ANALISIS SENTIMEN PADA ULASAN OBJEK WISATA PANTAI DI GOOGLE MAPS MENGGUNAKAN METODE LONG-SHORT TERM MEMORY

(Studi Kasus: Objek Wisata Pantai di Daerah Istimewa Yogyakarta)

TUGAS AKHIR

Tugas Akhir ini sebagai salah satu syarat untuk memperoleh gelar sarjana Informatika Universitas Pembangunan Nasional "Veteran" Yogyakarta

Disusun oleh:

Niken Oktaviana
123160053

PROGRAM STUDI INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK INDUSTRI

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

YOGYAKARTA

2020

HALAMAN PENGESAHAN PEMBIMBING

ANALISIS SENTIMEN PADA ULASAN OBJEK WISATA PANTAI DI GOOGLE MAPS MENGGUNAKAN METODE LONG-SHORT TERM MEMORY

Disusun oleh :

Niken Oktaviana

123160053

Telah diuji dan dinyatakan lulus oleh pembimbing

pada tanggal : 06 Oktober 2020

Menyetujui,

Pembimbing I	Pembimbing II
Herry Sofyan, S.T.,M.Kom.	Heru Cahya Rustamaji, S.Si., M.T.
NIK. 2 6404 96 0139 1	NIK. 2 7106 96 0065 1

Mengetahui,

Ketua Program Studi

Dr. Heriyanto, A.Md., S.Kom., M.Cs.
NIK. 2 7706 11 0301 1

HALAMAN PENGESAHAN PENGUJI

ANALISIS SENTIMEN PADA ULASAN OBJEK WISATA PANTAI DI GOOGLE MAPS MENGGUNAKAN METODE LONG-SHORT TERM MEMORY

Disusun oleh:

Niken Oktaviana
123160053

Telah diuji dan dinyatakan lulus pada tanggal 06 Oktober 2020 oleh :

Menyetujui,

Penguji I	Penguji II
Herry Sofyan, S.T.,M.Kom.	Heru Cahya Rustamaji, S.Si., M.T.
NIK. 2 6404 96 0139 1	NIK. 2 7106 96 0065 1

Penguji I	Penguji II
Dr. Herlina Jayadianti, S.T., M.T.	Rifki Indra Perwira, S.Kom., M.Eng
NIK. 2 7708 02 0235 1	NIK. 2 8307 12 0418 1

SURAT PERNYATAAN

KARYA ASLI TUGAS AKHIR

Sebagai mahasiswa Program Studi Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta, yang bertanda tangan dibawah ini, saya :

Nama : Niken Oktaviana

No Mhs : 123160053

Menyatakan bahwa karya ilmiah saya yang berjudul :

ANALISIS SENTIMEN PADA ULASAN OBJEK WISATA PANTAI DI GOOGLE MAPS MENGGUNAKAN METODE LONG-SHORT TERM MEMORY

merupakan karya asli saya dan belum pernah dipublikasikan dimanapun. Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, maka saya bersedia menerima konsekuensi apa pun yang diberikan Program Studi Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Yogyakarta

Pada tanggal : 30 Oktober 2020

Yang menyatakan

(Niken Oktaviana)

PERNYATAAN BEBAS PLAGIAT

Saya yang bertanda tangan dibawah ini :

Nama : Niken Oktaviana

No. Mhs : 123160053

Fakultas/Prodi : Teknik Industri/ Informatika

Dengan ini saya menyatakan bahwa judul Tugas Akhir

ANALISIS SENTIMEN PADA ULASAN OBJEK WISATA PANTAI DI GOOGLE MAPS MENGGUNAKAN METODE LONG-SHORT TERM MEMORY

adalah hasil kerja saya sendiri dan benar bebas dari plagiat kecuali cuplikan serta ringkasan yang terdapat di dalamnya telah saya jelaskan sumbernya (sitasi) dengan jelas. Apabila pernyataan ini terbutki tidak benar maka saya bersedia menerima sanksi sesuai peraturan Mendiknas RI No 17 Tahun 2010 dan Peraturan Perundang-undangan yang berlaku.

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab.

Yogyakarta, 30 Oktober 2020

Yang membuat pernyataan

Niken Oktaviana
NIM 123160053

HALAMAN PERSEMBAHAN

Bismillah...

Tugas akhir ini saya persembahkan untuk saya sendiri, Ayah, Ibu, adik-adik, teman-teman, dan semua pihak yang sering bertanya "kapan lulus?".

Kalian adalah alasan saya segera menyelesaikan tugas akhir ini.

ABSTRAK

Dinas Pariwisata Daerah Istimewa Yogyakarya (DIY) melakukan evaluasi dalam rangka menyusun strategi sebagai upaya meningkatkan pengelolaan obyek wisata di Yogyakarta. Namun teknis pelaksanaan evaluasi yang selama ini diterapkan dinilai masih kurang efektif. Hal tersebut disebabkan karena evaluasi masih fokus dilakukan secara langsung dan kondisional serta belum melibatkan opini masyarakat umum. Untuk mempermudah Dinas Pariwisata DIY dalam melakukan evaluasi melalui opini masyarakat, maka perlu dilakukan analisis sentimen. Opini diklasifikasikan ke dalam sentimen positif atau negatif, serta diklasifikasikan ke dalam kategori daya tarik, aksesbilitas, kebersihan, atau fasilitas.

Penelitian ini menggunakan algoritma *Long-Short Term Memory (*LSTM) yang dikombinasikan dengan pembobotan *word2vec*. LSTM merupakan salah satu algoritma *deep learning* yaitu pengembangan dari RNN yang menangani data *sequential*. Data yang digunakan bersumber dari ulasan *google maps* dan diberi label secara manual. Proses yang dilakukan adalah mengumpulkan data latih dan data uji, melakukan tahap *preprocessing, sentence conversion,* pembobotan dengan *word2vec*, kemudian dilakukan tahap klasifikasi. Lalu pada tahap terakhir dilakukan pengujian menggunakan tabel *confusion matrix* dan kurva ROC untuk mencari nilai akurasi, presisi, *recall*, dan nilai AUC. Pengujian menggunakan persentase data latih 80% dan data uji 20% dengan pendekatan K-*fold cross validation*.

Hasil pengujian dengan *confusion matrix* menghasilkan nilai akurasi rata-rata sebesar 84%, presisi sebesar 76%, dan *recall* sebesar 0,73% untuk model sentimen. Sedangkan untuk model kategori diperoleh akurasi rata-rata sebesar 76%, presisi sebesar 75%, dan *recall* sebesar 0,74%. Hasil pengujian dengan kurva ROC diperoleh nilai rata-rata AUC sebesar 0.73 (*fair classification*) untuk model sentimen. Sedangkan untuk model kategori diperoleh nilai rata-rata AUC sebesar 0.83 (*good classification*).

Kata kunci : Dinas Pariwisata, analisis sentimen, ulasan *google maps,* LSTM

KATA PENGANTAR

Segala puji syukur penulis panjatkan kehadirat Allah SWT, yang telah memberikan petunjuk dan kekuatan dalam menyelesaikan Tugas Akhir yang berjudul "Analisis Sentimen pada Ulasan Obyek Wisata Pantai di *Google Maps* menggunakan Metode *Long-Short Term Memory*". Tugas akhir ini merupakan syarat terakhir yang harus ditempuh untuk menyelesaikan pendidikan pada jenjang Strata Satu (S1) Program Studi Informatika Universitas Pembangunan Nasional "Veteran" Yogyakarta.

Didalam penyusunan tugas akhir ini, penulis menyadari bahwa penulisan ini tidak lepas dari bimbingan, bantuan serta dukungan dari berbagai pihak baik secara langsung maupun tidak langsung. Oleh karena itu, penulis mengucapkan terimakasih kapada:

1. Allah SWT yang senantiasa memberikan kesehatan, petunjuk dan kemudahan selama menyelesaikan tugas akhir.
2. Ayah dan Ibu yang telah sabar untuk selalu mendukung dan mendoakan penulis dalam menyelesaikan tugas akhir ini.
3. Bapak Herry Sofyan, S.T., M.Kom., selaku dosen pembimbing I atas waktu, pengertian, segala bantuan dalam memberikan referensi dan inovasi, pemberian kritik dan saran yang membangun, dan juga pemberian semangat kepada penulis.
4. Bapak Heru Cahya Rustamaji, S.Si, M.T., selaku dosen pembimbing II atas waktu, pengertian, segala bantuan dalam memahami materi dan memecahkan permasalahan, pemberian kritik dan saran, dan juga pemberian semangat kepada penulis.
5. Ibu Yuli Fauziah, S.T.,M.T., sebagai dosen wali atas dukungan dan semangat luar biasa kepada penulis.
6. Seluruh Dosen dan Pegawai jurusan Informatika atas kebaikan, dukungan, dan bantuan yang diberikan kepada penulis dalam menyelesaikan tugas akhir.

7. Ibu Ellya Shari, S.ST.Par., selaku Seksi Sarana dan Prasarana Bidang Destinasi Wisata Dinas Pariwisata DIY atas waktu, dukungan, dan kesabaran dalam berdiskusi dan memberikan informasi yang dibutuhkan dalam penyusunan tugas akhir.
8. Mas Vincent, Agus, Yahdi, Sugeng, Arif. Terimakasih atas ilmu dan bantuan dalam mengerjakan tugas akhir.
9. Teman-teman semua yang telah memberikan semangat serta bantuan dalam penyusunan tugas akhir.

Penulis menyadari penyusunan tugas akhir ini tentunya tidak lepas dari kekurangan dan masih jauh dari sempurna. Oleh karena itu penulis mengharapkan masukan berupa kritik, saran, dan tanggapan yang bersifat membangun dalam upaya pembelajaran lebih lanjut.

Akhir kata semoga tugas akhir ini bermanfaat bagi semua pihak, dan penulis sendiri pada khususnya.

Yogyakarta, Oktober 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR MODUL PROGAM

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Provinsi Daerah Istimewa Yogyakarta menjadi salah satu provinsi di Indonesia yang memiliki daya tarik untuk berwisata. Hal ini ditunjukkan dengan pernyataan bahwa Yogyakarta mendapat predikat sebagai daerah tujuan wisata kedua di Indonesia setelah Bali (Hermawan, 2017). Daerah Istimewa Yogyakarta memiliki destinasi wisata yang cukup beragam, salah satu yang terkenal adalah pantai (Dinas Pariwisata DIY, 2018). Demi menjaga dan meningkatkan kualitas serta menyusun strategi dalam pengelolaan obyek wisata di Yogyakarta, diperlukan berbagai upaya termasuk peran Dinas Pariwisata sebagai pihak pemerintah yang bertanggung jawab dalam sektor pariwisata. Salah satu upaya penting yaitu melakukan analisis kondisi objek wisata yang ada untuk mengetahui evaluasi serta menyiapkan solusi demi terwujudnya sektor pariwisata yang maju. Ibu Ellya Shari sebagai Seksi Sarana dan Prasarana Bidang Destinasi Wisata Dinas Pariwisata Provinsi Daerah Istimewa Yogyakarta, menyampaikan bahwa analisis kondisi objek wisata yang selama ini dilakukan oleh pemerintah dinas pariwisata Daerah Istimewa Yogyakarta adalah dengan cara mendatangi langsung lokasi wisata yang diambil jumlah sampel random setiap bulannya. Dalam praktek monitoring tersebut, jumlah sampel dan rentang waktu setiap kunjungan tidak ditentukan secara pasti. Sehingga cara yang selama ini diterapkan dinilai masih kurang efektif. Ibu Ellya Shari juga mengatakan bahwa selama ini, Pemerintah Dinas Pariwisata DIY masih terfokus dengan monitoring secara langsung ke lokasi. Pemerintah belum melibatkan masyarakat umum dalam melakukan evaluasi pelayanan dan sarana prasarana tempat wisata yang ada di Yogyakarta. Menurut Ibu Ellya Shari, sebenarnya Pemerintah Dinas perlu mendengar aspirasi dari masyarakat dalam melakukan evaluasi, namun saat ini masih belum diterapkan.

Di era digital saat ini, banyak fasilitas digital yang disediakan untuk masyarakat dalam menyampaikan opini yang berkaitan dengan obyek wisata, salah satunya kolom ulasan pada *google maps*. Ulasan-ulasan yang dituliskan secara *online* dinilai memiliki peran lebih, dibuktikan dengan hasil *local consumer review survey* yang menyebutkan bahwa 78% konsumen lebih mempercayai ulasan *online* daripada rekomendasi dari seseorang (Bright Local, 2018). Hasil dari opini yang dituliskan berperan penting bagi Dinas Pariwisata untuk mengetahui evaluasi dan menyiapkan strategi melalui tingkat kepuasan pengunjung (Koesumaningrum, 2018). Namun kalimat ulasan yang ada pada *google maps* belum cukup membantu untuk mengetahui kekurangan dan kelebihan suatu tempat wisata karena ulasannya mengandung berbagai aspek penilaian (Koesumaningrum, 2018). Maka dari itu perlu adanya analisis sentimen ulasan *google maps* pada tempat wisata khususnya pantai sehingga dapat diketahui penilaian sentimen dari masing-masing tempat.

Tantangan yang dihadapi dalam proses analisis sentimen yaitu adanya penilaian positif maupun negatif dalam satu ulasan yang menyebabkan sulit menemukan pola klasifikasinya. Maka dari itu, untuk meminimalisir kendala tersebut dapat dilakukan dengan menggunakan metode *deep learning*. Hal tersebut karena metode *deep learning* sangat efektif dan lebih mudah dalam mengidentifikasi pola-pola dari data yang diproses. *Deep learning* memiliki algoritma dengan struktur dan jumlah jaringan syaraf yang banyak sebagai jaringan tersembunyi (*hidden layer*) yang dapat digunakan untuk mengidentifikasi pola untuk klasifikasi ulasan. Selain itu, kelebihan lain dari *deep learning* yaitu mampu menangani data dengan skala yang besar dan mengekstraksi fitur dari data tersebut secara otomatis. Salah satu algoritma yang termasuk dalam metode *deep learning* dan mampu mengatasi permasalahan tersebut adalah *Long-Short Term Memory*.

Long-Short Term Memory (LSTM) merupakan salah satu variasi dari *Recurrent Neural Network* (RNN) yang dibuat untuk menghindari masalah ketergantungan jangka

panjang. Selain itu, LSTM terbukti memiliki tingkat akurasi yang tinggi dalam mengatasi masalah analisis sentimen. Seperti penelitian yang dilakukan oleh Dan Li dan Jiang Oian (2016) tentang analisis sentimen dengan LSTM yang hasilnya lebih tinggi tingkat akurasinya dibandingkan dengan RNN konvensional. Metode LSTM juga pernah dibandingkan dengan metode *Naive Bayes Classifier* dan menghasilkan akurasi yang lebih tinggi yaitu selisih 4,97% (Nurrohmat dan SN, 2019).

Berdasarkan permasalahan yang telah dipaparkan di atas, maka penerapan metode *deep learning* dengan menggunakan *Long-Short Term Memory* menjadi solusi yang ditawarkan pada penelitian ini. Data yang digunakan bersumber dari ulasan *google maps*, dimana data tersebut akan menjadi data latih dan juga data uji. Data latih digunakan sebagai bahan belajar untuk menghasilkan model klasifikasi yang mampu mengenali pola-pola sentimen, sedangkan data uji digunakan untuk mengukur performa dari algoritma *Long-Short Term Memory*. Sebelumnya ulasan akan dikelompokkan ke dalam empat kategori yang diambil berdasarkan pedoman dari Analisis Daerah Operasi Objek dan Daya Tarik Wisata Alam (ADO-ODTW) Dirjen PHKA tahun 2003, yaitu kategori daya tarik, aksesbilitas, fasilitas, dan kebersihan. Dengan menerapkan algoritma LSTM diharapkan dapat menghasilkan suatu model yang dapat dimanfaatkan untuk melakukan klasifikasi ulasan sehingga dapat diketahui penilaian sentimen dan penggolongan kategori.

1.2 Perumusan Masalah

Berdasarkan penjelasan latar belakang yang telah dipaparkan, maka dapat diambil rumusan masalah sebagai berikut:

1. Bagaimana menganalisis sentimen ulasan pantai di Yogyakarta yang bersumber dari *google maps* sehingga dapat memberikan informasi mengenai evaluasi pantai dari berbagai kategori menggunakan algoritma *Long-Short Term Memory*?

2. Bagaimana tingkat performa algoritma *Long-Short Term Memory* dalam melakukan analisis sentimen ulasan pantai di Yogyakarta?

1.3 Batasan Masalah

Guna menghindari terjadinya topik permasalahan yang lebih luas, maka diberikan batasan masalah sebagai berikut:

1. Data penelitian didapatkan dari ulasan *google maps* dalam lingkup 35 tempat wisata pantai di Provinsi Daerah Istimewa Yogyakarta.
2. Dataset yang digunakan dari tahun 2017 sampai dengan tahun 2019.
3. Data diberi label sentimen dan label kategori secara manual.
4. Pemberian label hanya salah satu yang paling dominan (bukan multilabel).
5. Data penelitian yang digunakan berbahasa Indonesia.
6. Persentase pembagian data latih dan data uji yang digunakan adalah 80:20.

1.4 Tujuan Penelitian

Tujuan pada penelitian ini adalah sebagai berikut:

1. Dapat memberikan informasi mengenai evaluasi pantai dari berbagai kategori melalui analisis sentimen ulasan pantai di Yogyakarta yang bersumber dari *google maps* menggunakan algoritma *Long-Short Term Memory*.
2. Dapat mengetahui tingkat performa algoritma *Long-Short Term Memory* dalam melakukan analisis sentimen ulasan pantai di Yogyakarta.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah untuk membantu pemerintah Dinas Pariwisata Daerah Istimewa Yogyakarta dalam menganalisis kondisi pantai-pantai yang ada di Yogyakarta guna mengetahui evaluasi dari berbagai kategori agar dapat segera dilakukan tindakan perbaikan.

1.6 Metodologi Penelitian

1.6.1 Metodologi Penelitian

Metodologi penelitian yang digunakan adalah metode kualitatif dengan tahapan wawancara bersama pihak Dinas Pariwisata Provinsi Daerah Istimewa Yogyakarta guna mencari informasi mengenai permasalahan yang dialami oleh dinas pariwisata dalam melakukan monitoring analisis kondisi objek wisata khususnya pantai di Yogyakarta. Tahapan lainnya adalah studi kepustakaan untuk melakukan studi perbandingan metode dan mencari solusi dari permasalahan penelitian. Studi kepustakaan bersumber dari buku teks, jurnal ilmiah, dan sumber lain yang dapat dipertanggungjawabkan.

1.6.2 Metodologi Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam penelitian ini adalah metode *prototype*. Memilih metode *prototype* karena metode ini lebih menekankan pada komunikasi antara pengembang dan pelanggan sehingga kebutuhan pengguna dapat diterjemahkan dalam bentuk model *prototype*. Selain itu, metode *prototype* juga dapat dievaluasi dan dimodifikasi kembali hingga mencapai model yang diinginkan. Beberapa tahapan yang dilakukan pada metode *prototype* yaitu (Pressman, 2010):

1. *Communication*

Pada tahapan ini dilakukan proses komunikasi antara pengembang dan pelanggan mengenai tujuan dibuatnya perangkat lunak. Selain itu, dilakukan analisis tentang kebutuhan yang diperlukan selama proses pembuatan perangkat lunak.

2. *Quick Plan* dan *Modeling Quick Design*

Pada tahapan ini dilakukan perancangan dan pemodelan berdasarkan kebutuhan yang telah didefinisikan pada tahapan sebelumnya. Sedangkan dari sisi pemodelan akan membuat desain model yang merepresentasikan aspek-aspek sesuai keinginan pelanggan.

3. *Construction of Prototype*

Pada tahap *construction of prototype*, pengembang akan mulai membuat perangkat lunak berdasarkan rencana dan model yang telah dibuat sebelumnya. Pada tahap ini juga termasuk pengujian atau *testing* dari perangkat lunak yang telah dibuat.

4. *Deployment Delivery & Feedback*

Setelah perangkat lunak telah dibuat dan sudah melalui proses *testing*, proses selanjutnya adalah tahap *deployment delivery & feedback*. Perangkat lunak yang dibuat akan disampaikan kepada pelanggan untuk dilakukan pengecekan. Jika mendapatkan *feedback* baik, maka perangkat lunak akan diterima oleh pelanggan dan pembuatan perangkat lunak telah selesai. Namun apabila *feedback* yang diberikan tidak sesuai dengan keinginan pelanggan, maka kembali ke tahap pertama yaitu *communication* dengan membahas perbaikan yang perlu dilakukan.

1.7 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam menyusun laporan penelitian ini adalah sebagai berikut :

Bab I Pendahuluan

Pada bagian ini membahas tentang latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

Bab II Tinjauan Pustaka

Tinjauan literatur memuat tentang dasar teori yang sudah ada sebagai bahan referensi terkini dan pondasi untuk memperkuat argumentasi dalam penelitian ini. Teori-teori yang sesuai dengan penelitian ini dan akan dijabarkan dalam tinjauan pustaka antara lain ulasan *google maps*, wisata pantai, *web scraping*, analisis sentimen, *text mining*, *deep learning*, *word embedding*, metode *Long-Short Term Memory*, dan pengujian.

Bab III Metodologi Penelitian dan Pengembangan Sistem

Bab ini membahas tentang tahap perancangan kebutuhan, tahap analisis, dan tahap perancangan serta memberikan gambaran garis besar penyusunan program.

Bab IV Hasil, Pengujian dan Pembahasan

Pada bab ini akan menyajikan hasil penelitian yang berisi hasil implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Selain itu berisi pengujian terhadap hasil penelitian beserta pembahasannya.

Bab V Kesimpulan dan Saran

Pada bagian ini berisi kesimpulan dari hasil penelitian dan saran yang diajukan oleh penulis untuk pengembangan penelitian selanjutnya.

BAB II

TINJAUAN PUSTAKA

2.1 Wisata Pantai

Wisata adalah kegiatan perjalanan yang dilakukan secara sukarela serta bersifat sementara waktu untuk menikmati objek dan daya tarik wisata. Sedangkan objek wisata adalah tempat yang memiliki daya tarik untuk berkunjung yang bertujuan menikmati, memenuhi rasa ingin tahu, yang bersifat rekreatif dan edukatif (Ali, 2016). Tujuan wisata yang dapat dikunjungi sangat beragam, salah satunya pantai. Pantai adalah sebuah bentuk geografis yang terdiri dari pasir, dan terdapat di daerah pesisir laut. Daerah pantai menjadi batas antara daratan dan perairan laut (Rif'an, 2018). Karena lokasi pantai yang indah dan masing-masing pantai memiliki keunikan tersendiri, maka wisata pantai menjadi salah satu tujuan destinasi wisata favorit bagi masyarakat.

Daerah Istimewa Yogyakarta mempunyai puluhan pantai yang tersebar di tiga kabupaten, yaitu Kabupaten Kulon Progo, Bantul, dan Gunung Kidul. Masing-masing kabupaten memiliki dinas pariwisata yang berperan dalam meningkatkan kualitas dan pelayanan di seluruh obyek wisata. Seluruh Dinas Pariwisata di Kabupaten/Kota akan di kontrol langsung oleh Dinas Pariwisata tingkat provinsi Daerah Istimewa Yogyakarta. Sehingga pemerintah dinas pariwisata baik tingkat kabupaten maupun provinsi mempunyai kewajiban dan tugas masing-masing dalam rangka mencapai pariwisata yang maju. Menurut Ellya Shari sebagai seksi sarana dan prasarana bidang destinasi wisata Provinsi Daerah Istimewa Yogyakarta, salah satu upaya penting dalam menjaga eksistensi kualitas wisata yang ada di Yogyakata yaitu dengan cara melakukan analisis kondisi objek wisata yang ada untuk mengetahui evaluasi serta menyiapkan solusi demi terwujudnya sektor pariwisata yang maju.

2.2 Ulasan *Google Maps*

Google maps merupakan jasa peta atau globe virtual dan *online* yang disediakan oleh perusahaan *google*. *Google maps* mempunyai fitur untuk merencanakan sebuah rute dan mencari sebuah alamat. *Google* juga menambahkan fitur rating dalam *google maps* sehingga penggunanya dapat menambahkan ulasan langsung pada tempat yang dikunjunginya (Wilianto dkk, 2017). Ulasan yang diisi oleh pengguna *google maps* bisa berupa pujian, kritik, dan juga saran untuk lokasi yang sedang dikunjungi.

2.3 Analisis Sentimen

Analisis sentimen juga dikenal sebagai *opinion mining* atau *emotion artificial intelligence* adalah penggunaan pemprosesan bahasa alami, analisis teks, komputasi linguistik dan biometrik untuk mengidetifikasi, mengekstrak, menghitung dan mempelajari informasi subjektif secara sistematis (Perdana, 2017). Analisis Sentimen mengacu pada bidang yang luas dari pengolahan bahasa alami, komputasi linguistik dan *text mining* yang bertujuan menganalisa pendapat, sentimen, evaluasi, sikap, penilaian dan emosi seseorang apakah pembicara atau penulis berkenaan dengan suatu topik, produk, layanan, organisasi, individu, ataupun kegiatan tertentu (Rizal, 2017)

Tugas dasar dalam analisis sentimen adalah mengelompokkan polaritas dari teks yang ada dalam dokumen, kalimat, atau fitur tingkat aspek dan menentukan apakah pendapat yang dikemukakan dalam dokumen, kalimat atau fitur entitas atau aspek bersifat positif, negatif atau netral. Selain itu, analisis sentimen dapat menyatakan emosional sedih, gembira, atau marah (Murnawan, 2017). Ekspresi atau sentimen mengacu pada fokus topik tertentu, pernyataan pada satu topik memiliki kemungkinan akan berbeda makna dengan pernyataan yang sama pada subjek yang berbeda.

2.4 *Web Scraping*

Web scraping adalah proses pengambilan sebuah dokumen semi-terstruktur dari suatu website yang bertujuan untuk mengambil informasi dari halaman website tersebut baik secara keseluruhan atau sebagian guna dimanfaatkan untuk kepentingan lain (Priyanto dan Ma'arif, 2018).

Proses *web scraping* pertama kali dilakukan dengan cara manual yaitu dengan *copy paste* dari website ke tempat penyimpanan lokal. Namun, apabila data yang ingin diambil dalam jumlah banyak, maka cara ini kurang efektif karena memerlukan tenaga untuk menyalin dan juga menguras banyak waktu. Selain dengan cara manual, *web scraping* dapat dilakukan otomatis dengan beberapa cara, diantaranya menggunakan *coding*, aplikasi ataupun *extension browser*. Pada penelitian ini menggunakan *coding* dalam proses *web scraping*.

Secara umum, ada empat tahapan dalam penggunaan *web scraping* untuk mengambil data secara otomatis dalam website sebagai berikut (Priyanto dan Ma'arif, 2018) :

1. Mempelajari dokumen HTML dari website untuk tag HTML yang mengapit informasi yang akan diambil.
2. Menelusuri mekanisme navigasi pada website yang akan diambil informasinya untuk ditirukan pada aplikasi *web scraper* yang akan dibuat.
3. Berdasarkan informasi yang didapat pada tahap pertama dan kedua, aplikasi *web scraper* dibuat untuk mengotomatisasi pengambilan informasi dari website yang ditentukan.
4. Informasi yang diambil dari hasil langkah ketiga akan disimpan dalam format data tertentu, misalnya csv atau *database*.

Dalam penelitian ini *web scraping* digunakan untuk mengambil data dari website *google maps* berupa ulasan menggunakan *chrome driver* dan bahasa pemrograman *python*.

2.5 *Text Mining*

Text Mining atau penambangan teks adalah istilah yang mendeskripsikan sebuah teknologi yang mampu menganalisis data teks semi-terstruktur maupun tidak terstruktur untuk mendapatkan suatu informasi. Dalam pengaplikasiannya *text mining* digunakan untuk menangani masalah yang berkaitan dengan klasifikasi, *clustering*, *information retrieval,* dan *information extraction* (Berry dan Kogan, 2010).

Proses *text mining* dimulai dengan *preprocessing* teks dokumen. Selanjutnya dokumen teks akan ditransformasikan menjadi data yang berupa angka agar dapat diolah oleh komputer. Tiap kata pada dokumen dalam *text mining* biasa disebut sebagai *term* atau fitur. Proses terakhir pada *text mining* adalah evaluasi.

2.6 Klasifikasi Teks

Klasifikasi teks merupakan bagian penting dari *text mining*. Klasifikasi teks bekerja dengan menganalisis atau mempelajari himpunan dokumen teks yang belum terklasifikasi untuk memperoleh suatu model atau fungsi yang dapat digunakan untuk mengelompokkan dokumen teks lain yang belum diketahui kelasnya ke dalam satu atau lebih kelas yang sudah didefinisikan (Sebastiani, 2002). Klasifikasi teks terdiri atas dua macam pelabelan, yaitu *single*-label (pelabelan tunggal) yang berarti hanya ada satu kelas, dan *multi*-label yang berarti lebih dari satu kelas.

Berdasarkan keluaran hasil klasifikasi, proses dokumen dapat dibagi menjadi:

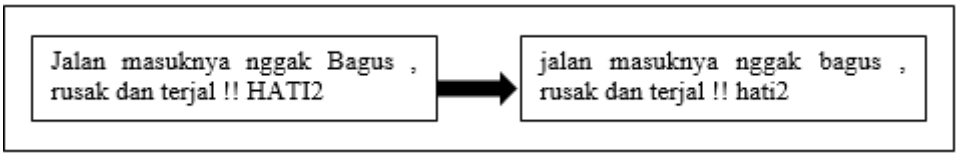
1. Klasifikasi *binary*, misalnya untuk klasifikasi fakta-hoax. Sehingga bernilai 0 dan 1.
2. Klasifikasi *multiclass*, misalnya klasifikasi judul buku, sehingga penggolongan kelas didasarkan lebih dari dua kelas.
3. Klasifikasi *multiclass* dalam skala besar, misalnya klasifikasi untuk penggolongan ribuan kelas pada suatu perusahaan besar.

2.7 *Text Preprocessing*

Text preprocessing merupakan tahap awal dari *text* mining. *Text preprocessing* adalah proses untuk membersihkan dan mempersiapkan data sebelum dilakukannya proses klasifikasi (Haddi dkk, 2013). Tujuan dari text preprocessing adalah untuk mendapatkan bentuk data yang siap olah. Tahapan yang dilakukan pada *text preprocessing* ini yaitu *case folding, cleansing, tokenizing, stemming,* dan *stopword removal* atau *filtering*.

2.7.1 *Case Folding (lowercase)*

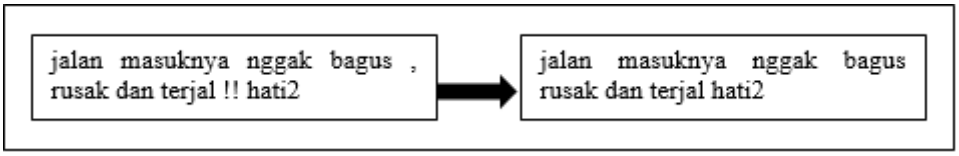
Tahap *case folding* adalah proses mengubah semua huruf pada dokumen menjadi satu bentuk, misalnya huruf kapital menjadi huruf kecil dan sebaliknya. Pada penelitian ini akan menyamakan semua huruf dalam dokumen menjadi huruf kecil (*lowercase*). Huruf yang diterima adalah ‘a’ sampai dengan ‘z’. Contoh *case folding* untuk *lowercase* dapat dilihat pada gambar 2.1.



Gambar 2.1 Proses *Case Folding (Lowercase)*

2.7.2 *Remove Punctuation*

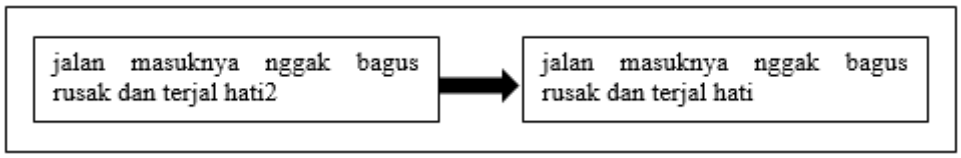
Tahap *remove punctuation* merupakan proses untuk menghapus tanda baca yang terdapat pada data. Penghapusan tanda baca pada suatu data akan mengurangi beban pemrosesan. Beberapa contoh tanda baca seperti titik (.), koma (,), tanda tanya (?), *hashtag* (#), dan lain-lain. Contoh *remove punctuation* dapat dilihat pada gambar 2.2.



Gambar 2.2 Proses *Remove Punctuation*

2.7.3 *Remove Number*

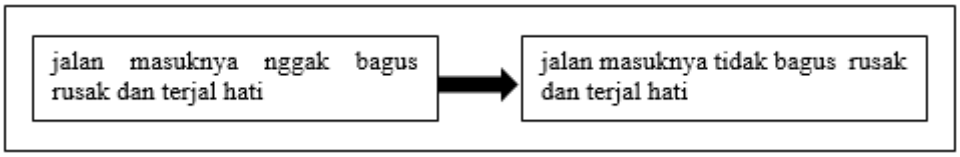
Tahap *remove number* merupakan tahapan yang hampir sama dengan *remove punctuaion*. Bedanya terletak pada objek yang di hapus. Jika pada *remove punctuation* yang dihapus adalah tanda baca, namun jika *remove number* yang dihapus adalah angka pada suatu data. Angka dihapus karena dianggap tidak memiliki arti dan termasuk *delimiter*. Contoh *remove number* dapat dilihat pada gambar 2.3.



Gambar 2.3 Proses *Remove Number*

2.7.4 *Spelling Correction*

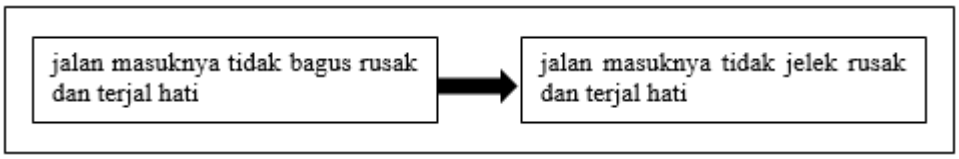
Spelling correction atau koreksi ejaan adalah tahapan *preprocessing* untuk mengganti kata yang berupa singkatan, salah penulisan, maupun kata tidak baku dengan kata yang ada di dalam kamus buatan. Contohnya yaitu “gk”, “nggak”, “engga” diubah menjadi “tidak”. Contoh *spelling correction* dapat dilihat pada gambar 2.4.



Gambar 2.4 Proses *spelling correction*

2.7.5 *Negation Word Conversion*

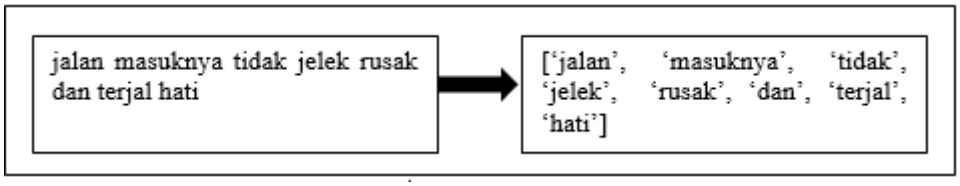
Negation word conversion atau konversi kata negasi adalah proses mengubah kata yang berlawanan atau antonim. Kamus negasi diperoleh dari pencarian kata setelah kata “tidak” pada datasets. Kemudian kata-kata yang telah dikumpulkan akan diseleksi sesuai kebutuhan penelitian, selanjutnya dicari kata negasi atau antonimnya. Kata yang diganti menjadi antonim kata tersebut adalah kata yang berada pada setelah kata “tidak”. Contoh *negation word conversion* dapat dilihat pada gambar 2.5.



Gambar 2.5 Proses *Negation Word Conversion*

2.7.6 *Tokenizing*

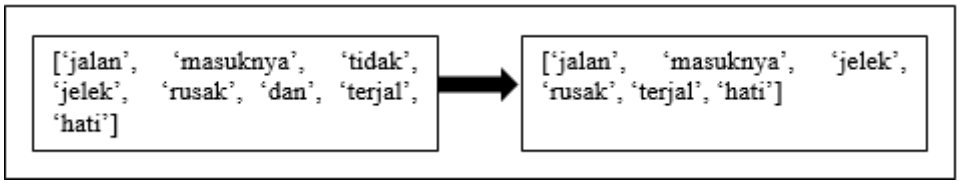
Tahap *tokenizing* atau tokenisasi adalah tahap pemotongan string input berdasarkan tiap kata yang menyusunnya (Murnawan, 2017). Setiap kata teridentifikasi dengan kata yang lain oleh karakter spasi, sehingga proses *tokenizing* akan mengendalikan karakter spasi pada dokumen untuk pemisahan kata. Contoh dari proses *tokenizing* dapat dilihat pada gambar 2.6.



Gambar 2.6 Proses *Tokenizing*

2.7.7 *Stopword Removal*

Tahap *stopword removal* atau disebut juga filtering adalah proses pengambilan kata-kata penting atau membuang kata kata kurang penting dari token yang telah dihasilkan pada tahap sebelumnya (Nurrohmat dan SN, 2019). Pembuangan *stopword* tidak akan mengubah makna dan isi dari suatu data. Beberapa contoh stopword seperti yang, juga, aku, kamu, ini, dan itu. Pada penelitian ini digunakan *stopword* dari *library* Sastrawi. Contoh dari proses *stopword removal* dapat dilihat pada gambar 2.7.



Gambar 2.7 Proses *stopword removal*

2.7.8 *Stemming*

Tahap *stemming* merupakan proses untuk mendapatkan kata dasar dari suatu kata dalam kalimat. Proses *stemming* dilakukan dengan cara memisahkan masing-masing kata dari kata dasar dan imbuhannya, baik awalan maupun akhiran. Salah satu tujuan proses *stemming* adalah untuk meningkatkan efisiensi cara memilah isi dokumen menjadi unit-unit kecil. Sebagai contoh, kata memandang, dipandang, pandangan, akan di *stem* ke kata dasarnya yaitu "pandang".

Algoritma *stemming* yang sering digunakan untuk penelitian klasifikasi teks berbahasa Indonesia diantaranya ada algoritma *stemming* Porter dan Nazief & Adriani. Dari penelitian Rezalina (2016) yang membandingkan beberapa algoritma *stemming* untuk dokumen berbahasa Indonesia dapat diketahui bahwa algoritma Nazief & Adriani lebih unggul dibanding lainnya dalam hal kecepatan dan akurasi, termasuk algoritma *stemming* Porter.

