, harus memiliki akun twitter terlebih dahulu dan login ke dalam website.
  2. Langkah berikutnya adalah membuat aplikasi baru yang dilakukan dengan menekan tombol *Create an app.* Tombol ini akan mendirect ke halaman form *App details* kemudian mengisi form yang tersedia dengan data-data yang valid. Data – data yang harus diisi diantaranya:

1. **App detail**

Berisi tentang detail dari aplikasi yang ingin dibuat

1. **App name**

Adalah nama dari app yang ingin dibuat

1. **Description,**

Adalah deskripsi tentang app yang ingin dibuat

1. **Organisation name**

Nama dari organisasi pembuat(jika ada)

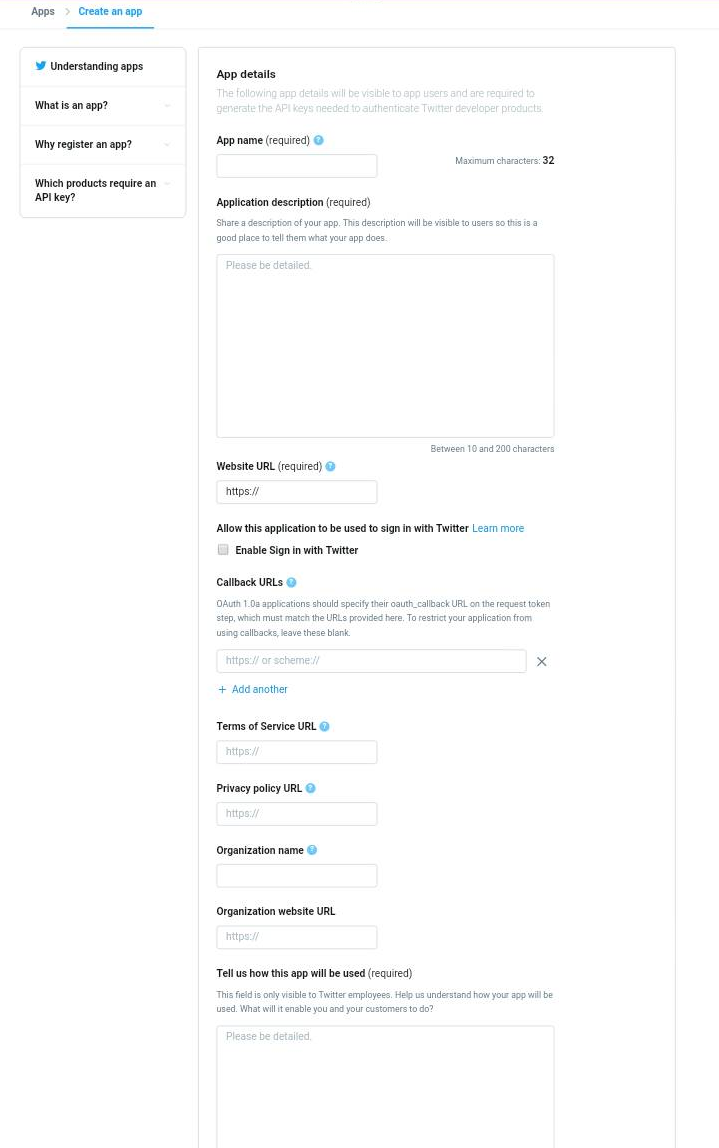
1. **Organisation website URL**

URL dari halaman web organisasi(jika ada)

1. **App usage**

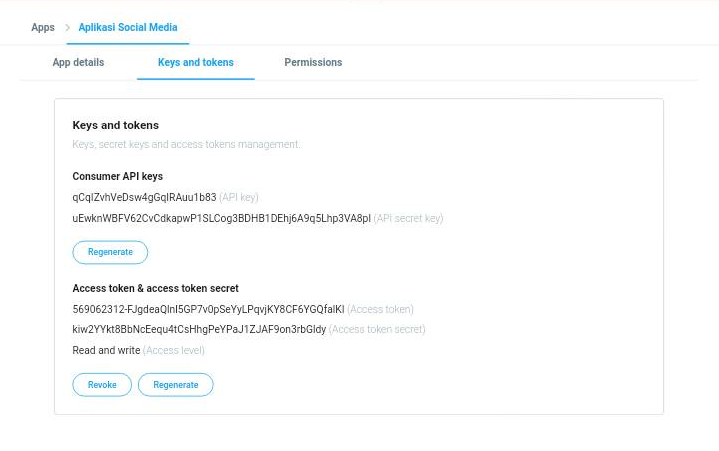
kegunaan dari app yang ingin dibuat

Contoh gambar dapat dilihat pada gambar 3.4.



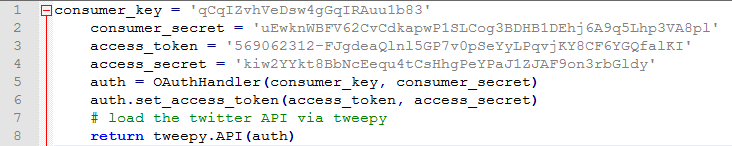
**Gambar 3.4** Halaman Application Details

* 1. Setelah form diisi dan disimpan maka akan dialihkan ke halaman selanjutnya yang menunjukan detail aplikasi yang telah dibuat, selanjutnya menekan tab keys and token maka akan muncul semua key yang dibutuhkan yakni *Consumer API keys, Acces token & Acces Token Secret* seperti pada gambar 3.5. seluruh key ini bersifat rahasia dan harus didapatkan untuk mengambil data dari *Twitter.*



**Gambar 3.5** Kunci dan Token dari Twitter API

Setelah mendapatkan *key* yang akan digunakan untuk mendapatkan hak akses dari Twitteryang selanjutnya akan digunakan untuk melakukan pengambilan data, pengguna dapat menyalin seluruh key tersebut dan memindahkannya ke *text* *editor* dengan berberapa sentuhan bahasa *python* seperti pada gambar 3.6.



**Gambar 3.6** Baris kode Autentifikasi Twitter API

**3.2.2 Pengambilan Data Twitter**

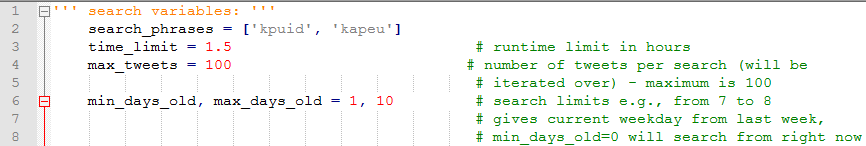
Setelah mendapatkan *key* untuk autentifikasi, kemudian dilakukan berberapa langkah untuk melakukan pengambilan data Twitter. Langkah – langkahnya adalah:

1. Menginstal library Tweepy

Tweepy adalah library bahasa python untuk mengakses Twitter API yang sangat mudah untuk digunakan. Untuk menginstallnya pengguna harus sudah meng-install python versi 3 ke-atas. Kemudian menjalankan perintah pip install tweepy pada command prompt.

1. Menjalankan Script

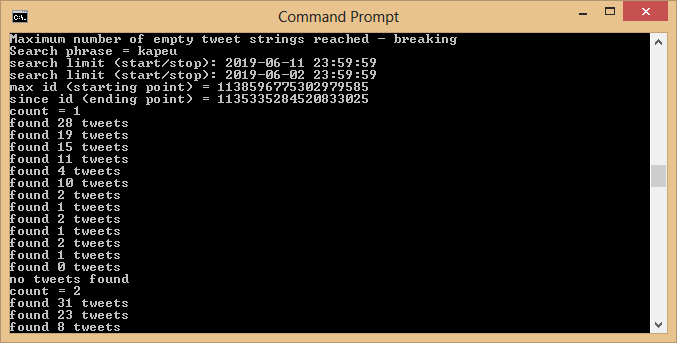
Untuk mempermudah proses pencarian data, penulis menggunakan script python. Pada *script* dilakukan pengisian *validation* *keys* yang telah di dapat pada proses sebelumnya serta kata kunci pencarian untuk melakukan pencarian sesuai kata kunci yang diinginkan pada script twitter\_search.py seperti pada gambar 3.7.