Pada penelitian ini akan menggunakan *stemmer* Sastrawi sebagai implementasi dari algoritma Nazief & Adriani. *Stemmer* Sastrawi dinilai cocok untuk klasifikasi teks pada penelitian ini karena akurasi *stemming*nya tergolong tinggi untuk teks berbahasa Indonesia (Luhrie, 2019). Adapun langkah-langkah proses pada algoritma Nazief & Adriani adalah sebagai berikut:

1. Kata dicari pada kamus, jika ditemukan maka kata tersebut dianggap sebagai kata dasar yang benar dan algoritma dihentikan. Jika tidak maka lanjut langkah kedua.
2. Hilangkan *Influectional suffeixes*, yaitu menghilangkan partikel seperth", "kah", "tah", atau "pun", kemudian hilangkan *inflectional possesive pronoun suffiex* seperti "ku", "mu", "nya" jika memang ada. Cek kata dalam kamus kata dasar, jika ditemukan maka algoritma dihentikan. Jila tidak lanjut ke langkah yang ketiga.

3. Hapus *Derivational suffiex* seperti “i”,”-an”, atau “kan”. Jika kata ditemukan dalam kata kamus kata dasar, maka algoritma akan dihentikan. Jika tidak, maka lanjut ke langkah 3.a.
   a. Jika akhiran “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lanjut ke langkah 3.b.
   b. Akhiran yang dihapus (“-i”,”-an”, atau “-kan”) dikembalikan, lanjut ke langkah 4.
4. Hapus *Derivational Prefix* (awalan turunan) seperti “be-“,”di-“,”me-“,”pe-”,”se-” dan “te-“. Jika pada langkah ketiga ada *suffix* yang dihapus maka lanjut ke langkah 4.a, jika tidak maka pergi ke langkah 4.b.
   a. Periksa tabel kombinasi awalan-akhiran yang tidak diizinkan. Jika ditemukan maka algoritma berhenti, jika tidak maka melakukan proses 4.b. Kombinasi awalan dan akhiran yang tidak diijinkan dapat dilihat pada Tabel 2.1.

Tabel 2.1 Kombinasi awalan dan akhiran yang tidak diijinkan (Nazief dan Adriani, 1996)

Awalan	**Akhiran yang tidak diijinkan**
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan

   b. Untuk i sama dengan 1 sampai 3, tentukan tipe awalan kemudian hapus awalan. Jika kata dasar belum ditemukan, maka lanjut ke langkah selanjutnya. Jika sudah menemukan maka algoritma berhenti. Proses ini harus dilihat terlebih dahulu pada aturan kata pada Tabel 2.2 dan Tabel 2.3.

Tabel 2.2 Aturan peluruhan kata dasar (Nazief dan Adriani, 1996)

Aturan	Awalan	Peluruhan
1	berV. . .	ber-V.. | be-rV..
2	Belajar	bel-ajar
3	berClerC2	be-ClerC2.. dimana C1!= {'r'|'l'}
4	terV. . .	ter-V.. | te-rV..
5	terCer. . .	terCer. . . dimana C!='r'
6	teClerC2	te-CleC2. . . dimana C1!='r'
7	me{I|r|w|y}V. . .	me-{I|r|w|y}V. . .
8	mem{b|fv}. . .	mem-{b|f|v}. . .
9	mempe. . .	m-pe. .
10	mem{r|V|V}. . .	me-m{r|V|V}. . . | me-p{r|V|V}. . .
11	men{c|d|j|z}. . .	Men-{c|d|j|z}. . .

Tabel 2.3 Lanjutan aturan peluruhan kata dasar (Nazief dan Adriani, 1996)

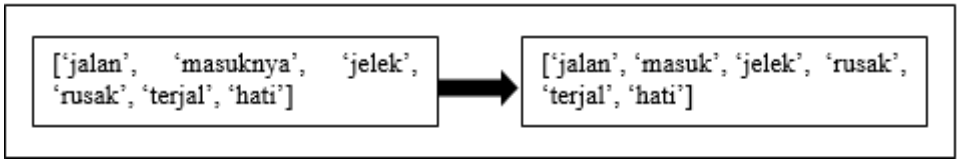
Aturan	Awalan	Peluruhan
12	menV. . .	Me-nV. . . | me-tV. . .
13	meng{g|h|q|k}. . .	Meng-{g|h|q|k}. . .
14	mengV. . .	Meng-V. . . | meng-kV. . .
15	mengeC. . .	Meng-C. . .
16	menyV. . .	Me-ny. . . | men-sV. . .
17	memV. . .	Mem-pV. . .
18	pe{w|y}V. . .	Pe-{w|y}V. . .
19	perV. . .	Per-V. . . | pe-rV. . .
20	pem{b|f|v}. . .	Pem-{b|f|v}. . .
21	pem{rV|V}. . .	Pe-m{rV|V}. . . | pe-p{rV|V}. . .
22	pen{c|d|j|z}. . .	Pen-{c|d|j|z}. . .
23	penV. . .	Pe-nV. . . | pe-tV. . .
24	peng{g|h|q}. . .	Peng-{g|h|q}
25	pengV. . .	Peng-V | peng-kV
26	penyV. . .	Pe-nya | peny-sV
27	pelV. . .	Pe-lV. . .; kecuali untuk kata "pelajar"
28	peCP. . .	Pe-CP. . . dimana C!={r|w|y|l|m|n} dan P!='er'
29	perCerV. . .	Per-CerV. . . dimana C!={r|w|y|l|m|n}

Tipe awalan ditentukan melalui langkah-langkah berikut:

1. Jika awalannya adalah "di-","ke-" atau "se-" maka tipe awalannya secara berturut-turut adalah "di-", "ke-" atau "se-".
2. Jika awalannya adalah ""te-", "me-", "be-" atau "pe-" maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3. Jika dua karakter pertama bukan "di-", "ke-", "se-", "te-", "be-", "me-" atau "pe-" maka berhenti.
4. Jika tipe awalan adalah "none" maka berhenti. Hapus awalan jika ditemukan.

5. Langkah selanjutnya adalah *recording*. Tahapan ini dihentikan jika memenuhi beberapa kondisi seperti terdapat kombinasi awalan yang dihilangkan sebelumnya, tiga awalan telah dihilangkan. Proses *recording* dilakukan dengan menambah karakter *recording* di awal kata yang dipenggal dengan mengacu pada Tabel 2.2 dan Tabel 2.3.
6. Jika semua langkah telah selesai dilakukan tetapi kata dasar tersebut tidak ditemukan pada kamus, maka algoritma akan mengembalikan kata yang asli sebelum dilakukan *stemming*.

Contoh proses *stemming* dapat dilihat pada gambar 2.8.



Gambar 2.8 Proses *Stemming*

2.8 *Tokenizer*

Tokenizer adalah salah satu *class* yang disediakan oleh *library* Keras. *Tokenizer* ini berfungsi untuk mengkonversi teks menjadi urutan *integer* indeks kata atau vektor *binary*. Masing-masing kata akan dihitung jumlah kemunculannya kemudian diberi indeks integer sesuai jumlah urutan kemunculan terbanyak. Misalnya terdapat data teks : "pantai bersih sekali","lokasi sangat bersih", "bersih pantai indah". Maka hasil indeks masing-masing kata setelah melalui proses Tokenizer adalah : {'bersih':1,'pantai':2,'sekali':3,'lokasi':4,'sangat' :5,'indah':6}. Dari hasil indeks tersebut, maka sudah dapat dijadikan bahan acuan untuk mengubah teks baru menjadi *integer*. Contohnya ketika diberikan teks baru "pantai indah lokasi bersih" maka jika di ubah menjadi *integer* akan menghasilkan indeks : [2 6 4 1].

2.9 *Word Embedding*

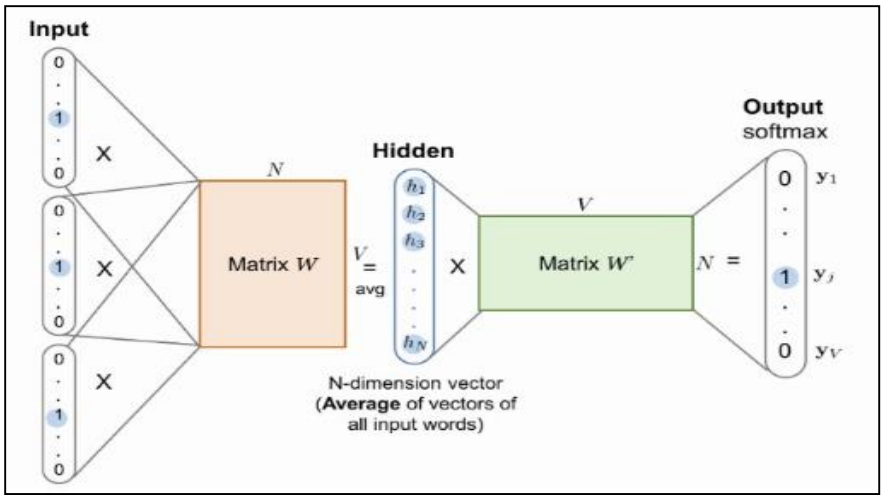
Word embedding adalah proses mengubah suatu kata menjadi sebuah vektor yang terdiri dari angka-angka. *Word embedding* biasa digunakan untuk algoritma *deep learning* karena algoritma *deep learning* tidak dapat memproses data dalam bentuk *string* atau teks.

Oleh karena itu harus mengkonversi ke dalam bentuk angka terlebih dahulu sebelum masuk pada proses *deep learning*. Contoh sederhana merubah kata menjadi vektor angka, misalnya terdapat kalimat "pantai bagus dan indah". Sebuah kamus akan berisi list selurut kata yang *unique*. Sehingga kamus yang terbentuk adalah: ["pantai","bagus","dan","indah"]. Menggunakan metode *one-hot encoding* akan menghasilkan vektor dimana 1 merepresentasikan posisi kata tersebut berada, sedangkan 0 untuk posisi kata lainnya. Sehingga hasil vektor representasi dari kata-kata pada contoh yaitu: kata "pantai" adalah [1,0,0,0], kata "bagus" adalah [0,1,0,0], kata "dan" adalah [0,0,1,0], dan kata "indah" adalah [0,0,0,1]. Selain metode *one-hot encoding*, masih ada metode lain untuk *word* embedding, salah satunya yaitu *word2vec*.

Word2vec merupakan metode yang digunakan untuk mentransformasikan teks menjadi suatu vektor angka. Proses *word embedding* dengan metode *word2vec* menggunakan konsep *neural network* yang memetakan kata ke variabel target. Sama halnya dengan *neural network*, proses yang dilakukan dengan metode *word2vec* menggunakan *weight* sebagai representasi vektor kata. *Word2vec* akan menangkap hubungan semantik atau sintatikal antar kata diruang vektor. Misalnya terdapat kata "kids" dan "child", keduanya memiliki kata yang berbeda namun memiliki hubungan konteks yang sama. Sehingga *word2vec* akan memberikan vektor embedding yang sama untuk kedua kata tersebut (Abdullah 2018). Pada penerapannya, *word2vec* terdiri dari dua teknik untuk melakukan proses *word embedding* yaitu *Continous Bag of Words* dan *Skip-Gram Model.*

2.9.1 *Continous Bag of Words*

Alur kerja dari teknik *Continous Bag of Word* (CBOW) dapat diilustrasikan seperti pada gambar 2.9.



Gambar 2.9 Ilustrasi Alur Kerja Teknik CBOW (Abdullah, 2018)

Misalnya ingin membuat model dari data training yang terdiri nilai input kata "king" dan "brave" maka hasilnya adalah "man". Alur kerja yang dilakukan untuk membuat model tersebut apabila menggunakan *continous bag of words* adalah sebagai berikut (Abdullah, 2018):

1. Lakukan proses konversi pada layer input dan layer output (target) menggunakan *one-hot encoding* sehingga vektor yang dihasilkan berukuran $[1\ x\ V]$. Dimana nilai V adalah banyaknya kata dari hasil *tokenizing* atau jumlah *vocabulary*. Misalnya pada contoh ini memiliki 6 kata maka matriks input berukuran $2[1\ x\ V]$ dimana $V = 6$ dan 2 adalah jumlah kata yaitu "king" dan "brave". Bentuk matriks input yang dihasilkan dapat dilihat pada persamaan 2.1.

$$king = (1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0)$$
$$brave = (0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0) \quad \text{.....(2.1)}$$

Sedangkan matriks dari *output layer* berukuran $1[1\ x\ V]$ dapat dilihat pada persamaan 2.2.

$$man = (0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0) \quad \text{.....(2.2)}$$

2. Langkah selanjutnya yaitu melakukan pembobotan. Proses pembobotan dilakukan dua set yaitu di antara *input layer* dengan *hidden layer* dan di antara *hidden layer* dengan *output layer*. Matriks *input hidden layer* berukuran $[V\ x\ N]$ sedangkan matriks *output*

hidden layer berukuran $[N\ x\ V]$, dimana nilai *N* merupakan jumlah *neuron*. Misalnya jumlah parameter *neuron*nya adalah 4 maka matriks *input hidden layer* yang terbentuk memiliki ukuran $[6\ x\ 4]$ seperti pada persamaan 2.3.

$$Input\ hidden\ layer = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 \end{pmatrix} \text{..... (2.3)}$$

Sedangkan ukuran matriks dari *output hidden layer* yang terbentuk adalah $[4\ x\ 6]$, dapat dilihat pada persamaan 2.4.

$$Output\ hidden\ layer = \begin{pmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 & 1.0 & 1.1 & 1.2 \\ 1.3 & 1.4 & 1.5 & 1.6 & 1.7 & 1.8 \\ 1.9 & 2.0 & 2.1 & 2.2 & 2.3 & 2.4 \end{pmatrix} \text{..... (2.4)}$$

3. Berikutnya *input layer* dikalikan dengan *weight* dari *input hidden layer*. Hasil dari proses ini disebut dengan *hidden activation*. Persamaan untuk menghasilkan *hidden activation* dapat dilihat pada persamaan 2.5. Pada iterasi pertama, bobot akan diberikan secara random.

$$Hidden\ activation = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} \text{.....(2.5)}$$

Kemudian, cari nilai rata-rata dari vektor *hidden activation*. Hasil rata-rata *hidden activation* dapat dilihat pada persamaan 2.6.

$$Avg\ hidden\ activation = \begin{pmatrix} 3 & 4 & 5 & 6 \end{pmatrix} \text{..... (2.6)}$$

4. Langkah berikutnya adalah melakuan perkalian antara *hidden activation* dengan *hidden output weight* untuk memperoleh *output*. Proses perkalian dapat dilihat pada persamaan 2.7.

$$output = \begin{pmatrix} 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 & 1.0 & 1.1 & 1.2 \\ 1.3 & 1.4 & 1.5 & 1.6 & 1.7 & 1.8 \\ 1.9 & 2.0 & 2.1 & 2.2 & 2.3 & 2.4 \end{pmatrix}$$

$$output = \begin{pmatrix} 21 & 22.8 & 24.6 & 26.4 & 28.2 & 30 \end{pmatrix} \ldots\ldots (2.7)$$

5. Setelah mendapat nilai *output* pada langkah sebelumnya, selanjutnya *output* akan ditransformasikan menggunakan fungsi $softmax$ untuk mendapatkan nilai probabilitas. Berikut hasil probabilitas dari *output* dapat dilihat pada persamaan 2.8.

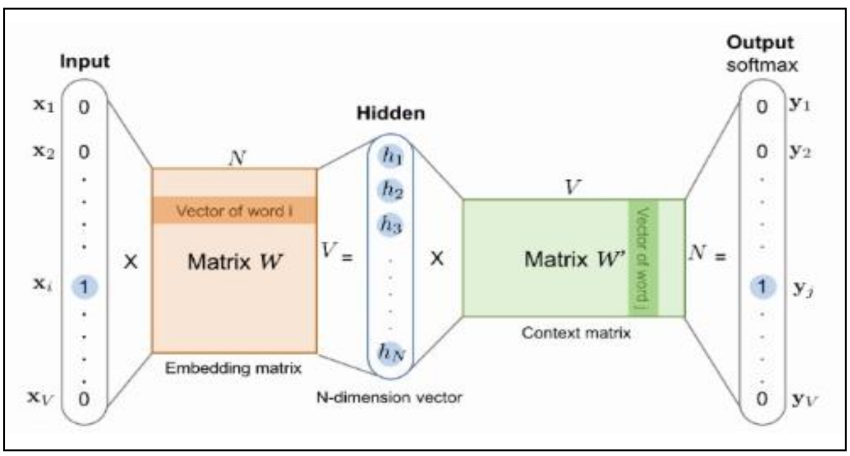
$$output\ softmax = \begin{pmatrix} 0.0001030124 \\ 0.0006231887 \\ 0.003770072 \\ 0.02280761 \\ 0.137978 \\ 0.8347181 \end{pmatrix} \ldots\ldots (2.8)$$

6. Selanjutnya menghitung nilai *error* antara *output* dan target kata. Kemudian dilakukan *backpropagation* untuk *re-adjust weight*nya guna memperbaiki nilainya. Cara menghitung yaitu matriks target dikurang matriks *output* seperti pada persamaan 2.9.

$$Error = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 0.0001030124 \\ 0.0006231887 \\ 0.003770072 \\ 0.02280761 \\ 0.137978 \\ 0.8347181 \end{pmatrix} = \begin{pmatrix} -0.0001030124 \\ -0.0006231887 \\ 0.9962299 \\ -0.02280761 \\ -0.137978 \\ -0.8347181 \end{pmatrix} \ldots\ldots (2.9)$$

2.9.2 *Skip-Gram Model*

Alur kerja dari teknik *skip-gram model* dapat diilustrasikan seperti pada gambar 2.10.



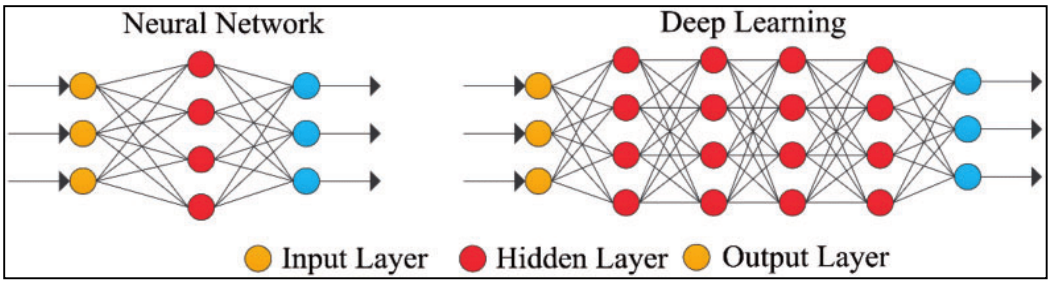
Gambar 2.10 Ilustrasi Alur Kerja Teknik Skip-Gram Model (Abdullah, 2018)

Dengan menggunakan contoh model sebelumnya yaitu input kata “king” dan “brave”, alur kerja yang dilakukan apabila menggunakan *skip-gram model* adalah sebagai berikut (Abdullah, 2018):

1. Langkah pertama yaitu mentransformasikan nilai input menjadi berukuran $[1\ x\ V]$, matriks *input hidden weight* berukuran $[V\ x\ N]$ dimana N merupakan jumlah *neuron* dalam *hidden layer* dan ukuran matriks dari *hidden output* adalah $C[1\ x\ V]$ dengan C adalah jumlah konteks katanya.
2. Kemudian nilai matriks dari *hidden activation* dikalikan dengan *weight* antara *hidden layer* dan *output layer* untuk menghasilkan *output* dari prediksi.
3. Hasil dari proses sebelumnya berupa nilai *output* ditransformasikan dengan fungsi $softmax$ untuk mendapatkan nilai probabilitas.
4. Hitung *error* antara nilai *output* dengan target. Kemudian dilakukan proses *backpropagation* untuk *re-adjust weight*nya.

2.10 *Deep Learning*

Deep learning merupakan salah satu bidang dari pembelajaran mesin (*machine learning*) yang berbasis jaringan syaraf tiruan. *Deep learning* adalah sebuah pendekatan dalam penyelesaian masalah pada sistem pembelajaran komputer yang menggunakan konsep hirarki (Goodfellow dkk, 2016). Hirarki pada deep learning terdiri dari banyaknya struktur dan jumlah jaringan syaraf pada algoritmanya hingga ratusan lapisan yang setiap lapisannya memiliki tanggung jawabnya masing-masing. *Deep learning* adalah teknik dalam *neural network* yang menggunakan teknik tertentu untuk mempercepat proses pembelajaran dalam *neural network* yang menggunakan jumlah lapis yang banyak. Dengan adanya *deep learning*, waktu yang dibutuhkan untuk training akan semakin sedikit karena masalah hilangnya gradien pada propagasi balik akan semakin rendah (Ahmad, 2017). Untuk mengetahui gambaran diagram *deep learning*, dapat dilihat pada gambar 2.11.



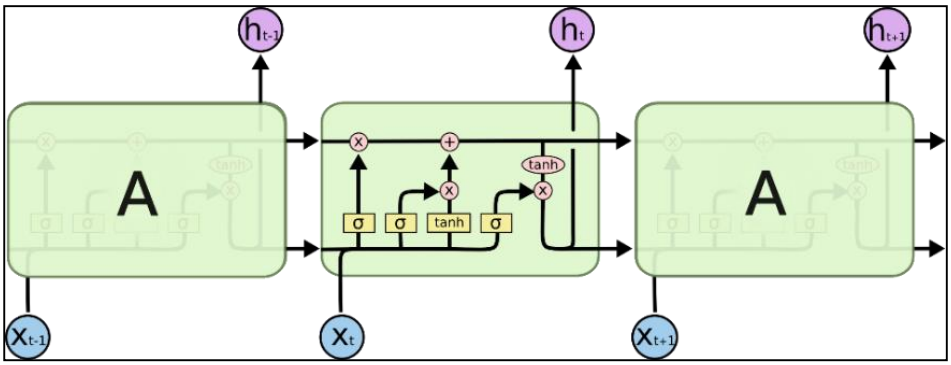
Gambar 2.11 Diagram *Deep Learning* (Xing dan Du, 2018)

Kelebihan *deep learning* yaitu memiliki *feature engineering* yang berfungsi untuk mengekstrak pola yang penting dari suatu data sehingga dapat memudahkan model untuk membedakan kelas. Model merupakan hasil dari proses pelatihan pada *deep learning* yang nantinya akan digunakan untuk melakukan prediksi. Kelebihan lain dari *deep learning* adalah mampu mengubah data dari *non-linieraly separable* menjadi *linearly separable* melalui serangkaian transformasi (*hidden layer*). Selain itu, *deep learning* juga mampu menangani berbagai permasalahan dengan data berskala besar seperti *computer vision, speech recognition* dan *natural language processing* (Putra, 2019).

2.11 *Long Short-Term Memory*

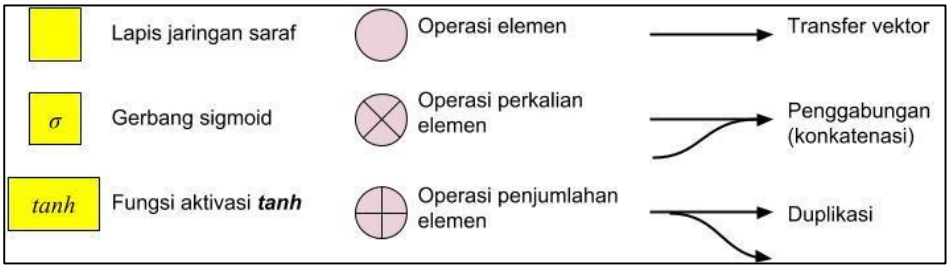
Long Short Term Memory (LSTM) adalah salah satu algoritma *deep learning* yang dapat digunakan dalam pemrosesan data *sequential*. LSTM merupakan sebuah evolusi dari arsitektur *Recurrent Neural Network* (RNN). RNN tidak dapat untuk belajar menghubungkan informasi karena memori lama yang tersimpan akan semakin tidak berguna dengan seiringnya waktu berjalan karena tertimpa atau tergantikan dengan memori baru, permasalahan ini ditemukan oleh Bengio, et al. (1994). Berbeda dengan RNN, LSTM tidak memiliki kekurangan tersebut karena LSTM dapat mengatur memori pada setiap masukannya dengan menggunakan *memory cells* dan *gate units* (Nurrohmat dan SN, 2019).

Seperti RNN, LSTM juga memiliki struktur jaringan berantai namun dengan struktur yang berbeda. Jika dalam RNN hanya menggunakan satu lapisan tunggal, maka dalam LSTM terdiri dari empat lapisan yang saling berkaitan seperti pada gambar 2.12.



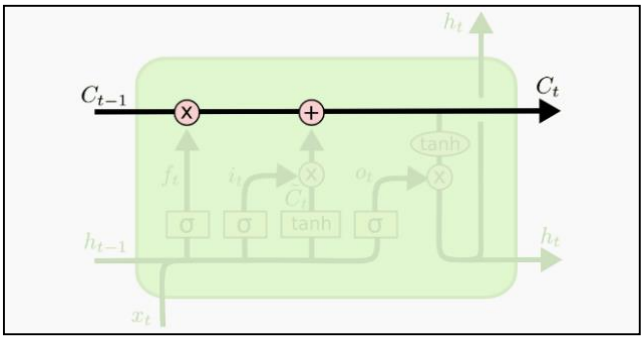
Gambar 2.12 Struktur Jaringan LSTM (Olah, 2015)

Pada gambar 2.12, terdapat beberapa notasi yang digunakan dalam membangun jaringan LSTM. Notasi tersebut dijelaskan pada gambar 2.13.



Gambar 2.13 Notasi pada Jaringan LSTM

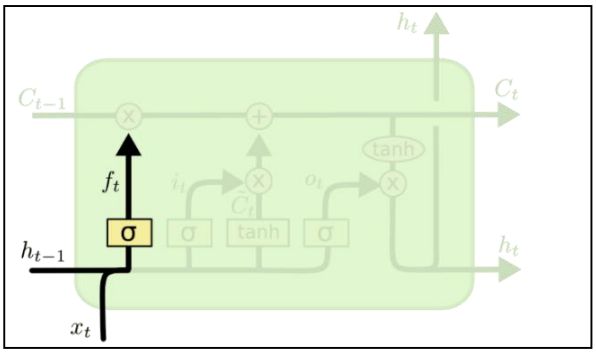
Jaringan LSTM terdiri dari blok memori yang disebut dengan sel. Terdapat dua jalur yang digunakan untuk menyampaikan informasi ke sel selanjutnya, yaitu melalui *cell state* dan *hidden state*. Kunci utama dalam LSTM adalah *cell state*, yaitu garis aliran utama yang menghubungkan semua *gate*. Data yang dibawa oleh *cell state* dapat ditambahkan maupun dihapus melalui gerbang *sigmoid*. Struktur *cell state* pada LSTM dapat dilihat pada gambar 2.14.



Gambar 2.14 *Cell State* pada LSTM (Olah, 2015)

Terdapat tiga jenis unit gerbang atau *gate* berbeda yang digunakan dalam LSTM, yaitu *input gate*, *forget gate*, dan *output gate*. *Input gate* berfungsi untuk menentukan sebuah masukan akan ditambahkan ke dalam *memory cell state* saat itu atau tidak. *Forget gate* berguna untuk menentukan sebuah memori pada waktu sebelumnya harus dilupakan atau tidak. Sedangkan *output gate* berguna untuk menentukan seberapa besar pengaruh memori *cell state* terhadap hasil prediksi yang akan dihasilkan (Habibie, 2018).

Langkah pertama jalannya metode LSTM adalah memutuskan informasi yang akan dihapus dari *cell state*. Keputusan ini dibuat oleh *sigmoid layer* yang bernama *forget gate*. *Forget gate* akan memproses h_{t-1} dan x_t sebagai input, lalu menghasilkan keluaran berupa angka 0 atau 1 pada *cell state* C_{t-1} seperti pada gambar 2.15.



Gambar 2.15 *Forget gate* (Olah, 2015)

Persamaan *forget gate* diuraikan pada persamaan 2.10.

$$f_t = \sigma\left(x_t W_f + h_{t-1} U_f + b_f\right) \quad \text{.....(2.10)}$$

Keterangan:

f_t : *forget gate*

σ : fungsi *sigmoid*

W_f : *weight* pada *forget gate*

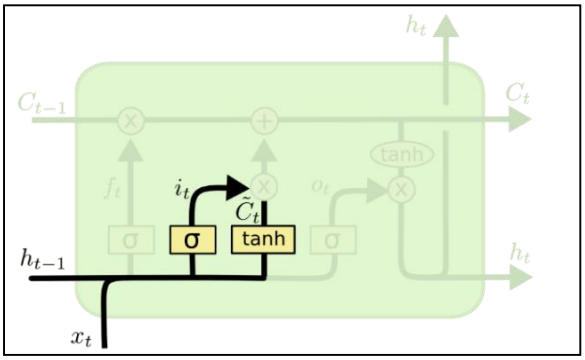
U_f : *reccurent weight* pada *forget gate*

h_{t-1} : nilai keluaran sebelum orde ke t

x_t : nilai *input* pada orde ke t

b_f : nilai bias pada *forget gate*

Langkah kedua adalah memutuskan informasi yang akan disimpan pada *cell state*. Terdapat dua bagian pada proses ini, pertama sigmoid *layer* yang bernama *input gate* memutuskan nilai mana yang akan diperbaharui. Selanjutnya, *tanh layer* membuat satu kandidat konteks dengan nilai baru yang akan ditambahkan ke *cell state*. Langkah selanjutnya adalah *output* dari *input gate layer* dan *tanh layer* akan digabungkan untuk memperbarui *cell state*. Proses untuk langkah kedua yaitu *input gate* dan kandidat konteks digambarkan pada gambar 2.16.



Gambar 2.16 *Input gate* dan kandidat nilai baru (Olah, 2015)

Terdapat dua operasi dalam gambar 2.16, operasi pertama yaitu perhitungan *input gate* dan penentuan kandidat konteks yang akan ditambahkan pada *cell state* yang ditunjukkan pada persamaan 2.11 dan 2.12.

$$i_t = \sigma\left(x_t W_i + h_{t-1} U_i + b_i\right) \quad \text{.....(2.11)}$$

$$\check{C}_t = tanh\left(x_t W_c + h_{t-1} U_c + b_c\right) \quad \text{.....(2.12)}$$

Keterangan:

i_t : *input gate*

$\check{C}_t$: kandidat konteks

σ : fungsi *sigmoid*

W_f : *weight* pada *input gate*

U_f : *reccurent weight* pada *input gate*

b_i : nilai bias pada *input gate*

W_c : *weight* pada kandidat konteks

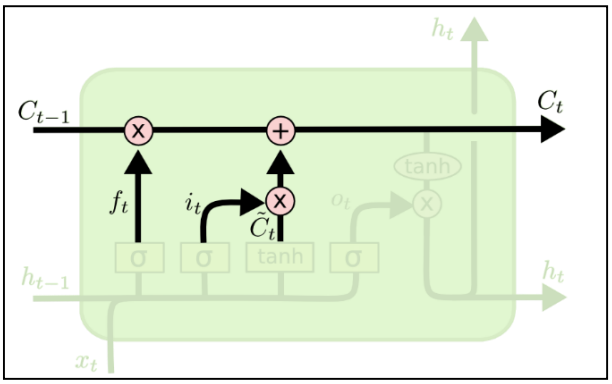
U_c : *reccurent weight* pada kandidat konteks

b_c : nilai bias pada kandidat konteks

h_{t-1} : nilai keluaran sebelum orde ke t

x_t : nilai *input* pada orde ke t

Langkah ketiga adalah memperbarui *cell state* yang lama menjadi *cell state* yang baru. Setelah nilai *input gate* dan kandidat konteks ditemukan, terjadi mekanisme perkalian yang ditunjukkan dengan notasi operasi perkalian elemen. Kedua nilai yang telah dihasilkan sebelumnya kemudian akan dikalikan untuk memperbarui *cell state*. Memori baru ini kemudian akan ditambahkan ke memori lama yang dilambangkan dengan C_{t-1} dan akan menghasilkan memori yang baru dengan lambang C_t. Proses tersebut diuraikan dalam gambar 2.17.



Gambar 2.17 Proses memperbarui *memory cell* (Olah, 2015)

Berdasarkan gambar 2.17, persamaan yang terjadi dalam proses memperbarui *memory cell state* dituliskan pada persamaan 2.13.

$C_t = f_t * C_{t-1} + i_t * \check{C}_t$ ..... (2.13)

Keterangan:

C_t : *Cell state*

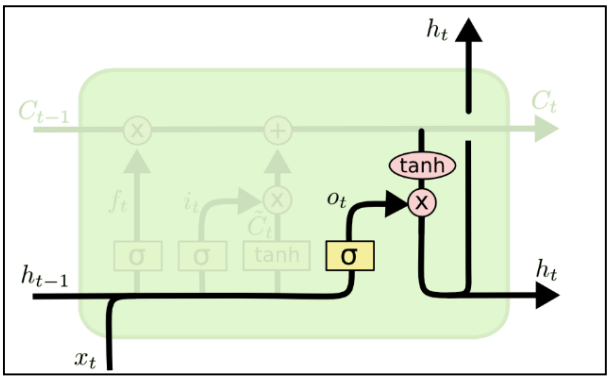
f_t : *forget state*

C_{t-1} : *Cell state* sebelum orde ke t

i_t : nilai input pada orde ke t

$\check{C}_t$: kandidat konteks

Langkah terakhir adalah memutuskan hasil *output*. Pertama, lapisan *sigmoid* memutuskan bagian sel mana yang menjadi *output*. Selanjutnya, *output* dari gerbang *sigmoid* (O_t) dikalikan dengan nilai baru yang dihasilkan oleh lapisan *tanh* dari *cell state* (C_t), nilai yang dihasilkan berada antara -1 sampai 1. Proses ini dijelaskan pada gambar 2.18.



Gambar 2.18 *Output gate* dan *Hidden state* (Olah, 2015)

Berdasarkan gambar 2.18, maka proses yang terjadi dapat dijelaskan dalam persamaan 2.14 dan 2.15.

$$O_t = \sigma\left(x_t W_o + h_{t-1} U_o + b_o\right) \quad \text{.....(2.14)}$$

$$h_t = O_t tanh(C_t) \quad \text{.....(2.15)}$$

Keterangan:

O_t : *output gate*

σ : fungsi sigmoid

W_o : nilai *weight* untuk *output gate*

U_o : *reccurent weight* pada *output gate*

h_{t-1} : nilai keluaran sebelum orde ke t

x_t : nilai input pada orde ke t

b_o : nilai bias pada *output gate*

C_t : *Cell state*

Beberapa fungsi aktivasi yang digunakan pada proses jaringan LSTM, yaitu *sigmoid* dan *tanh* yang diuraikan pada persamaan 2.16 dan 2.17.

$\sigma(\mathrm{x}) = \frac{1}{1+e^{-x}}$ ..... (2.16)

$tanh(\mathrm{x}) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ ..... (2.17)

Keterangan:

σ = fungsi *sigmoid*

x = data *input*

e = fungsi eksponensial

2.12 Validasi dan Pengujian

Validasi dan pengujian diterapkan untuk mengukur kinerja sebuah sistem, khusus dalam penelitian ini digunakan untuk mengukur keakuratan metode pengklasifikasian dokumen teks. Validasi yang dilakukan menggunakan *K-Fold Cross Validation*, sedangkan pengujian menggunakan tabel *Confusion Matrix* dan kurva *Receiver Operating Characteristic* (ROC).

2.12.1 K-*Fold Cross Validation*

Validasi pada penelitian ini menggunakan *k-fold cross validation* merupakan salah satu metode pengujian dalam pembelajaran mesin. Pengujian pada *k-fold cross validation*, dataset dibagi menjadi beberapa bagian atau *fold* secara acak. Bagian pertama pada dari hasil tersebut digunakan sebagai data pengujian dan sisanya digunakan sebagai data pelatihan. Hasil pengujian berupa nilai rata-rata dari semua hasil pengujian yang dilakukan (Rampeng, 2018). K-*fold cross validation* sering digunakan dalam pengujian data karena mampu memberikan perkiraan kesalahan dari klasifikasi yang terbaik. Pembagian data dalam k-*fold*

cross validation bersifat random, sehingga pengujian diperlukan berkali-kali untuk menghasilkan data uji terbaik.

2.12.2 Tabel *Confusion Matrix*

Teknik pengujian yang diterapkan pada penelitian ini adalah akurasi, presisi, dan *recall*. Sehingga dibutuhkan sebuah *matrix* yang akan menghitung empat kemungkunan yang terjadi dalam proses klasifikasi suatu baris data (Luhrie, 2019). Keempat kemungkinan tersebut akan direpresentasikan ke dalam bentuk matriks 2x2 yang disebut *confusion matrix*. Menurut Han dan Kambar (2006) *confusion matrix* merupakan alat yang digunakan untuk menganalisis seberapa baik *classifier* mengenali *tuple* dari kelas yang berbeda. TP dan TN memberikan informasi ketika *classifier* benar, sedangkan FP dan FN memberitahu ketika *classifier* salah. *Confusion matrix* adalah sebuah tabel yang menyatakan tingkat kenenaran proses klasifikasi. Tabel *confusion matrix* dapat dilihat pada Tabel 2.4.

Tabel 2.4 Tabel Confusion Matrix

		True Class	
		Positive	Negative
Predicted Class	Positive	true positives count (TP)	false negatives count (FP)
	Negative	false positives count (FP)	true negatives count (TN)

Keterangan:

TP atau *True Positive* : jumlah prediksi yang benar bahwa yang diprediksi hasilnya positif.

FP atau *False Positif* : jumlah prediksi yang salah, yang seharusnya nilai negatif diprediksi positif

TN atau *True Negative* : jumlah prediksi yang benar bahwa hasil prediksi bernilai negative.

FN atau *False Negative* : jumlah prediksi yang salah, harusnya nilai positif diprediksi negatif.

Berikut perhitungan akurasi, presisi dan *recall* menurut Kamber dan Han (2006) :

a. Akurasi

Akurasi merupakan nilai yang merepresentasikan tingkat kedekatan antara nilai prediksi dengan nilai aktual. Perhitungan akurasi dilakukan dengan cara membagi jumlah data yang diklasifikasikan secara benar dengan total data *testing* yang diuji. Rumus perhitungan nilai akurasi dapat dilihat pada persamaan 2.18.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \text{ X } 100\% \quad \text{.....(2.18)}$$

b. Presisi

Presisi merupakan proporsi jumlah dokumen teks yang relevan terkenali diantara semua dokumen teks yang dipilih oleh terpilih oleh sistem. Nilai presisi menggambarkan jumlah data kategori positif yang diklasifikasikan secara benar dibagi dengan jumlah data yang diklasifikasikan positif. Untuk menghitung presisi dapat dilihat pada persamaan 2.19.

$$Precision = \frac{TP}{TP+FP} \text{ X } 100\% \quad \text{.....(2.19)}$$

c. Recall

Recall merupakan tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Rumus perhitungan *recall* dapat dilihat pada persamaan 2.20.

$$Recall = \frac{TP}{TP+FN} \text{ X } 100\% \quad \text{.....(2.20)}$$

2.12.3 Kurva *Receiver Operating Characteristic* (ROC)

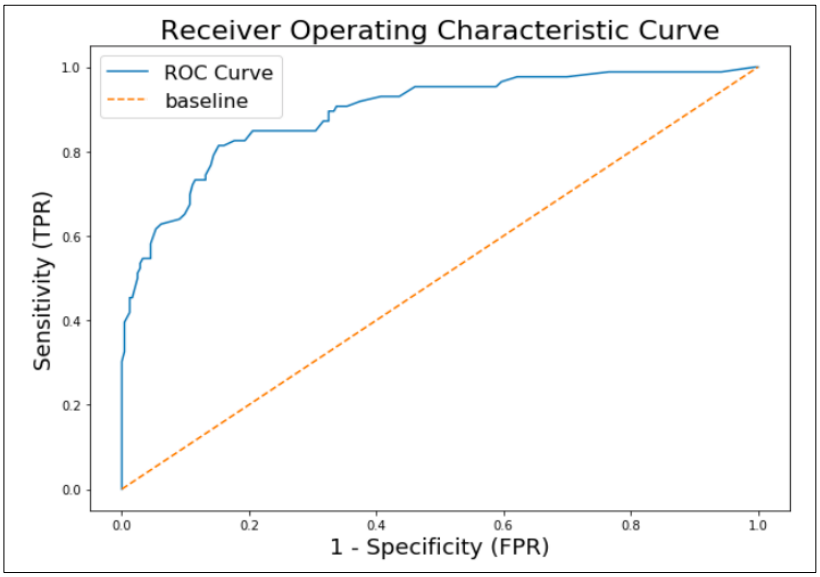
Kurva ROC digunakan untuk memvisualisasi dan menguji kinerja pengklasifikasian berdasarkan performanya (Gorunescu, 2011). Kurva ROC diplot pada grafik dengan nilai *True Positive Rate* (sensitivitas) pada sumbu Y dan nilai *False Positive Rate* (1 - sensitivitas) pada sumbu X (Chazhoor, 2019). Nilai-nilai TPR dan FPR ditemukan untuk banyak ambang batas dari 0 hingga 1. Nilai-nilai yang didapatkan kemudian diplot pada kurva. Persamaan untuk menghitung nilai FPR dapat dilihat pada persamaan 2.21.

$$TPR\ (Sensitivitas) = \frac{TP}{TP+FN} \quad \text{.....(2.21)}$$

Sedangkan persamaan menghitung TPR dapat dilihat pada persamaan 2.22.

$$FPR\ (1 - Spesifisitas) = \frac{TN}{TN+FP} \quad \text{.....(2.22)}$$

Contoh kurva ROC dapat dilihat pada gambar 2.19.



Gambar 2.19 Contoh Kurva ROC (Chazhoor, 2019)

Akurasi klasifikasi ROC dilakukan dengan cara menghitung luas daerah di bawah kurva ROC yang disebut dengan *Area Under Curve* (AUC). Rumus untuk mencari nilai AUC dapat dilihat pada persamaan 2.29 (Suwarno dan Abdillah, 2016).

$$AUC = \frac{1}{2}\sum_{i=1}^{n}(x_{i+1} - x_i)(y_{i+1} - y_i) \quad \text{.....(2.10)}$$

Kriteria keakuratan pengujian menggunakan AUC disajikan pada tabel 2.5.

Tabel 2.5 Tabel kriteria AUC (Suwarno dan Abdillah, 2016)

Nilai AUC	Interpretasi
0.90 – 1.00	*Excellent classification*
0.80 – 0.90	*Good classification*
0.70 – 0.80	*Fair classification*
0.60 – 0.70	*Poor classification*
0.50 – 0.60	*Failure*

2.13 Studi Pustaka (*State of the Art*)

Kealsian penelitian yang dilakukan berdasarkan pada penelitian terdahulu yang mempunyai karakteristik relatif sama dalam hal tema, objek, dan metode yang digunakan. Penelitian yang berkaitan dengan rekomendasi tempat wisata pernah dilakukan dengan

sistem pendukung keputusan menggunakan metode *Fuzzy Tahani* dengan hasil rekomendasi objek wisata sesuai dengan kriteria yang dipilih. Dalam penelitian tersebut, pengguna memasukkan beberapa kriteria yang nantinya akan diklasifikasi lalu menghasilkan sebuah rekomendasi objek wisata (Purnomo, 2013). Selain itu, ada penelitian lain yang memberikan rekomendasi tempat wisata dengan cara analisis sentimen menggunakan metode *Naive Bayes Classifier*. Penelitian tersebut mengklasifikasikan opini berupa komentar ke dalam dua kelas yaitu positif dan negatif dengan akurat (Wilianto dkk, 2017).

Penelitian yang berkaitan dengan rekomendasi tempat wisata juga pernah dilakukan dengan analisis sentimen melalui pendekatan *sentiment lexicon* dengan tingkat akurasi 93,8% (Kurniawati, 2016). Penerapan analisis sentimen juga pernah dilakukan dengan beberapa metode seperti metode K-NN dengan tingkat akurasi klasifikasi sentimen mencapai 83,33% dalam analisis *tweet* (Zuhdi dkk, 2019). Ada juga metode *Deep Belief Network* dengan akurasi sebesar 93,31% dan metode *Support Vector Machine* dengan akurasi sebesar 92,18% dalam analisis sentimen *tweet* berbahasa Indonesia (Zulfa & Winarko, 2017). Metode *Ontology Supported Polarity Mining* (OSPM) juga pernah diterapkan dalam penelitian tentang analisis sentimen komentar di *website TripAdvisor* dengan tingkat akurasi sebesar 87% (Koesumaningrum, 2018).

Adapun beberapa penelitian yang telah dilakukan dan dijadikan referensi dalam penelitian ini dapat dilihat pada Tabel 2.6 dan Tabel 2.7.

Tabel 2.6 Studi Pustaka

No	Penulis	Judul	Intisari
1	Oni Harnantyo (2019)	Analisis Sentimen Tempat Wisata di Ypgyakarta Menggunakan Metode *Recurrent Neural Network* dengan *Algoritma Long Short-Term Memory*	Tujuan penelitian ini adalah untuk mengklasifikasikan tweet berbahasa Indonesia menjadi 3 kelas yaitu positif, negatif, dan netral. Tingkat akurasi pelatihan yang didapatkan sebesar 99,31%.
2	Oman Somantri, Dairoh (2019)	Analisis Sentimen Penilaian Tempat Tujuan Wisata Kota Tegal Berbasis Text Mining	Tujuan penelitian tersebut adalah mencari model sistem untuk memberikan sebuah informasi pendukung keputusan bagi para wisatawan dan pengelola tempat wisata untuk dijadikan sumber informasi terhadap tempat wisata yang ada. Hasil penelitian menunjukkan bahwa model yang didapatkan setelah dilakukan eksperimen didapatkan tingkat akurasi *naive bayes* sebesar 77,50% lebih baik dibandingkan dengan menggunakan *decision tree* yang menghasilkan tingkat akurasi 60,83%.
3	Muh Amin Nurrohmat, Azhari SN (2019)	*Sentiment Analysis of Novel Review Using Long Short-Term Memory Method*	Penelitian ini bertujuan untuk melakukan pengklasifikasian terhadap review novel berbahasa Indonesia berdasarkan sentimen positif, netral dan negatif dengan menggunakan metode *Long Short-Term Memory* (LSTM). Dataset yang digunakan adalah data review novel berbahasa Indonesia yang diambil dari situs goodreads.com. Hasil pengujian memperlihatkan bahwa metode Long Short-Term Memory memiliki hasil akurasi yang lebih baik dibandingkan dengan metode naïve bayes dengan nilai akurasi 72.85%, dibandingkan dengan hasil akurasi metode *Naïve Bayes* dengan nilai akurasi 67.88%.
4	Dessy Koesumaningrum (2018)	Analisis Sentimen Ulasan TripAdvisor pada Tem[at Wisata menggunakan *Ontology Supported Polarity Mining* (OSPM) (Studi Kasus Bandung)	OSPM digunakan untuk pengklasifikasian ulasan pengunjung kedalam beberapa aspek tertentu berserta dengan penilaian pada setiap aspek tersebut.
5	Shalfa Fitriga Luhrie	Klasifikasi Informasi dan Keluhan Masyarakat Menggunakan Algoritma Fuzzy K-Nearest Neighbor	Tujuan penelitian tersebut adalah mengetahui performa dari algoritma Fuzzy K-Nearest Neighbor dalam mengklasifikasikan informasi dan keluhan masyarakat. Hasil pengujian menunjukkan nilai akurasi optimum untuk klasifikasi bidang sebesar 96,528% dengan K=11 pada persentase data 80 (train) : 20 (test) dan untuk klasifikasi unit kerja sebesar 97,375% dengan K=9 pada persentase data 70 (train) : 30 (test). Sedangkan nilai presisi dan recall optimum untuk klasifikasi bidang sebesar 56,597% dengan K=11 pada persentase data 80 (train) : 20 (test). Sedangkan untuk klasifikasi unit kerja sebesar 58,052% dengan K=5 pada persentase data 90 (train) : 10 (test).

Tabel 2.7 Lanjutan Studi Pustaka

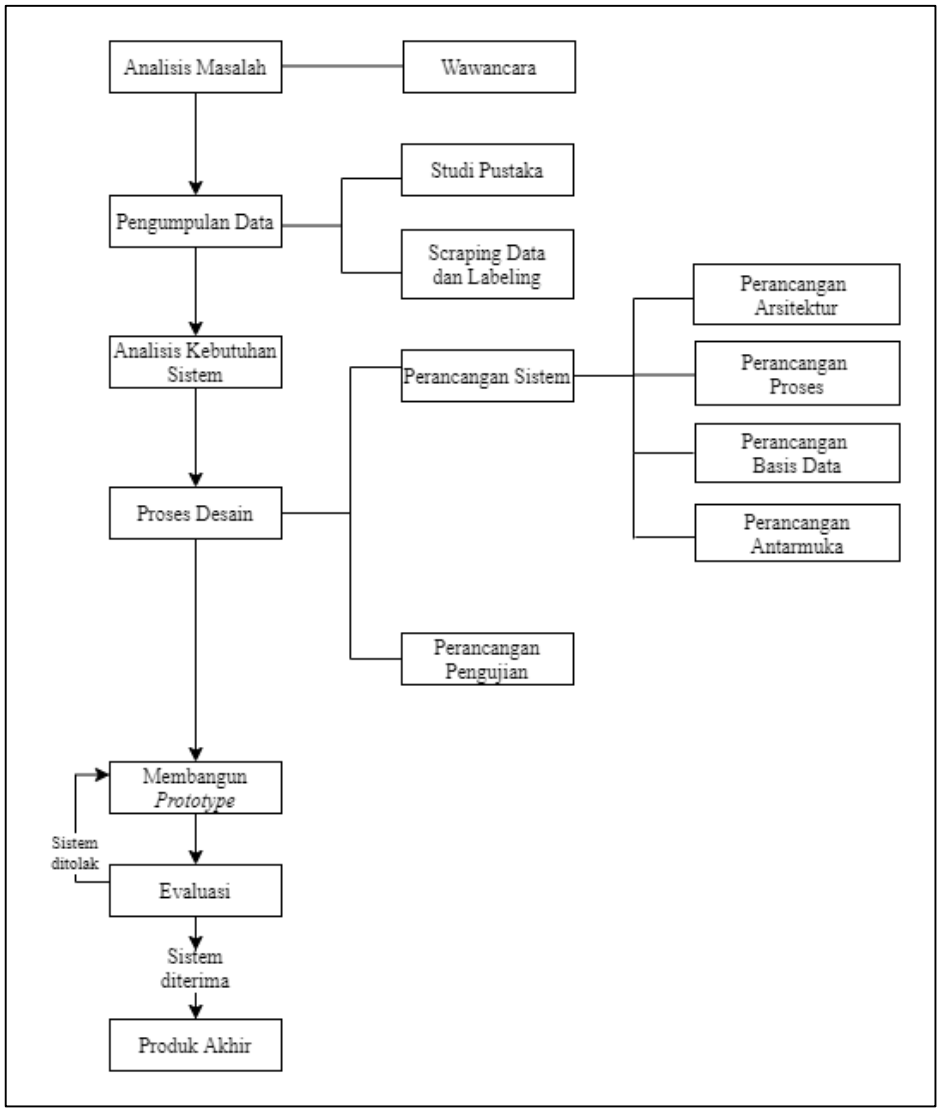
6	Debby Gybson Putri Deri Desya Rampeng (2018)	Penerapan Analisis Sentimen pada Media Sosial menggunakan Metode Lexicon Based dan Support Vector Machine (SVM) sebagai Rekomendasi Oleh-Oleh Favorit	Tujuan penelitian tersebut adalah membuat sistem penerapan analisis sentimen pada media sosial Twitter dan Instagram untuk mengetahui oleh-oleh mana yang memiliki review positif yang paling banyak dari masyarakat atau para wisatawan yang pernah membeli oleh-oleh di Kota Yogyakarta. Adapun hasil akurasi terbesar adalah menggunakan metode Lexicon Based sebesar 87,78% sedangkan akurasi menggunakan metode Support Vector Machine sebesar 86%.
7	Ibnu Habibie (2018)	Identifikasi Judul Berita Clickbait Berbahasa Indonesia dengan *Algoritma Long Short-Term Memory* (LSTM) *Recurrent Neural Network*	Tujuan penelitian ini adalah untuk mengklasifikasikan judul berita berbahasa Indonesia menjadi 2 kelas yaitu clickbait, dan nonclickbait. Tingkat akurasi model yang dihasilkan sebesar 82%.
8	Lio Wilianto, Tacbir Hendro Pudjiantoro, Fajri Rakhmat Umbara (2017)	Analisis Sentimen Terhadap Tempat Wisata dari Komentar Pengunjung dengan Menggunakan Metode *Naive Bayes Classifier* Studi Kasus Jawa Barat	Tujuan penelitian tersebut adalah untuk memberikan informasi tentang kualitas sebuah tempat wisata yanga da di Jawa Barat. Sumber data berasal dari komentar pengunjung dan rating tempat di *google maps*. Dari hasil penelitian yang telah dilakukan terlihat bahwa algoritma *naive bayes classifier* dapat mengklasifikasikan komentar ke dalam dua kelas, yaitu positif dan negatif dengan akurat.
9	Zaiying Wang, Bahao Song (2019)	*Research on Hot News Classification Algorithm based on Deep Learning*	LSTM efektif mengatasi urutan informasi teks yang panjang. Dibandingkan dengan pembelajaran mesin tradisional dan CNN, RNN memiliki akurasi yang lebih tinggi. Metode pembelajaran yang mendalam memberikan kerangka kerja yang fleksibel untuk menghasilkan klasifikasi yang lebih baik. (Wang & Song, 2019)
10	Ali Alwchaibi, Kaushik Roy (2018)	*Comparison of Pre-Trained Word Vectors for A Arabic Text Classification using Deep learning Approach*	Penambahan word embedding membuat analisis sentimen teks bahasa arab menjadi lebih baik. LSTM lebih efektif untuk klasifikasi teks dibandingkan dengan CNN dan RNN konvensional. Penelitian terkini menunjukkan bahwa LSTM berkinerja relatif baik di sebagian besar NLP (Alwehaibi, 2018)

BAB III

METODOLOGI PENELITIAN DAN PENGEMBANGAN SISTEM

3.1 Metodologi Penelitian

Dalam bagian ini akan membahas tentang metodologi penelitian dan metodologi pengembangan sistem yang digunakan dalam penelitian ini. Metodologi penelitian berdasarkan data dan memanfaatkan teori pustaka untuk mendukung penelitian dan sebagai penguat argumentasi. Instrumen yang digunakan dalam penelitian ini adalah wawancara lisan terhadap pihak terkait. Adapun tahapan metodologi penelitian yang akan dilakukan dapat dilihat pada gambar 3.1.



Gambar 3.1 Metodologi Penelitian

Sedangkan metode pengembangan sistem yang digunakan pada penelitian ini adalah metode *prototype*. Memilih metode *prototype* karena metode ini lebih menekankan pada komunikasi antara pengembang dan pelanggan sehingga kebutuhan pengguna dapat diterjemahkan dalam bentuk model *prototype*. Keunggulan lain dari model *prototype* adalah lebih efisien untuk diterapkan pada sistem perangkat lunak yang bersifat *custemize* yang bertujuan untuk mengimplementasikan algoritma tertentu pada suatu kasus (Aldhiansyah, 2020). Metode *prototype* terdiri dari empat tahapan yaitu pengumpulan kebutuhan, proses desain, membangun *prototype* serta evaluasi dan perbaikan (Purnomo, 2017).

Pada tahapan pengumpulan kebutuhan dilakukan pengumpulan data yang bersumber dari studi pustaka, dan *web scraping*. Selanjutnya dilanjutkan dengan analisa dari data tersebut serta analisa kebutuhan sistem. Kemudian pada proses desain akan dibuat rancangan sistem seperti rancangan arsitektur, rancangan basis data, rancangan antarmuka, dan rancangan pengujian. Setelah itu membangun *prototype* berdasarkan rancangan yang dibuat dengan penulisan program atau *coding*. *Prototype* yang dihasilkan akan diuji untuk mengetahui performa dari algoritma *long-short term memory* dalam menangani permasahan analisis sentimen. Terakhir akan dievaluasi dan dilakukan perbaikan sesuai keinginan pelanggan.

3.2 Analisis Masalah

Sebelumnya telah dilakukan kegiatan wawancara terhadap Ibu Ellya Shari, S.ST.Par, MM selaku Kepala Seksi Sarana dan Prasarana Bidang Destinasi Wisata Dinas Pariwisata Provinsi DIY pada tanggal 26 Februari 2020 dan 26 Agustus 2020 terkait peran pemerintah dalam melakukan monitoring tempat wisata, khususnya pantai. Dari hasil wawancara, didapatkan hasil bahwa dalam memonitoring guna mengetahui evaluasi dari kondisi obyek wisata di Provinsi DIY dilakukan oleh dua lembaga terkait, yaitu Dinas Pariwisata Provinsi DIY dan Dinas Pariwisata Kabupaten/Kota. Dinas Pariwisata DIY berfungsi sebagai pusat

pengelolaan yang bertugas mengawasi dan memantau kondisi tempat wisata melalui Dinas Pariwisata Kabupaten/Kota yang mengelola secara langsung di daerah masing-masing. Namun, pemerintah Dinas Pariwisata DIY tidak hanya mengandalkan laporan dari Dinas Pariwisata tingkat kabupaten/kota saja, melainkan ikut terjun langsung ke lokasi guna memastikan kondisi tempat wisata yang sebenarnya. Monitoring oleh pemerintah provinsi dilakukan sekitar sebulan sekali dengan mengambil beberapa sampel secara random lokasi wisata mana saja yang akan di kunjungi. Dalam pelaksanaan monitoring tersebut, jumlah sampel dan rentang waktu setiap kunjungan tidak ditentukan secara pasti, melainkan kondisional sesuai kebutuhan. Sehingga cara yang selama ini diterapkan dinilai masih kurang efektif. Ibu Ellya Shari juga mengatakan bahwa selama ini, Pemerintah Dinas Pariwisata DIY masih terfokus dengan monitoring secara langsung ke lokasi. Pemerintah belum melibatkan masyarakat umum dalam melakukan evaluasi pelayanan dan sarana prasarana tempat wisata yang ada di Yogyakarta. Menurut Ibu Ellya Shari, sebenarnya Pemerintah Dinas perlu mendengar aspirasi langsung dari masyarakat, namun saat ini masih belum diterapkan.

Dari wawancara yang dilakukan, timbul pemikiran untuk membuat sistem yang dapat membantu pihak Dinas Pariwisata DIY dalam memonitoring kondisi obyek wisata khususnya pantai berdasarkan opini masyarakat umum. Opini yang dipilih diambil dari ulasan tentang obyek wisata pantai pada *google maps*. Wawancara terhadap Ibu Ellya Shari juga membahas terkait pengelompokan kategori dalam melakukan evaluasi kondisi obyek wisata. Menurut Ibu Ellya Shari, aspek terpenting yang perlu dievaluasi adalah SDM dari segi pengelola obyek wisata. Namun aspek SDM dapat dijabarkan lagi menjadi beberapa kategori, yaitu daya tarik, aksesbilitas, kebersihan, dan fasilitas. Sehingga dalam penelitian ini mengelompokkan aspek ulasan ke empat kategori, yaitu kategori daya tarik, aksesbilitas,

kebersihan dan fasilitas. Adapun deskripsi untuk penjelasan dari masing-masing kategori dapat dilihat pada Tabel 3.1.

Tabel 3.1 Deskripsi kategori

Kategori	Deskripsi	Contoh
Daya Tarik	Segala sesuatu tentang keindahan, keunikan, dan nilai daya tarik dari suatu tempat wisata	lokasinya bagus/jelek, menarik/tidak menarik, unik/biasa saja, keren, menakjubkan, kelebihan dari lokasi wisata (spt: ombak indah, pemandangan hijau, udara sejuk).
Aksesbilitas	Mencakup akses menuju lokasi wisata, seperti ketersediaan transportasi, kondisi jalan, jarak tempuh yang diukur dari pusat kota, kemudahan memperoleh informasi tentang destinasi	jalan rusak, terjal, bagus, halus, kasar, banyak batu, licin, sempit, hanya cukup dilewati satu kendaraan, okasinya jauh dari pusat kota, pelosok, susah diakses, susah dicari melalui *google maps.*
Kebersihan	Mencakup tingkat kebersihan dari area tempat wisata dan juga ketersediaan petugas kebersihan di lokasi wisata.	bersih, kotor, banyak sampah, jorok, kumuh, tidak terawat.
Fasilitas	Sesuatu yang menunjang kemudahan dan kenyamanan wisatawan, seperti ketersediaan tempat ibadah, kamar mandi, area bermain, penjual makanan, area parkir dalam radius tertentu dan sarana wisata pendukung lainnya	fasilitas lengkap, ada kamar mandi, ada tempat ibadah (mushollah, gereja, dan lain-lain), tersedia tempat makan, area parkir luas, taman bermain, tempat duduk, tempat istirahat, area *camping*, persediaan air, wastafel (tempat cuci tangan), fasilitas penunjang untuk penderita difabel.

3.3 Pengumpulan Kebutuhan

Pada bagian ini akan membahas mengenai kebutuhan-kebutuhan baik berupa data maupun informasi untuk dianalisis. Pengumpulan kebutuhan terdiri dari beberapa tahapan yaitu pengumpulan data, analisis data dan analisis kebutuhan sistem.

3.3.1 Pengumpulan Data

Data berperan penting dalam suatu penelitian untuk diolah menjadi sebuah informasi yang dapat membantu dalam pengambilan keputusan. Pada penelitian ini terdiri dari dua bagian dalam pengumpulan kebutuhan, yaitu studi pustaka dan *scraping* data.

3.1.2.1 Studi Pustaka

Studi pustaka adalah tahap yang dilakukan untuk mendapatkan informasi yang berkaitan dengan masalah penelitian, metode, dan teknik serta kelebihan dan kekurangan pada penelitian-penelitian sebelumnya. Studi pustaka dilakukan sebagai referensi untuk

memperkuat argumentasi pada penelitian ini. Studi pustaka bersumber dari buku teks, jurnal ilmiah, dan sumber lain yang dapat dipertanggungjawabkan.

3.1.2.2 *Scraping* Data dan *Labeling*

Data pada penelitian ini menggunakan data primer yang berupa teks. Data didapatkan dari ulasan 35 pantai di Provinsi Daerah Istimewa Yogyakarta dari *google maps*. Data diambil dengan menggunakan teknik *web scraping* mulai tahun 2017 sampai 2019. *Scraping* dilakukan dengan teknik *coding* menggunakan bahasa pemrograman *python* dan dengan bantuan *chromedriver*.

Data yang sudah terkumpul dari *scraping*, selanjutnya diberi label secara manual baik label sentimen (positif atau negatif) maupun label kategori (daya tarik, aksesbilitas, kebersihan, atau fasilitas). Sehingga diperoleh *dataset* dengan total 3135 ulasan yang terdiri dari 2466 ulasan sentimen positif dan 669 ulasan sentimen negatif. Sedangkan untuk label kategori, 1352 ulasan daya tarik, 526 ulasan aksesbilitas, 536 ulasan kebersihan, dan 721 ulasan fasilitas. Sehingga data yang digunakan pada penelitian ini merupakan data *imbalanced* atau tidak seimbang. Rincian jumlah data pada penelitian ini dapat dilihat pada Tabel 3.2.

Tabel 3.2 Jumlah Data

No	Nama Pantai	Jumlah Data	Jumlah Data Label Sentimen		Jumlah Data Label Kategori			
			Positif	Negatif	Daya Tarik	Aksesbilitas	Kebersihan	Fasilitas
1	Indrayanti	89	68	21	25	16	35	13
2	Drini	89	87	2	39	14	12	24
3	Siung	92	78	14	46	19	16	11
4	Ngandong	92	83	9	40	7	22	23
5	Nglambor	86	73	13	46	18	9	13
6	Tanjung Kesirat	92	79	13	50	31	3	8
7	Greweng	96	73	23	41	36	8	11
8	Ngrawe	87	74	13	27	9	16	35
9	Wedi Ombo	94	79	15	43	21	17	13
10	Sadeng	81	70	11	52	5	8	16
11	Sedahan	94	47	47	27	44	11	12

Tabel 3.3 Lanjutan Jumlah Data

12	Ngedan	96	65	31	27	30	15	24
13	Sadranan	98	91	7	44	6	22	26
14	Krakal	89	78	11	42	9	18	20
15	Timang	85	52	33	40	37	0	8
16	Gesing	82	62	20	31	13	11	27
17	Wohkudu	80	64	16	19	42	6	13
18	Watu Kodok	80	67	13	25	11	13	31
19	Ngrumput	90	77	13	30	15	21	24
20	Sanglen	99	66	33	37	13	29	20
21	Watulawang	95	84	11	36	21	24	14
22	Sepanjang	100	87	13	29	14	24	33
23	Nglolang	94	81	13	58	8	12	16
24	Baron	93	72	21	32	15	14	32
25	Ngrenehan	95	82	13	44	11	12	28
26	Pandansari	89	61	28	51	3	19	16
27	Samas	84	50	34	48	5	21	10
28	Depok	91	72	19	37	4	13	37
29	Kuwaru	87	47	40	37	9	19	22
30	Baru	93	80	13	25	11	14	43
31	Parangtritis	84	71	13	48	5	14	17
32	Goa Cemara	88	81	7	43	5	12	28
33	Congot	84	43	41	43	6	20	15
34	Glagah	84	72	12	46	7	6	25
35	Bugel Peni	83	50	33	44	6	20	13
Total data		3135	2466	669	1352	526	536	721

Contoh data ulasan hasil *scraping* beserta label sentimen dan label kategori dapat dilihat pada tabel 3.4.

Tabel 3.4 Contoh Data beserta Label

No	Ulasan	Label Sentimen	Label Kategori
1	Kurang suka .. Masih sepi bgt.. Cocok buat camp kyknya..."	Negatif	Daya Tarik
2	akses susah tapi dibayar ketika sampai sana	Negatif	Aksesbilitas
3	Kotor, mending ke landasan pacu bandara kalo mau foto	Negatif	Kebersihan
4	Fasilitas minim, kebanyakan toilet, jajanan mahal	Negatif	Fasilitas
5	Pantai ini sangat bagus, ombaknya tidak terlalu besar di tepi pantainya dan banyak hewan laut	Positif	Daya Tarik
6	Akses ke pantai sangat mudah. Jalan sudah aspal.	Positif	Aksesbilitas
7	Pantainya bersih, indah ombaknya besar sekali	Positif	Kebersihan
8	Panas tp banyak fasilitas. Ada kolam renang banyak untuk anak2, cukup adem di bagian fasilitasnya, ada banyak penjual ikan udang	Positif	Fasilitas

3.3.2 Analisis Kebutuhan Sistem

Pada bagian ini akan dilakukan analisis terhadap kebutuhan-kebutuhan yang diperlukan untuk mengembangkan sistem pada penelitian ini. Analisis kebutuhan terbagi menjadi dua bagian yaitu kebutuhan fungsional dan kebutuhan perangkat.

3.2.1.1 Kebutuhan Fungsional

Kebutuhan fungsional adalah gambaran besar proses dari keseluruhan proses yang dapat dilakukan oleh sistem. Kebutuhan fungsional yang akan dilakukan sistem adalah sebagai berikut:

a. Sistem dapat menampilkan persentase ulasan positif masing-masing pantai berupa grafik yang sebelumnya diklasifikasikan terlebih dahulu.
b. Sistem dapat melakukan *input* teks ulasan.
c. Sistem dapat melakukan proses klasifikasi ulasan.
d. Sistem dapat menampilkan hasil akhir *preprocessing*.
e. Sistem dapat menampilkan hasil klasifikasi sentimen dan kategori.
f. Sistem dapat menampilkan hasil detail klasifikasi yang berupa angka prediksi.
g. Sistem dapat menampilkan hasil pengujian algoritma berupa akurasi, presisi, dan *recall* berupa grafik.

3.2.1.2 Kebutuhan Perangkat

Kebutuhan perangkat pada penelitian ini akan menjelaskan mengenai spesifikasi kebutuhan perangkat yang meliputi perangkat keras dan perangkat lunak sebagai penunjang pengembangan sistem. Adapun perangkat keras maupun perangkat lunak yang digunakan pada penelitian ini dapat dilihat pada tabel 3.5.

Tabel 3.5 Spesifikasi Kebutuhan Perangkat Keras dan Perangkat Lunak

Perangkat Keras	**Spesifikasi**
Sistem Operasi	Windows 10
Tipe Sistem	64-bit
Prosesor	Intel Core i3-6006U, 2.0 GHz
Memori (RAM)	4.00 GB
Harddisk	500 GB
Perangkat Lunak	**Spesifikasi**
Bahasa Pemrograman	Python 3.6
Python Distribution	Anaconda Navigator
IDE	Sublime Text, Jupyter
DBMS	MySQL 7.2.0
Web Browser	Google Chrome
GUI	PyQt5

3.4 Proses Desain

Proses desain adalah tahapan yang membahas proses pembuatan rancangan untuk disajikan kepada pelanggan berdasarkan hasil komunikasi antara pengembang dan pelanggan sebelumnya. Perancangan yang dibuat bertujuan untuk memberikan gambaran dan mempermudah proses pembuatan sistem yang nantinya akan dibangun. Pada proses desain ini akan dibuat rancangan sistem dan rancangan pengujian.

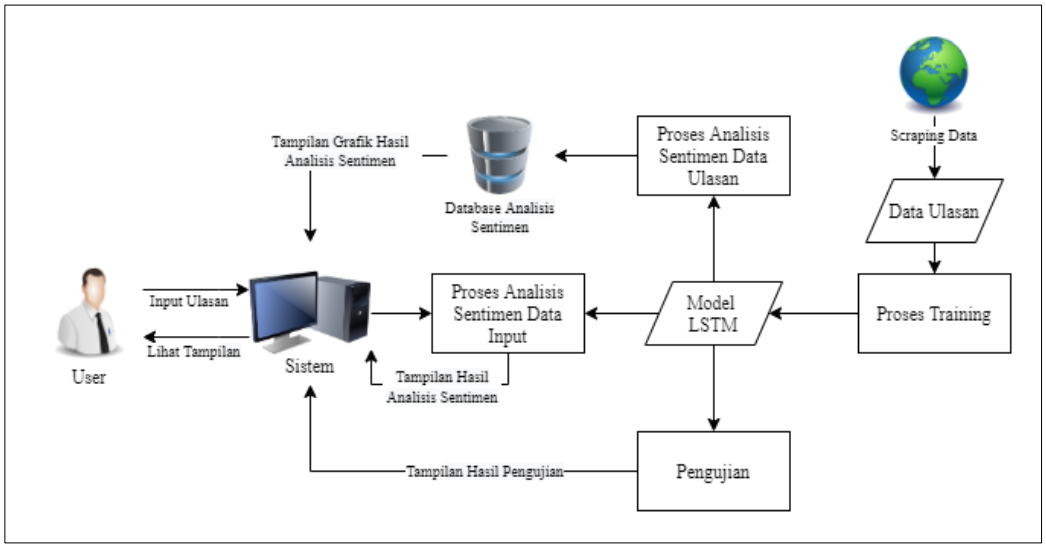
3.4.1 Perancangan Sistem

Pada bagian ini akan membahas tentang rancangan sistem yang akan dibangun pada penelitian ini. Perancangan yang dibuat untuk membangun sistem analisis sentimen ulasan antara lain perancangan arsitektur, perancangan proses, perancangan basis data dan perancangan antar muka.

3.4.2 Arsitektur Sistem

Arsitektur sistem yang akan dibuat terdiri dari pengguna, *database*, proses *scraping*, model LSTM, proses klasifikasi, dan pengujian. Terdapat satu pengguna sistem yang disebut dengan *user*. Secara garis besar, *user* akan melakukan *input* ulasan pada halaman klasifikasi. Setelah proses *input, user* dapat menekan *button* "prediksi" untuk memberikan perintah pada

sistem untuk melakukan proses analisis sentimen terhadap teks ulasan yang telah diinputkan sebelumnya. Sehingga ketika *button* "prediksi" sudah dipilih maka lanjut ke proses teks *preprocessing*, mengubah teks menjadi angka atau *integer*, dan proses klasifikasi dengan *long-short term memory*. Setelah rangkaian proses tersebut selesai maka akan ditampilkan sebagai *output* kepada *user* berupa hasil klasifikasi sentimen dan kategori. Selain itu, *user* juga dapat melihat data yang telah tersimpan dalam *database*. Data yang ditampilkan berupa grafik persentase jumlah data ulasan pantai bernilai positif, yang mana sebelumnya merupakan hasil dari klasifikasi ulang seluruh dataset. Terakhir, *user* dapat melihat hasil pengujian berupa akurasi, presisi, dan *recall* dari model klasifikasi yang dibuat. Ilustrasi dari arsitektur sistem dapat dilihat pada gambar 3.2.



Gambar 3.2 Arsitektur Sistem

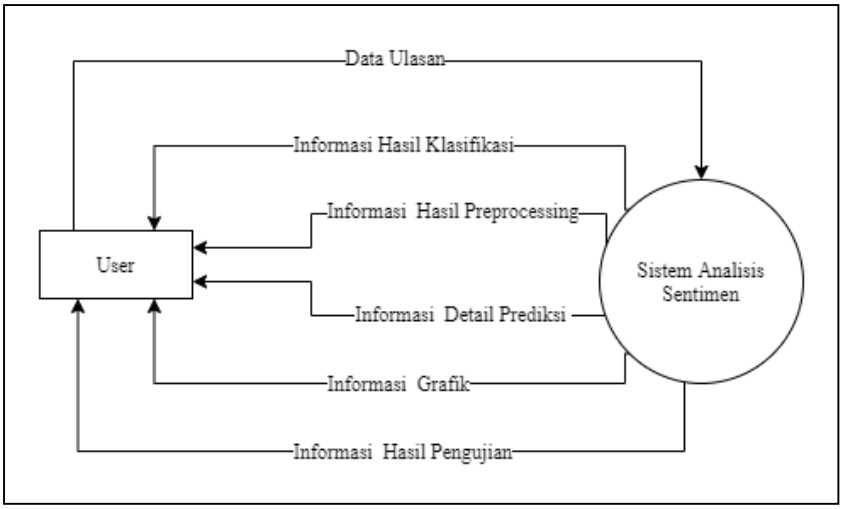
3.4.3 Perancangan Proses

Pada tahap ini akan menggambarkan alur dari proses yang ada pada sistem. Perancangan proses digambarkan dengan *Data Flow Diagram* (DFD) dari level 0 hingga level 1 serta *Flowchart* pada beberapa proses.

a. *Data Flow Diagram Level 0*

Data Flow Diagram (DFD) level 0 menjelaskan gambaran besar mengenai proses dan alur pergerakan data pada sistem ini. *User* dapat memberikan *input* kepada sistem berupa

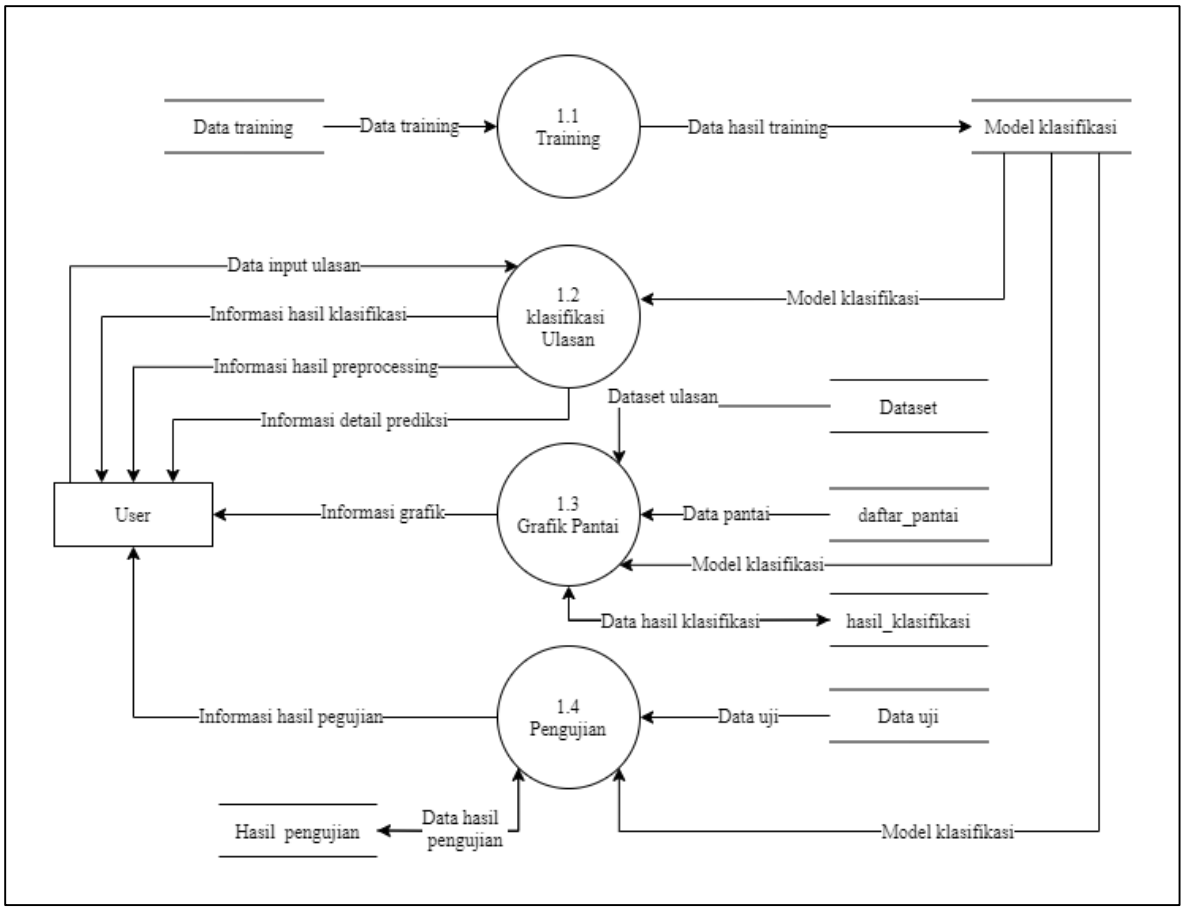
data ulasan dan menerima *output* dari sistem berupa informasi hasil klasifikasi, informasi hasil *preprocessing*, informasi detail prediksi, informasi grafik, dan informasi hasil pengujian. Gambaran dari *data flow diagram* level 0 dapat dilihat pada gambar 3.3.