**Gambar 3.7** Memasukan kata kunci pada kode

1. Melakukan Pencarian dan Pengambilan Data

Setelah langkah – langkah diatas selesai dilakukan, langkah selanjutnya adalah melakukan pencarian dan pengambilan data twitter atau bisa disebut crawling. Hal yang harus dilakukan adalah menjalankan file twitter\_search.py pada cmd dan tunggu hingga prosesnya selesai.

****

**Gambar 3.8** Proses pengambilan data dari Twitter

Jika proses pencarian data berhasil, maka hasil pencarian akan menghasilkan sebuah file json yang berisi seluruh data pencarian dari hastag twitter yang menggunakan bahasa *python*. Proses pencarian memakan waktu tergantung dari jumlah data yang ingin diambil. Hasil dari pencarian dan pengambilan data dapat dilihat pada Tabel 3.1.

**Tabel 3.1** Hasil Ekstraksi Data Twitter

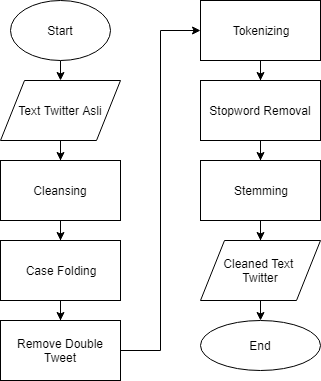
|  |  |
| --- | --- |
| **No** | **Text Tweet** |
| 1. | Bersihkan KPU dari praktek demokrasi kotor, https://t.co/07RXap5JOH |
| 2. | RT @KPU\_ID: Anak Kelas ll MTS Bobol Situs KPU Pemilu 2019 |
| 3. | @reno\_2 @Dody\_Lucas\_\_ @AkunTofa @KPU\_ID @bawaslu\_RI Mantap... |
| 4. | @Nadiku18 Yaa itu mah 02 nya gak legowo sama hasil situng |
| 5. | RT @mansiank83: KOMPLEK MEGA PERMAI 1 MANA SUARANYA??\n\nKESALAHAN INPUT KPU DI TPS 5 KELURAHAN PADANG SARAI KOMPLEK MEGA PERMAI 1 KOTA PADA\u2026 |
| 6. | @RadioElshinta @KPU\_ID @republikaonline https://t.co/XrnX1ecCtz |
| 7. | RT @b0y4n\_05: Palembang :\n350.539 suara (38,52%) : 559.422 suara (61,48%)\n\nhttps://t.co/seqAqbRzgx |
| 8. | RT @Yswn\_Kawilarang: TPS 10 Gerung utara kec Gerung, Lombok barat NTB. Input di web KPU tidak sesuai dengan C1 asli |
| 9. | Ini \ud83d\udc4e malah lebih dahsyat salah inputnya:\n\nhttps://t.co/7lrEZTrcbV" |
| 10. | Kesalahan KPU Input Data 5 Formulir C1 Itu\u00a0Manusiawi https://t.co/l52MGashcr https://t.co/xDZArQaYwB |
| 11. | RT @iqrozen: @AkunTofa @KPU\_ID Nih sedikit bocoran nya https://t.co/VMtZPoStbY |
| 12. | RT @imayadewi: Saya tidak salahkan @KPU\_ID KPU. Saya pertanyakan Komisi II DPR yang memilih manusia goblok kayak gini jadi komisioner KPU !\u2026 |
| 13. | "RT @alan\_borut: rakyat surabaya menuntut suara mereka yg sdh dirampok. inilah hasil dr kecurangan yg dibiarkan.rakyat pun melawan.mrk merin\u2026 |
| 14. | RT @helmifelis: Pemilu kali ini gak beres.\nTidak cuma situngnya yang gak beres |
| 15. | RT @mediaindonesia: Tolak Narasi Kecurangan, PMII Gelar Aksi Dukung KPU dan Bawaslu\n\n https://t.co/hgGDrk5Vml |
| 16. | @EndangPRS @KPK\_RI Aduh pak BAP nya udah di revisi, kirain cuma situng KPU aja yang bisa direvisi \ud83d\ude01\nBener-bener sem\u2026 |
| 17. | RT @CNNIndonesia: Relawan Prabowo Minta Situng KPU Dihentikan https://t.co/R6AM3lqL4k |
| 18. | RT @b0y4n\_05: Kab.Rejang Lebong, Bengkulu : \n69.543 suara : 90.540 suara\n\nhttps://t.co/a7IydpfqEt |
| 19. | RT @rmolco: KPU, Situng Bikin Gaduh, Buat Apa Dipertahankan? #KlikRMOL https://t.co/kZ7tZnRWw9 |
| 20. | **.....** |

**3.3 Pre-*Processing* *Data***

Pada pre-*processing* *data* adalah tahapan yang harus dilakukan sebelum melakukan klasifikasi. Pre-*processing* dilakukan untuk melakukan perubahan bentuk data menjadi lebih terstruktur dan sesuai dengan kebutuhan untuk pemrosesan lebih lanjut. Berberapa tujuan dari pre-*processing* *data* adalah :

1. Untuk mempermudah sehingga mempermudah pemilihan teknik dan metode *data* *mining* yang tepat
2. Untuk meningkatkan kualitas *data* sehingga hasil dari data mining menjadi lebih baik
3. Untuk meningkatkan efisiensi dan kemudahan proses penambangan *data*

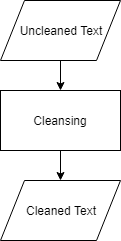
Pre-processing data terdiri dari berberapa langkah, langkah – langkah dalam pre-*processing* dapat dilihat pada gambar 3.9.



**Gambar 3.9** Alur Proses Pre-Processing Data

**3.3.1 *Cleansing* *Data***

*Cleansing* *data* adalah salah satu langkah dalam proses pre-*processing* untuk membersihkan *data* yang ‘kotor’ supaya dapat digunakan untuk proses lebih lanjut. Langkah – langkah untuk melakukan *cleansing* data dapat dilihat pada gambar 3.10 :



**Gambar 3.10** Alur Proses Cleansing Data

Pada proses cleansing, yang dilakukan untuk membersihkan tweet adalah sebagai berikut:

1. Menghapus URL pada tweet
2. Menghapus hashtag
3. Menghapus RT
4. Menghapus @
5. Menghapus *Double* *Tweet*

Setelah cleansing contoh hasil dari cleansing *tweet* dapat di lihat pada tabel 3.2

**Tabel 3.2** Contoh Cleansing Data

|  |  |
| --- | --- |
| **Data Twitter Sebelum Cleansing** | **Sesudah Cleansing** |
| RT @CNNIndonesia: Relawan Prabowo Minta Situng KPU Dihentikan <https://t.co/R6AM3lqL4k> | Relawan Prabowo Minta Situng KPU Dihentikan |
| RT @do\_ra\_dong: Sesuai kicauan kami belum lama ini\n\nPakar ITB sebut situng dikendalikan intruder \n\nKami sebut itu berasal dari depan gambir\u2026 | Sesuai kicauan kami belum lama ini Pakar ITB sebut situng dikendalikan intruder Kami sebut itu berasal dari depan gambir |
| RT @PDI\_Perjuangan: TERBARU Hasil Real Count KPU Pilpres 2019 Jokowi vs Prabowo Hari Ini Jumat 3 Mei Data Masuk 63%. Jokowi-Ma'ruf 55.94% | TERBARU Hasil Real Count KPU Pilpres 2019 Jokowi vs Prabowo Hari Ini Jumat 3 Mei Data Masuk 63 Jokowi Ma ruf 55 94 |

* + 1. ***Case Folding***

*Case* *folding* adalah tahapan merubah seluruh data menjadi huruf kecil atau lowercase. Hal ini dilakukan untuk memudahkan program menjalankan fungsi – fungsinya dan terhindar dari case sensitive error. Sebuah contoh *case* *folding* dapat dilihat pada tabel 3.3.