Gambar 3.3 *Data Flow Diagram* Level 0

b. *Data Flow Diagram Level 1*

DFD level 1 merupakan penjelasan lebih rinci dari DFD level 0. Pada DFD level 1 menggambarkan semua proses yang ada di dalam sistem yang dibuat. Terdapat tiga proses yang digambarkan di dalam diagram, yaitu klasifikasi ulasan, grafik pantai, dan pengujian. Pada proses klasifikasi ulasan menjelaskan bagaimana sistem menerima *input* ulasan dari *user*. Hasil klasifikasi berupa informasi hasil klasifikasi, informasi hasil *preprocessing*, dan informasi detail prediksi akan ditampilkan kepada *user*. Kemudian pada proses grafik pantai, proses yang terjadi adalah *user* dapat melihat data yang ditampilkan berupa grafik persentase jumlah data ulasan pantai bernilai positif, dimana sebelumnya merupakan hasil dari klasifikasi ulang seluruh dataset yang tersimpan dalam tabel hasil_klasifikasi. Terakhir yaitu proses pengujian, proses ini menggambarkan bagaimana sistem menampilkan hasil pengujian akurasi, presisi, dan *recall* dari model algoritma berupa grafik. Gambaran dari *data flow diagram* level 1 dapat dilihat pada gambar 3.4.

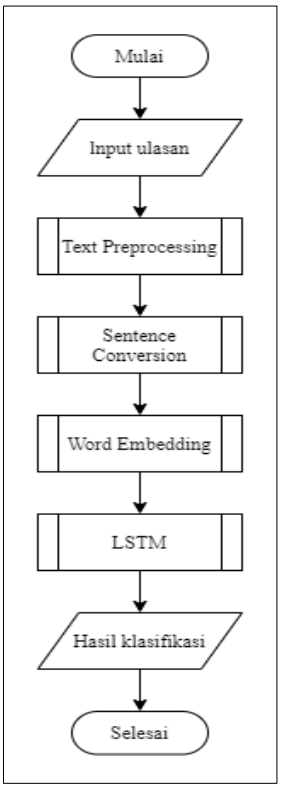


Gambar 3.4 *Data Flow Diagram* Level 1

c. *Flowchart* dan Analisis Klasifikasi Ulasan

Flowchart yang ditampilkan pada penelitian ini merupakan proses secara garis besar, yaitu proses memasukkan teks ulasan pada halaman klasifikasi. Kemudian akan dilakukan proses *preprocessing* teks, yang terdiri dari tahapan-tahapan dalam pembersihan dan penyeleksian data. Hasil dari *preprocessing* teks akan diubah menjadi numerik agar dapat diproses oleh algoritma atau biasa disebut dengan *sentence* conversion. Selain itu, hasil dari *preprocessing* teks juga dilakukan proses pembobotan fitur menggunakan metode *word embedding* dengan *word2vec*. Hasil dari proses *word embedding* tersebut akan digunakan sebagai bobot pada *hidden layer* saat proses *training*. Klasifikasi dilakukan dua kali, yang pertama untuk klasifikasi dalam menentukan nilai sentimen (positif atau negatif) dan yang kedua untuk klasifikasi dalam menentukan nilai kategori (daya tarik, aksesbilitas, kebersihan, atau fasilitas). *Output* dari sistem ini adalah hasil klasifikasi ulasan baik

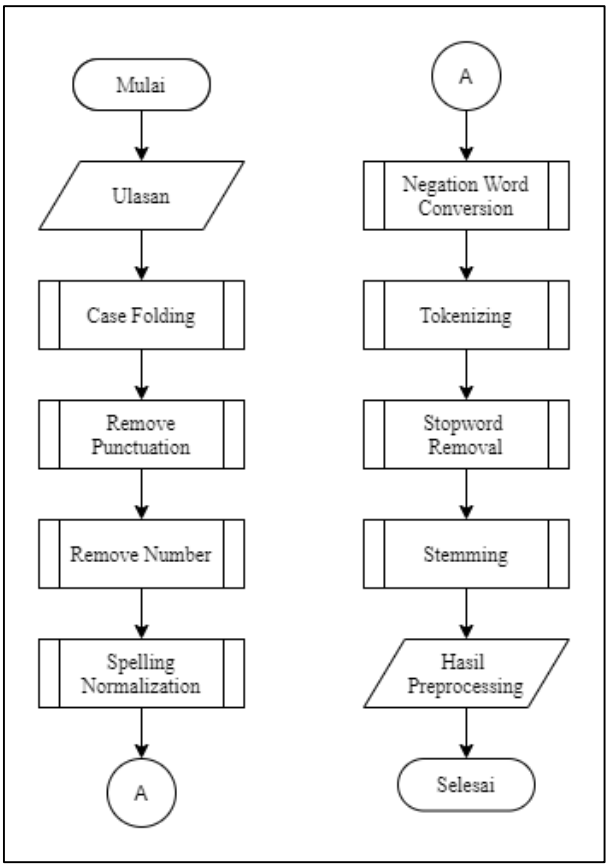
sentimen maupun kategori dari suatu ulasan. *Flowchart* dari proses klasifikasi dapat dilihat pada gambar 3.5.



Gambar 3.5 *Flowchart* Klasifikasi Ulasan

d. *Flowchart* dan Analisis *Text Preprocessing*

Flowchart yang akan ditampilkan merupakan proses seleksi data ulasan yang diinputkan oleh *user* agar data yang akan digunakan menjadi lebih terstruktur sebelum diklasifikasikan nanti. *Preprocessing* pada penelitian ini terdiri dari beberapa tahapan, antara lain: *case folding, remove punctuation, remove number, spelling correction, negation word conversion, tokenizing, stopword removal,* dan *stemming*. Pemilihan tahapan *preprocessing* bergantung pada kebutuhan setiap penelitian. *Flowchart* dari *preprocessing* dapat dilihat pada gambar 3.6.



Gambar 3.6 *Flowchart Text Preprocessing*

Berikut adalah contoh hasil *text preprocessing* terhadap sebuah ulasan setelah melalui semua tahapan proses *preprocessing*.

Teks awal :

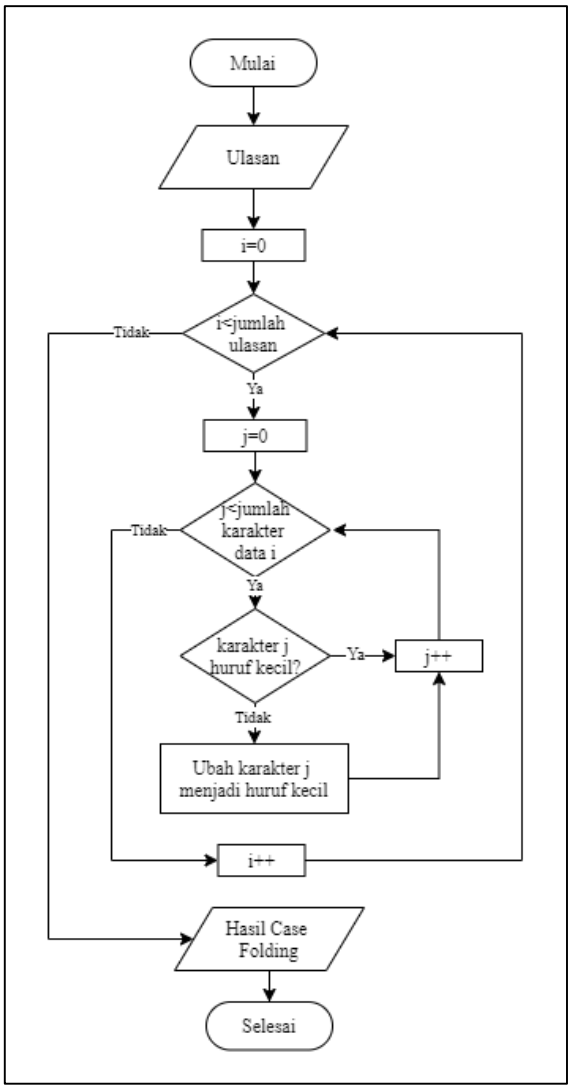
“Pantainya Indrayanti benar2 indah dan tdk kotor!!”

Hasil *preprocessing*:

“[‘pantai’,’indrayanti,’indah’,’bersih’]”

e. *Flowchart* dan Analisis *Case Folding*

Flowchart pada tahap *case folding* berfungsi untuk mengetahui bagaimana proses ulasan yang diinputkan sebelumnya diubah karakternya. Karakter yang terdapat huruf kapital akan diganti menjadi huruf kecil semua dengan menggunakan fungsi `.lower()`. *Flowchart* dari proses *case folding* dapat dilihat pada gambar 3.7.



Gambar 3.7 *Flowchart Case Folding*

Teks awal :

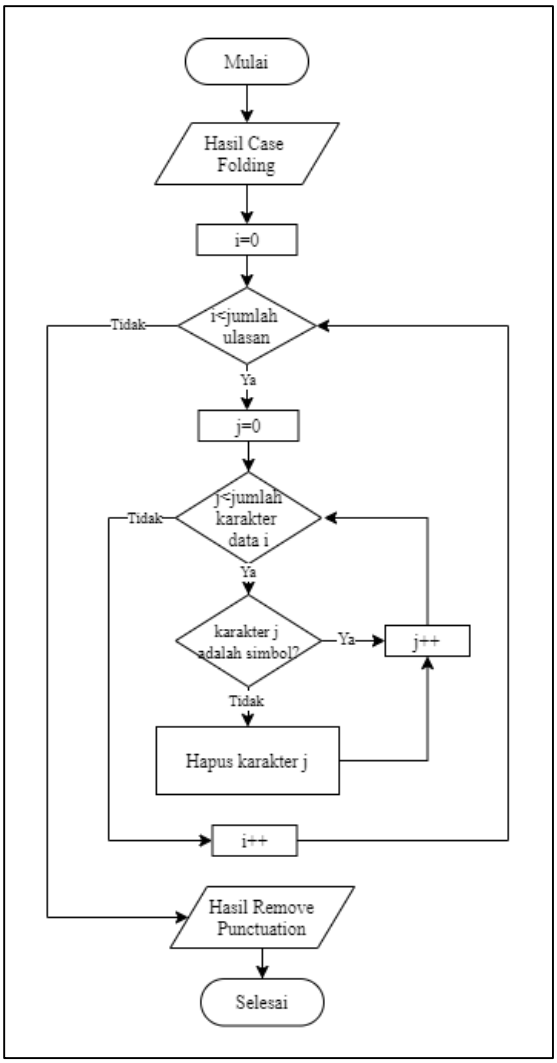
“Pantainya Indrayanti benar2 indah dan tdk kotor!!”

Hasil *case folding*:

“pantainya indrayanti benar2 indah dan tdk kotor!!”

f. *Flowchart* dan Analisis *Remove Punctuation*

Flowchart pada tahap *remove punctuation* berfungsi untuk mengetahui bagaimana proses menghapus tanda baca atau simbol dari hasil tahapan *preprocessing* sebelumnya. Contoh simbol yang dihapus adalah tanda titik (.), koma (,), tanda tanya (?), dan lain-lain. *Flowchart* dari proses *remove punctuation* dapat dilihat pada gambar 3.8.



Gambar 3.8 *Flowchart Remove Punctuation*

Teks hasil *case folding* :

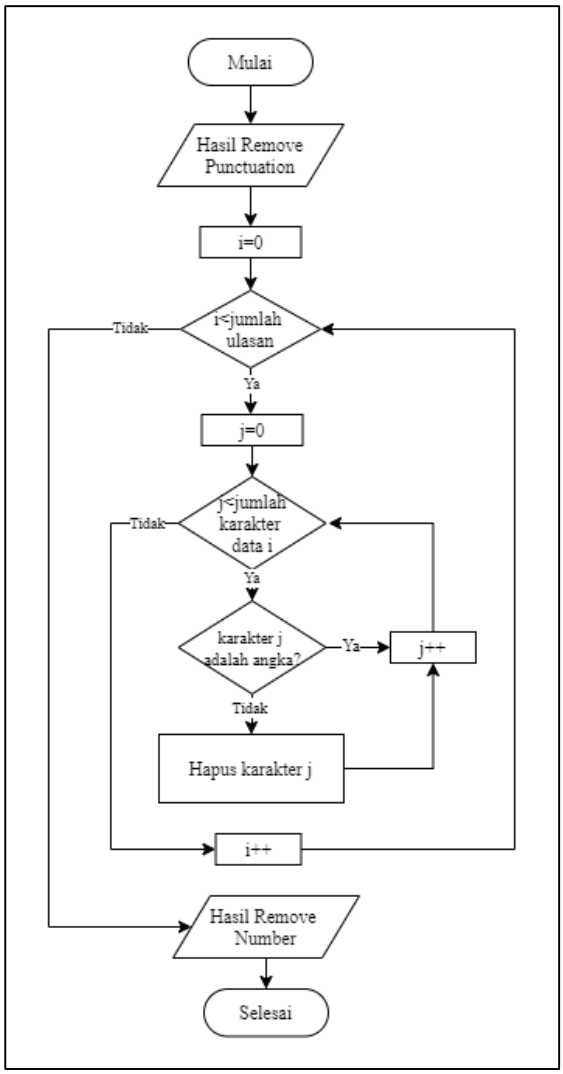
“pantainya indrayanti benar2 indah dan tdk kotor!!”

Hasil *remove punctuation*:

“pantainya indrayanti benar2 indah dan tdk kotor”

g. *Flowchart* dan Analisis *Remove Number*

Flowchart pada tahap *remove number* berfungsi untuk mengetahui bagaimana proses menghapus angka dari hasil tahapan *preprocessing* sebelumnya. *Flowchart* dari proses *remove number* dapat dilihat pada gambar 3.9.



Gambar 3.9 *Flowchart Remove Number*

Teks hasil *remove punctuation*:

“pantainya indrayanti benar2 indah dan tdk kotor”

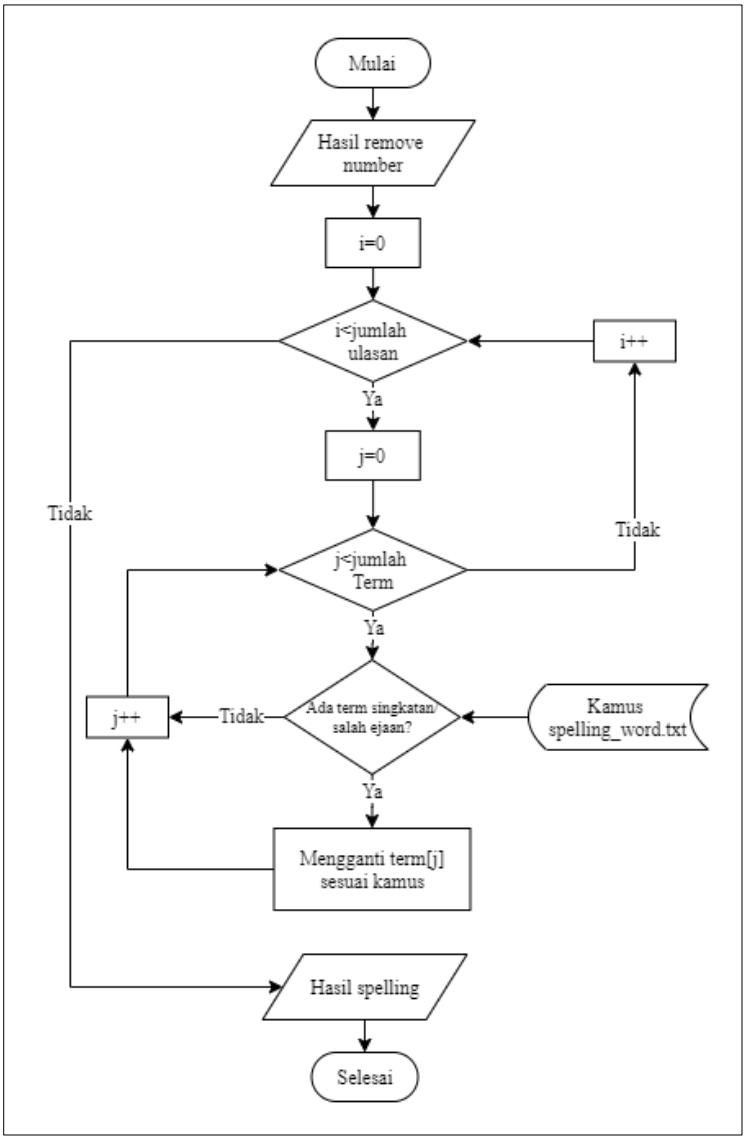
Hasil *remove number* :

“pantainya indrayanti benar indah dan tdk kotor”

h. *Flowchart* dan Analisis *Spelling Correction*

Spelling correction merupakan proses untuk mengganti kata yang tidak sesuai ejaan atau kata singkatan menjadi kata aslinya. Pergantian tersebut menggunakan fungsi sinonim. Jadi sebelumnya telah dibuat daftar kata sinonim dari kata singkatan yang telah disimpan di dalam *file* dengan nama Kamus spelling_word.txt. Proses ini dilakukan guna meminimalisir penggunaan kata singkatan ata kata sehari-hari agar mudah untuk melakukan proses

klasifikasi. Tidak menutup kemungkinan jika ulasan-ulasan yang ada pada kolom komentar *google maps* banyak yang menggunakan kata singkatan untuk efisiensi waktu. Selain itu ulasan yang ada di *google maps* juka tidak menutup kemungkinan ada kesalahan dalam menulis atau biasa disebut *typo*. *Flowchart* dari proses *spelling correction* dapat dilihat pada gambar 3.10.



Gambar 3.10 *Flowchart Spelling Correction*

Teks hasil *remove number*:

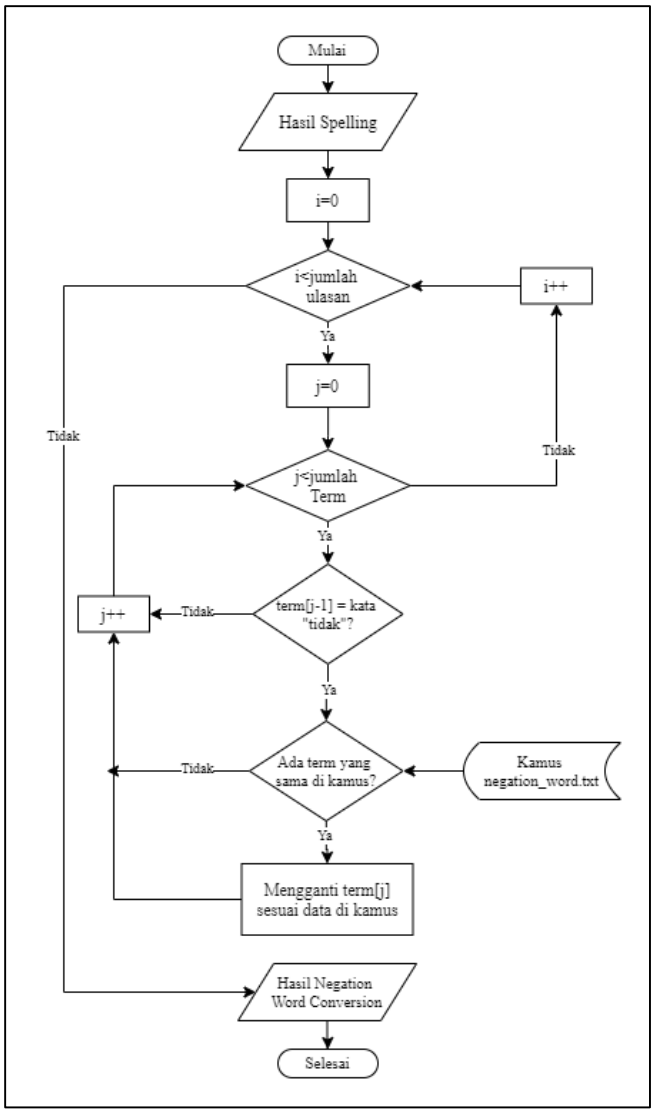
“pantainya indrayanti benar indah dan tdk kotor”

Hasil *spelling correction*:

“pantainya indrayanti benar indah dan tidak kotor”

i. *Flowchart* dan Analisis *Negation Word Conversion*

Negation word conversion merupakan proses untuk mengganti kata negasi yang terbentuk dari setelah kata "tidak". Hal ini dilakukan agar tidak memberikan makna ganda ketika dibiarkan dalam bentuk kata yang terpisah. Misalnya terdapat teks "tidak bersih" maka jika dibiarkan akan mengandung dua kata yaitu "tidak" dan "bersih" yang memiliki makna berbeda. Sehingga perlu dilakukan proses mengubah dua kata tersebut menjadi satu kata tanpa mengubah makna. Misal dari kata "tidak bersih" akan diubah menjadi kata "kotor". Pergantian tersebut menggunakan fungsi antonim. Jadi sebelumnya telah dibuat daftar kata antonim yang disimpan di dalam *file* dengan nama Kamus negation_word.txt. kata yang diganti adalah kata yang letak posisi *term*nya di setelah kata "tidak". Nantinya kata "tidak" akan dihapus saat proses *stopword removal. Flowchart* dari proses *negation word conversion* dapat dilihat pada gambar 3.11.



Gambar 3.11 *Flowchart Negation Word Conversion*

Teks hasil *spelling correction*:

“pantainya indrayanti benar indah dan tidak kotor”

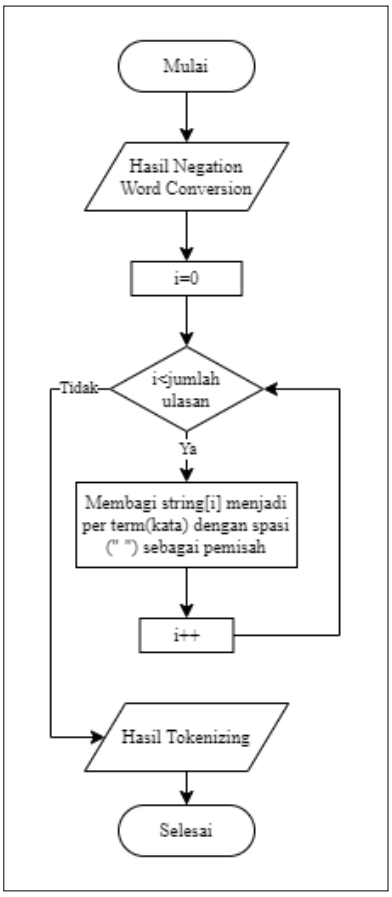
Hasil *negation word conversion* :

“pantainya indrayanti benar indah dan tidak bersih”

j. *Flowchart* dan Analisis *Tokenizing*

Proses *tokenizing* atau tokenisasi adalah proses pemisahan kata dengan memotong *string* terhadap kalimat atau teks hasil dari tahapan *preprocessing* sebelumnya. Pemisahan kata menggunakan kunci karakter spasi. Sehingga pemotongan *string* didapatkan

berdasarkan penemuan spasi yang memisahkan antar kata pada teks. *Flowchart* dari proses *tokenizing* dapat dilihat pada gambar 3.12.



Gambar 3.12 *Flowchart Tokenizing*

Teks hasil *negation word conversion*:

“pantainya indrayanti benar indah dan tidak bersih”

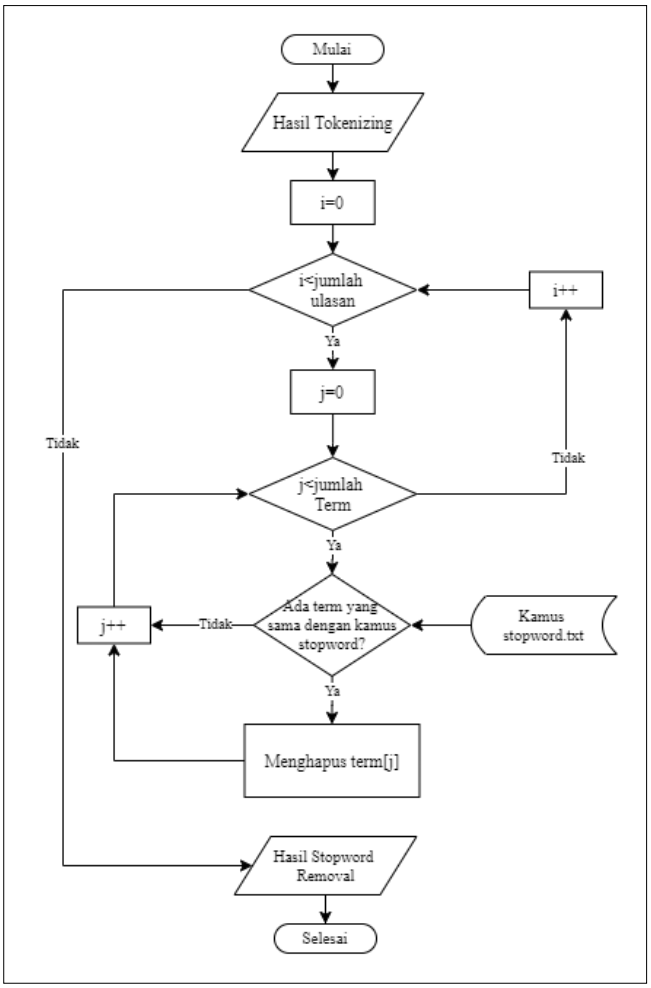
Hasil *tokenizing* :

“[‘pantainya’, ‘indrayanti’, ‘benar’, ‘indah’, ‘dan’, ‘tidak’, ‘bersih’]”

k. *Flowchart* dan Analisis *Stopword Removal*

Proses *stopword removal* bertujuan untuk menghapus kata-kata yang tidak memiliki arti atau tidak berpengaruh terhadap nilai klasifikasi nantinya. Sehingga prosesnya adalah dengan membuang *term* yang diperoleh dari tahap pengecekan pada daftar *stopword*. Apabila sebuah kata terdapat pada daftar *stopword* maka kata tersebut tidak akan diproses lebih lanjut atau dihapus. Sebaliknya apabila sebuah kata tidak terdaftar pada daftar

stopword maka kata tersebut akan masuk ke proses berikutnya. Daftar *stopword* disimpan dalam bentuk *file* dokumen dengan nama kamus_stopword. Dalam penelitian ini menggunakan daftar *stopword* yang tersedia dalam *library* Sastrawi namun dengan menghapus kata baik dan jauh, karena dua kata tersebut dinilai sangat mempengaruhi dalam proses klasifikasi karena memiliki makna tersendiri. *Flowchart* dari proses *stopword removal* dapat dilihat pada gambar 3.13.



Gambar 3.13 *Flowchart Stopword Removal*

Teks hasil *tokenizing*:

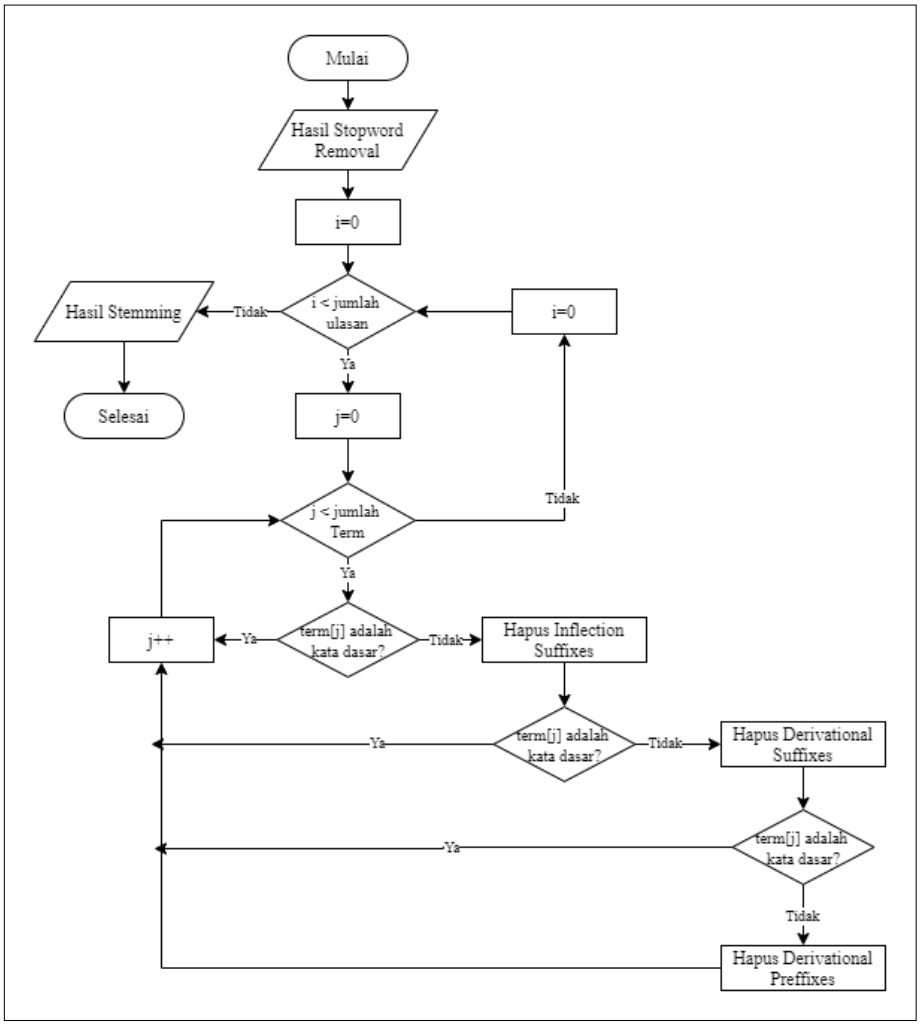
"['pantainya', 'indrayanti', 'benar', 'indah', 'dan', 'tidak', 'bersih']"

Hasil *stopword removal* :

"['pantainya', 'indrayanti', 'indah', 'bersih']"

l. *Flowchart* dan Analisis *Stemming*

Proses *stemming* merupakan proses pembentukan kata dasar atau menghilangkan imbuhan dari sebuah kata. *Term* yang diperoleh dari tahap pembuangan *stopword* akan dilakukan proses *stemming*. *Stemming* digunakan untuk mereduksi bentuk *term* guna menghindari ketidakcocokan yang dapat mengurangi *recall*, dimana *term-term* yang berbeda namun memiliki makna dasar yang sama akan direduksi menjadi satu bentuk (Aldhiansyah, 2020). Penghapusan dimulai pada *inflection suffixes* (-lah, -kah, -ku, dll), kemudian menghapus *deRivational suffix* (-i, -kan, -an) dan terakhir menghapus *devrivational prefix* (-be, -di, -me, -pe, -se, dan –te). Pada penelitian ini, proses *stemming* menggunakan *library* Sastrawi. *Flowchart* dari proses *stemming* dapat dilihat pada gambar 3.14.



Gambar 3.14 *Flowchart Stemming*

Teks hasil *stopword removal*:

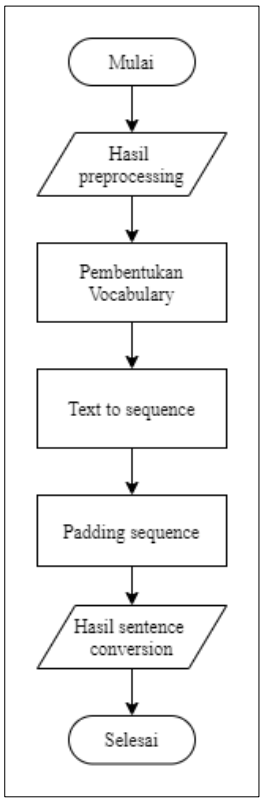
“[‘pantainya’, ‘indrayanti’, ‘benar’, ‘indah’, ‘dan’, ‘tidak’, ‘bersih’]”

Hasil *stemming* :

“[‘pantai’, ‘indrayanti’, ‘indah’, ‘bersih’]”

m. *Flowchart* dan Analisis *Sentence Conversion*

Bagian ini akan dijelaskan mengenai *sentence conversion* atau konversi kalimat menjadi bentuk angka. Perubahan ini dilakukan karena algoritma yang digunakan tidak dapat melakukan pemrosesan terhadap teks secara langsung. Terdapat tiga proses dalam bagian ini antara lain: pembentukan *vocabulary*, *text to sequence*, dan *pedding sequence*. *Flowchart* untuk proses *sentence conversion* dapat dilihat pada gambar 3.15.



Gambar 3.15 *Flowchart sentence conversion*

Data hasil *preprocessing* akan dilanjutkan ke proses *tokenizing* yaitu pemecahan kata dalam dokumen, proses ini bertujuan untuk membantuk *vocabulary* yang akan digunakan

dalam perubahan kalimat ke dalam urutan angka. Proses ini akan dicontohkan menggunakan dataset pada tabel 3.6.

Tabel 3.6 Contoh *datasets*

No	Kalimat
1	pantai indah dengan udara sejuk dan bersih
2	pantai indrayanti dan pantai drini airnya sangat bersih
3	bagus sejuk dan bersih pantai disana

Setiap kalimat pada tabel 3.6 akan di *split* (pisah) menjadi perkata. Kemudian kata tersebut akan dikelompokkan sesuai dengan kata yang sama, jumlah kata dengan frekuensi tertinggi akan mendapatkan *index* yang kecil. Hasil pengelompokan kata dapat dilihat pada tabel 3.7.

Tabel 3.7 Frekuensi kemunculan kata

No	Kata	Kemunculan	Index
1	pantai	4	1
2	indah	1	5
3	dengan	1	6
4	udara	1	7
5	sejuk	2	4
6	dan	3	2
7	bersih	3	3
8	indrayanti	1	8
9	drini	1	9
10	airnya	1	10
11	sangat	1	11
12	bagus	1	12
13	disana	1	13

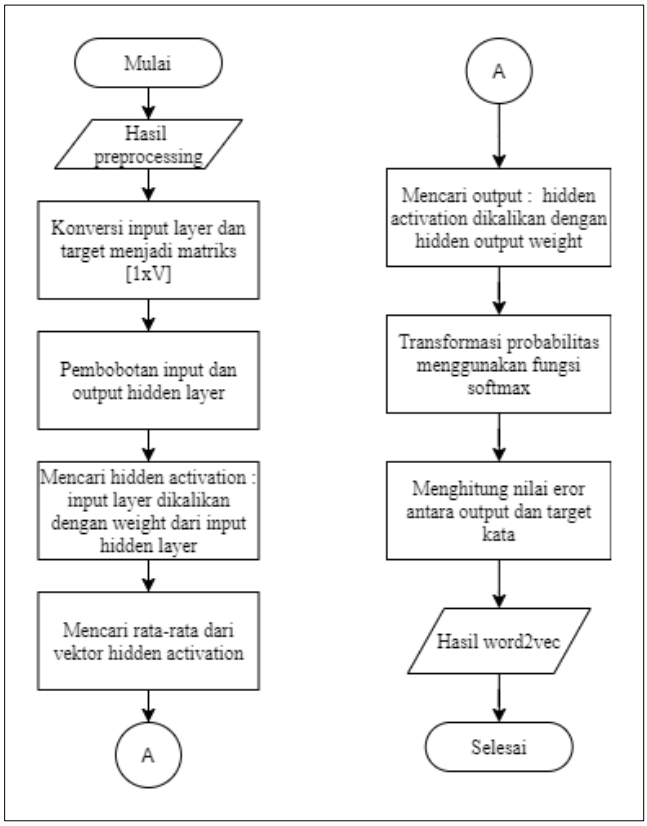
Setelah proses pembentukan *vocabulary*, selanjutnya akan masuk ke proses *text to sequence*. Pada mekanisme ini, kalimat hasil preprocessing pada tahap sebelumnya akan diubah sesuai *index* kata penyusun kalimat pada tabel 3.7. Terakhir yaitu menambahkan *padding sequence* dengan tujuan untuk menyamakan ukuran data. Pada contoh ini diberikan panjang maksimal ukuran data sebesar 10. Hasil dari proses *text to sequence* dan *padding* dapat dilihat pada tabel 3.8.

Tabel 3.8 *Text to sequence* dan *padding*

Hasil preprocesing	Text to sequence	Padding sequence
pantai indrayanti indah bersih	[1 8 5 3]	[0 0 0 0 0 0 1 8 5 3]

n. *Flowchart* dan Analisis *Word Embedding*

Bagian ini akan dijelaskan mengenai proses dari *word embedding*. Proses ini akan mengubah suatu kata menjadi vektor. Vektor yang dihasilkan dari proses ini akan merepresentasikan dari setiap kata yang ada. Kata yang menjadi *input* pada *word embedding* berasal dari hasil akhir keseluruhan *preprocessing*. Vektor yang dihasilkan memiliki panjang atau *dimensi* tertentu yang dapat ditentukan, semakin panjang suatu vektor maka akan semakin teliti representasi yang dihasilkan. Pada penelitian ini metode yang digunakan pada *word embedding* adalah *word2vec*. Hasil dari *word2vec* akan dijadikan sebagai parameter *weight* untuk *embedding layer*. Berikut *Flowchart* proses pembentukan vektor dengan metode *word2vec* dapat dilihat pada Gambar 3.16.



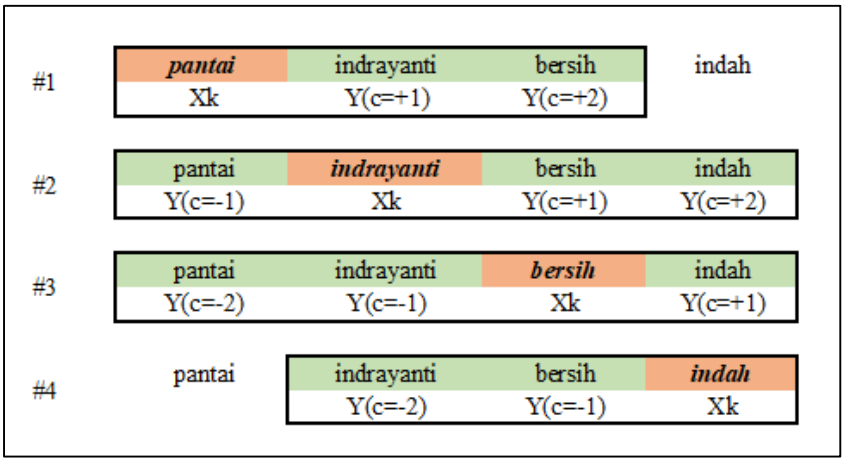
Gambar 3.16 *Flowchart* Proses *Word2Vec*

Pada penelitian ini, tiap-tiap kata akan direpresentasikan menjadi 100 dimensi. Namun, untuk memudahkan dalam memberikan contoh maka akan ditampilkan vektor dari

setiap kata dengan panjang dimensi 5. Berikut adalah contoh analisis langkah-langkah dalam perhitungan *word2vec* dengan panjang 5 dimensi:

1. *Data Preparation*

Contoh data yang akan diproses diambil dari hasil akhir *preprocessing* yaitu data yang terdiri dari 4 kata, yaitu "pantai indrayanti indah bersih". Dimana pada data tersebut digunakan parameter d=5 dan c=2. Nilai c merupakan batas untuk menentukan kata-kata yang bertetangga dengan kata target. Karena nilai c sama dengan dua, maka diartikan bahwa dua kata yang berada di kiri dan kanan dari kata target dianggap sebagai kata konteks atau tetangga. Berikut penentuan kata target beserta tetangganya dapat dilihat pada Gambar 3.17.



Gambar 3.17 Penentuan Kata Tetangga

Kemudian dari data tersebut dilakukan proses *one-hot encoding*, yaitu mengubah kata target menjadi 1 (satu) dan lainnya menjadi 0 (nol). Dalam proses ini, yang diubah menjadi *one-hot encoding* adalah data target dan juga tetangga (y) untuk masing-masing target. Penentuan *one-hot encoding* untuk data *preparation* dapat dilihat pada Tabel 3.9 dan Tabel 3.10.

Tabel 3.9 Penentuan *one-hot encoding*

#	**Token**	**#1**			**#2**			
0	pantai	**1**	**0**	**0**	**0**	**1**	**0**	**0**
1	indrayanti	0	1	0	1	0	0	0
2	bersih	0	0	1	0	0	1	0
3	indah	0	0	0	0	0	0	1
		Xk	y(c=+1)	y(c=+2)	Xk	y(c=-1)	y(c=+1)	y(c=+2)

Tabel 3.10 Lanjutan Penentuan *one-hot encoding*

#	**Token**	**#3**				**#4**		
0	pantai	**0**	**1**	**0**	**0**	**0**	**0**	**0**
1	indrayanti	0	0	1	0	0	1	0
2	bersih	1	0	0	0	0	0	1
3	indah	0	0	0	1	1	0	0
		Xk	Y(c=-2)	Y(c=-1)	Y(c=+1)	Xk	Y(c=-2)	Y(c=-1)

2. Perhitungan *Hidden Layer* dan *y_pred*

Proses pembobotan pada *word2vec* berada pada *hidden layer*, terdapat dua pembobotan yaitu W1 dan W2. Dimana pada *default word2vec*, kedua bobot tersebut diinisialisasikan sebagai *random number* sesuai jumlah kata (*m*) dan dimensi (*d*). Pada contoh ini, bobot W1 yang digunakan mengambil dari *numpy random* dengan ukuran kata (*m*) x dimensi (*d*) yaitu 4 x 5. Bobot (W1) dapat dilihat pada Tabel 3.11.

Tabel 3.11 Bobot *random* (W1)

W1				
0.70506098	-0.10871205	0.65437026	0.72073724	0.59730889
0.07102115	0.19979316	-0.78093917	0.64489623	-0.26134024
-0.3271392	0.14303545	0.87448123	-0.63874215	0.92498628
0.12994038	0.20938897	0.56970817	-0.60357684	0.58286786

Setelah menginisialisasi *bobot* (W1), selanjutnya menentukan *hidden layer* dengan cara menghitung *dot* dari *input* (*one-hot encoding* data target) dan *weight* (W1). Karena ukuran dari wt adalah 1x4 dan *weight* 4x5, maka hasil *dot* dari keduanya memiliki ukuran 1x5. Berikut hasil perhitungan *hidden layer* (h) dapat dilihat pada Tabel 3.12.

Tabel 3.12 Hasil perhitungan *hidden layer*

Hidden layer (h)
0.70506098
-0.10871205
0.65437026
0.72073724
0.59730889

Selanjutnya menghitung *output layer*, yaitu hasil perkalian dari *hidden layer* (*h*) dengan bobot untuk W2. Ukuran panjang dari W2 kebalikan dari ukuran panjang W1, yaitu dimensi (*d*) x jumlah kata (*m*) sehingga panjangnya 5 x 4. Pada contoh ini, bobot (W2) yang digunakan mengambil dari *numpy random*. Bobot (W2) dapat dilihat pada Tabel 3.13.

Tabel 3.13 Bobot random (W2)

W2			
0.28051035	0.70953748	0.32623082	0.28924548
0.00454899	0.92550979	0.70937339	0.82615607
0.25331587	0.59994193	0.96747514	0.31045799
0.73702188	0.71642267	0.50797742	0.3142267
0.35612519	0.28311639	0.40213934	0.09657033

Setelah menginisialisasi *bobot* (W2), selanjutnya menentukan *output layer* dengan cara menghitung *dot* dari *hidden layer* (h) dan *weight* (W2). Karena ukuran dari adalah 1x5 dan W2 5x4, maka hasil *dot* dari keduanya memiliki ukuran 1x4. Berikut hasil perhitungan *output layer* dapat dilihat pada Tabel 3.14.

Tabel 3.14 *Output Layer*

Output layer
1.10794966
0.55554166
0.60610861
0.14848449

Selanjutnya hasil *output layer* akan ditransformasikan menggunakan fungsi *softmax* untuk mendapatkan nilai probabilitas. Hasil *softmax* dari *output layer* dapat dilihat pada Tabel 3.15.

Tabel 3.15 Hasil *Softmax*

#	**Token**	***Output layer***	***Softmax***
0	pantai	1.10794966	0.3900042
1	indrayanti	0.55554166	0.22447167
2	bersih	0.60610861	0.23611441
3	indah	0.14848449	0.14940972

3. Mencari *Error* dan *Sum of Different*.

Langkah selanjutnya adalah mencari jumlah *Sum of Different* pada waktu *wt*. Sebagai contoh pada kata "pantai", mempunyai tetangga kata "indrayanti" dan "indah". Maka pada kata "pantai" akan dicari jumlah *error* yang diperoleh dari kata tetangga "indrayanti" dan "indah". Sedangkan untuk mencari *error* diperoleh dari hasil pengurangan *softmax* dan nilai *one-hot encoding*. Perhitungan *diff* untuk kata tetangga "indrayanti" dapat dilihat pada Tabel 3.16. Sedangkan untuk kata tetangga "bersih" dapat dilihat pada Tabel 3.17.

Tabel 3.16 Perhitungan *diff* untuk kata tetangga "indrayanti"

y_pred softmax	**wc=+1**	**Token**	***Diff* (wc=+1)**
0.3900042	0	pantai	0,3900042
0.22447167	1	**indrayanti**	-0,775528
0.23611441	0	bersih	0,2361144
0.14940972	0	indah	0,1494097

Tabel 3.17 Perhitungan *diff* untuk kata tetangga "bersih"

y_pred softmax	**wc=+2**	**Token**	***Diff* (wc=+2)**
0.3900042	0	pantai	0,3900042
0.22447167	0	indrayanti	0,22447167
0.23611441	1	**bersih**	-0,76388559
0.14940972	0	indah	0,14940972

Kemudian dilanjutkan dengan menjumlahkan nilai *diff* untuk masing-masing kata tetangga. Hasil penjumlahan nilai *diff* dapat dilihat pada Tabel 3.18.

Tabel 3.18 Hasil penjumlahan nilai *diff*

***Diff* (wc=+1)**	***Diff* (wc=+2)**	**EI (Sum of Diff)**
0,3900042	0,3900042	0,780008
-0,775528	0,22447167	-0,55106
0,2361144	-0,76388559	-0,52777
0,1494097	0,14940972	0,298819

4. *Back Propagation*

Proses *back propagation* bertujuan untuk menyesuaikan kembali setiap bobot atau *weight* dan bias berdasarkan *error* yang didapatkan. Pada tahap ini akan dicari nilai *delta* dari masing-masing W. Pertama, mencari delta pada W2 dengan cara mengalikan nilai *hidden layer* dengan *sum of diff*. Karena *hidden layer* memiliki panjang 5x1 dan *sum of diff* memiliki panjang 4x1, maka perkalian keduanya dengan menghasilkan nilai *delta* W2 dengan ukuran 5x4. Hasil perkalian *hidden layer* dengan *sum of diff* dapat dilihat pada Tabel 3.19.

Tabel 3.19 Hasil perkalian *hidden layer* dengan *sum of diff*

***Delta for* W2**			
0.54995349	-0.38852855	-0.37211087	0.21068593
-0.08479631	0.0599065	0.05737509	-0.03248527
0.5104143	-0.36059509	-0.34535776	0.19553855
0.5621811	-0.39716706	-0.38038434	0.2153703
0.46590595	-0.32915104	-0.31524242	0.17848751

Sedangkan untuk mencari *delta* W1, langkah pertama yang dilakukan adalah mencari *dot* dari *sum off diff* (EI) dan W2. Hasil dari dot W2 dan EI dapat dilihat pada tabel 3.20.

Tabel 3.20 Hasil *dot* W2 dan EI

np.dot(W2,EI)
0.08641247
1.12771841
0.11035589
-1.8189912
0.67488846

Kemudian untuk membuat hasil *dot* tersebut menjadi *delta* W1, hasil *dot* dikalikan dengan *one-hot encoding* dari *wt*. Hasil *delta for* W1 dapat dilihat pada Tabel 3.21.

Tabel 3.21 Hasil *delta for* W1

delta for W1				
0.08641247	1.12771841	0.11035589	-0.18189912	0.67488846
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

5. *Update Weight*

Setelah diperoleh nilai *delta* dari masing-masing *weight*, selanjutnya meng*update* nilai dari masing-masing *weight* dengan cara *weight - learning rate * delta weight*. *Learning rate* pada contoh ini sebesar 0.025. Hasil *update* W1 dapat dilihat pada Tabel 3.22.

Tabel 3.22 *Update* WI

Update Weight (W1)				
0,702900668	**-0,13690501**	**0,651611363**	**0,725284718**	**0,580436679**
0,07102115	0,19979316	-0,78093917	0,64489623	-0,26134024
-0,3271392	0,14303545	0,87448123	-0,63874215	0,92498628
0,12994038	0,20938897	0,56970817	-0,60357684	0,58286786

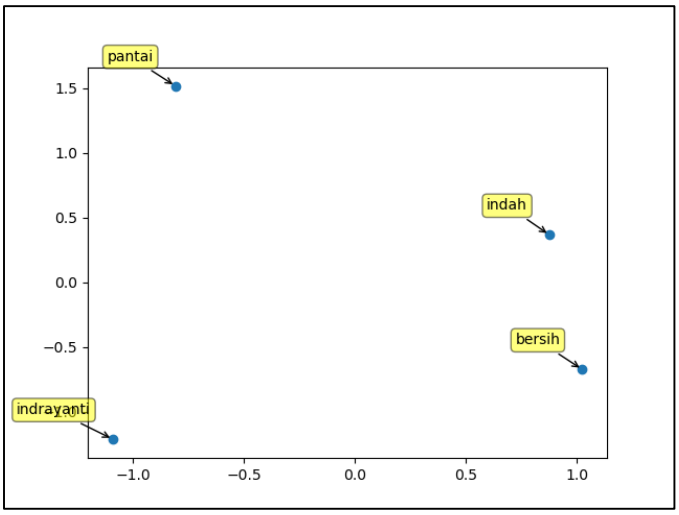
Setelah diperoleh nilai *update*, kemudian dilakukan pengulangan sebanyak yang diperlukan. Sehingga hasil akhir dari proses *training* adalah nilai dengan ukuran *m x d*. Berdasarkan proses perhitungan di atas untuk kata "pantai" akan memiliki vektor sebesar [0.702900668, -0.13690501, -0.651611363, 0.725284718, 0.580436679].

Setelah perhitungan untuk kata target pertama ("pantai") selesai, maka dilanjutkan untuk kata target berikutnya dengan cara yang sama. Berikut hasil dari proses *word2vec* untuk data "pantai indrayanti indah bersih" dapat dilihat pada Tabel 3.23.

Tabel 3.23 Contoh hasil *word2vec*

Kata	**Dimensi**				
	1	**2**	**3**	**4**	**5**
pantai	0.702900668	-0.13690501	0.651611363	0.725284718	0.580436679
indrayanti	0.8127969	2.364095	-0.6084724	0.6790205	-0.65735227
indah	2.7809677	3.9450598	-1.4909214	0.26488346	-0.89886343
bersih	2.928585	2.9040217	-1.6556750	-0.00391697	-2.49438300

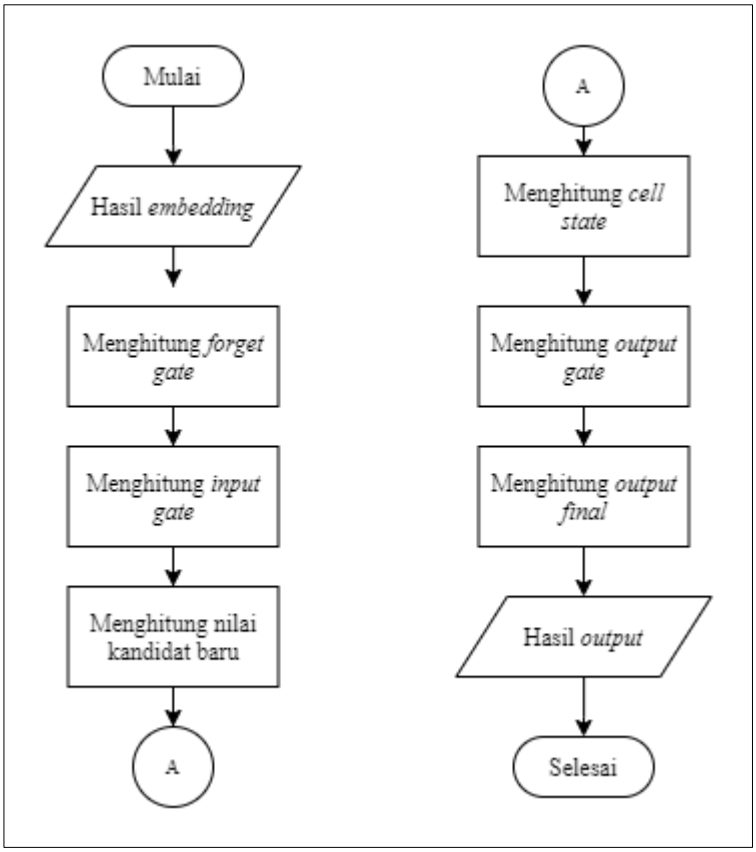
Berdasakan vektor pada tabel 3.23 maka dapat dibuat visualisasi yang merepresentasikan kedekatan antar kata. Berikut hasil visualisasi dari vektor *word embedding* dilihat pada gambar 3.18.



Gambar 3.18 Visualisasi *Word Embedding*

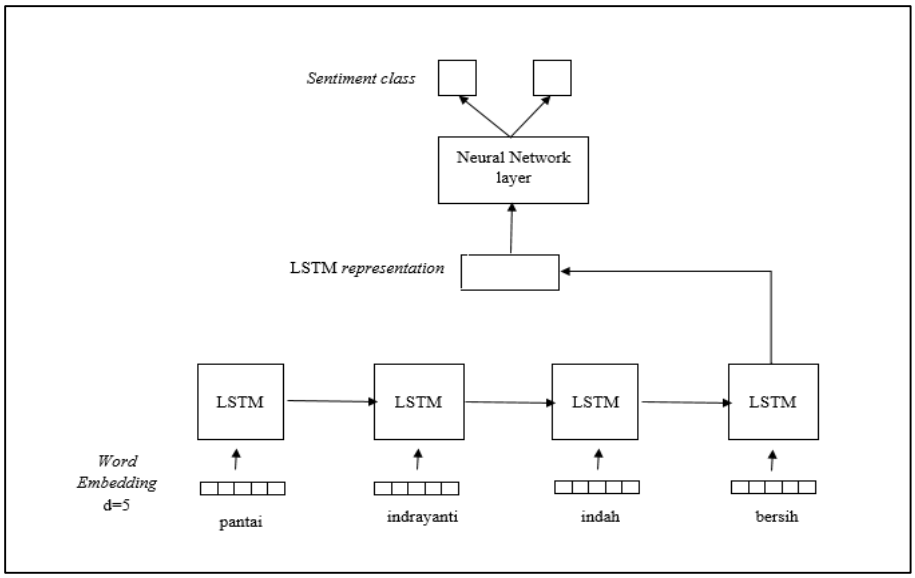
o. *Flowchart* dan Analisis *Long-Short Term Memory* (LSTM)

Pada bagian ini menggambarkan proses klasifikasi menggunakan algoritma LSTM. Hasil dari proses *embedding* akan menjadi data *input* yang akan dimasukkan pada *layer* LSTM. Kemudian masing-masing input akan masuk ke tahapan LSTM, yaitu memutuskan informasi yang akan dihapus dari *cell state*, memutuskan informasi yang akan disimpan pada *cell state*, memperbarui *cell state* yang lama menjadi *cell state* yang baru, dan memutuskan hasil *output*. Hasil masing-masing *output* nantinya akan masuk ke *layer output* dengan menggunakan fungsi *softmax* sehingga diketahui probabilitas dari hasil klasifikasi. Berikut adalah *Flowchart* dari algoritma LSTM dapat dilihat pada gambar 3.19.



Gambar 3.19 *Flowchart* LSTM

Setelah didapatkan nilai *ouput* sesuai dengan *Flowchart* pada gambar 3.18, selanjutnya nilai *output* akan diproses untuk tahap LSTM berikutnya pada *input* yang lain. Begitu seterusnya sampai pada *input* yang terakhir. Selanjutnya hasil tersebut akan masuk ke *layer neural network* untuk diproses hingga manghasilkan nilai probabilitas di masing-masing kelas klasifikasi. Berikut adalah contoh ilustrasi arsitektur dari algoritma LSTM untuk klasifikasi sentimen ke dalam dua kelas yaitu kelas positif dan kelas negatif dapat dilihat pada gambar 3.20.



Gambar 3.20 Ilustrasi Arsitektur LSTM Klasifikasi Sentimen

Untuk analisis algoritma LSTM lebih dalam, berikut adalah contoh perhitungan algoritma LSTM berdasarkan contoh data yang telah melalui tahapan *text preprocessing* dan *word embedding* yang dapat dilihat pada Tabel 3.20.

Data yang akan diproses terdiri dari 4 kata yaitu "pantai", "indrayanti", "indah", dan "bersih". Masing-masing kata akan menjadi input dari LSTM secara berurutan. Sehingga kata pertama yang diproses adalah kata "pantai", kemudian hasil *output* dari proses tersebut akan menjadi nilai h_{t-1} untuk proses input selanjutnya yaitu kata "indrayanti", begitu seterusnya hingga kata terakhir. Karena LSTM tidak mengenali data berupa *text*, maka yang menjadi input adalah angka-angka *word embedding* yang merepresentasikan dari masing-masing kata. Pada contoh ini, *hidden unit* LSTM yang digunakan sebesar lima. *Hidden unit* digunakan dalam pembentukan matriks pada bobot tiap gerbang (*W*), *recurrent weight* (*U*) serta bias (*b*). Perhitungan proses *input* pertama yaitu "pantai" memiliki tahapan algoritma LSTM sebagai berikut :

1. Menghitung *Forget Gate* (f_t)

Proses perhitungan *forget gate* (f_t) berdasarkan persamaan 2.12. Terdapat beberapa variabel yang digunakan dalam perhitungan ini diantaranya *weight forget gate* (W_f),

reccurent weight gate (U_f), dan bias *forget gate* (b_f). *Shape* pada setiap variabel disesuaikan dengan *input* dimensi serta *hidden unit* dengan nilai pembentukan secara *random*. Contoh nilai variabel pada saat t=0 atau saat proses kata pertama "pantai" sebagai berikut:

$$h_{t-1} = [0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$x_t = [0.702900668 \quad -0.13690501 \quad 0.651611363 \quad 0.725284718 \quad 0.580436679]$$

$$W_f = \begin{bmatrix} 0.40812321 & -0.08924346 & -0.59908337 & 0.13585656 & -0.21089559 \\ 0.75411028 & -0.99579461 & -0.99579461 & 0.52794679 & -0.27336747 \\ 0.42183167 & -0.28461212 & -0.26954448 & -0.18463889 & -0.27634445 \\ -0.07616175 & -0.13537449 & -0.19740540 & -0.33538216 & 0.13820167 \\ -0.49762996 & 0.79362659 & -0.76840949 & 0.40920501 & 0.60462960 \end{bmatrix}$$

$$U_f = \begin{bmatrix} -0.72565495 & -0.19702300 & -0.63358108 & 0.23999763 & -0.14133422 \\ -0.49446265 & -0.09840754 & -0.04553243 & 0.13584866 & 0.12456008 \\ 0.26532187 & -0.21847749 & -0.16607277 & -0.34502667 & -0.05585357 \\ -0.18400017 & -0.01344236 & 0.18728717 & -0.28626925 & -0.47900931 \\ -0.91419699 & 0.06878985 & -0.32550184 & -0.67543081 & -0.76229993 \end{bmatrix}$$

$$b_f = [-0.00068824 \quad 1.0009696 \quad 1.0009884 \quad 1.0009965 \quad 0.0999007]$$

Dalam perkalian matriks, syarat dua buah matriks dapat dikalikan adalah jumlah kolom matriks A harus sama dengan jumlah baris pada matriks B sehingga hasil akhirnya adalah matriks C dengan ordo baris matriks A x kolom matriks B. Berdasarkan syarat tersebut, maka perhitungan *forget gate* (f_t) berdasarkan persamaan 2.12 dimulai dengan perkalian antara input pada saat ke t (x_t) dan bobot untu *forget gate* (W_f). Hasil perkalian x_t dan W_f adalah sebagai berikut:

$$x_t * W_f = [0.11441728 \quad 0.25060843 \quad -1.14622271 \quad -0.10282765 \quad 0.1603023]$$

Langkah selanjutnya adalah mengalikan *reccurent weight gate* (U_f) dengan nilai input pada orde t (S_{t-1}). Hasil perkalian keduanya adalah sebagai berikut:

$$h_{t-1} * U_f = [0 \quad 0 \quad 0 \quad 0 \quad 0]$$

Selanjutnya hasil dari $W_f * x_t$ akan ditambahkan dengan hasil $U_f * h_{t-1}$ dan juga nilai bias pada *forget gate* (b_f). Syarat penjumlahan pada matriks adalah jumlah baris dan kolom setiap matriks harus sama. Sehingga hasil dari penjumlahan pada tahap *forget gate* adalah:

$x_t W_f + h_{t-1} U_f + b_f = [0.11372903 \quad 1.25157803 \quad -0.14523431 \quad 0.89816885 \quad 1.15930929]$

Terakhir, untuk memperoleh nilai *forget gate* maka hasil perhitungan yang telah diperoleh sebelumnya harus di aktivasi dengan fungsi *sigmoid* untuk memperoleh nilai antara 0 dan 1. Sehingga nilai *forget gate* adalah sebagai berikut:

$f_t = \sigma\left(x_t W_f + h_{t-1} U_f + b_f\right)$

$f_t = \sigma([0.11372903 \quad 1.25157803 \quad -0.14523431 \quad 0.89816885 \quad 1.15930929])$

$f_t = [0.52840165 \quad 0.77757291 \quad 0.46375511 \quad 0.71057306 \quad 0.76120719]$

2. Menghitung *Input Gate* (i_t)

Berdasarkan Persamaan 2.13, pada contoh perhitungan ini variabel yang digunakan dituliskan sebagai berikut:

$$W_i = \begin{bmatrix} -0.08384067 & 0.05389094 & -0.16728953 & -0.04402454 & -0.24824556 \\ -0.14289487 & 0.38901460 & -0.50905692 & 0.01007607 & -0.63259120 \\ 0.32864657 & -0.20383239 & 0.27974045 & -0.30355830 & 0.34484547 \\ -0.13876237 & 0.43230888 & 0.15773965 & -0.25042242 & -0.18264823 \\ -0.74698850 & 0.79080627 & 0.18194573 & 0.83778325 & 0.04702878 \end{bmatrix}$$

$$U_i = \begin{bmatrix} -0.78114249 & 0.20248958 & -0.16181301 & -0.78598100 & 0.09579491 \\ -0.01982183 & 0.19947390 & -0.00218896 & -0.34185440 & 0.10404024 \\ -0.02342916 & -0.19378471 & 0.20666523 & 0.08170953 & 0.05257188 \\ 0.15744441 & 0.11748825 & 0.27381027 & -0.03372682 & -0.29804187 \\ 0.47337926 & -0.67635669 & -0.14493155 & -0.08870842 & -0.65075383 \end{bmatrix}$$

$b_i = [0.00098822 \quad -0.00099665 \quad 0.00099452 \quad 0.00099469 \quad 0.00099273]$

Proses perhitungan pada *gate* ini sama seperti perhitungan pada *gate* sebelumnya. Proses perhitungan pada *input gate* dijelaskan sebagai berikut:

$x_t * W_i = [-0.35944055 \quad 0.62436242 \quad 0.35440071 \quad 0.07452619 \quad 0.03164342]$

$h_{t-1} * U_i = [0 \quad 0 \quad 0 \quad 0 \quad 0]$

Kemudian nilai tersebut dijumlahkan dengan menambahkan nilai *bias* untuk *input gate* (b_i). Hasil penjumlahan dari ketiganya adalah sebagai berikut:

$x_t W_i + h_{t-1} U_i + b_i = [-0.35845233 \quad 0.62336577 \quad 0.35539523 \quad 0.07552089 \quad 0.03263615]$

Terakhir, untuk memperoleh nilai *input gate* maka hasil perhitungan yang telah diperoleh sebelumnya di aktivasi dengan fungsi *sigmoid* untuk memperoleh nilai antara 0 dan 1. Sehingga nilai *input gate* adalah sebagai berikut:

$i_t = \sigma\,(x_t W_i + h_{t-1} U_i + b_i)$

$i_t = \sigma\,([-0.35845233 \quad 0.62336577 \quad 0.35539523 \quad 0.07552089 \quad 0.03263615])$

$i_t = [0.41133427 \quad 0.65098365 \quad 0.58792529 \quad 0.51887125 \quad 0.50815831]$

3. Menghitung Nilai Kandidat Baru ($\check{C}_t$)

Berdasarkan Persamaan 2.14, pada contoh perhitungan ini variabel yang digunakan dituliskan sebagai berikut:

$$W_C = \begin{bmatrix} 0.92604747 & -0.06616451 & 0.36310227 & -0.79046286 & 0.81499577 \\ -0.55812204 & -0.22692451 & 0.23074913 & 0.46490959 & 0.85258090 \\ -0.35916275 & 0.09782788 & 0.46956800 & 0.11810622 & 0.38800818 \\ 0.29614687 & 0.04033977 & 0.20984042 & 0.04260591 & -0.18922086 \\ -0.79026545 & -0.46473017 & -0.54532653 & -0.92516799 & -0.65253428 \end{bmatrix}$$

$$U_C = \begin{bmatrix} -0.79651941 & -0.16652455 & 0.44165317 & -0.88569451 & 0.30678813 \\ -0.22113278 & -0.11974911 & 0.01106774 & -0.04848960 & 0.01709633 \\ -0.39116687 & 0.00177022 & 0.20922771 & 0.11332781 & -0.04900297 \\ 0.04681184 & -0.22821266 & 0.13414979 & -0.18060955 & -0.00232818 \\ 0.03934869 & 0.04671448 & 0.24460776 & -0.00362939 & -0.25552842 \end{bmatrix}$$

$b_C = [-0.00099993 \quad -0.00099997 \quad 0.00099993 \quad 0.00099991 \quad -0.00099989]$

Proses perhitungan pada *gate* ini sama seperti perhitungan pada *gate* sebelumnya. Proses perhitungan pada *input gate* dijelaskan sebagai berikut:

$x_t * W_C = [0.24938631 \quad -0.19218284 \quad 0.36527649 \quad -1.04840599 \quad 0.19297519]$

$h_{t-1} * U_C = [0 \quad 0 \quad 0 \quad 0 \quad 0]$

Kemudian nilai tersebut dijumlahkan dengan menambahkan nilai *bias* untuk *input gate* (b_C). Hasil penjumlahan dari ketiganya adalah sebagai berikut:

$x_t W_C + h_{t-1} U_C + b_C = [0.24838638 \quad -0.1931828 \quad 0.36627642 \quad -1.04740608 \quad 0.19197529]$

Terakhir, nilai ini akan doproses dengan fungsi *tanh* untuk memperoleh nilai antara 1 dan -1. Sehingga nilai kandidat *cell state* baru adalah sebagai berikut:

$\check{C}_t = tanh\,(x_t W_C + h_{t-1} U_C + b_C)$

$\check{C}_t = tanh\,([0.24838638 \quad -0.1931828 \quad 0.36627642 \quad -1.04740608 \quad 0.19197529])$

$\check{C}_t = [0.24340123 \quad -0.19081498 \quad 0.35073045 \quad -0.78079585 \quad 0.18965116]$

4. Menghitung *Cell State*

Hasil dari *forget gate, input gate,* dan kandidat *cell state* yang telah dihitung sebelumnya akan digunakan untuk mengganti nilai *cell state* yang lama menjadi *cell state* baru. Nilai *cell state* didapatkan dari hasil perkalian *forget gate* (f_t) dengan *cell state* sebelumnya (C_{t-1}) lalu menambahkannya dengan hasil perkalian *input gate* (i_t) dengan nilai kandidat baru ($\check{C}_t$). Karena dalam contoh ini menggunakan t=0 untuk proses kata pertama yaitu "pantai", maka nilai *cell state* sebelumnya (C_{t-1}) masih bernilai 0. Berdasarkan Persamaan 2.15, perhitungan *cell state* dijelaskan sebagai berikut:

$C_{t-1} = [0 \quad 0 \quad 0 \quad 0 \quad 0]$

$C_{t-1} * f_t = [0 \quad 0 \quad 0 \quad 0 \quad 0]$

$i_t * \check{C}_t = [0.10011927 \quad -0.12421743 \quad 0.20620331 \quad -0.40513252 \quad 0.09637281]$

$C_t = C_{t-1} * f_t + i_t * \check{C}_t$

$C_t = [0.10011927 \quad -0.12421743 \quad 0.20620331 \quad -0.40513252 \quad 0.09637281]$

5. Menghitung *Output Gate* (o_t)

Berdasarkan Persamaan 2.16, dalam menentukan nilai *output gate* diperlukan beberapa variabel sebagai berikut:

$$W_o = \begin{bmatrix} -0.67650780 & 0.16697072 & 0.99581742 & 0.10917386 & -0.02733255 \\ -0.31256801 & -0.41493682 & -0.09581164 & -0.65348942 & 0.86822200 \\ -0.26541660 & 0.13074106 & -0.27597720 & 0.20468484 & -0.43865386 \\ -0.18691759 & -0.29926628 & 0.31149325 & -0.35591918 & -0.26953852 \\ 0.94784690 & -0.85529163 & 0.34871932 & 0.86990796 & 0.01563211 \end{bmatrix}$$

$$U_o = \begin{bmatrix} -0.24249850 & -0.28964251 & -0.14982719 & 0.39583957 & 0.24798371 \\ -0.07348764 & -0.06738382 & 0.00877949 & 0.36337010 & 0.00691418 \\ 0.09444071 & 0.42943755 & 0.02961049 & 0.44220633 & -0.21021531 \\ -0.01019727 & -0.00211894 & -0.37785919 & 0.00822705 & 0.37503898 \\ -0.15103883 & 0.00998165 & -0.00746319 & -0.06825706 & -0.55064251 \end{bmatrix}$$

$b_o = [0.00044157 \quad -0.00099684 \quad -0.00050602 \quad 0.00099475 \quad 0.00099281]$

Proses perhitungan pada *output gate* dijelaskan sebagai berikut:

$x_t * W_o = [-0.1910775 \quad -0.45413277 \quad 0.96157872 \quad 0.54636307 \quad -0.61032657]$

$h_{t-1} * U_o = [0 \quad 0 \quad 0 \quad 0 \quad 0]$

Kemudian nilai tersebut dijumlahkan dengan menambahkan nilai *bias* untuk *output gate* (b_o). Hasil penjumlahan dari ketiganya adalah sebagai berikut:

$x_t W_o + h_{t-1} U_o + b_o = [-0.1906359 \quad -0.4551296 \quad 0.9610727 \quad 0.54735782 \quad -0.60933376]$

Terakhir, untuk memperoleh nilai *output gate* maka hasil perhitungan yang telah diperoleh sebelumnya di aktivasi dengan fungsi *sigmoid* untuk memperoleh nilai antara 0 dan 1. Sehingga nilai *input gate* adalah sebagai berikut:

$O_t = \sigma\,(x_t W_o + h_{t-1} U_o + b_o)$

$O_t = \sigma\,([-0.1906359 \quad -0.4551296 \quad 0.9610727 \quad 0.54735782 \quad -0.60933376])$

$O_t = [0.45248483 \quad 0.38814185 \quad 0.72333653 \quad 0.63352237 \quad 0.35221119]$

6. Menghitung *Output Final*

Tahap terakhir adalah menghitung *output final* (h_t) , yaitu mengalikan *output gate* (O_t) dengan nilai *tanh* dari *cell state baru* (C_t). Hasil perhitungan dijelaskan sebagai berikut:

$tanh\,(C_t)$ = $[0.09978608 \quad -0.12358246 \quad 0.2033296 \quad -0.38433196 \quad 0.09607556]$

$h_t = O_t\, tanh\,(C_t)$

$h_t = [0.04515169 \quad -0.04796752 \quad 0.14707572 \quad -0.24348289 \quad 0.03383889]$

Dari hasil perhitungan di atas untuk input pertama (kata “pantai”), diperoleh nilai *output final* $[0.04515169 \quad -0.04796752 \quad 0.14707572 \quad -0.24348289 \quad 0.03383889]$. Nilai tersebut nantinya akan digunakan untuk proses LSTM input selanjutnya sebagai pertimbangan seberapa besar *memory* sebelumnya yang akan diteruskan atau dibuang. Setelah proses pada orde ke t berdasarkan *input* “pantai” selesai diproses, selanjutnya adalah memproses orde kedua berdasarkan input kata berikutnya yaitu “indrayanti”. Tahapan pada proses ini sama dengan proses pada input kata sebelumnya, yang mana nantinya hasil dari

output dan *cell state* yang telah diperbaharui akan digunakan untuk proses pada orde berikutnya. Begitu seterusnya hingga orde yang terakhir. Hasil perhitungan dari setiap langkah LSTM untuk data yang dicontohkan dapat dilihat pada Tabel 3.24, 3.25, 3.26, 3.27, dan 3.28.