**Tabel 3.3** Contoh Case Folding

|  |  |
| --- | --- |
| **Tweet** | **Hasil** |
| Relawan Prabowo Minta Situng KPU Dihentikan | relawan prabowo minta situng kpu dihentikan |

* + 1. ***Tokenizing***

*Tokenizing* adalah proses pemecahan sebuah kalimat menjadi kata, kata – kata pembentuk sebuah kalimat akan dipecah menjadi sebuah kata tunggal. Langkah ini dilakukan untuk melakukan pelabelan atau *scoring* yang membutuhkan sebuah kata untuk parameternya. Satu contoh dari *tokenizing* *tweet* dapat dilihat pada tabel 3.4

**Tabel 3.4** *Tokenizing* Tweet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tweet** | **Tokenized Tweet** | | | |
| relawan prabowo minta situng kpu dihentikan | relawan | prabowo | minta | situng |
| kpu | dihentikan |  |  |

Pada proses *tokenizing* kalimat dipecah menjadi bentuk per-kata. Karakter spasi dihilangkan karena dianggap sebagai delimiter atau pemisah antar kata.

* + 1. ***Stopword Removal***

*Stopword Removal* adalah proses penghapusan daftar kata yang tidak memiliki makna. Langkah ini dilakukan untuk mengurangi jumlah kata yang disimpan oleh sistem. Stopwords yang digunakan adalah milik masdevid yang dapat di download pada link <https://github.com/masdevid/ID-Stopwords> Yang kemudian dilakukan penambahan kata yang terdapat dalam dokumen dan membuat *list stopword* dari daftar kata yang ada kemudian dilakukan penghapusan *stopword*.

Berikut adalah daftar stopword yang akan dihilangkan :

**Tabel 3.5** List Stopword

|  |  |
| --- | --- |
| **No** | **Stopword** |
| 1. | Loh |
| 2. | Milu |
| 3. | Mts |
| 4. | Kec |
| 5. | Itu |
| 6. | kayak |
| 7. | Gini |
| 8. | Yg |
| 9. | Ini |
| 10. | Biar |
| 11. | Gak |
| 12. | cuma |
| 13. | merin |
| 14. | Aduh |
| 15. | kirain |
| 16. | Kab |
| 17. | Pas |
| …. | ….. |
| 167. | Dong |

Penghapusan *stopword* dilakukan dengan menghilangkan kata *stopword* pada kalimat tweet. Pada proses penghapusan *stopword*, kata-kata yang terdapat pada list *stopwords* dihilangkan dari kalimat data tweet. Contoh dapat dilihat pada tabel 3.6.

**Tabel 3.6** Tweet setelah stopword

|  |  |  |
| --- | --- | --- |
| **No** | **Tweet** | **Tweet Setelah Stopword** |
| 1 | bersihkan kpu dari praktek demokrasi kotor | bersihkan kpu praktek demokrasi kotor |

* + 1. ***Stemming***

*Stemming* adalah proses pencarian kata dasar, yakni mengubah kata yang berimbuhan menjadi kata dasarnya. Pada proses *stemming* ini digunakan *library* *python* yang bernama Sastrawi. *Library* ini bisa diinstall dengan menggunakan perintah ‘pip install sastrawi’ pada terminal. Hasil yang didapatkan dengan melakukan proses stemming adalah pengubahan data *tweet* menjadi kata dasar. Contoh dapat dilihat pada tabel 3.7

**Tabel 3.7** Contoh Proses *Stemming*

|  |  |
| --- | --- |
| **Tweet** | **Tweet Stemmed** |
| Sesuai kicauan belum lama pakar sebut situng kendali intruder kami sebut sebut berasal depan gambir | sesuai kicau lama pakar sebut situng kendali intruder sebut asal depan gambir |

* + 1. **Klasifikasi dengan *Lexicon* *Based* *Features***

Dalam proses labeling ini, dilakukan berdasarkan kata positif dan negatif yang terdapat pada tweet yang telah di bersihkan melalui tahap pre-*processing*. Klasifikasi ini dilakukan dengan kata-kata yang terdapat dalam database kata positif dan database kata negatif. Klasifikasi dilakukan dengan cara mencari jumlah kata positif dan negatif di dalam tweet, apabila dalam satu kalimat jumlah kata positif lebih besar dari kata bersentimen negatif, akan digolongkan sebagai sentimen positif dan diberi skor 1. Namun, jika sebaliknya kalimat akan digolongkan sebagai sentimen negatif dan diberi skor -1. Database untuk leksikon bahasa Indonesia yang digunakan dalam penelitian ini adalah leksikon milik masdevid yang dapat di download pada link <https://github.com/masdevid/ID-OpinionWords> , serta berberapa tambahan dari penelitian terdahulu. Rumus untuk klasifikasi lexicon base adalah:

Sentencesentiment  (3.1)

Apabila rumus diatas menghasilkan nilai sentimen lebih besar dari 0, maka tweet akan digolongkan sebagai tweet positif. Sebaliknya, apabila nilai sentimen lebih kecil dari 0, maka tweet akan digolongkan sebagai tweet negatif. Daftar kalimat positif pada lexicon dapat dilihat pada tabel 3.8 dan daftar kalimat negatif dapat dilihat pada tabel 3.9.

**Tabel 3.8** Daftar *Lexicon* Positif

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Kata Positif** | **No** | **Kata Positif** | **No** | **Kata Positif** |
| 1 | Penuh | 45 | Lagi | 89 | Selamat |
| 2 | Syarat | 46 | Tenang | 90 | Undang |
| 3 | Riah | 47 | Tentu | 91 | Unggul |
| 4 | Favorit | 48 | Terang | 92 | Untung |
| 5 | Pihak | 49 | Terima | 93 | Nikmat |
| 6 | Mikat | 50 | Terobos | 94 | Rasa |
| 7 | Hati | 51 | Kaget | 95 | Cinta |
| 8 | Pimpin | 52 | Kagum | 96 | Antusias |
| 9 | Baik | 53 | Agung | 97 | Tinggi |
| 10 | Akhlak | 54 | Akal | 98 | Tingkat |
| 11 | Cantik | 55 | Anggap | 99 | Titip |
| 12 | Percaya | 56 | Angkat | 100 | Jamin |
| 13 | Jelas | 57 | Asyik | 101 | Janji |
| 14 | Kaya | 58 | Anjur | 102 | Tonjol |
| 15 | Kuat | 59 | Atas | 103 | Mentereng |
| 16 | Pesona | 60 | Kecoh | 104 | Tunjuk |
| 17 | Muas | 61 | Kejut | 105 | Gejala |
| 18 | Mudah | 62 | Kesah | 106 | Turut |
| 19 | Puja | 63 | Kesan | 107 | Hak |
| 20 | Puji | 64 | Gairah | 108 | Sanjung |
| 21 | Lebih | 65 | Gelenyar | 109 | Sambut |
| 22 | Pukau | 66 | Geli | 110 | Selesai |
| 23 | Mulia | 67 | Gelitik | 111 | Sederhana |
| 24 | Pulih | 68 | Gembira | 112 | Segar |
| 25 | Murni | 69 | Gempar | 113 | Selaras |
| 26 | Takjub | 70 | Getar | 114 | Seringai |
| 27 | Tambah | 71 | Giur | 115 | Rangsang |
| 28 | Menang | 72 | Goda | 116 | Renung |
| 29 | Tantang | 73 | Gugah | 117 | Susul |
| 30 | Tarik | 74 | Hadiah | 118 | Subsidi |
| 31 | Tawan | 75 | Halus | 119 | Rela |
| 32 | Cengang | 76 | Harga | 120 | Mewah |
| 33 | Cukup | 77 | Haru | 121 | Yakin |
| 34 | Dalam | 78 | Heran | 122 | Minim |
| 35 | Damai | 79 | Hibur | 123 | Modern |
| 36 | Dapat | 80 | Hidup | 124 | Modis |
| 37 | Debar | 81 | Hormat | 125 | Montok |
| 38 | Dewa | 82 | Ideal | 126 | Monumental |
| 39 | Dominasi | 83 | Idola | 127 | Moralitas |
| 40 | Dorong | 84 | Ilham | 128 | Muda |
| 41 | Dukung | 85 | Ingin | 129 | Mudah |
| 42 | Tebus | 86 | Isyarat | 130 | Mujarab |
| 43 | Tegak | 87 | Kuasa | 131 | Mukjizat |
| 44 | Tegas | 88 | Ucap | 132 | .... |