Tabel 3.24 Hasil Perhitungan *Forget gate*

Periode t	**Input (x_t)**	***Forget Gate* (f_t)**				
0	pantai	0.52840165	0.77757291	0.46375511	0.71057306	0.76120719
1	indrayanti	0.89810945	0.13046388	0.56368545	0.8782094	0.5289985
2	indah	0.97886377	0.02857141	0.29726848	0.95772233	0.29067572
3	bersih	0.98144897	0.02494726	0.67427909	0.89226741	0.17583526

Tabel 3.25 Hasil Perhitungan *Input gate*

Periode t	**Input (x_t)**	***Input Gate* (i_t)**				
0	pantai	0.41133427	0.65098365	0.58792529	0.51887125	0.50815831
1	indrayanti	0.43334514	0.68563438	0.17228436	0.36612332	0.11884249
2	indah	0.35284345	0.77713899	0.04729078	0.39069872	0.02024824
3	bersih	0.66448652	0.38978626	0.05350733	0.16320698	0.03538368

Tabel 3.26 Hasil Perhitungan kandidat *cell state*

Periode t	**Input (x_t)**	**Kandidat *cell state* ($\check{C}_t$)**				
0	pantai	0.24340123	-0.19081498	0.35073045	-0.78079585	0.18965116
1	indrayanti	0.2714178	-0.25640796	0.79418953	0.78012405	0.99168141
2	indah	0.92590814	-0.65563607	0.95166258	0.29409388	0.99996966
3	bersih	0.9986257	0.16279728	0.98233369	0.82013468	0.99998259

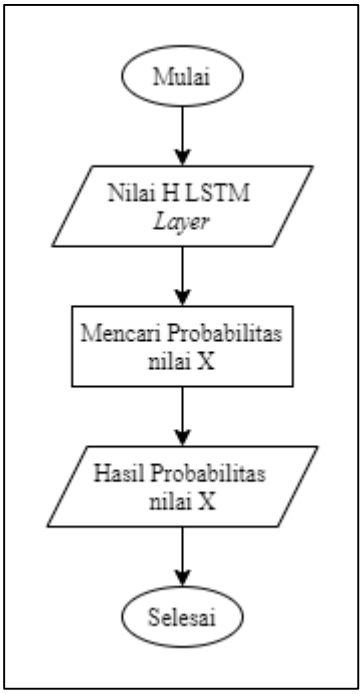
Tabel 3.27 Hasil Perhitungan *output gate*

Periode t	**Input (x_t)**	***Output gate* (O_t)**				
0	pantai	0.45248483	0.38814185	0.72333653	0.63352237	0.35221119
1	indrayanti	0.1331931	0.37458467	0.69455581	0.08850155	0.87814137
2	indah	0.02590588	0.35280553	0.92910369	0.03217611	0.9782385
3	bersih	0.00813754	0.77853955	0.90223564	0.0163728	0.95574397

Tabel 3.28 Hasil Perhitungan *output final*

Periode t	**Input (x_t)**	***Output* (h_t)**				
0	pantai	0.04515169	-0.04796752	0.14707572	-0.24348289	0.03383889
1	indrayanti	0.02725218	-0.07105223	0.17210619	-0.00619994	0.14686811
2	indah	0.01257125	-0.1671722	0.11117254	0.00153361	0.06770683
3	bersih	0.0067426	0.03936684	0.11985476	0.00285876	0.04543299

Setelah diperoleh hasil representasi LSTM, selanjutnya diproses dengan *dense layer* untuk memperoleh nilai probabilitas dari setiap kelas (positif dan negatif). *Output* dari *layer* LSTM (h_t) pada periode terakhir akan digunakan sebagai input yang kemudian akan dihitung dengan bobot serta *bias* pada *dense layer*. Hasil pada perhitungan tersebut kemudian akan menjadi input pada fungsi *sigmoid* sesuai pada persamaan 2.17. *Flowchart* pada proses ini ditunjukkan pada gambar 3.21.



Gambar 3.21 *Flowchart dense layer*

Variabel pada *dense layer* diinisiasi secara *random*, contoh bobot (W_y) yang digunakan adalah sebagai berikut:

$$W_y = \begin{bmatrix} -0.03224891 & -0.90037138 \\ -0.28031669 & -1.03261221 \\ 0.55466795 & 1.18619224 \\ 0.53400163 & -0.88365553 \\ 0.64554507 & -1.05343887 \end{bmatrix}$$

Shape yang terbentuk pada bobot (W_y) memiliki formula (*hidden state*, jumlah label *class*). Dalam contoh ini, *hidden state* sesuai dengan *layer* LSTM yaitu lima dan jumlah label *class* yang digunakan adalah dua (positif dan negatif). Sehingga matriks yang terbentuk

berukuran 5x2 (W_{5x2}). Sedangkan *shape* yang terbentuk untuk *bias* sesuai dengan jumlah label *class*, sehingga nilai bias (b_y) adalah sebagai berikut:

$b_y = [-0.82886214 \quad 0.44722233]$

Proses perhitungan probabilitas dari *dense layer* dijelaskan sebagai berikut:

$h_t = [0.00674261 \quad 0.03936684 \quad 0.11985476 \quad 0.00285876 \quad 0.04543299]$

$Y = \sigma\left(h_t W_y + b_y\right)$

$Y = [0.32239663 \quad 0.62064446]$

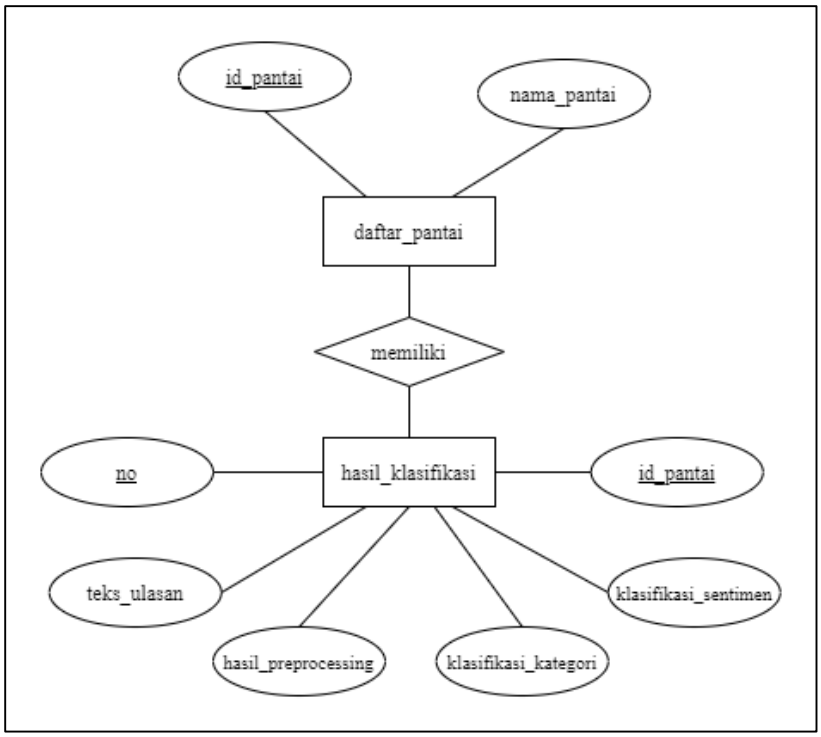
Berdasarkan hasil yang diperoleh (Y) tersebut, nilai probabilitas tertinggi berada pada *index* ke 1 atau kolom kedua. Sehinggu *input* pada contoh yang diberikan yaitu "pantai indrayanti bersih indah" memiliki hasil probabilitas tertinggi pada label kedua. Proses ini kemudian akan berjalan untuk semua *input dataset* yang selanjutnya akan masuk proses *training* dalam pembentukan model.

3.2.2.1 Perancangan Basis Data

Penelitian ini menggunakan *database MySql* sebagai sarana untuk menyimpan data, baik data pantai dan data hasil klasifikasi. Penyimpanan pada *database* akan mempermudah pengembang dalam memvisualisasikan data-data yang perlu untuk ditampilkan dalam sistem. Pada *database* yang bernama "db_analisis_sentimen" ini didalamnya terdapat dua tabel, yaitu tabel daftar pantai dan tabel hasil klasifikasi.

a. *Entity Relationship Diagram* (ERD)

ERD dari *database* yang dibuat pada penelitian ini dapat dilihat pada gambar 3.22.



Gambar 3.22 ERD analisis_sentimen

b. Struktur Tabel

1. Tabel daftar_pantai

Pada tabel daftar_pantai mempunyai dua buah atribut yaitu id_pantai dan nama_pantai dimana id_pantai sebagai *primary key*. Struktur tabel daftar_pantai dapat dilihat pada Tabel 3.29.

Tabel 3.29 Struktur Tabel daftar_pantai

No	Nama Atribut	Tipe Data	Panjang	Keterangan
1	id_pantai	int	11	*Primary Key*
2	nama_pantai	varchar	30	Nama-nama pantai

2. Tabel hasil_klasifikasi

Pada tabel hasil_klasifikasi mempunyai 6 buah atribut yaitu no, id_panta, hasil_preprocessing, klasifikasi_sentimen dan klasifikasi_kategori dimana no sebagai *primary key auto increment*. Struktur tabel hasil_klasifikasi dapat dilihat pada Tabel 3.30.

Tabel 3.30 Struktur Tabel hasil_klasifikasi

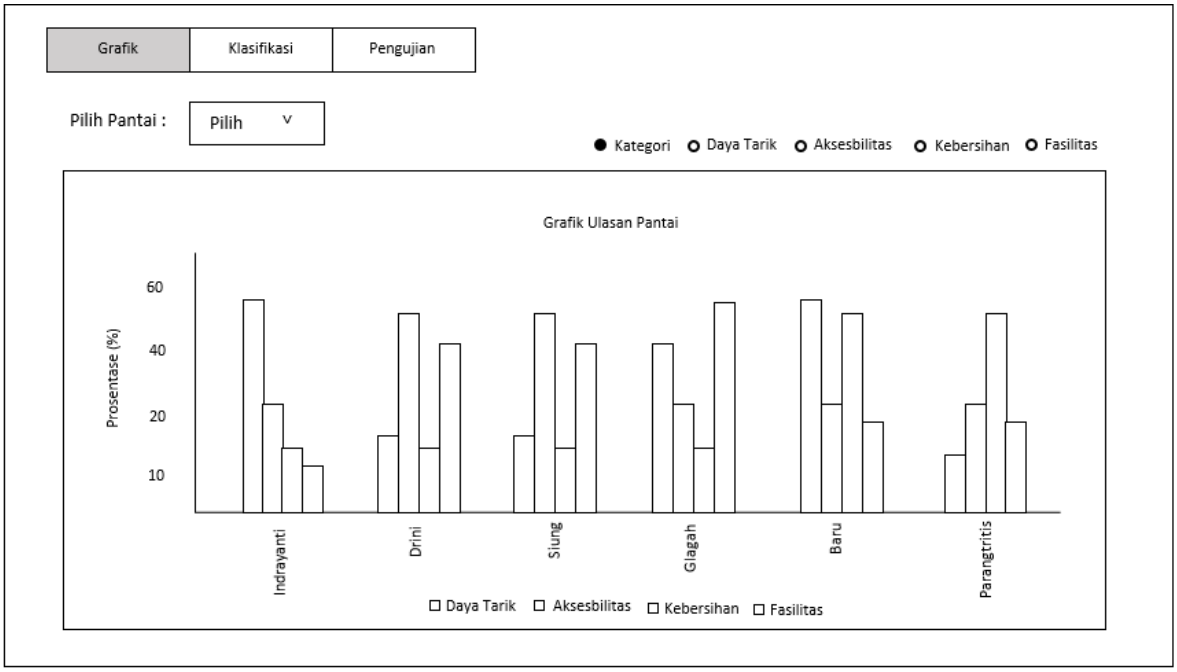
No	Nama Atribut	Tipe Data	Panjang	Keterangan
1	no	int	11	*Primary Key*
2	id_pantai	int	11	*Foreign key* dengan id_pantai pada tabel daftar_pantai.
3	hasil_preprocessing	text		Hasil akhir dari seluruh tahapan *preprocessing.*
4	klasifikasi_sentimen	varchar	10	Hasil klasifikasi sentimen
5	klasifikasi_kategori	varchar	20	Hasil klasifikasi kategori

3.4.4 Perancangan Antarmuka Sistem

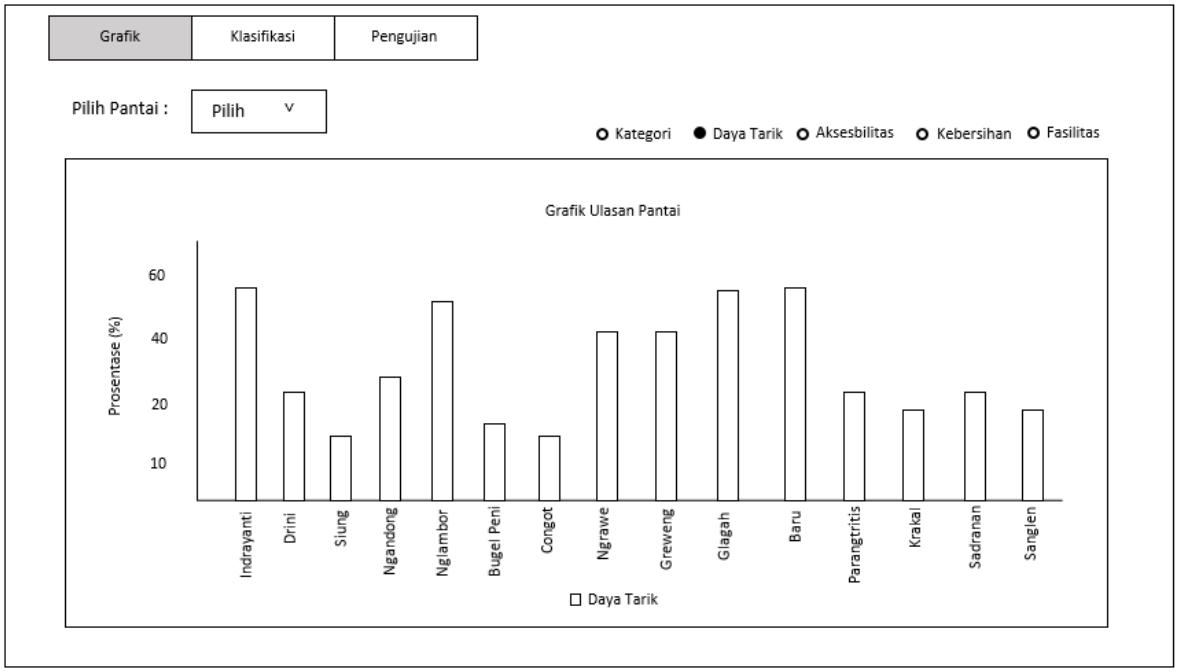
Rancangan antarmuka sistem atau *user Interface* adalah rancangan model mekanisme komunikasi antara *user* dengan sistem. Rancangan *user Interface* pada penelitian ini terdiri dari beberapa halaman sebagai berikut:

1. Halaman Grafik

Halaman grafik adalah halaman pertama muncul saat membuka aplikasi. Pada halaman ini *user* dapat melihat grafik ulasan pantai untuk masing-masing kategori secara keseluruhan dari 35 pantai yang ada. Grafik menampilkan persentase jumlah ulasan positif dari data pantai. Kemudian *user* dapat memilih *radio button* yang berisi opsi untuk menampilkan ulasan pantai berdasarkan kategori yang dipilih. Selain itu, *user* juga dapat memilih nama pantai yang ingin dilihat secara jelas persentasenya. *User* memilih nama pantai melalui menu *combo box*. Rancangan halaman grafik pantai secara keseluruhan dapat dilihat pada gambar 3.23 sedangkan rancangan halaman grafik berdasarkan kategori yang dipilih dapat dilihat pada gambar 3.24.

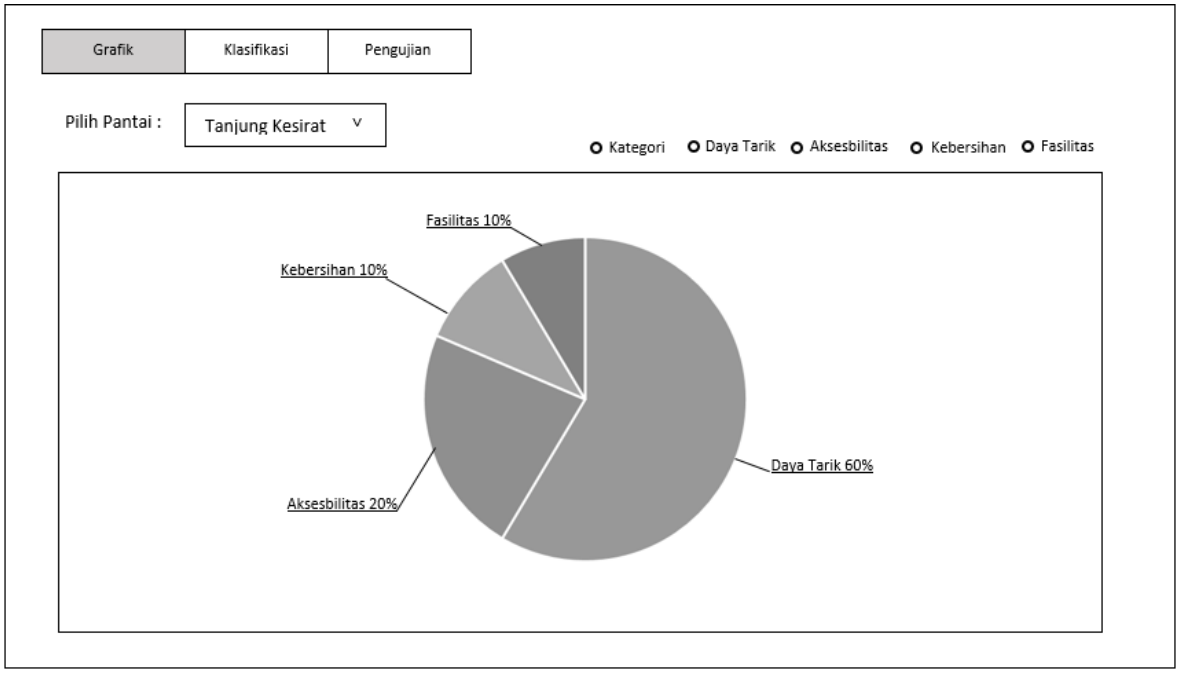


Gambar 3.23 Rancangan *Interface* Halaman Grafik



Gambar 3.24 Rancangan *Interface* Halaman Grafik per Kategori

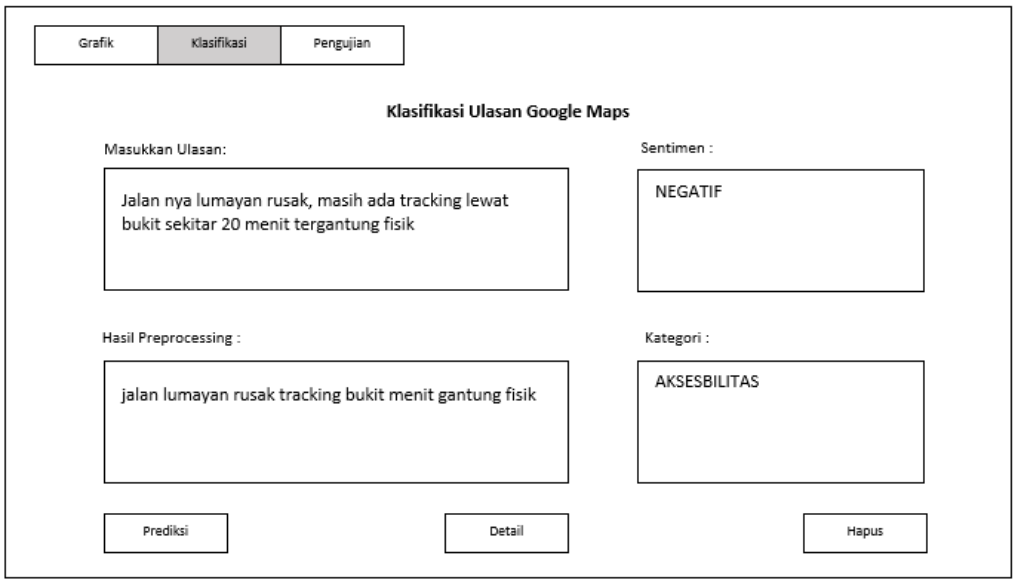
Sedangkan rancangan halaman grafik untuk pantai yang dipilih pada *combo box* dapat dilihat pada gambar 3.25.



Gambar 3.25 Rancangan *Interface* Halaman Grafik per Pantai

2. Halaman Klasifikasi

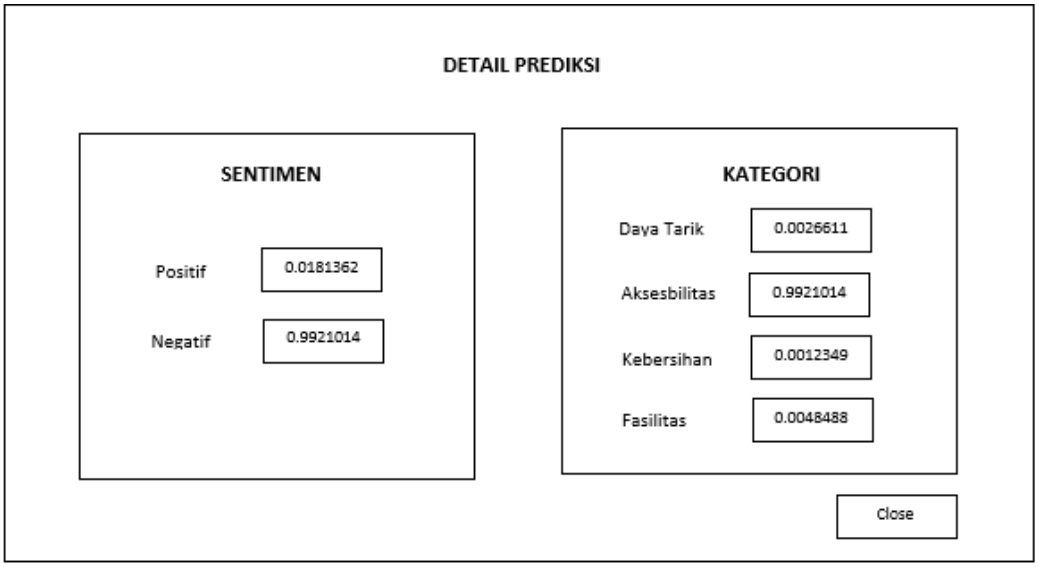
Halaman klasifikasi adalah halaman yang digunakan untuk mengetahui hasil klasifikasi dari sebuah ulasan. Pada halaman ini *user* dapat menginputkan teks berupa ulasan atau komentar untuk dapat diketahui hasil klasifikasinya, baik sentimen maupun kategori. Rancangan halaman klasifikasi dapat dilihat pada gambar 3.26.



Gambar 3.26 Rancangan *Interface* Halaman Klasifikasi

3. Halaman Detail

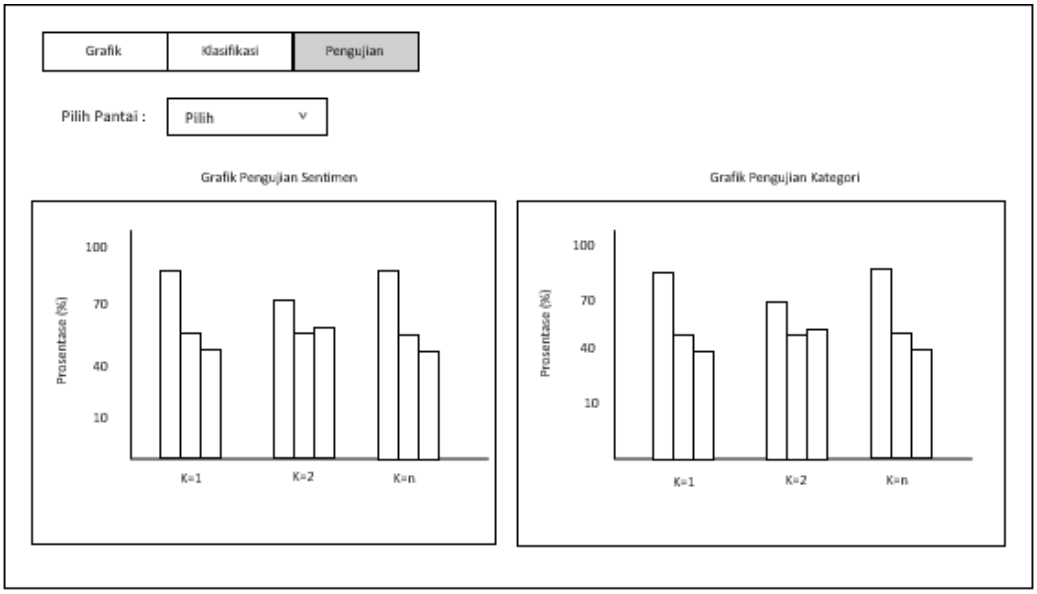
Halaman detail muncul pada saat *user* menekan tombol detail. Halaman detail dapat digunakan *user* untuk melihat detail probabilitas hasil klasifikasi algoritma *Long-Short Term Memory* pada tiap-tiap sentimen maupun kategori. Rancangan halaman detail dapat dilihat pada gambar 3.27.



Gambar 3.27 Rancangan *Interface* Halaman *Detail*

4. Halaman Pengujian

Halaman pengujian adalah halaman yang dapat digunakan *user* untuk melihat hasil pengujian algoritma *Long-Short Term Memory*. Pengujian yang ditampilkan berupa grafik dari *akurasi*, *presisi*, *recall* dari hasil pengujian model sentimen dan model kategori. Rancangan halaman pengujian dapat dilihat pada gambar 3.28.



Gambar 3.28 Rancangan *Interface* Halaman Pengujian

3.4.5 Rancangan Pengujian

Pengujian bertujuan untuk mengetahui tingkat keberhasilan proses klasifikasi yang telah dilakukan. Pengujian yang akan dilakukan yaitu dengan melakukan pendekatan *k-fold cross validation.* Pengujian dilakukan dengan membuat *Confusion Matrix* yang merupakan tabel untuk menyatakan tingkat kebenaran proses klasifikasi. Melalui *confusion matrix* hasil pengujian akurasi, presisi, *recall* dapat diketahui. Selain *confusion matrix*, pengujian juga menggunakan kurva ROC (*Receiver Operating Characteristics*). Kurva ROC diperoleh dari perhitungan nilai TPR dan FPR yang didapatkan dari hasil *confusion matrix*. Kurva ROC dan *Confusion matrix* diterapkan untuk model klasifikasi sentimen dan model klasifikasi kategori. Rancangan tabel yang akan digunakan pada pengujian K-*fold cross validation, confusion matrix,* kurva ROC dapat diihat pada Tabel 3.31, 3.32, 3.33, 3.34, 3.35, 3.36, 3.37 dan 3.38.

Tabel 3.31 Tabel *Confusion Matrix* model klasifikasi sentimen

Fold	***Confusion Matrix***			
	TP	**FN**	**FP**	**TN**
1				
2				
Ke-n				
Rata-rata				

Tabel 3.32 Tabel *Confusion Matrix* model klasifikasi kategori

Fold	***Confusion Matrix***															
	TP				**FN**				**FP**				**TN**			
	DT	**AK**	**KB**	**FS**	**DT**	**AK**	**KB**	**FS**	**DT**	**AK**	**KB**	**FS**	**DT**	**AK**	**KB**	**FS**
1																
2																
Ke-n																

Tabel 3.33 Tabel pengujian K-*fold cross validation* model klasifikasi sentimen

Fold	Akurasi	Presisi	Recall
1			
2			
Ke-n			
Rata-rata			

Tabel 3.34 Tabel pengujian K-*fold cross validation* model klasifikasi kategori

Fold	Akurasi	Presisi	Recall
1			
2			
Ke-n			
Rata-rata			

Tabel 3.35 Tabel nilai FPR dan TPR model sentimen

Fold	FPR	TPR
1		
2		
Ke-n		
Rata-rata		

Tabel 3.36 Tabel nilai FPR dan TPR model kategori

Fold	FPR				TPR			
	DT	**AK**	**KB**	**FS**	**DT**	**AK**	**KB**	**FS**
1								
2								
Ke-n								
Rata-rata								

Tabel 3.37 Tabel nilai AUC model klasifikasi sentimen

Fold	Nilai AUC
1	
2	
Ke-n	
Rata-rata	

Tabel 3.38 Tabel nilai AUC model klasifikasi kategori

Label	**Nilai AUC**
Daya Tarik	
Aksesbilitas	
Kebersihan	
Fasilitas	
Rata-rata	

BAB IV

HASIL, PENGUJIAN, DAN PEMBAHASAN

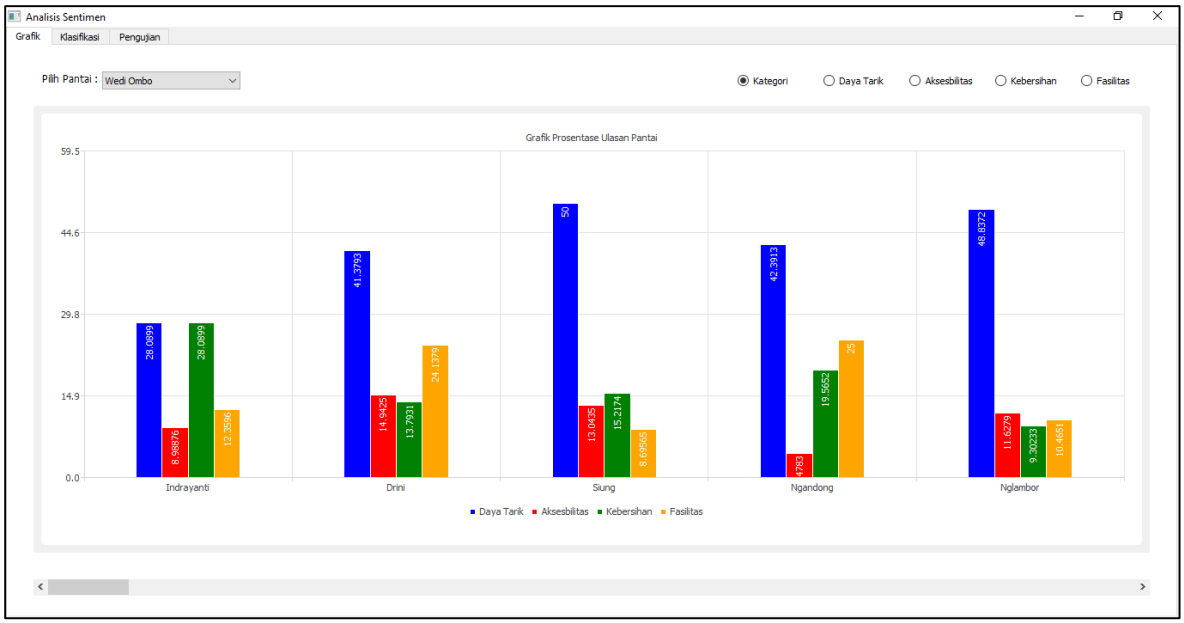
Dalam bab ini akan dijelaskan mengenai tahapan implementasi hasil perancangan yang terdapat pada bab tiga untuk menghasilkan *output* hasil analisis sentimen secara otomatis. Pada tahap ini juga akan dilakukan pembahasan untuk mengetahui apakah aplikasi yang dibuat benar-benar dapat menghasilkan tujuan yang diinginkan berdasarkan pada analisis dan perancangan. Setelah tahap implementasi selesai maka dilakukan tahap pengujian terhadap sistem untuk mengetahui kemampuan fitur dan algoritma yang digunakan. Pengujian menggunakan pendekatan *k-fold cross validation, confusion matrix* dan kurva ROC.

4.1 Implementasi Aplikasi

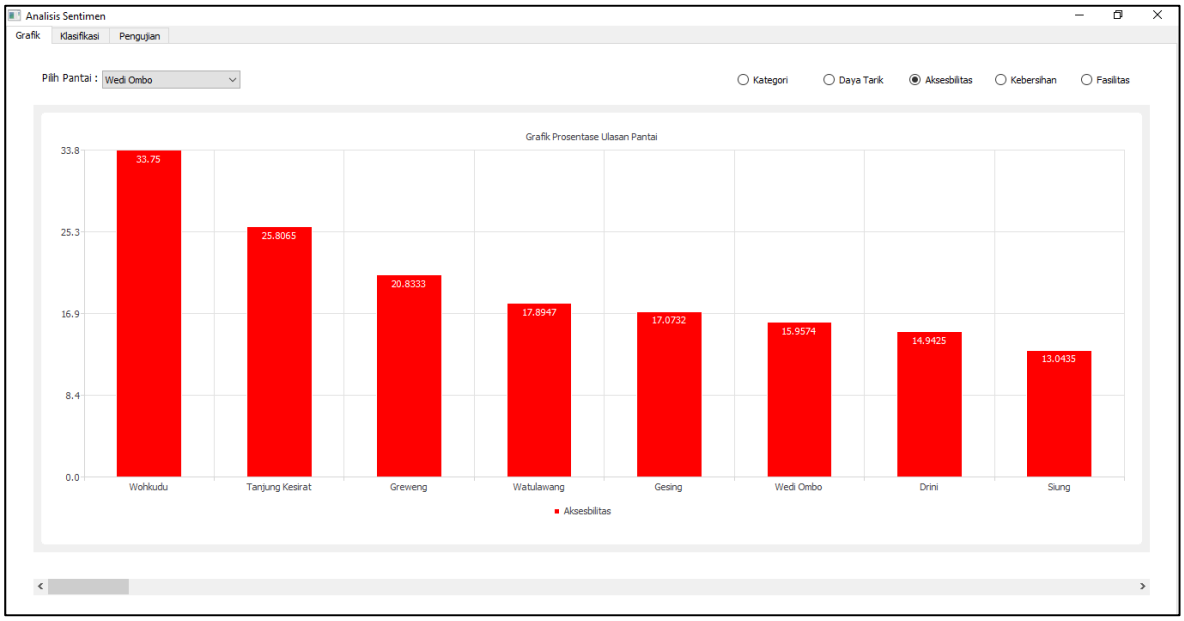
Implementasi aplikasi merupakan tampilan program saat aplikasi sedang berjalan. Berikut ini akan ditampilkan *screenshot* dari *Interface* aplikasi yang telah dibangun beserta *Source code* pembentuknya.

4.1.1 Halaman Grafik

Halaman grafik merupakan halaman awal ketika aplikasi dijalankan. Pada halaman ini *user* dapat melihat grafik ulasan pantai untuk masing-masing kategori secara keseluruhan dari 35 pantai yang ada. Grafik menampilkan persentase jumlah ulasan positif dari data pantai. Kemudian *user* dapat memilih *radio button* yang berisi opsi untuk menampilkan ulasan pantai berdasarkan kategori yang dipilih. Ulasan pantai berdasarkan kategori ditampilkan berupa grafik secara berurutan dari persentase tertinggi hingga paling rendah. Selain itu, *user* juga dapat memilih nama pantai yang ingin dilihat secara jelas persentasenya. *User* memilih nama pantai melalui menu *combo box*. Tampilan halaman grafik pantai secara keseluruhan dapat dilihat pada gambar 4.1 sedangkan tampilan halaman grafik berdasarkan kategori yang dipilih dapat dilihat pada gambar 4.2.

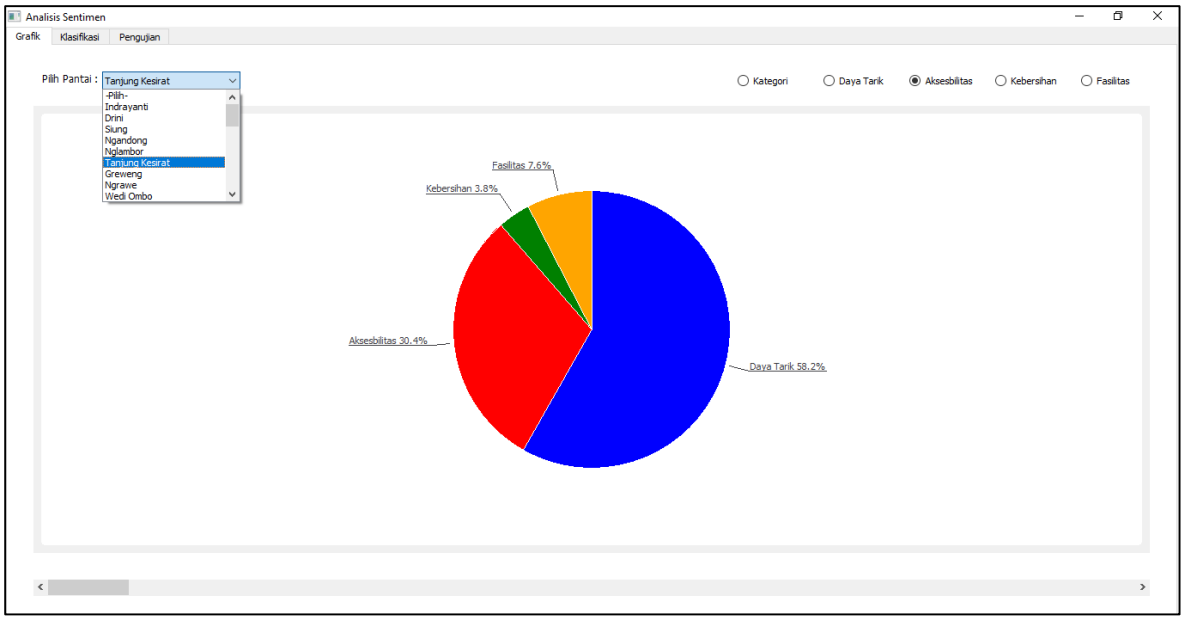


Gambar 4.1 Halaman Grafik Semua Pantai



Gambar 4.2 Halaman Grafik per kategori

Sedangkan tampilan halaman grafik untuk pantai yang dipilih pada *combo box* dapat dilihat pada gambar 4.3.



Gambar 4.3 Halaman Grafik per pantai

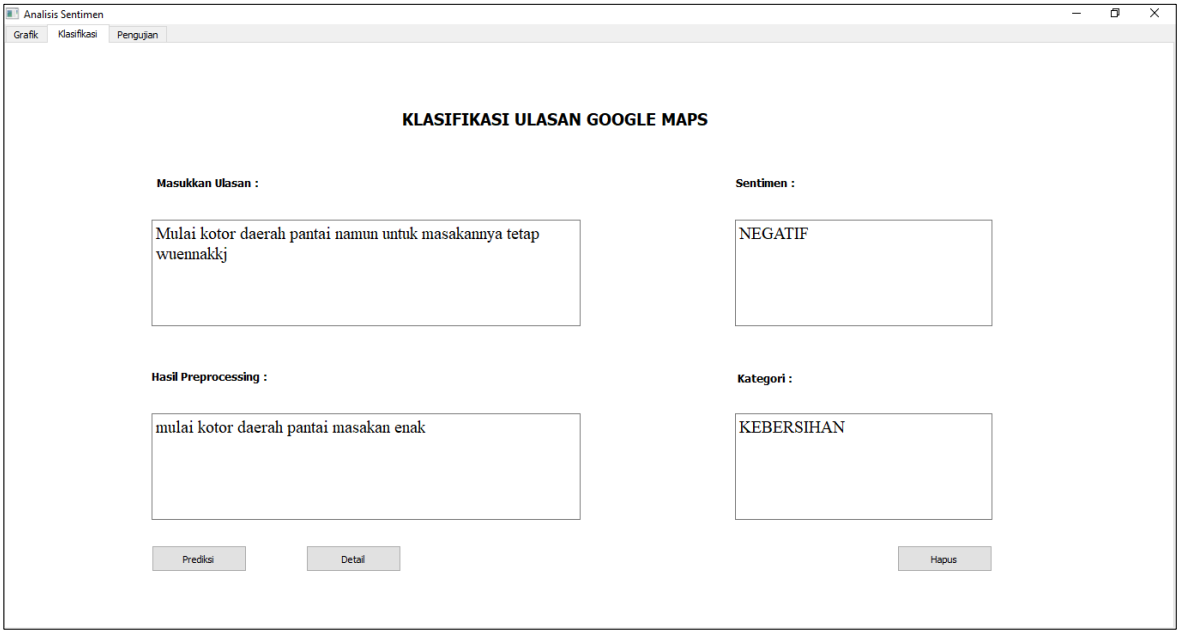
Nilai grafik diambil dari *database* yang berisi *datasets* yang telah diklasifikasikan sentimen dan kategori menggunakan model LSTM. Berikut *Source code* fungsi untuk menampilkan nilai positif dari masing-masing pantai dapat dilihat pada modul program 4.1.

```
def jenis_kategori(p, k):
    db = pymysql.connect(
    host='localhost',
    user='root',
    passwd='',
    port=3306,
    db='db_anaisis_sentimen'
    )
      pantai=p
      kategori=k
      tot_ positif =0
    cursor = db.cursor()
    sql= "SELECT * from hasil_klasifikasi JOIN daftar_pantai ON
   hasil_klasifikasi.id_pantai=daftar_pantai.id WHERE
   hasil_klasifikasi.klasifikasi_sentimen='Positif' and
   daftar_pantai.nama_pantai='%s' and
   hasil_klasifikasi.klasifikasi_kategori= '%s'"%(pantai,kategori)
    try:
            cursor.execute(sql)
            results = cursor.fetchall()
            for row in results:
                  tot_ positif =tot_ positif +1
    except :
            print("error")
     return(tot_positif)
```

Modul Program 4.1 *Source code* menampilkan nilai positif

4.1.2 Halaman Klasifikasi

Halaman klasifikasi digunakan *user* untuk memasukkan data uji berupa ulasan. Data uji baru itulah yang nanti akan diprediksi oleh sistem untuk diketahui sentimen dan kategorinya. Tampilan dari halaman klasifikasi dapat dilihat pada gambar 4.4.



Gambar 4.4 Halaman Klasifikasi

Ketika *user* menekan tombol prediksi maka data uji tersebut akan diproses dalam sistem sehingga menghasilkan keluaran berupa hasil klasifikasi sentimen dan kategori. Hasil dari *preprocessing text* sebagai langkah awal dalam melakukan klasifikasi juga akan ditampilkan pada halaman klasifikasi. Dalam melakukan klasifikasi terdapat beberapa proses dengan cara kerja yang berbeda-beda. Berikut merupakan *Source code* dari masing-masing proses awal (*text preprocessing*) hingga prediksi model dengan algoritma LSTM:

4.1.2.1 Implementasi *Text Preprocessing*

Data ulasan yang diinputkan oleh *user* pertama-tama akan dilakukan proses *text preprocessing*. Proses ini terdiri atas 8 tahap yaitu *case folding, remove punctuation, remove number, spelling correction, negation word conversion, tokenizing, stopword removal,* dan *stemming*.

1. Implementasi *Case Folding*

Pada proses ini, data ulasan akan dilakukan penyamarataan bentuk huruf menjadi huruf kecil semua (*lowercase*). Sehingga jika ada huruf kapital maka akan diganti dengan huruf kecil. *Source code* proses *case folding* ditunjukkan pada modul program 4.2.

```
def case_folding(tokens):
     return tokens.lower()

data_casefolding=[]
for i in range(0, len(data)):
   data_casefolding.append(case_folding(data[i]))
```

Modul Program 4.2 *Source code* proses *case folding*

2. Implementasi *Remove Punctuation*

Setelah proses *case folding,* data ulasan hasil proses *case folding* akan dibersihkan dari tanda baca dan simbol. *Source code* pada proses *remove punctuation* ditunjukkan pada modul program 4.3.

```
def remove_punct(text):
      text_nopunct = ''
      text_nopunct = re.sub('['+string.punctuation+']', '', text)
      return text_nopunct

data_remove=[]
for i in range(0, len(data)):
   data_remove.append(remove_punct(data_casefolding[i]))
```

Modul Program 4.3 *Source code* proses *remove punctuation*

3. Implementasi *Remove Number*

Setelah proses *remove punctuation,* data ulasan hasil proses *remove punctuation* akan dibersihkan juga dari angka. *Source code* pada proses *remova number* ditunjukkan pada modul program 4.4.

```
def remove_num(text):
      text_nonum = ''
      text_nonum = re.sub(r'\d+','', text)
      return text_nonum

data_remove_num=[]
for i in range(0, len(data)):
   data_remove_num.append(remove_num(data_remove[i]))
```

Modul Program 4.4 *Source code* proses *remove number*

4. Implementasi *Spelling Correction*

Setelah proses *remove number*, data ulasan hasil proses *remove number* akan dilakukan pengecekan terhadap kamus *spelling_word*. Pengecekan tersebut dilakukan jika terdapat kesalahan penulisan kata, kata yang merupakan singkatan, atau sinonim agar dapat diperbaiki dengan kata asli atau kata baku. *Source code* pada proses *spelling correction* ditunjukkan pada modul program 4.5.

```
def open_kamus_prepro(x):
   kamus={}
   with open(x,'r') as file :
      for line in file :
         spelling=line.replace("'","").split(':')
         kamus[spelling[0].strip()]=spelling[1].rstrip('\n').lstrip()
   return kamus
kamus_spelling=open_kamus_prepro('Kamus spelling_word.txt')

def spelling(text):
   sentence_list = text.split()
   new_sentence = []
   for word in sentence_list:
      for candidate_replacement in kamus_spelling:
        if candidate_replacement == word:
            word = word.replace(candidate_replacement,kamus_spelling
            [candidate_replacement])
      new_sentence.append(word)
    return " ".join(new_sentence)

spell=[]
for i in range(0, len(data)):
   spall.append(spalling(data_remove_num[i]))
```

Modul Program 4.5 *Source code* proses *spelling correction*

5. Implementasi *Negation Word Conversion*

Setelah proses *spelling correction*, data ulasan hasil proses *spelling correction* akan dilakukan pengecekan apakah setelah kata “tidak” terdapat kata yang ada di kamus *negation_word*. Pengecekan tersebut dilakukan untuk mengganti kata tersebut menjadi negasinya. *Source code* pada proses *negation word conversion* ditunjukkan pada modul program 4.6.

```
kamus_negasi=open_kamus_prepro('Kamus negation_word.txt')
def ganti_negasi(w):
   w_splited = w.split(' ')
   if 'tidak' in w_splited:
      index_negasi = w_splited.index('tidak')
      for i,k in enumerate(w_splited):
         if k in kamus_negasi and w_splited[i-1] == 'tidak':
            w_splited[i] = kamus_negasi[k]
   return ' '.join(w_splited)

negasi=[]
for i in range(0, len(data)):
  negasi.append(ganti_negasi(spall[i]))
```

Modul Program 4.6 *Source code* proses *Negation Word Conversion*

6. Implementasi *Tokenizing*

Setelah proses *negation word conversion*, data ulasan hasil proses *negation word conversion* akan dipecah menjadi perkata atau token-token. Proses tersebut dilakukan dengan cara mendeteksi spasi antar kata. Jika ditemukan spasi maka kata tersebut akan dipotong sehingga menjadi token-token. *Source code* pada proses *tokenizing* ditunjukkan pada modul program 4.7.

```
from nltk import word_tokenize

tokens = [word_tokenize(sen) for sen in negasi]
```

Modul Program 4.7 *Source code* proses *tokenizing*

7. Implementasi *Stopword Removal*

Setelah proses *tokenizing*, data ulasan hasil proses *tokenizing* akan dilakukan pengecekan terhadap daftar kata di kamus *stopword*. Jika kata pada data ulasan ditemukan pada daftar kata *stopword*, maka kata tersebut akan dihapus. *Source code* pada proses *stopword removal* ditunjukkan pada modul program 4.8.

```
kamus_stopword=[]
   with open('Kamus stopword.txt','r') as file :
      for line in file :
         stop=line.replace("'","").strip()
         kamus_stopword.append(stop)
def remove_stop_words(tokens):
    return [word for word in tokens if word not in kamus_stopword]

filtered_words = [remove_stop_words(sen) for sen in tokens]
```

Modul Program 4.8 *Source code* proses *stopword removal*

8. Implementasi *Stemming*

Setelah proses *stopword removal*, data ulasan hasil proses *stopword removal* akan dilakukan perubahan kata yang memiliki imbuhan menjadi kata dasar. Proses *stemming* ini menerapkan metode terbaru yaitu menggunakan Sastrawi. *Stemmer* Sastrawi mengimplementasikan algoritma Nazief & Adriani. *Source code* pada proses *stemming* ditunjukkan pada modul program 4.9.

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

factory = StemmerFactory()
stemmer = factory.create_stemmer()

def stemming(tokens):
   data_stem =[]
   for i in tokens:
      kata = stemmer.stem(i)
      data_stem.append(kata)
   return data_stem

stem=[]
for i in range(0, len(data)):
   stem.append(stemming(filtered_words[i]))
```

Modul Program 4.9 *Source code* proses *Stemming*

4.1.2.2 Implementasi *Sentence Conversion*

Pada tahap ini, data ulasan yang sudah melalui proses *text preprocessing* akan di ubah menjadi angka dengan menggunakan fungsi *Tokenizer*. Pada tahap ini tidak membuat *index* dengan *tokenizer* baru, melainkan memakai hasil *tokenizer* dari *datasets* yang sebelumnya disimpan dengan nama *tokenize.pickle*. *Tokenizer* tersebut dijadikan sebagai acuan *index* pada proses mengubah data ulasan menjadi angka. Setelah diubah menjadi angka, selanjutnya ditambahkan padding angka 0 agar panjang ukuran data ulasan sama dengan data latih, yaitu dengan panjang 50. *Source code* pada proses *tokenizer* ditunjukkan pada modul program 4.10.

```
import pickle

with open('Tokenize.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)

MAX_SEQUENCE_LENGTH = 50
sequences = tokenizer.texts_to_sequences(stem)

test_data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)
```

Modul Program 4.10 *Source code* proses *Tokenizer*

4.1.2.3 Prediksi

Setelah data ulasan telah diubah menjadi angka dan ditambahkan *padding*, maka tahap selanjutnya adalah prediksi menggunakan model LSTM. Karena model telah dibuat sebelumnya melalui proses *training*, maka pada tahap ini model yang telah disimpan akan di *load* dan dijadikan acuan untuk melakukan prediksi sentimen dan kategori. Model terdiri dari model klasifikasi sentimen dan model klasifikasi kategori. Modul program 4.11 merupakan *Source code* untuk mengunggah file *model* klasifikasi.

```
json_file_sentimen = open('saved_model_sentimen.json', 'r')
model_json_sentimen = json_file_sentimen.read()
json_file_sentimen.close()
model_sentimen = model_from_json(model_json_sentimen)
model_sentimen.load_weights('saved_model_sentimen.h5')

json_file_kategori = open(saved_model_kategori.json', 'r')
model_json_kategori = json_file_kategori.read()
json_file_kategori.close()
model_kategori = model_from_json(model_json_kategori)
model_kategori.load_weights('saved_model_kategori.h5')

with open('saved_tokenize.pickle', 'rb') as handle:
      tokenizer = pickle.load(handle)
```

Modul Program 4.11 *Source code load model*

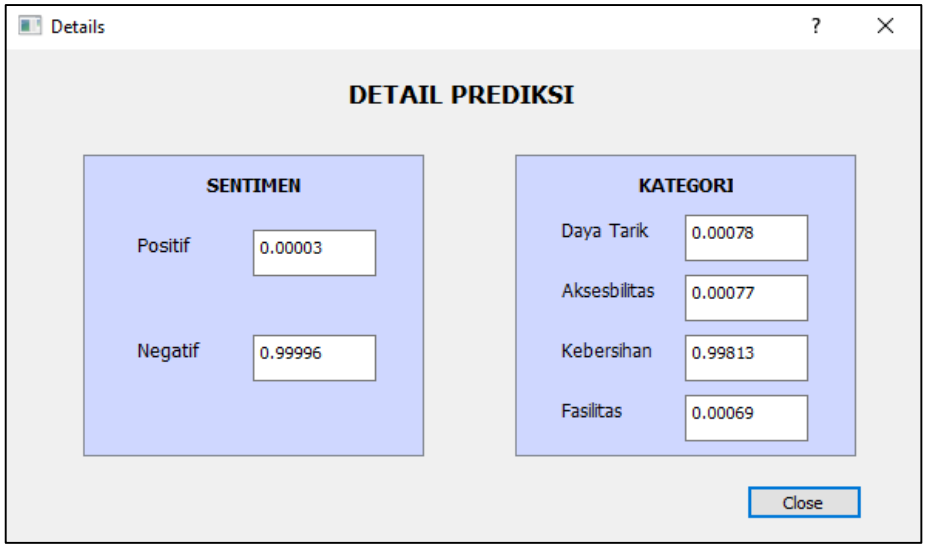
Proses prediksi data baru yang diinputkan oleh *user* dilakukan dengan fungsi *predict*. Prediksi dilakukan sebanyak dua kali, pertama menggunakan model sentimen dan kedua menggunakan model kategori. Selanjutnya hasil prediksi dan nilai probabilitas masing-masing *class* disimpan dalam variabel. Modul program 4.12 merupakan *Source code* untuk melakukan prediksi data input ulasan berdasarkan model.

```
prediksi_kategori = model_kategori.predict(x_test, batch_size=1,
verbose=1)
prediksi_sentimen = model_sentimen.predict(x_test, batch_size=1,
verbose=1)

predict_sentimen_positif=prediksi_sentimen[0][0]
predict_sentimen_negatif=prediksi_sentimen[0][1]

predict_DayaTarik=prediksi_kategori[0][0]
predict_Aksesbilitas=prediksi_kategori[0][1]
predict_Kebersihan=prediksi_kategori[0][2]
predict_Fasilitas=prediksi_kategori[0][3]

class_category = ['DAYA TARIK', 'AKSESBILITAS','KEBERSIHAN','FASILITAS']
class_sentimen = ['POSITIF', 'NEGATIF']

for i in range(prediksi_kategori.shape[0]):
      print(str(i) + " " + class_sentimen[prediksi_sentimen[i].argmax()]
" , " + class_category[prediksi_kategori[i].argmax()] + " : " + data[i])
      hasil_sentimen = class_sentimen[prediksi_sentimen[i].argmax()]
      hasil_kategori = class_category[prediksi_kategori[i].argmax()]

return (hasil_sentimen,hasil_kategori,prepro[0],predict_sentimen_positif,
predict_sentimen_negatif,predict_DayaTarik,predict_Aksesbilitas,predict_K
ebersihan,predict_Fasilitas)
```

Modul Program 4.12 *Source code* prediksi

4.1.3 Halaman Detail

Halaman detail merupakan halaman yang muncul ketika *user* menekan tombol detail pada halaman klasifikasi. Pada halaman ini berisi nilai probabilitas perlabel pada sentimen dan kategori. Tampilan dari halaman detail dapat dilihat pada gambar 4.5.



Gambar 4.5 Halaman Detail Prediksi

Nilai probabilitas yang akan ditampilkan pada detail prediksi diperoleh dari hasil prediksi pada proses sebelumnya. Kemudian masing-masing nilai tersebut, baik pada klasifikasi sentimen maupun kategori akan ditampilkan pada halaman detail prediksi sesuai dengan kelasnya. Modul program 4.13 merupakan *Source code* fungsi menampilkan nilai pada halaman detail prediksi.

```
Def setDetail(self,set_positif, set_negatif, set_dayatarik,  set_aksesbi
litas, set_kebersihan,set_fasilitas):
        self.setPositif.setText(str('%.5f' %set_positif))
        self.setNegatif.setText(str('%.5f' %set_negatif))
        self.setDayaTarik.setText(str('%.5f' %set_dayatarik))
        self.setAksesbilitas.setText(str('%.5f' %set_aksesbilitas))
        self.setKebersihan.setText(str('%.5f' %set_kebersihan))
        self.setFasilitas.setText(str('%.5f' %set_fasilitas))
```

Modul Program 4.13 *Source code* fungsi menampilkan nilai detail prediksi

4.1.4 Halaman Pengujian

Halaman pengujian merupakan halaman bagi *user* untuk melihat hasil pengujian performa algoritma klasifikasi *long short term memory* dalam bentuk grafik. Pengujian ini menggunakan data latih dan data uji dengan perbandingan 80:20. Pada pengujian ini terdapat 2 tahapan yaitu tahap implementasi *training* (latih) dan *testing* (uji). Sehingga *datasets* hasil *preprocessing* dari *datasets* akan di *split* sebesar 20% untuk data uji. Proses *split* menggunakan *library train_test_split* dengan melakukan pengacakan data sebanyak 42 kali. Modul program 4.14 merupakan *Source code* proses *split* atau pembagian data.

```
from sklearn.model_selection import train_test_split

data_train,data_test=train_test_split(data,test_size=0.2,random_state=42)
```

Modul Program 4.14 *Source code* split data latih dan data uji

4.1.4.1 Implementasi Training (Latih)

Proses *training* menggunakan 80% *datasets* yang disebut dengan data latih. Data latih merupakan data yang sudah diberi label sentimen dan kategori. Data latih nantinya akan digunakan sebagai acuan dalam pengklasifikasian ulasan dari data uji (*testing*). Proses *training* terdiri dari beberapa langkah. Pertama, hasil *preprocessing* diubah menjadi angka

dengan *tokenizer*. Kedua menyiapkan *word embedding*, ketiga membuat model, keempat melakukan proses *training*, dan terakhir menyimpan model klasifikasi.

a. Tahap *Tokenizer*

Data latih yang berupa teks akan dikonversi menjadi urutan *integer index* dengan menggunakan fungsi *tokenizer* yang tersedia di *library Keras*. Nantinya hasil *tokenizer* akan disimpan pada file dengan ekstensi *pickle*. File tersebut akan digunakan untuk menyimpan data token kata beserta bobot sekuensinya guna keperluan untuk prediksi data uji. Implementasi *Tokenizer* dapat dilihat pada modul program 4.15.

```
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer()
tokenizer.fit_on_texts(data_train["Text_Final"].tolist())

with open('Tokenize.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
model_json = model.to_json()
```

Modul Program 4.15 *Source code* implementasi *tokenizer*

Setelah urutan *index* pada data latih sudah dibuat, kemudian hasil tersebut digunakan sebagai acuan dalam mengubah teks menjadi angka pada data latih. Implementasi proses mengubah teks menjadi angka dan menambahkan *padding* pada data latih dapat dilihat pada modul program 4.16.

```
from keras.preprocessing.sequence import pad_sequences

training_sequences=tokenizer.texts_to_sequences(data_train["Text_Final"].
tolist())

train_lstm_data=pad_sequences(training_sequences,maxlen=MAX_SEQUENCE_LENG
TH)
```

Modul Program 4.16 *Source code* mengubah ke angka dan menambah *padding*

b. Tahap *Word Embedding*

Model *embedding* yang digunakan dalam penelitian ini membuat sendiri dengan *datasets* menggunakan fungsi *word2vector* dengan 100 dimensi. Model disimpan dengan ektensi .txt yang selanjutnya akan digunakan untuk *word embedding*. Implementasi proses *word2vec* dan *load* model *embedding* dapat dilihat pada modul program 4.17.

```
from gensim.models import Word2Vec

model = Word2Vec(data['tokens']  , size=100, min_count=1)
model.wv.save_word2vec_format('word2vec.txt',binary=False)

word2vec_path = 'word2vec.txt'
word2vec = models.KeyedVectors.load_word2vec_format(word2vec_path,
binary=True)
```

Modul Program 4.17 *Source code* implementasi *word2vec*

Setelah model *word2vec* di simpan dalam variabel *word2vec*, maka selanjutnya membuat variabel baru dengan nama *train_embedding_weights* untuk menyimpan bobot-bobot vektor kata pada seluruh kata yang ada di *datasets*. Dengan menggunakan fungsi perulangan, satu persatu kata akan diperiksa, jika kata tersebut terdapat pada *word2vec* maka bobot akan disimpan. Jika tidak ada maka akan mencari bobot dengan fungsi *random*. Selanjutnya *train_embedding_weights* nantinya akan dimasukkan dalam parameter bobot untuk *embedding layer* pada arsitektur model. *Source code* proses perulangan untuk menyimpan bobot kata berdasarkan *word2vec* dapat dilihat pada modul program 4.18.

```
train_embedding_weights=np.zeros((len(train_word_index)+1,EMBEDDING_DIM))

for word,index in train_word_index.items():
train_embedding_weights[index,:] = word2vec[word] if word in word2vec else
np.random.rand(EMBEDDING_DIM)
```

Modul Program 4.18 *Source code* implementasi *word2vec*

c. Pembuatan Arsitektur Model

Arsitektur model dibuat dengan memilih jenis dan *hyperparameter layer* pada tiap lapisan. Lapisan pertama adalah *layer input*. *Layer input* memiliki parameter *shape* dengan angka 50 dan tipe data *int32*. *Layer* selanjutnya adalah *layer embedding* yang sudah diinisialisasi sebelum pembuatan model. Setelah *layer input* dan *embedding*, selanjutnya adalah *layer* LSTM. *Layer* LSTM menggunakan 256 unit *neuron* dan menambahkan *dropout* dengan probabilitas 20. Kemudian *dense output layer* dengan 2 unit *neuron* untuk model klasifikasi sentimen dan 4 unit *neuron* untuk model klasifikasi kategori, serta fungsi aktivasi

menggunakan *sigmoid*. Implementasi pembuatan arsitektur model dapat dilihat pada modul program 4.19.

```
from keras.layers import Dense, Dropout, Input, Embedding
from keras.layers.recurrent import LSTM

def recurrent_nn(embeddings,max_sequence_length,num_words, embedding_dim,
labels_index):
     embedding_layer = Embedding(num_words,
                                 embedding_dim,
                                 weights=[embeddings],
                                 input_length=max_sequence_length,
                                 trainable=True)

    sequence_input = Input(shape=(max_sequence_length,), dtype='int32')
    embedded_sequences = embedding_layer(sequence_input)

    lstm = LSTM(256)(embedded_sequences)

    x = Dense(128, activation='relu')(lstm)
    x = Dropout(0.2)(x)
    preds = Dense(labels_index, activation='sigmoid')(x)

    model = Model(sequence_input, preds)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['acc'])
    model.summary()
    return model

model    =    recurrent_nn(train_embedding_weights,    MAX_SEQUENCE_LENGTH,
len(train_word_index)+1, EMBEDDING_DIM,len(list(label_names)))
```

Modul Program 4.19 *Source code* implementasi pembuatan arsitektur model

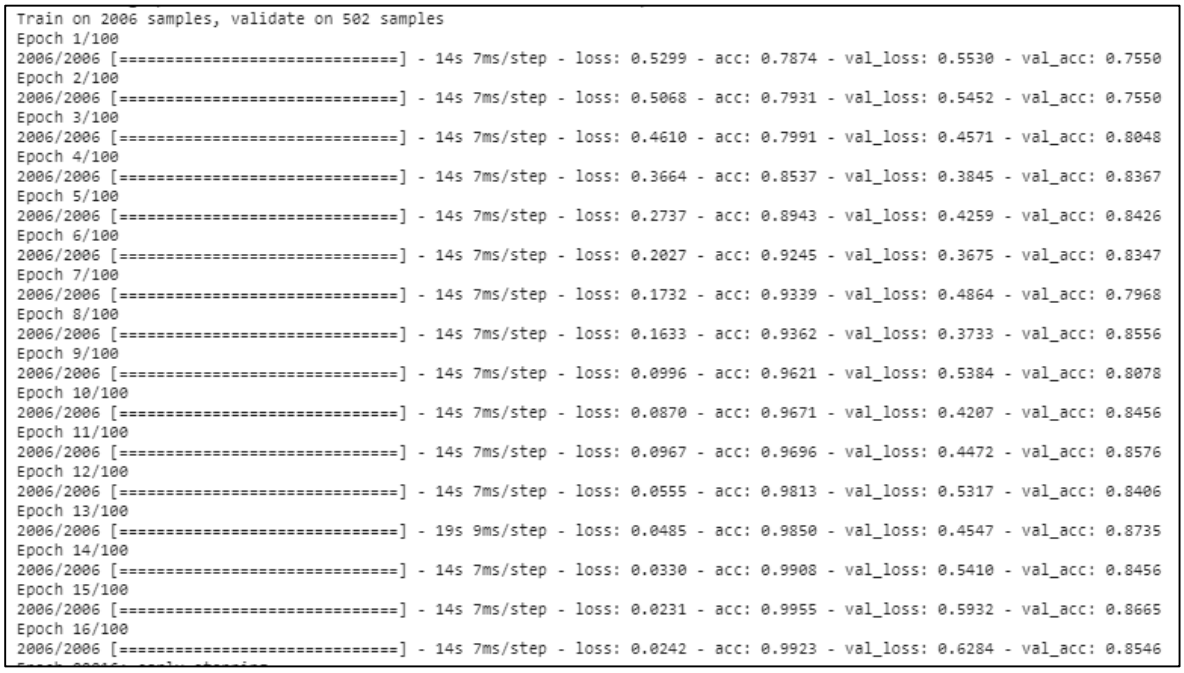
d. *Training* Model

Dari hasil percobaan, jumlah *batch size* yang menghasilkan akurasi terbaik adalah 32. Sedangkan jumlah *epoch* yang dipakai adalah 50. Namun untuk mengurangi *overfitting*, iterasi akan dihentikan ketika model *overfitting* dengan menggunakan fungsi *Early Stopping*. Proses *training* menggunakan data validasi 20% dari data uji. Implementasi *training* model dapat dilihat pada modul program 4.20.

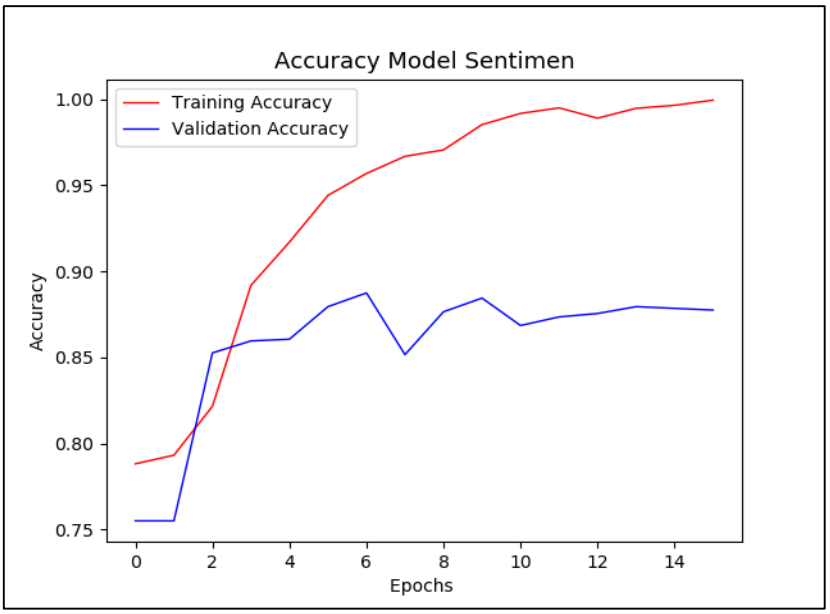
```
from keras.callbacks import EarlyStopping

num_epochs = 50
batch_size = 32

es_callback =EarlyStopping(monitor='val_loss', patience=10, verbose=1)

hist      =      model.fit(x_train,      y_train,      epochs=num_epochs,
validation_split=0.2, shuffle=True , batch_size=batch_size)
```

Modul Program 4.20 *Source code* implementasi *training* model

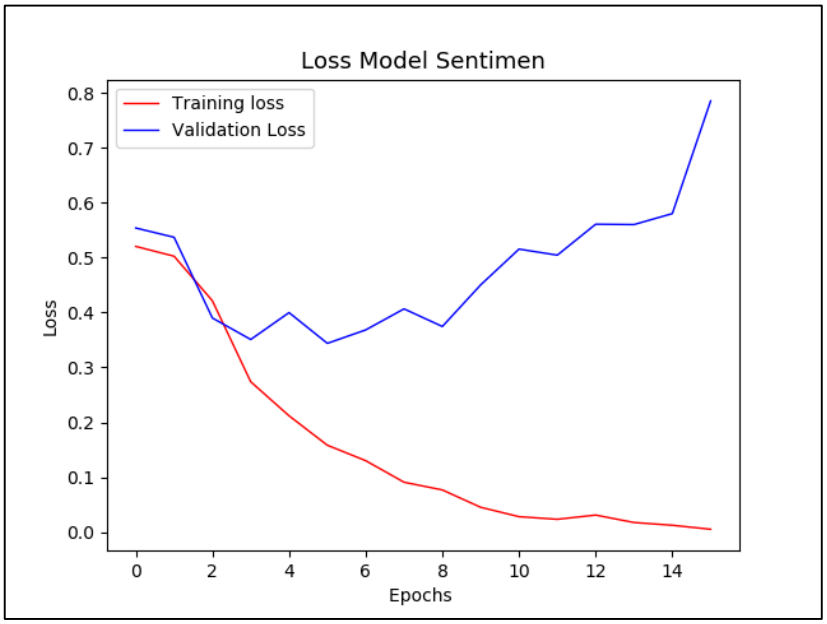
Model yang dibuat dan diberi *input* akan mempelajari pola yang ada pada data latih dan melakukan validasi pada setiap akhir *epoch*. Proses *training* model sentimen dapat dilihat pada gambar 4.6, sedangkan grafik *accuracy* dan *loss* model sentimen dapat dilihat pada gambar 4.7 dan 4.8.



Gambar 4.6 Proses *Training* Sentimen



Gambar 4.7 Grafik *Accuracy* Sentimen

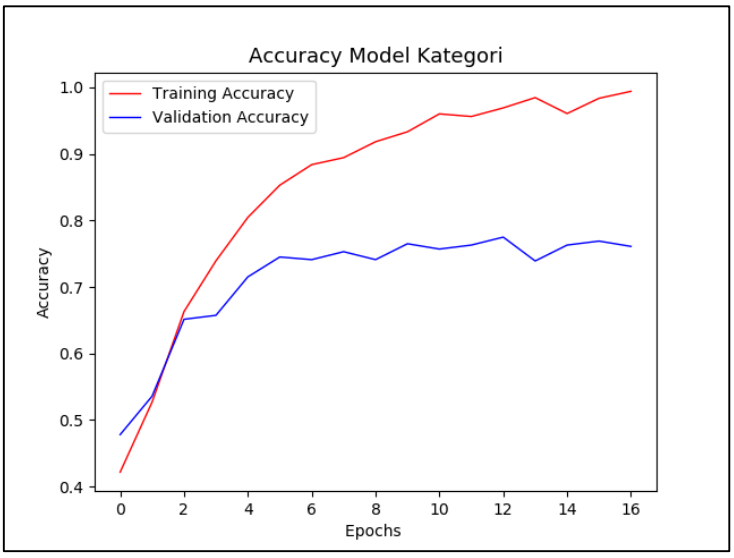


Gambar 4.8 Grafik *Loss* Sentimen

Proses *training* model kategori dapat dilihat pada gambar 4.9, sedangkan grafik *accuracy* dan *loss* model kategori dapat dilihat pada gambar 4.10 dan 4.11.

```
Train on 2006 samples, validate on 502 samples
Epoch 1/100
2006/2006 [==============================] - 15s 7ms/step - loss: 0.5365 - acc: 0.7547 - val_loss: 0.5117 - val_acc: 0.7659
Epoch 2/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.5080 - acc: 0.7668 - val_loss: 0.5084 - val_acc: 0.7654
Epoch 3/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.4583 - acc: 0.7951 - val_loss: 0.4338 - val_acc: 0.8073
Epoch 4/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.3926 - acc: 0.8278 - val_loss: 0.4358 - val_acc: 0.7869
Epoch 5/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.3008 - acc: 0.8772 - val_loss: 0.3333 - val_acc: 0.8566
Epoch 6/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.2554 - acc: 0.8962 - val_loss: 0.3322 - val_acc: 0.8561
Epoch 7/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.2007 - acc: 0.9242 - val_loss: 0.2755 - val_acc: 0.8964
Epoch 8/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.1646 - acc: 0.9391 - val_loss: 0.2915 - val_acc: 0.8815
Epoch 9/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.1349 - acc: 0.9514 - val_loss: 0.3051 - val_acc: 0.8894
Epoch 10/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.1172 - acc: 0.9554 - val_loss: 0.3294 - val_acc: 0.8904
Epoch 11/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.1158 - acc: 0.9566 - val_loss: 0.3351 - val_acc: 0.8810
Epoch 12/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.0899 - acc: 0.9652 - val_loss: 0.3287 - val_acc: 0.8780
Epoch 13/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.0696 - acc: 0.9741 - val_loss: 0.3527 - val_acc: 0.8810
Epoch 14/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.0449 - acc: 0.9834 - val_loss: 0.4590 - val_acc: 0.8715
Epoch 15/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.0390 - acc: 0.9870 - val_loss: 0.4134 - val_acc: 0.8909
Epoch 16/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.0320 - acc: 0.9888 - val_loss: 0.4317 - val_acc: 0.8870
Epoch 17/100
2006/2006 [==============================] - 14s 7ms/step - loss: 0.0232 - acc: 0.9926 - val_loss: 0.4757 - val_acc: 0.8884
Epoch 00017: early stopping
```

Gambar 4.9 Proses *Training* Kategori



Gambar 4.10 Grafik *Accuracy* Kategori

Gambar 4.11 Grafik *Loss* Kategori

e. Penyimpanan Model

Model akan disimpan di file berekstensi .json dan .h5. Model.json digunakan untuk menyimpan konfigurasi model pada *keras*. Model.h5 digunakan untuk menyimpan bobot-bobot dari sinapsis jaringan syaraf tiruan. Modul program 4.21 merupakan *Source code* untuk mmenyimpan model.

```
import json

with open('model_sentimen.json','w') as json_file:
  json_file.write(model_json)
model.save_weights("model_sentimen.h5")
print("Model saved to disk")
```

Modul Program 4.21 *Source code* implementasi proses menyimpan model

4.1.4.2 Implementasi *Testing* (Uji)

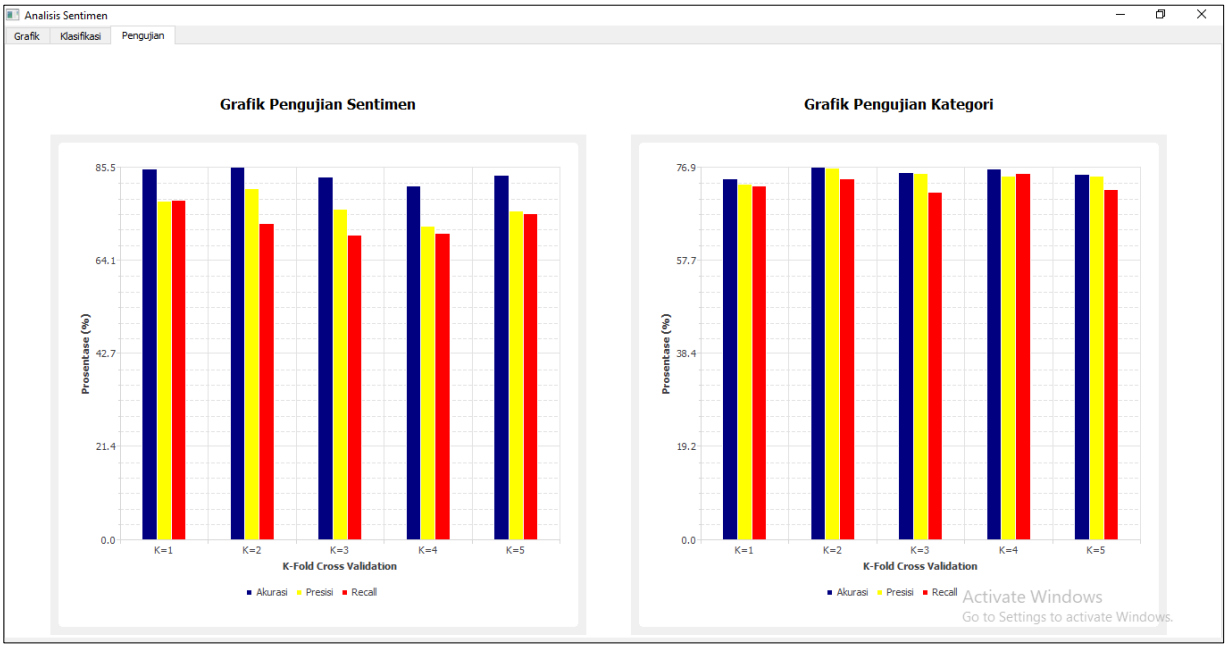
Data uji merupakan data yang dilakukan prediksi klasifikasi oleh sistem. Pada proses *testing* akan menggunakan 20% jumlah dari *datasets* ulasan dengan pendekatan k-fold *cross validation*. Proses pengujian menggunakan 5-*fold*, sehingga proses pengujian berlangsung sebanyak lima kali iterasi kemudian hasil akan dihitung nilai rata-ratanya. Setelah melalui proses *text preprocessing*, selanjutnya data uji diubah menjadi angka. Kemudian satu persatu data uji akan di prediksi dengan model training yang sudah dibuat sehingga dapat diketahui nilai probabilitas di setiap kelas sentimen dan kategorinya. Proses prediksi menggunakan

predict berdasarkan model *training* yang dibuat sebelumnya. Proses implementasi *testing* dapat dilihat pada modul program 4.22.

```
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)

for train, test in kfold.split(x, y):
   test_sequences=tokenizer.texts_to_sequences(data_test["Text_Final"].To
list())
   test_lstm_data=pad_sequences(test_sequences,maxlen=MAX_SEQUENCE_LENGT
H)
   predictions = model.predict(test_lstm_data, batch_size=1, verbose=1)
```

Modul Program 4.22 *Source code* implementasi *testing*

Pengujian ini diukur dengan menghitung nilai akurasi, presisi, dan *recall* dengan *confusion matrix*. Tampilan halaman pengujian dapat dilihat pada gambar 4.12.



Gambar 4.12 Halaman Pengujian

Perhitungan akurasi, presisi, dan *recall* menggunakan tabel *confusion matrix*. *Source code* pembentukan *confusion matrix* untuk sentimen dapat dilihat pada modul program 4.23, sedangkan untuk kategori dapat dilihat pada modul program 4.24.