**Tabel 3.9** Daftar Lexicon Negatif

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Kata Negatif** | **No** | **Kata Negatif** | **No** | **Kata Negatif** |
| 1 | Murah | 45 | Ejan | 89 | Gugur |
| 2 | Musnah | 46 | Ejek | 90 | Guling |
| 3 | Memutarbalikkan | 47 | Mengejutkan | 91 | Gusar |
| 4 | Putus | 48 | Kejut | 92 | Hadap |
| 5 | Tabrak | 49 | Keluar | 93 | Halang |
| 6 | Tahan | 50 | Keluh | 94 | Halau |
| 7 | Takut | 51 | Kelupas | 95 | Hambat |
| 8 | Tampar | 52 | Embara | 96 | Hancur |
| 9 | Tanam | 53 | Eksploitasi | 97 | Hantu |
| 10 | Tarik | 54 | Emis | 98 | Hapus |
| 11 | Cabut | 55 | Endap | 99 | Hasut |
| 12 | Caci | 56 | Kepung | 100 | Heboh |
| 13 | Cekik | 57 | Erang | 101 | Heran |
| 14 | Cela | 58 | Keras | 102 | Hilang |
| 15 | Cemas | 59 | Kerdil | 103 | Hina |
| 16 | Cibir | 60 | Kerut | 104 | Hindar |
| 17 | Cicit | 61 | Kesal | 105 | Hujat |
| 18 | Ciut | 62 | Gagal | 106 | Igau |
| 19 | Colok | 63 | Ganggu | 107 | Kikis |
| 20 | Copot | 64 | Ganti | 108 | Indoktrinasi |
| 21 | Curi | 65 | Gantung | 109 | Ingkar |
| 22 | Curiga | 66 | Garuk | 110 | Injak |
| 23 | Cubit | 67 | Gelepar | 111 | Intai |
| 24 | Dakwa | 68 | Gasak | 112 | Intimidasi |
| 25 | Deklamasi | 69 | Gelap | 113 | Intip |
| 26 | Derita | 70 | Gelegar | 114 | Isap |
| 27 | Desak | 71 | Menggeliat | 115 | Khawatir |
| 28 | Desis | 72 | Geli | 116 | Khianat |
| 29 | Didih | 73 | Gelisah | 117 | Kritik |
| 30 | Distorsi | 74 | Gempar | 118 | Oceh |
| 31 | Tegur | 75 | Gentar | 119 | Oles |
| 32 | Tekan | 76 | Gerak | 120 | Olok |
| 33 | Tentang | 77 | Geram | 121 | Omel |
| 34 | Tetes | 78 | Gerenyet | 122 | Korban |
| 35 | Kalah | 79 | Gerogot | 123 | Kotor |
| 36 | Kacau | 80 | Gertak | 124 | Oyak |
| 37 | Amputasi | 81 | Gerutu | 125 | Kurang |
| 38 | Ammuk | 82 | Gigil | 126 | Kuasa |
| 39 | Ancam | 83 | Gigit | 127 | Kutuk |
| 40 | Aniaya | 84 | Gila | 128 | Tiada |
| 41 | Kebiri | 85 | Giling | 129 | Tidur |
| 42 | Kecam | 86 | Goda | 130 | Timbul |
| 43 | Kecewa | 87 | Goyang | 131 | Benci |
| 44 | Kecil | 88 | Gugat | 132 | .... |

Apabila rumus diatas menghasilkan nilai sentimen lebih besar dari 0, maka *tweet* akan digolongkan sebagai *tweet* positif. Sebaliknya, apabila nilai sentimen lebih kecil dari 0, maka *tweet* akan digolongkan sebagai *tweet* negatif. Contoh kalimat dapat dilihat pada tabel 3.10.

**Tabel 3.10** Lexicon Based Features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tweet** | **Pencocokan Dengan Leksikon** | | | | **Score** |
| kapok malu kpuid diberitakan media mainstream | kapok | Malu | Kpuid | Diberitakan | S = 0 + (-2) = -2  S = Negatif |
| Media | mainstream |  |  |
| bubar kpuid isi sampah dapur | Bubar | Kpuid | Isi | Sampah | S = 0 + (-2) = -2  S = Negatif |
| Dapur |  |  |  |
| percaya kpuid usah ngatur segala dukung | **percaya** | kpuid | usah | ngatur | S = 0 + 2 = 2  S = Positif |
| Segala | **dukung** |  |  |

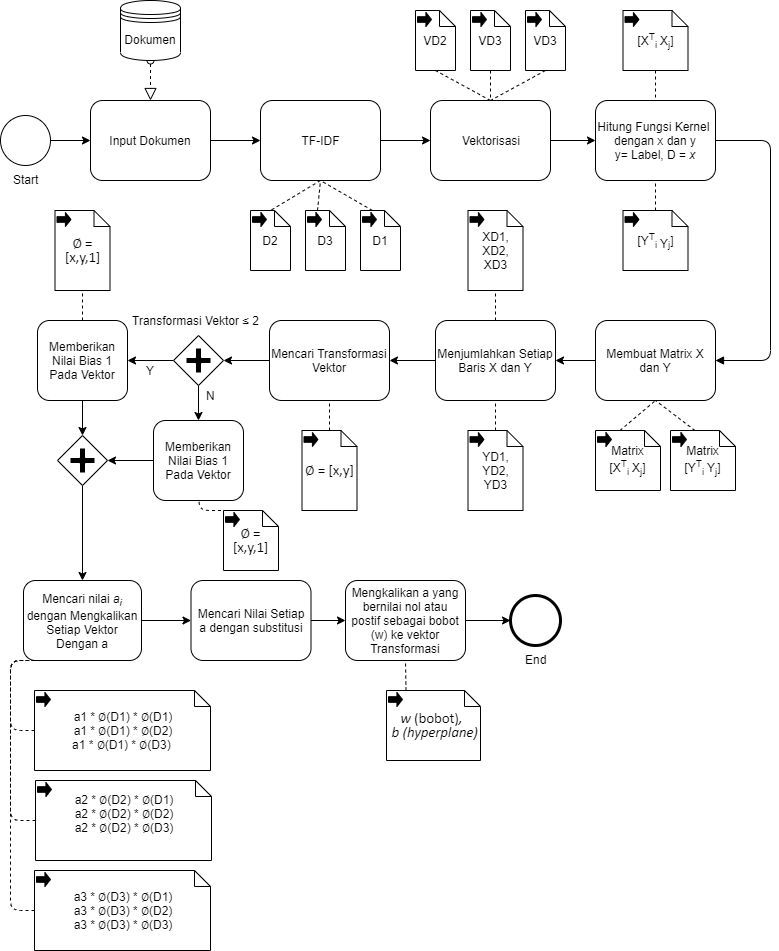
Ket :

= Kata Negatif

= Kata Positif

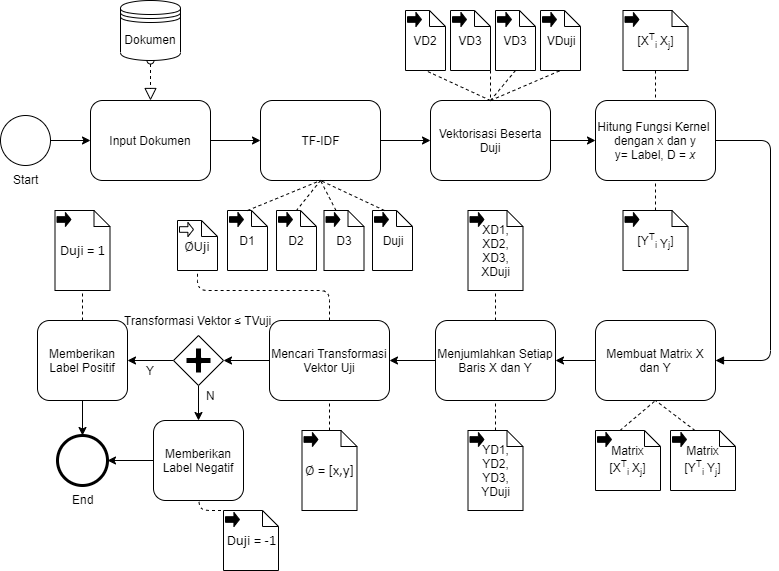
S = Sentimen

* 1. **Diagram *BPNM Proses Support Vector Machine***

****

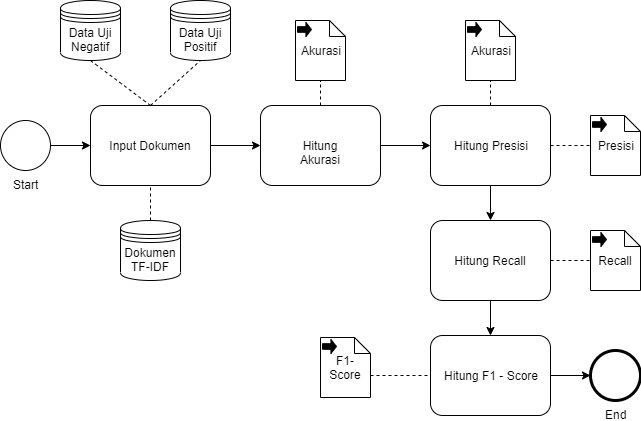
**Gambar 3.11** Diagram *BPNM Support Vector Machine*

Pada Gambar 3.11 diatas, dijelaskan bahwa dilakukan serangkaian proses perhitungan yang akan menghasilkan nilai hyperplane dan bobot yang digunakan sebagai patokan uji untuk menghitung data uji. Alur untuk pengujian dapat dilihat pada gambar 3.12.

****

**Gambar 3.12** Diagram Uji *BPNM Support Vector Machine*

Data hasil pengujian kemudian dihitung menggunakan *confusion* *matrix* untuk mendapatkan hasil pengujian berupa akurasi, presisi, recall dan f1-score. Diagram BPNM untuk proses *confusion* *matrix* dapat dilihat pada gambar 3.13.



**Gambar 3.13** Diagram *BPNM Confusion Matrix*

* 1. **Pembobotan TF-IDF**

*Term Frequency – Inverse Document Frequency* (TF-IDF) adalah metode yang digunakan untuk menghitung bobotan setiap kata yang telah diekstrak. TF-IDF berfungsi untuk menghitung seluruh jumlah kata dan total kemunculannya pada sebuah dokumen serta menghasilkan sebuah kamus data yang kemudian akan digunakan untuk proses training pada SVM.

Dalam pencarian informasi, TF-IDF (*Term Frequency - Inverse Document Frequency*) adalah statistik numerik yang dimaksudkan untuk mencerminkan betapa pentingnya sebuah kata dalam koleksi atau *corpus* [4]. Rumus yang digunakan adalah :

(3.2)

TFIDF = TF x IDF = TF x loge

Di mana,

TF = Term Frequency, banyaknya sebuah kata/istilah muncul

DF = Document Frecuency, jumlah dokumen di mana kata/istilah tersebut muncul. Minimal 1 dokumen.

Pada penelitian ini menggunakan metode TF-IDF (*Term Frequency – Inverse Document Frequency*) untuk melakukan pembobotan kata. Akan digunakan 3 kalimat sebagai contoh perhitungan TF-IDF seperti pada tabel 3.11.

**Tabel 3.11** Kalimat Testing

|  |  |
| --- | --- |
| **Dokumen** | **Terms** |
| D1 | Kpu berkerja sangat baik |
| D2 | Dukung kawal perhitungan kpu semangat |
| D3 | Situng kpu dinilai tambah kacau brutal |

Tabel 3.10 berisi data hasil pre-processing yang akan dilatih. Data akan dihitung bobot katanya pada masing – masing dokumen. Berikut adalah penjelasan mengenai perhitungan TF-IDF yang akan dilakukan:

1. D adalah jumlah dokumen, karena yang akan dilakukan pembobotan adalah sebuah kalimat maka digunakan Sentence sebagai Dokumen. Terdapat 3 Dokumen di dalam koleksi (D1, D2, D3).
2. Dilakukan perhitungan jumlah kata yang muncul di dalam suatu dokumen (*Term Frequencies*), perhitungan TF (*Term Frequencies*) dapat dilihat pada tabel 3.11.
3. Melakukan perhitungan DF (*Document Frequencies*), yaitu menghitung logaritma antara jumlah koleksi dokumen dengan jumlah kata yang muncul di setiap dokumen. Perhitungan dapat dilihat pada tabel 3.12

**Tabel 3.12** Proses TF dan DF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Term** | **TF** | | | **DF** |
| **D1** | **D2** | **D3** |
| Kpu | 1 | 1 | 1 | Log(3/3)= 0.159 |
| Berkerja | 1 | 0 | 0 | Log(3/1)= 0.477 |
| Sangat | 1 | 0 | 0 | 0.477 |
| Baik | 1 | 0 | 0 | 0.477 |
| Dukung | 0 | 1 | 0 | 0.477 |
| Kawal | 0 | 1 | 0 | 0.477 |
| Perhitungan | 0 | 1 | 0 | 0.477 |
| Semangat | 0 | 1 | 0 | 0.477 |
| Situng | 0 | 0 | 1 | 0.477 |
| Dinilai | 0 | 0 | 1 | 0.477 |
| Tambah | 0 | 0 | 1 | 0.477 |
| Kacau | 0 | 0 | 1 | 0.477 |
| Brutal | 0 | 0 | 1 | 0.477 |

1. Setelah nilai TF dan IDF ditemukan maka dilakukan pencarian bobot tiap kata dengan mengkalikan TF dengan IDF seperti pada tabel 3.13

**Tabel 3.13** Perhitungan TF-IDF

|  |  |  |  |
| --- | --- | --- | --- |
| **Term** | **W = TF x IDF** | | |
| **D1** | **D2** | **D3** |
| Kpu | 0.159 | 0.159 | 0.159 |
| Berkerja | 0.477 | 0 | 0 |
| Sangat | 0.477 | 0 | 0 |
| Baik | 0.477 | 0 | 0 |
| Dukung | 0 | 0.477 | 0 |
| Kawal | 0 | 0.477 | 0 |
| Perhitungan | 0 | 0.477 | 0 |
| Semangat | 0 | 0.477 | 0 |
| Situng | 0 | 0 | 0.477 |
| Dinilai | 0 | 0 | 0.477 |
| Tambah | 0 | 0 | 0.477 |
| Kacau | 0 | 0 | 0.477 |
| Brutal | 0 | 0 | 0.477 |

* 1. **Tahap Pelatihan dengan *Support Vector Machine***

Metode yang dilakukan pada penelitian ini adalah dengan menggunakan *Support Vector Machine* (SVM). Alasan digunakannya metode klasifikasi *Support Vector Machine* pada penelitian ini karena metode SVM berusaha menemukan satu hyperplane dengan memaksimalkan jarak antar kelas (margin). Dengan memaksimalkan margin, SVM akan memiliki kemampuaan generalisasi yang tinggi terhadap data – data yang akan datang. Secara umum, SVM digunakan untuk klasifikasi dua kelas walaupun dapat digunakan untuk menyelesaikan masalah multi kelas.

Pada tahap pelatihan ini bertujuan untuk menemukan nilai a, nilai w dan nilai konstanta b sebagai parameter untuk menemukan hyperplane terbaik. Penelitian ini mengklasifikasikan data kedalam 2 kelas yaitu, positif dan negatif. Sebelum dilakukan perhitungan SVM, data latih dari tahap pembobotan TF-IDF diubah menjadi bentuk *vector*, kalimat yang masuk kedalam kelas positif dinyatakan dalam angka 1 dan kelas negatif dinyatakan dalam angka -1. Data adalah seperti tabel 3.14:

**Tabel 3.14** Pembentukan Vektor setelah TF-IDF

|  |  |  |
| --- | --- | --- |
| **Dokumen** | **Bentuk Vector Setelah TF-IDF** | **Y** |
| D1 | [0.159,0.477,0.477,0.477,0,0,0,0,0,0,0,0,0] | 1 |
| D2 | [0.159,0,0,0,0.477,0.477,0.477,0.477,0,0,0,0,0] | 1 |
| D3 | [0.159,0,0,0,0,0,0,0,0.477,0.477,0.477,0.477,0.477] | -1 |

Tahap selanjutnya adalah menghitung fungsi kernel. *Kernel* yang digunakan pada proses ini adalah kernel linear. Fungsi *kernel* *linear* adalah sebagai berikut:

(3.3)

*K(xi , xj ) = xj*

*Kernel* ini adalah *kernel* sederhana yang digunakan untuk mengklasifikasikan data yang terbagi kedalam 2 kelas secara linear yang dipisahkan dengan *hyperplane.*

Setiap dokumen diinisialisasikan sebagai *x* dengan nilai terms dan vector yang terdapat didalamnya, maka D1 = *x1, D2=x2,* dan D3=x3 seperti pada tabel 3.15.

**Tabel 3.15** Nilai *x1,x2,x3*

|  |  |
| --- | --- |
| **Dokumen** | **Bentuk Vector Setelah TF-IDF** |
| X1 | [0.159,0.477,0.477,0.477,0,0,0,0,0,0,0,0,0] |
| X2 | [0.159,0,0,0,0.477,0.477,0.477,0.477,0,0,0,0,0] |
| X3 | [0.159,0,0,0,0,0,0,0,0.477,0.477,0.477,0.477,0.477] |

*Kernel* akan melakukan iterasi dan mengkalikan setiap x dengan x lainnya, contoh perhitungan pertama dapat dilihat pada tabel 3.16.

**Tabel 3.16** Contoh Perhitungan  **xj**

|  |  |  |
| --- | --- | --- |
|  | **x1** | **xj** |
| [0.159, 0.477, 0.477, 0.477 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0] | [0.159, 0.477, 0.477, 0.477 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0] | 0.707868 |

Setelah proses iterasi selesai, hasil dari iterasi perkalian setiap x adalah seperti pada tabel 3.17

**Tabel 3.17** Hasil Iterasi Perkalian *x*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **x1** | **x2** | **x3** | **x1** | **x2** | **x3** | **x1** | **x2** | **x3** |
| 0.707 | 0.025 | 0.025 | 0.025 | 0.935 | 0.025 | 0.025 | 0.025 | 1.162 |

hasil dari perkalian seluruh nilai x, maka hasil perkalian dibuatkan matrix  **x1.**

**xj =**

**xj=**

Tahap selanjutnya adalah melakukan perhitungan terhadap nilai y dengan cara yang sama. Nilai y adalah kelas atau label, untuk kelas positif diberikan nilai 1 dan kelas negatif diberikan nilai -1 seperti terlihat pada tabel 3.18.

**Tabel 3.18** Nilai y1,y2,y3

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | 1 | -1 |

Nilai y dikalikan dengan seluruh nilai y yang lain. Contoh perhitungan adalah seperti terlihat pada tabel 3.19.

**Tabel 3.19** Contoh Perhitungan  **yj**

|  |  |  |
| --- | --- | --- |
|  | **y1** | **y1** |
| 1 | 1 | 1 |

Setelah proses iterasi selesai, hasil dari iterasi perkalian setiap y adalah seperti pada tabel 3.20.

**Tabel 3.20** Hasil Iterasi Perkalian *y*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **y1** | **y2** | **y3** | **y1** | **y2** | **y3** | **y1** | **y2** | **y3** |
| 1 | 1 | -1 | 1 | 1 | -1 | -1 | -1 | 1 |

Setelah perhitungan selesai, hasil dari perkalian seluruh nilai y, maka hasil perkalian dibuatkan matrix  **yj.**

**yj =**

**yj =**

Tahap selanjutnya adalah mencari nilai x dan y untuk setiap dokumen dengan menjumlahkan baris matrix pada  **xj** dan  **yj** dengan persamaan (3.4) sebagai berikut:

xi , (*i, j =* 1 ,..,*n*) (3.4)

Perhitungan untuk x adalah sebagai berikut :

XD1 =  **x1 +  x2** +  **x3** = 0.707 + 0.025 + 0.025= 0.758

XD2 =  **x1 +  x2** +  **x3** = 0.025 + + 0.025 = 0.985

XD3 =  **x1 +  x2** +  **x3** = 0.025 + 0.025 + 1.162 = 1.21

Hasil dari perhitungan x disajikan dalam tabel 3.21 :

Tabel 3.21 Hasil Perhitungan nilai *x*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| X | 0.758 | 0.985 | 1.21 |

Selanjutnya dilakukan perhitungan untuk y. Perhitungan menggunakan rumus (3.5) sebagai berikut :

yi , (*i, j =* 1 ,..,*n*) (3.5)

Perhitungan *y* adalah sebagai berikut:

YD1 =  **y1 + y2** +  **y3** = 1 + 1 + (-1) = 1

YD2 =  **y1 + y2** +  **y3** = 1 + 1 + (-1) = 1

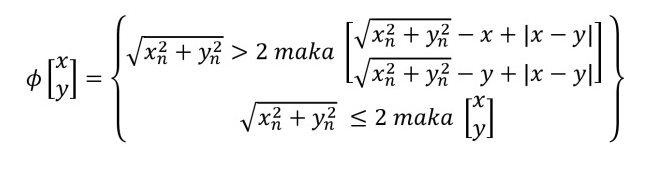
YD3 =  **y1 + y2** +  **y3** = (-1) + (-1) + 1 = -1

Hasil dari perhitungan y disajikan dalam tabel 3.22 :

Tabel 3.22 Hasil Perhitungan *y*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Y | 1 | 1 | -1 |

Setelah nilai x dan y diketahui, langkah selanjutnya adalah mengetahui nilai *vector* transformasi dengan memasukan nilai x dan y ke persamaan (3.6) berikut:



(3.6)

Untuk menghitung *vector* transformasi sebagai contoh apabila n = 1, maka Xn = XD1, dan Yn = YD1 . nilai – nilai ini kemudian dimasukan kedalam fungsi = = 1.254 . karena nilai < 2 maka nilai transformasi adalah = . dengan melakukan perhitungan dengan fungsi (3.6), didapatkan nilai transformasi untuk setiap dokumen sebagai berikut:

= , = , =

Langkah selanjutnya, masing – masing nilai transformasi *vector* diberi nilai bias sebesar 1 untuk membantu mendapatkan nilai *hyperplane*. Setiap *vector* transformasi yang diberi nilai bias adalah sebagai berikut :

= , = , =

Setelah mendapatkan nilai vektor transformasi diberikan bias, langkah selanjutnya adalah mencari nilai *a*i dengan mengkalikan setiap dokumen dengan nilai a. Persamaan yang digunakan adalah persamaan (3.7) sebagai berikut :

(3.7)

, *a*i Dj

Yang artinya akan dilakukan iterasi perkalian untuk setiap *a*i dengan setiap D. Proses dan hasil perhitungan persamaan diatas adalah sebagai berikut:

Perhitungan pada D1 :

a1  \* = 2.575

a1  \* = 2.747

a1  \* = 0.920

Perhitungan pada D2 :

a2  \* = 2.747

a2  \* = 2.972

a2  \* = 1.196

Perhitungan pada D3 :

a3 \* = 0.920

a3 \* = 1.196

a3 \*= 3.472

Perhitungan dilakukan untuk mendapatkan nilai parameter *a*i. Setelah perhitungan selesai dilakukan akan didapatkan persamaan sebagai berikut:

2.575 *a*1 + 2.747 *a*2 + 0.920 *a*3 = 1

2.747 *a*1 + 2.972 *a*2 + 1.196 *a*3= 1

0.920 *a*1 + 1.196 *a*2 + 3.472 *a*3= -1

Hasil dari persamaan diatas disubstitusikan sehingga mendapatkan nilai untuk setiap *a*. Hasil dari substitusi adalah sebagai berikut:

*a*1= -0.460, *a*2= 0.961, *a*3= -0.490,

nilai *a*i yang digunakan ke tahap selanjutnya adalah yang bernilai nol atau positif, karena nilai *a*1 dan *a*3 bernilai negatif maka, nilai yang digunakan hanya nilai *a*2 dan dappat ditentukan bila *a*2 adalah *support* *vector*

Selanjutnya nilai *support* *vector* dimasukan kedalam rumus untuk mendapatkan nilai w (bobot) dan *b* *hyperplane*. Persamaan yang digunakan adalah:

(3.8)

Di

*w* = 0.961 =

Maka, didapatkan bahwa nilai *w* = dan *b* = 0.961. nilai *b* adalah *hyperplane* untuk mengklasifikasikan nilai positif dan negatif.

* 1. **Pengujian Data**

Pengujian data dilakukan setelah nilai *w* dan *b* ditemukan, pengujian dilakukan guna mengetahui apakah data yang diuji masuk ke kelas positif atau masuk ke kelas negatif berdasarkan model yang telah dibuat. Sebagai contoh proses pengujian, akan digunakan sebuah data uji yang akan diklasifikasikan melalui model yang sudah dibuat. Data uji adalah sebagai berikut :

**Tabel 3.23** Data Uji

|  |  |
| --- | --- |
| **Dokumen** | **Terms** |
| Duji | Perhitungan kpu kawal sangat baik |

Data uji kemudian dibuatkan vektor dengan menggunakan tabel bobot TF-IDF yang dibuat sebelumnya. Hal ini bertujuan untuk mendapatkan nilai vektor milik Data uji. Pembuatan vektor dapat dilihat pada tabel 3.24.

**Tabel 3.24** Bobot Data Uji

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Term** | **W = TF x IDF** | | | |
| **D1** | **D2** | **D3** | **Duji** |
| Kpu | 0.159 | 0.159 | 0.159 | 0.159 |
| Berkerja | 0.477 | 0 | 0 | 0 |
| Sangat | 0.477 | 0 | 0 | 0.477 |
| Baik | 0.477 | 0 | 0 | 0.477 |
| Dukung | 0 | 0.477 | 0 | 0 |
| Kawal | 0 | 0.477 | 0 | 0.477 |
| Perhitungan | 0 | 0.477 | 0 | 0.477 |
| Semangat | 0 | 0.477 | 0 | 0 |
| Situng | 0 | 0 | 0.477 | 0 |
| Dinilai | 0 | 0 | 0.477 | 0 |
| Tambah | 0 | 0 | 0.477 | 0 |
| Kacau | 0 | 0 | 0.477 | 0 |
| Brutal | 0 | 0 | 0.477 | 0 |

Setiap dokumen termasuk data uji kemudian diubah menjadi bentuk vektor untuk selanjutnya dihitung menggunakan fungsi kernel linear. Nilai y untuk data uji diberikan nilai 0 karena belum masuk kedalam kelas apapun. Perubahan data dapat dilihat pada tabel 3.25.

**Tabel 3.25** Vektorisasi Duji

|  |  |  |
| --- | --- | --- |
| **Dokumen** | **Bentuk Vector Setelah TF-IDF** | **Y** |
| D1 | [0.159,0.477,0.477,0.477,0,0,0,0,0,0,0,0,0] | 1 |
| D2 | [0.159,0,0,0,0.477,0.477,0.477,0.477,0,0,0,0,0] | 1 |
| D3 | [0.159,0,0,0,0,0,0,0,0.477,0.477,0.477,0.477,0.477] | -1 |
| Duji | [0.159,0,0.477,0.477,0,0.477,0.477,0,0,0,0,0,0] | 0 |

Setiap dokumen kemudian diinisialisasikan sebagai *x* dengan nilai terms dan vector yang terdapat didalamnya, maka D1 = *x1,* D2=*x2,* D3=*x3* dan Duji = *x4* seperti pada tabel 3.26.

**Tabel 3.26** Nilai Vektor Pengujian

|  |  |
| --- | --- |
| **Dokumen** | **Bentuk Vector Setelah TF-IDF** |
| *x*1 | [0.159,0.477,0.477,0.477,0,0,0,0,0,0,0,0,0] |
| *x*2 | [0.159,0,0,0,0.477,0.477,0.477,0.477,0,0,0,0,0] |
| *x*3 | [0.159,0,0,0,0,0,0,0,0.477,0.477,0.477,0.477,0.477] |
| *x4* | [0.159,0,0.477,0.477,0,0.477,0.477,0,0,0,0,0,0] |

Mirip dengan tahap sebelumnya pada tahap pelatihan, nilai *x* kemudian dikalikan dengan nilai *x* yang lain dan melakukan iterasi hingga selesai. contoh perhitungan dapat dilihat pada tabel 3.27

**Tabel 3.27** Contoh Perhitungan x uji

|  |  |  |
| --- | --- | --- |
|  | **X4** | **xj** |
| [0.159, 0.477, 0.477, 0.477 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0] | [0.159,0,0.477,0.477,0,0.477,0.477,0,0,0,0,0,0] | 1.921 |

Setelah proses iterasi selesai, hasil dari iterasi perkalian setiap x adalah seperti pada tabel 3.28.

**Tabel 3.28** Hasil Perhitungan Iterasi x Uji

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **x1** | **x2** | **x3** | **x4** | **x1** | **x2** | **x3** | **x4** |
| 0.707 | 0.025 | 0.025 | 0.480 | 0.025 | 0.935 | 0.025 | 0.480 |
| **x1** | **x2** | **x3** | **x4** | **x1** | **x2** | **x3** | **x4** |
| 0.025 | 0.025 | 1.162 | 0.025 | 0.480 | 0.480 | 0.025 | 0.935 |

hasil dari perkalian seluruh nilai x, maka hasil perkalian dibuatkan matrix  **x1.**

**xj =**

**xj=**

Tahap selanjutnya adalah melakukan perhitungan terhadap nilai y dengan cara yang sama. Nilai y dapat dilihat pada tabel 3.29.

**Tabel 3.29** Nilai *y1,y2,y3,y4*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | 1 | -1 | 0 |

Nilai y dikalikan dengan seluruh nilai y yang lain. Contoh perhitungan adalah seperti terlihat pada tabel 3.30.

**Tabel 3.30** Contoh Perhitungan *y* Uji

|  |  |  |
| --- | --- | --- |
|  | **y1** | **y1** |
| 1 | 1 | 1 |

Setelah proses iterasi selesai, hasil dari iterasi perkalian setiap y adalah seperti pada tabel 3.31

**Tabel 3.31** Hasil Perhitungan Iterasi *y* Uji

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **y1** | **y2** | **y3** | **y4** | **y1** | **y2** | **y3** | **y4** |
| 1 | 1 | -1 | 0 | 1 | 1 | -1 | 0 |
| **y1** | **y2** | **y3** | **y4** | **y1** | **y2** | **y3** | **y4** |
| -1 | -1 | 1 | 0 | 0 | 0 | 0 | 0 |

Setelah perhitungan selesai, hasil dari perkalian seluruh nilai y, maka hasil perkalian dibuatkan matrix  **yj.**

**yj =**

**yj =**

Tahap selanjutnya adalah mencari nilai x dan y untuk setiap dokumen dengan menjumlahkan baris *matrix* pada  **xj** dan  **yj.**

Perhitungan untuk x adalah sebagai berikut :

XD1 =  **x1 +  x2** +  **x3 + x4** = 0.707 + 0.025 + 0.025 + 0.480= 1.238

XD2 =  **x1 +  x2** +  **x3 +** +  **x4** = 0.025 + + 0.025 + 0.480= 1.466

XD3 =  **x1 +  x2** +  **x3** +  **x4** = 0.025 + 0.025 + 1.162 + 0.025 = 1.238

XD4 =  **x1 + x2** +  **x3 +  x4** = 0.480 + 0.480 + 0.025 + 0.935 = 1.921

Hasil dari perhitungan x disajikan dalam tabel 3.32 :

**Tabel 3.32** Hasil Perhitungan Nilai *x* dan *x*Uji

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | ***D4*** |
| X | 1.238 | 1.466 | 1.238 | 1.921 |

Selanjutnya dilakukan perhitungan untuk y. Perhitungan adalah sebagai berikut :

YD1 =  **y1 + y2** +  **y3 + y4** = 1 + 1 + (-1) + 0 = 1

YD2 =  **y1 + y2** +  **y3** +  **y4** = 1 + 1 + (-1) + 0 = 1

YD3 =  **y1 + y2** +  **y3 + y4**= (-1) + (-1) + 0 = -1

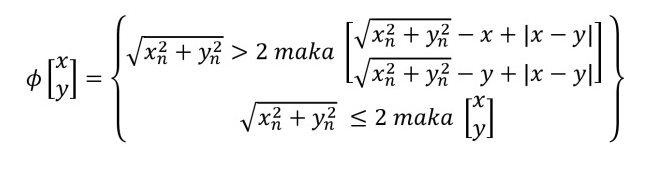
YD4 =  **y1 +**  **y2** +  **y3 +**  **y4** = 0 + 0 + 0 + 0 = 0

Hasil dari perhitungan y disajikan dalam tabel 3.33 :

**Tabel 3.33** Hasil Perhitungan Nilai y dan yUji

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | ***D4*** |
| Y | 1 | 1 | -1 | 0 |

Setelah nilai x dan y diketahui, langkah selanjutnya adalah mengetahui nilai *vector* transformasi dengan memasukan nilai x dan y ke persamaan (3.5) sebagai berikut:



Untuk menghitung data uji maka n = 4, maka Xn = XDuji, dan Yn = YDuji . nilai dihitung dengan fungsi = = 1.921 . karena nilai < 2 maka nilai transformasi adalah = . dengan perhitungan ini maka,

Kemudian untuk mengetahui kelas dari Duji, dilakukan perkalian dengan matriks bobot seperti berikut:

*wT* =[0.947, 0.961] \* =1.819

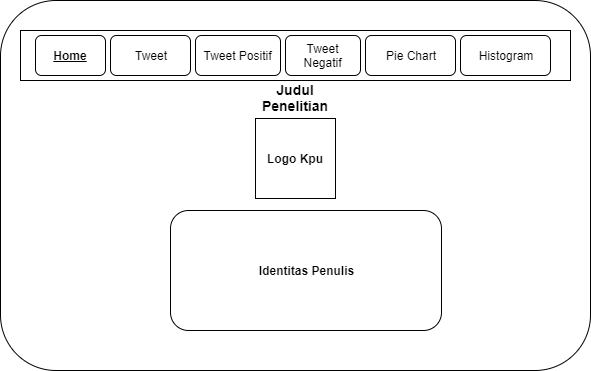
Hasil perhitungan Duji menghasilkan nilai 1.819 yang lebih besar dari nilai *b*(*hyperplane*). Karena nilai Duji lebih besar, maka Duji masuk ke kelas Positif.

* 1. **Perancangan Tampilan Website**

Perancangan tampilan website merupakan tahapan yang harus dikerjakan sebelum membuat suatu website. Rancangan tampilan ini menggambarkan keterkaitan antara setiap halaman dan juga menjelaskan arah komunikasinya. Rancangan tampilan aplikasi bertujuan agar website yang dihasilkan terlihat lebih menarik, mudah dimengerti dan dioperasikan. Rancangan website ini terdiri dari halaman utama (*home*), *Tweet*, *Tweet* positif, *Tweet* negatif, *Pie Chart* dan *Histogram.* Adapun tahapan pembuatan rancangan aplikasi adalah sebagai berikut:

1. Rancangan Tampilan Halaman *Home*

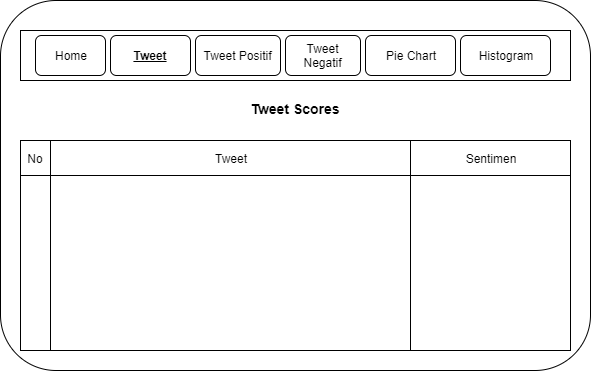
Halaman utama merupakan tampilan utama dari *website*. Halaman ini menampilkan tombol navigasi, logo kpu, judul penelitian dan identitas penulis seperti pada gambar 3.14.



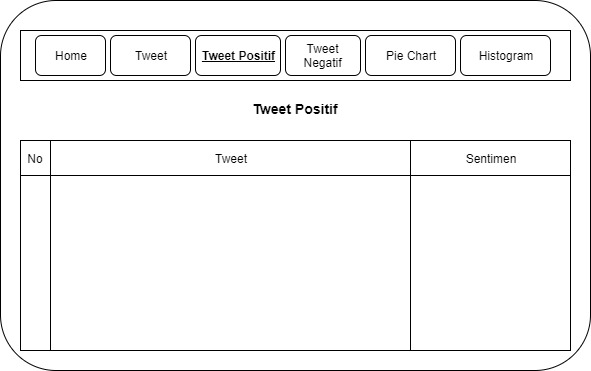
**Gambar 3.14** Rancangan Home Kpu

1. Rancangan Halaman *Tweet , Tweet* Positif, dan *Tweet* Negatif

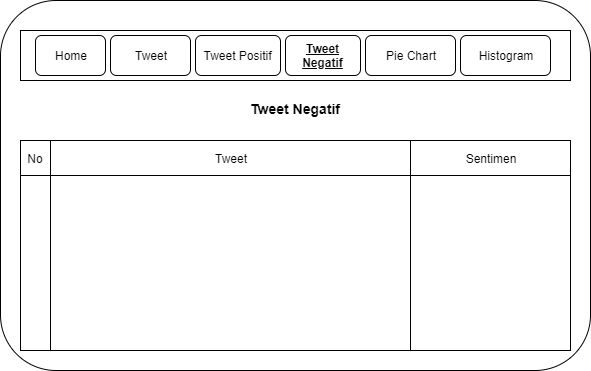
Halaman *Tweet* menampilkan seluruh *tweet* yang telah dicleansing yang berlabel positif maupun negatif, Halaman *Tweet* positif menampilkan seluruh *tweet* berlabel postifi, sedangkan halaman tweet negatif menampilkan seluruh data *tweet* dengan label negatif yang digunakan di dalam penelitian ini. Rancangan halaman *tweet* dapat dilihat pada gambar 3.15, rancangan halaman *Tweet* positif dapat diliha pada gambar 3.16, dan Rancangan halaman *Tweet* negatif dapat dilihat pada gambar 3.17.



**Gambar 3.15** Rancangan Halaman Tweet



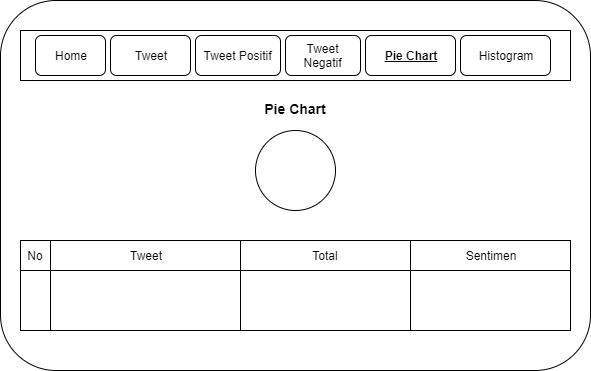
**Gambar 3.16** Rancangan Halaman Tweet Positif



**Gambar 3.17** Rancangan Halaman Tweet Negatif

1. Rancangan Halaman *Pie Chart*

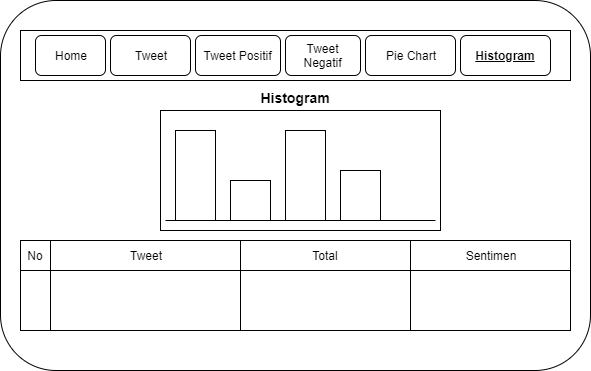
Halaman *Pie Chart* menampilkan diagram dalam bentuk pie, dimana diagram menunjukan prosentase tweet negatif dan tweet positif di dalam penelitian. Tweet positif akan akan diberikan warna biru dan tweet negatif akan diberikan warna merah. Rancangan halaman *Pie Chart* dapat dilihat pada gambar 3.18.



**Gambar 3.18** Rancangan Halaman *Pie Chart*

1. Rancangan Halaman *Histogram*

Halaman *Histogram* menampilkan diagram dalam bentuk *Histogram*, yang berisi perhitungan Akurasi, Presisi, Recall dan F1-Score. Rancangan halaman histogram dapat dilihat pada gambar 3.19.



**Gambar 3.19** Rancangan Halaman *Histogram*