```
from sklearn.metrics import confusion_matrix

cf_sentimen = pd.DataFrame(
                data=confusion_matrix(y_tes, y_pred, labels=labels),
                columns=labels,
                index=labels
            )
print(cf_sentimen)

tps_sentimen = {}
fps_sentimen  = {}
fns_sentimen  = {}
tns_sentimen  = {}
for label in labels:
  tps_sentimen[label] = cf_sentimen.loc[label, label]
  fps_sentimen[label] = cf_sentimen[label].sum() - tps_sentimen[label]
  fns_sentimen[label] = cf_sentimen.loc[label].sum() - tps_sentimen[label]

for label in set(y_tes):
  tns_sentimen[label]    =    len(y_tes)    -    (tps_sentimen[label]    +
fps_sentimen[label] + fns_sentimen[label])
```

Modul Program 4.23 *Source code confusion matrix* untuk sentimen

```
from sklearn.metrics import confusion_matrix

cf_kategori = pd.DataFrame(
    data=confusion_matrix(y_tes, y_pred, labels=labels),
    columns=labels,
    index=labels
)
cf_kategori

tps_kategori = {}
fps_kategori  = {}
fns_kategori  = {}
tns_kategori  = {}
for label in labels:
  tps_kategori[label] = cf_kategori.loc[label, label]
  fps_kategori[label] = cf_kategori[label].sum() - tps_kategori[label]
  fns_kategori[label] = cf_kategori.loc[label].sum() -
tps_kategori[label]

for label in set(y_tes):
  tns_kategori[label] = len(y_tes) - (tps_kategori[label] +
fps_kategori[label] + fns_kategori[label])
```

Modul Program 4.24 *Source code confusion matrix* untuk kategori

Source code perhitungan akurasi untuk sentimen dapat dilihat pada modul program 4.25, sedangkan untuk kategori dapat dilihat pada modul program 4.26.

```
accuracySentimen=sum(tps_sentimen.values())/len(y_tes)
```

Modul Program 4.25 *Source code* akurasi sentimen

```
accuracyKategori=sum(tps_kategori.values())/len(y_tes)
```

Modul Program 4.26 *Source code* akurasi kategori

Source code perhitungan presisi untuk sentimen dapat dilihat pada modul program 4.27, sedangkan untuk kategori dapat dilihat pada modul program 4.28.

```
tpfp_sentimen = [ai + bi for ai, bi in zip(list(tps_sentimen.values()),
list(fps_sentimen.values()))]

precision=[ai/bi if bi>0 else 0 for ai , bi in zip(list(tps_sentimen.values
()), tpfp_sentimen)]
precisionSentimen=sum(precision)/2
```

Modul Program 4.27 *Source code* presisi sentimen

```
tpfp_kategori = [ai + bi for ai, bi in zip(list(tps_kategori.values()),
list(fps_kategori.values()))]

precision=[ai/bi  if bi>0 else 0 for ai,bi in zip(list(tps_kategori.values
()),tpfp_kategori)]
precisionKategori=sum(precision)/4
print("Presisi : "+ str(precisionKategori))
```

Modul Program 4.28 *Source code* presisi kategori

Source code perhitungan *recall* untuk sentimen dapat dilihat pada modul program 4.29, sedangkan untuk kategori dapat dilihat pada modul program 4.30.

```
tpfp_sentimen = [ai + bi for ai, bi in zip(list(tps_sentimen.values()),
list(fps_sentimen.values()))]

precision=[ai/bi if bi>0 else 0 for ai,bi in zip(list(tps_sentimen.values
()),tpfp_sentimen)]
precisionSentimen=sum(precision)/2
```

Modul Program 4.29 *Source code recall* sentimen

```
tpfn_kategori = [ai + bi for ai, bi in zip(list(tps_kategori.values()),
list(fns_kategori.values()))]

recall=[ai/bi if bi>0 else 0 for ai, bi in zip(list(tps_kategori.values()),
tpfn_kategori)]
recallKategori=sum(recall)/4
print("Recall : "+ str(recallKategori))
```

Modul Program 4.30 *Source code recall* kategori

Selain menghitung nilai akurasi, presisi, dan *recall*, pengujian juga dilakukan dengan menggunakan kurva ROC dan menentukan nilai AUC. Kurva ROC dibuat dari hasil nilai TPR dan FPR dari masing-masing perulangan *fold*. Kemudian akan dirata-rata untuk diketahui nilai AUC rata-rata dari hasil pengujian. *Source code* untuk membuat kurva ROC dari pengujian model sentimen dapat dilihat pada modul program 4.31 dan 4.32.

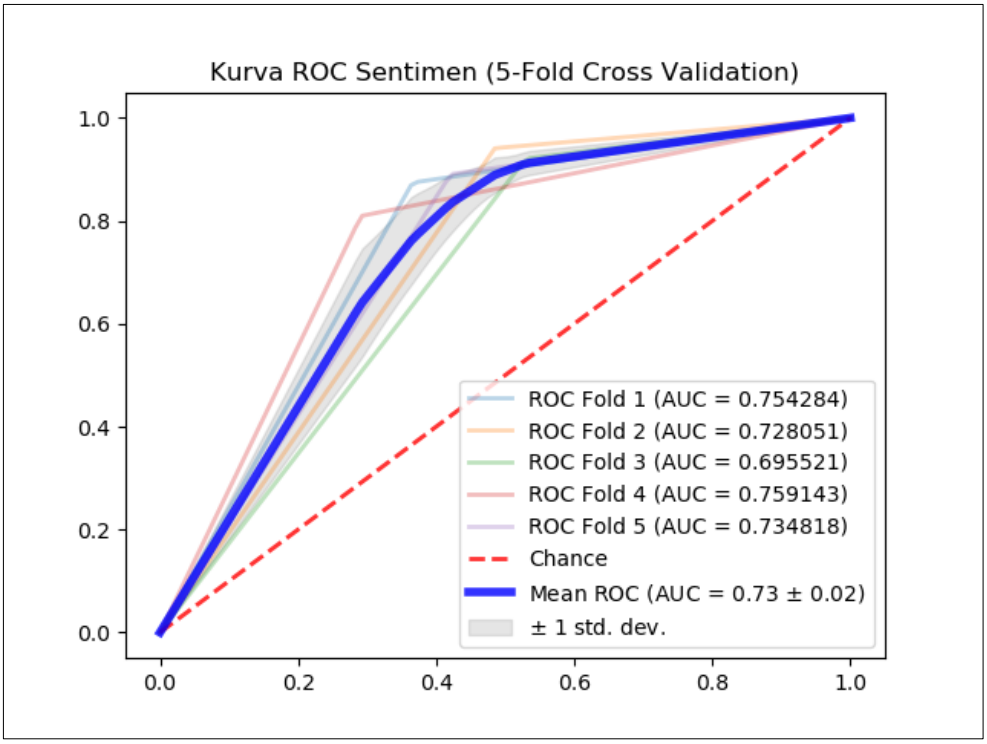
```
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)
fig, ax = plt.subplots()
fold=0

for train, test in kfold.split(x, y):
    labelsss=[1,0]
    y_predss=[]
    for p in predictions:
        y_predss.append(labelsss[np.argmax(p)])

    fpr, tpr, thresholds = roc_curve(y_tesss, y_predss)

    nilai_auc = auc(fpr, tpr)
    interp_tpr = interp(mean_fpr, fpr, tpr)
    interp_tpr[0] = 0.0
    tprs.append(interp_tpr)
    aucs.append(nilai_auc)

    ax.plot(mean_fpr, interp_tpr, label=r'ROC Fold %d (AUC = %f)' % (fold,
nilai_auc),
        lw=2, alpha=0.3)
    fold=fold+1
```

Modul Program 4.31 *Source code* kurva ROC sentimen

```
ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
        label='Chance', alpha=.8)

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)

ax.plot(mean_fpr, mean_tpr, color='b',
        label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
        lw=4, alpha=.8)

ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],
       title="Kurva ROC Sentimen (5-Fold Cross Validation)")

ax.legend(loc="lower right")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
fig.savefig('ROC Sentimen.png')
```

Modul Program 4.32 Lanjutan *Source code* kurva ROC sentimen

Kurva ROC model sentimen dapat dilihat pada gambar 4.13.



Gambar 4.13 Kurva ROC sentimen

Kurva ROC untuk model kategori tidak dapat dibuat secara langsung karena merupakan *multiclass*. Sehingga perlu melakukan pemisahan label dengan menggunakan fungsi *label_binarize* yang disediakan oleh modul *sklearn.preprocessing*. *Source code* untuk *label_binarize* dapat dilihat pada modul program 4.33.

```
labelsss=[1,2,3,4]
y_predss=[]
for p in predictions:
   y_predss.append(labelsss[np.argmax(p)])

y_binari=label_binarize(y[test],classes=['Daya Tarik','Aksesbilitas','Keb
ersihan','Fasilitas'])
n_classes=y_binari.shape[1]

y_prediksi=label_binarize(y_predss,classes=[1,2,3,4])
```

Modul Program 4.33 *Source code* fungsi *label_binarize*

Selanjutnya masing-masing label akan diplot dan disimpan dalam variabel yang nantinya digunakan untuk menghitung rata-rata. Proses tersebut berlangsung hingga lima kali perulangan *fold*. *Source code* untuk plot TFR dan FPR dapat dilihat pada modul program 4.34.

```
fpr = dict()
    tpr = dict()
    roc_auc = dict()
    for i in range(n_classes):
      fpr[i],tpr[i], _ = roc_curve(y_binari[:,i], y_prediksi[:,i])
      roc_auc[i]=auc(fpr[i],tpr[i])

colors=cycle(['blue','orange','black','red'])

for i, color in zip(range(n_classes),colors):
   if i == 0 :
      interp_tpr = interp(mean_fpr, fpr[i], tpr[i])
      interp_tpr[0] = 0.0
      kelas0_tpr.append(interp_tpr)
      kelas0_auc.append(roc_auc[i])
   elif i == 1 :
      interp_tpr = interp(mean_fpr, fpr[i], tpr[i])
      interp_tpr[0] = 0.0
      kelas1_tpr.append(interp_tpr)
      kelas1_auc.append(roc_auc[i])
   elif i == 2 :
      interp_tpr = interp(mean_fpr, fpr[i], tpr[i])
      interp_tpr[0] = 0.0
      kelas2_tpr.append(interp_tpr)
      kelas2_auc.append(roc_auc[i])
    elif i == 3 :
      interp_tpr = interp(mean_fpr, fpr[i], tpr[i])
      interp_tpr[0] = 0.0
      kelas3_tpr.append(interp_tpr)
      kelas3_auc.append(roc_auc[i])
    ax.plot(fpr[i],tpr[i], color=color, lw=1)
```

Modul Program 4.34 *Source code* plot TFR dan FPR

Selain mencari nilai TPR dan FPR untuk masing-masing label, nilai lain yang dicari adalah TPR dan FPR untuk *micro* dan *micro*. *Source code* untuk mencari TPR dan FPR *macro* dan *micro* dapat dilihat pada modul program 4.35.

```
fpr["micro"],tpr["micro"],_=roc_curve(y_binari.ravel(),y_prediksi.ravel()
)
roc_auc["micro"]=auc(fpr["micro"],tpr["micro"])
interp_tpr_micro = interp(mean_fpr, fpr["micro"], tpr["micro"])
interp_tpr_micro[0] = 0.0

micro_tpr.append(interp_tpr_micro)
micro_auc.append(roc_auc["micro"])
```

Modul Program 4.35 *Source code* plot TFR dan FPR *macro* dan *micro*

Saat proses perulangan *fold* sudah selesai, selanjutnya akan dihitung nilai rata-rata dari setiap label kemudian kurva ROC akan disimpan. *Source code* untuk mencari rata-rata setiap label serta nilai *macro* dan *micro* dapat dilihat pada modul program 4.36.

```
mean_tpr0 = np.mean(kelas0_tpr, axis=0)
mean_tpr0[-1] = 1.0
mean_auc0 = auc(mean_fpr, mean_tpr0)

mean_tpr1 = np.mean(kelas1_tpr, axis=0)
mean_tpr1[-1] = 1.0
mean_auc1 = auc(mean_fpr, mean_tpr1)

mean_tpr2 = np.mean(kelas2_tpr, axis=0)
mean_tpr2[-1] = 1.0
mean_auc2 = auc(mean_fpr, mean_tpr2)

mean_tpr3 = np.mean(kelas3_tpr, axis=0)
mean_tpr3[-1] = 1.0
mean_auc3 = auc(mean_fpr, mean_tpr3)

mean_micro=np.mean(micro_tpr,axis=0)
mean_micro[-1] = 1.0
mean_auc_micro = auc(mean_fpr, mean_micro)
```

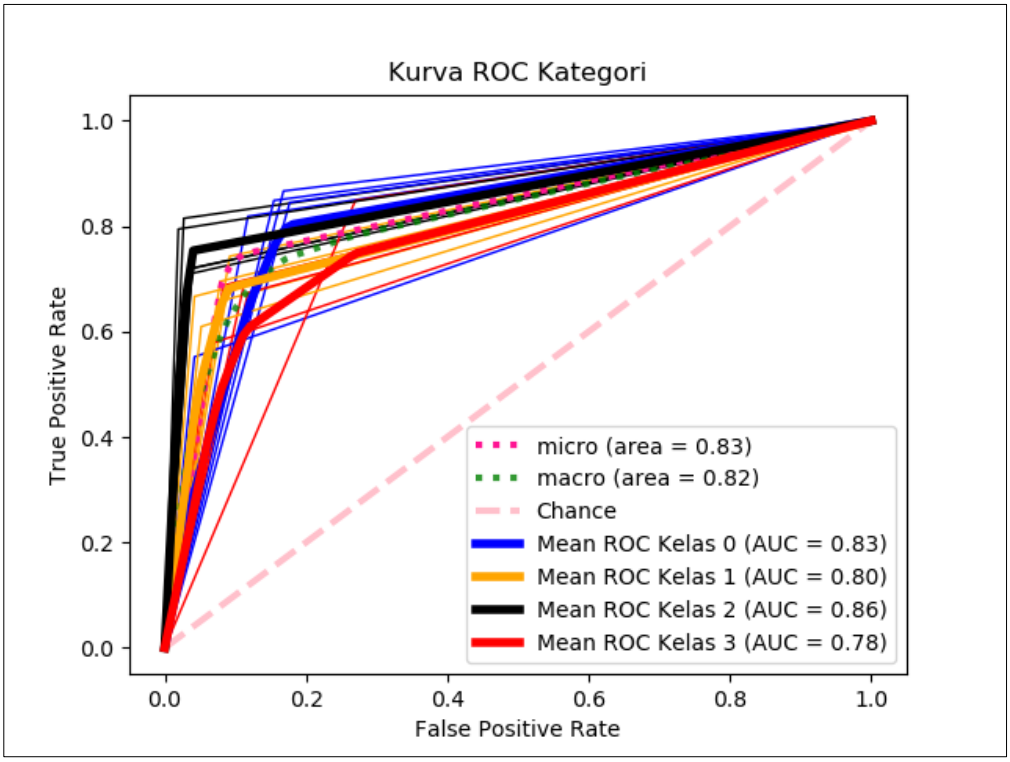
Modul Program 4.36 *Source code* mencari rata-rata

Terakhir, plot semua nilai yang dihasilkan sebelumnya kedalam kurva ROC dan disimpan. *Source code* untuk plot kurva ROC dapat dilihat pada modul program 4.37.

```
ax.plot(mean_fpr,      mean_micro,lw=3,      label='micro      (area      =
{0:0.2f})'''.format(mean_auc_micro),color='deeppink',linestyle=':')

mean_tpr = (mean_tpr0 + mean_tpr1 + mean_tpr2 + mean_tpr3)/4
mean_auc = (mean_auc0 + mean_auc1 + mean_auc2 + mean_auc3)/4
ax.plot(mean_fpr,     mean_tpr,     alpha=.8,label='macro     (area     =
{0:0.2f})'''.format(mean_auc),color='green',linestyle=':',linewidth=3)
ax.plot([0, 1], [0, 1], linestyle='--', lw=3, color='pink',
        label='Chance')
ax.plot(mean_fpr, mean_tpr0,
        label=r'Mean ROC Kelas 0 (AUC = %0.2f)' % (mean_auc0),
        lw=4,color='blue')
ax.plot(mean_fpr, mean_tpr1,
        label=r'Mean ROC Kelas 1 (AUC = %0.2f)' % (mean_auc1),
        lw=4, color='orange')
ax.plot(mean_fpr, mean_tpr2,
        label=r'Mean ROC Kelas 2 (AUC = %0.2f)' % (mean_auc2),
        lw=4, color='black')
ax.plot(mean_fpr, mean_tpr3,
        label=r'Mean ROC Kelas 3 (AUC = %0.2f)' % (mean_auc3),
        lw=4,color='red')

ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],
       title="Kurva ROC Kategori")
ax.legend(loc="lower right")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
fig.savefig('Kurva ROC Kategori.png')
```

Modul Program 4.37 *Source code* plot kurva ROC

Kurva ROC model kategori dapat dilihat pada gambar 4.14.



Gambar 4.14 Kurva ROC kategori

4.2 Pengujian

Tahap pengujian bertujuan untuk mengetahui tingkat keberhasilan proses klasifikasi yang telah dibangun. Implementasi pengujian pada penelitian ini dilakukan dengan membuat *confusion matrix* dan kurva ROC. Melalui *confusin matrix*, dapat diketahui nilai akurasi, presisi dan *recall*. Sedangkan kurva ROC digunakan untuk memvisualisasi dan menguji kinerja pengklasifikasian berdasarkan performanya. Pengujian metode LSTM pada penelitian ini dilakukan untuk dua model, yaitu model klasifikasi sentimen dan model klasifikasi kategori. Pengujian dilakukan pada 3135 dataset dengan metode evaluasi k-*fold cross validation*. Proses pengujian menggunakan pendekatan k=5, pada 5-*fold cross validation* data akan dibagi menjadi 5 *subset* dengan ukuran yang sama dan data yang berbeda. Pada setiap iterasi, satu bagian digunakan untuk data pengujian sedangkan sisanya digunakan untuk data pelatihan. Misalnya pada iterasi pertama *subset* pertama digunakan sebagai data pengujian, maka *subset* kedua sampai lima digunakan sebagai data pelatihan.

Pada iterasi kedua dan subset kedua sebagai data pengujian, maka *subset* pertama, ketiga sampai dengan lima sebagai data pelatihan dan begitu seterusnya hingga iterasi kelima.

4.2.1 Pengujian *Confusion Matrix*

Tabel *confusion matrix* digunakan sebagai acuan dalam melakukan pengujian akurasi, presisi dan *recall*. Melalui *confusion matrix* dapat diketahui nilai *True Positif* (TP), *False Negative* (FN), *False Positive* (FP) dan *True Negative* (TN) yang dihasilkan oleh model klasifikasi. Berikut hasil TP, FN, FP, dan TN *confusion matrix* model klasifikasi sentimen menggunakan 5-*fold cross validation* dapat dilihat pada tabel 4.1.

Tabel 4.1 Tabel *Confusion Matrix* model klasifikasi sentimen

Fold	***Confusion Matrix***			
	TP	**FN**	**FP**	**TN**
1	443	50	52	82
2	451	42	54	80
3	436	57	63	71
4	466	27	78	56
5	455	39	57	76

Berbeda dengan model klasifikasi sentimen yang merupakan *binaryclass*, model klasifikasi kategori merupakan *multiclass* dengan 4 kelas. Sehingga hasil TP, FP, FN, dan TN dijabarkan berdasarkan kelas label. Hasil *confusion matrix* model klasifikasi kategori dapat dilihat pada tabel 4.2.

Tabel 4.2 Tabel *Confusion Matrix* model klasifikasi kategori

Fold	***Confusion Matrix***															
	TP				**FN**				**FP**				**TN**			
	DT	**AK**	**KB**	**FS**	**DT**	**AK**	**KB**	**FS**	**DT**	**AK**	**KB**	**FS**	**DT**	**AK**	**KB**	**FS**
1	209	68	79	115	61	38	28	29	36	23	11	86	321	397	489	509
2	226	83	82	92	44	22	26	51	51	53	12	28	306	455	469	507
3	234	69	92	72	27	36	15	73	79	24	30	18	278	464	498	490
4	224	67	91	93	47	38	16	51	58	28	25	41	298	442	494	495
5	230	77	80	86	41	28	27	58	56	48	16	34	300	449	474	504

Selanjutnya akan dilakukan perhitungan nilai akurasi, presisi, dan *recall* untuk masing-masing iterasi. Berikut hasil pengujian model klasifikasi sentimen menggunakan 5-*fold cross validation* dapat dilihat pada tabel 4.3 dan hasil pengujian model klasifikasi kategori dapat dilihat pada tabel 4.4.

Tabel 4.3 Tabel pengujian K-*fold cross validation* model klasifikasi sentimen

Fold	Akurasi	Presisi	Recall
1	84%	76%	75%
2	85%	77%	76%
3	81%	71%	71%
4	83%	77%	69%
5	85%	78%	75%
Rata-rata	**84%**	**76%**	**73%**

Tabel 4.4 Tabel pengujian K-*fold cross validation* model klasifikasi kategori

Fold	Akurasi	Presisi	Recall
1	75%	76%	74%
2	77%	77%	76%
3	76%	76%	73%
4	76%	74%	74%
5	75%	74%	73%
Rata-rata	**76%**	**75%**	**74%**

4.2.2 Pengujian Kurva ROC

Kurva ROC diplot pada kurva dengan nilai *True Positive Rate* (TPR) pada sumbu Y dan nilai *False Positive Rate* (FPR) pada sumbu X. Berikut hasil TPR dan FPR model klasifikasi sentimen dan model klasifikasi kategori menggunakan 5-*fold cross validation* dapat dilihat pada tabel 4.5.

Tabel 4.5 Tabel nilai FPR dan TPR model sentimen

Fold	FPR	TPR
1	0.39	0.90
2	0.40	0.91
3	0.47	0.88
4	0.58	0.94
5	0.43	0.92
Rata-rata	**0.45**	**0.91**

Berbeda dengan kurva ROC yang dihasilkan pada model klasifikasi sentimen, untuk kurva ROC model klasifikasi kategori menggunakan *multiclass* sehingga perlu melakukan pemisahan label kelas. Berikut nilai FPR dan TPR pada masing-masing kelas pada model lasifikasi kategori dapat dilihat pada tabel 4.6.

Tabel 4.6 Tabel nilai FPR dan TPR model kategori

Fold	FPR				TPR			
	DT	AK	KB	FS	DT	AK	KB	FS
1	0.10	0.04	0.02	0.18	0.77	0.64	0.74	0.80
2	0.14	0.10	0.02	0.06	0.84	0.80	0.76	0.64
3	0.22	0.05	0.06	0.04	0.90	0.66	0.86	0.50
4	0.16	0.05	0.05	0.09	0.83	0.64	0.85	0.65
5	0.16	0.09	0.03	0.07	0.83	0.73	0.75	0.60
Rata-rata	**0.16**	**0.07**	**0.04**	**0.09**	**0.83**	**0.69**	**0.79**	**0.64**

Dari nilai FPR dan TPR yang diperoleh, selanjutnya di plot menjadi bentuk kurva. Dari kurva yang dihasilkan dapat diketahui akurasi klasifikasi ROC yaitu dengan cara menghitung luas daerah di bawah kurva ROC yang disebut dengan *Area Under Curve* (AUC). Nilai AUC dari kurva ROC model sentimen dapat dilihat pada tabel 4.7.

Tabel 4.7 Tabel nilai AUC model klasifikasi sentimen

Fold	Nilai AUC
1	0.75
2	0.76
3	0.71
4	0.68
5	0.75
Rata-rata	**0.73**

Berikut adalah nilai AUC yang dihitung dari rata-rata TPR dan FPR pada kurva ROC model klasifikasi kategori dapat dilihat pada tabel 4.8.

Tabel 4.8 Tabel nilai AUC model klasifikasi kategori

Label	Nilai AUC
Daya Tarik	0.84
Aksesbilitas	0.81
Kebersihan	0.88
Fasilitas	0.77
Rata-rata	**0.83**

4.2.3 Pembahasan

1. Pengujian Algoritma LSTM

Dari implementasi dan pengujian yang dilakukan, diperoleh hasil bahwa sistem dapat melakukan klasifikasi sentimen dan kategori dengan baik. Nilai rata-rata akurasi model klasifikasi sentimen sebesar 84%, presisi sebesar 76% dan *recall* sebesar 73% serta nilai AUC sebesar 0.73. Nilai rata-rata akurasi model klasifikasi kategori sebesar 76%, presisi sebesar 75% dan *recall* sebesar 74% serta nilai AUC sebesar 0.83.

Kondisi ideal dari suatu sistem adalah apabila suatu rasio *recall* dan presisi sama besarnya 1:1 (Pao, 1989). Jika dilihat perbandingan presisi dan *recall* pada penelitian ini adalah mendekati 1:1 walaupun tidak sama persis besarnya yaitu selisih 3% untuk model klasifikasi sentimen dan 4% untuk model klasifikasi kategori, maka sistem ini telah mendekati kondisi ideal dari keefektifan suatu sistem klasifikasi. Berdasarkan teori Lancaster dalam (Pendit, 2008) dijelaskan bahwa efektifitas sistem dikategorikan menjadi 2 yaitu efektif jika nilai presisi dan *recall* diatas 50% dan tidak efektif jika nilainya dibawah 50%. Kedua ukuran tersebut diukur dalam bentuk persentase 1-100%. Pada penelitian ini presisi dan *recall* yang didapat sudah diatas 50%, sehingga dapat dikatakan bahwa sistem yang dibangun sudah efektif.

Pengujian dengan menggunakan kurva ROC pada model klasifikasi sentimen dihasilkan nilai rata-rata akurasi klasifikasi atau biasa disebut AUC sebesar 0.73, sedangkan pada model klasifikasi kategori sebesar 0.83. Hasil AUC model klasifikasi sentimen berada pada nilai antara 0.70 – 0.80 sehingga masuk dalam kategori *fair classification*. Hasil AUC model klasifikasi kategori berada pada nilai antara 0.80 – 0.90 sehingga masuk dalam kategori *good classification* (Suwarno dan Abdillah, 2016).

2. Analisis Akurasi Prediksi

Nilai akurasi yang diperoleh pada pengujian algoritma LSTM masih belum sempurna, salah satu faktor adalah jumlah kelas data *training* yang tidak seimbang atau *imbalanced*. Pada klasifikasi sentimen, data positif memiliki jumlah yang dominan yaitu 2466 sedangkan data negatif hanya sebesar 669. Pada klasifikasi kategori, daya tarik memiliki jumlah kelas yang paling dominan yaitu sebesar 1352. Jumlah tersebut sangat jauh berbeda dari jumlah kelas yang lain, yaitu aksesbilitas sebesar 526, kebersihan sebesar 536, dan fasilitas sebesar 721. Klasifikasi data yang tidak seimbang atau *imbalanced* merupakan masalah yang krusial pada bidang *machine learning* dan *data mining*. Ketidakseimbangan data memberikan dampak yang buruk pada hasil klasifikasi dimana kelas minoritas sering disalah klasifikasikan sebagai kelas mayoritas (Siringoringo, 2018). Oleh karena itu proses klasifikasi data uji pada penelitian ini masih belum menghasilkan akurasi yang maksimal. Rata-rata akurasi sentimen sebesar 84% yang artinya masih ada 16% atau 100 dari 627 data uji yang diprediksi salah. Sedangkan rata-rata akurasi kategori sebesar 76% yang artinya masih ada 24% atau 150 dari 627 data uji yang diprediksi salah. Contoh hasil prediksi salah untuk klasifikasi sentimen dan kategori beserta nilai probabilitas dilihat pada Tabel 4.9 dan 4.10.

Tabel 4.9 Tabel prediksi salah pada data uji sentimen

No	Ulasan	Prediksi (Salah)	Label dataset (Benar)	Probabilitas	
				Positif	Negatif
1	jarak tempuh lumayan cepat banding pantai wonosari akses mudah roda roda larang kondisi jalan sempit sana alam bawa pikir jauh hiruk pikuk kota syahdu	Negatif	Positif	0,345885	0,621812
2	aneh ibuk ibuk usia jalan jauh banget hehehe terjal	Positif	Negatif	0,536531	0,452916
3	mobil sepi cocok nyari suasana tenang	Positif	Negatif	0,999308	0,000717
4	suka karang pas air laut nabrak karang liat bagus tebing sayang musti bayar sunset matahari halang tebing	Positif	Negatif	0,999378	0,000551
5	pantai greweng pantai letak ujung gunung kidul batas jogja wonogiri kesana kudu tenaga daki tanjak turun curam tantang pantai greweng pandang bagus nyaman ngecamp pribadi acara bareng pantaigreweng gunungkidul lestguide	Positif	Negatif	0,999791	0,000195

Tabel 4.10 Tabel prediksi salah pada data uji kategori

No	Ulasan	Prediksi (Salah)	Label dataset (Benar)	Probabilitas			
				DT	AK	KB	FS
1	pantai bagus bersih pasir putih serpih pecah karang cocok main air anak anak	Kebersihan	Daya Tarik	0,032611	0,001226	0,961837	0,004325
2	mobil sepi cocok nyari suasana tenang	Daya Tarik	Aksesbilitas	0,973809	0,024528	0,000162	0,001500
3	harus jalan kirang menit lupa bawa akua pakai sendal gunung sendal jepit sepatu bayar pas pandang sih tempat relatif sepi pinggir pantai diri tenda ombak lumayan agak kenceng	Aksesbilitas	Daya Tarik	0,012114	0,876309	0,004238	0,107338
4	pantai ku wisata ikan nge recommend in main air ombak lumayan sedih pasar ikan pindah isa milih ikan bawa warung dimasakin makan pinggir pantai huhuh warung makan nembak harga mentang wisata harga standart tau harga dasar ikan biar nawar turut tawar jangkau	Fasilitas	Daya Tarik	0,004831	0,011442	0,000543	0,983182
5	pandang pantai indah sayang renang patung ikan ukur puncak bukit kondisi penuh coret tangan tangan jahil guna kendara motor kendara motor atas bukit larang kelola bahaya kemudi motor unjung jalan kaki dasar informasi duduk lokal pandang matahari benam musim hujan	Aksesbilitas	Daya Tarik	0,032383	0,875305	0,003015	0,089297

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, maka diperoleh kesimpulan sebagai berikut:

1. Telah dibangun sistem pengklasifikasi ulasan objek wisata pantai yang diperoleh dari *google maps* menggunakan metode *long-short term memory*. Ulasan diklasiifikasikan ke dalam dua model, yaitu sentimen dan kategori.
2. Hasil pengujian analisis sentimen ulasan *google maps* menggunakan *confusion matrix* dengan pendekatan 5-*fold cross validation* menghasilkan rata-rata nilai akurasi untuk model klasifikasi sentimen sebesar 84%, presisi sebesar 76% dan *recall* sebesar 73% dalam waktu komputasi rata-rata selama 277 detik. Sedangkan untuk klasifikasi kategori diperoleh rata-rata nilai akurasi sebesar 76%, presisi sebesar 75% dan *recall* sebesar 74% dalam waktu komputasi rata-rata selama 279 detik.
3. Hasil pengujian analisis sentimen ulasan *google maps* menggunakan kurva ROC dengan pendekatan 5-*fold cross validation* pada model klasifikasi sentimen dihasilkan nilai rata-rata AUC sebesar 0.73, sedangkan pada model klasifikasi kategori sebesar 0.83. Hasil AUC model klasifikasi sentimen berada pada nilai antara 0.70 – 0.80 sehingga masuk dalam kategori *fair classification*. Hasil AUC model klasifikasi kategori berada pada nilai antara 0.80 – 0.90 sehingga masuk dalam kategori *good classification*

5.2 Saran

Penelitian ini masih terdapat beberapa keterbatasan dan kekurangan. Keterbatasan dan kekurangan ini bisa dijadikan acuan atau pertimbangan untuk penelitian selanjutnya. Pada penelitian ini didapatkan hasil pengujian dengan nilai akurasi yang belum tinggi karena jumlah data yang masih belum seimbang atau *imbalanced*. Saran untuk penelitian selanjutnya adalah dengan menambahkan jumlah data latih dengan data yang seimbang. Selain itu, untuk penelitian selanjutnya yang mengimplementasikan *deep learning* sebagai metode klasifikasi teks dapat mengatur kembali paarameter dalam pembentukan model atau *hyperparameter tuning* agar mendapatkan model klasifikasi yang lebih baik.

DAFTAR PUSTAKA

Ahmad, A. (2017). Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning. *Jurnal Teknologi Indonesia*, (October), 3.

Aldhiansyah, P. B. (2020). Algoritma Random Forest Decision Tree Untuk Klasifikasi Pesan Isu Suku Agama Ras Dan Antar Golongan (Sara) Di Twitter.

Alwehaibi, A. (2018). Comparison of Pre-trained Word Vectors for Arabic Text Classification using Deep Learning Approach. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1471–1474. https://doi.org/10.1109/ICMLA.2018.00239

Berry, M. W., & Kogan, J. (2010). *Text Mining : Applications and Theory*. United Kingdom: WILEY.

Bright Local. (2018). Local Consumer Review Survey.

Chazhoor, A. P. (2019). ROC Curve in Machine Learning. Retrieved May 10, 2020, from https://towardsdatascience.com/roc-curve-in-machine-learning-fea29b14d133

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. The MIT Press.

Habibie, I. (2018). Identifikasi Judul Berita Clickbait Berbahasa Indonesia dengan Algoritma Long Short Term Memory (LSTM) Recurrent Neural Network. *Repositori Institusi Universitas Sumatra Utara*, 59. Retrieved from https://repositori.usu.ac.id/handle/123456789/8874

Haddi, E., Liu, X., & Shi, Y. (2013). The Role of Text Pre-processing in Sentiment Analysis. *Procedia Computer Science*, (17), 26–32.

Hermawan, H. (2017). Pengembangan Destinasi Wisata pada Tingkat Tapak Lahan Dengan Pendekatan AnalisisSwot.

Koesumaningrum, D. (2018). Analisis Sentimen Ulasan TripAdvisor pada Tempat Wisata Menggunakan Ontology Supported Polaruty Mining (OSPM)(Studi Kasis Bandung).

Luhrie, S. F. (2019). Klasifikasi Informasi Dan Keluhan Masyarakat Menggunakan Algoritma Fuzzy K-Nearest Neighbor.

Murnawan, M. (2017). Pemanfaatan Analisis Sentimen Untuk Pemeringkatan Popularitas Tujuan Wisata. *Jurnal Penelitian Pos Dan Informatika*, *7*(2), 109. https://doi.org/10.17933/jppi.2017.070203

Nazief, B., & Adriani, M. (1996). Confix Stripping: Approach to Stemming Algorithm for Bahasa Indonesia.

Nurrohmat, M. A., & SN, A. (2019). Sentiment Analysis of Novel Review Using Long Short-Term Memory Method. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *13*(3), 209. https://doi.org/10.22146/ijccs.41236

Pao, M. L. (1989). *Concepts of Information Retrieval*. Englewood Colorado: Libraries Unlimited.

Pendit, P. L. (2008). *Perpustakaan digital dari A sampai Z*. Jakarta: Cita Karyakarsa Mandiri.

Priyanto, A., & Ma'arif, M. R. (2018). Implementasi Web Scrapping dan Text Mining untuk Akuisisi dan Kategorisasi Informasi dari Internet (Studi Kasus: Tutorial Hidroponik). *Indonesian Journal of Information Systems*, *1*(1), 25–33. https://doi.org/10.24002/ijis.v1i1.1664

Purnomo, D. (2017). Model Prototyping Pada Pengembangan Sistem Informasi. *JIMP - Jurnal Informatika Merdeka Pasuruan*, *2*(2), 54–61.

Putra, J. W. G. (2019). Pengenalan Konsep Pembelajaran Mesin dan Deep Learning, 1–235. Retrieved from https://www.researchgate.net/publication/323700644

Rampeng, D. G. P. D. D. (2018). Penerapan Analisis Sentimen pada Media Sosial Menggunakan Metode lexicon Based dan Support Vector Machine (SVM) sebagai Rekomendasi Oleh-oleh Favorit.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, *34*(1), 1–47.

Siringoringo, R. (2018). Klasifikasi Data Tidak Seimbang Menggunakan Algoritma SMOTE dan k-Nearest Neighbor. *Jurnal ISD*, *3*(1), 44–49.

Suwarno, & Abdillah, A. A. (2016). Penerapan Algoritma Bayesian Regularization Backpropagation untuk Memprediksi Penyakit Diabetes.

Wang, Z., & Song, B. (2019). Research on hot news classification algorithm based on deep learning. *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, (Itnec), 2376–2380. https://doi.org/10.1109/ITNEC.2019.8729020

Wilianto, L., Pudjiantoro, T. H., & Umbara, F. R. (2017). Analisis Sentimen Terhadap Tempat Wisata dari Komentar Pengunjung dengan Menggunakan Metode Naive Bayes Classifier Studi Kasus Jawa Barat, 439–448.

Xing, W., & Du, D. (2018). Dropout Prediction in MOOCs : Using Deep Learning for Personalized Intervention. *Journal of Educational Computing Research*, (March). https://doi.org/10.1177/0735633118757015

Yogyakarta, D. P. D. I. (n.d.). *Statistik Kepariwisataan 2017*.

Zuhdi, A. M., Utami, E., & Raharjo, S. (2019). Analisis Sentiment Twitter Terhadap Capres Indonesia 2019 Dengan Metode K-NN. *Jurnal Informa Politeknik Indonusa Surakarta*, *5*, 1–7.

LAMPIRAN

Lampiran 1. Contoh Labeling Ulasan

Data ke-	Nama responden
1-1200	Septi Nur Indrawati
	Faizal Aji Kurniawan
	Rasyid Prayoga
	Yahdi Indrawan
	Hafidz Amarul Ma'rufi
1201-2400	Risma Stela Putri
	Laelatuz Zahro
	Muhammad Iqbal
	Candra Juni C.
	Asrina Yuni Ikhsanti
2401-3500	Arif Kurniawan
	Aryandi
	Pramudavardani Khansaraswati
	Andre Saputra
	M. Rafif Azzaki

No	Ulasan	Sentimen						Kategori					
		R1	R2	R3	R4	R5	Hasil	R1	R2	R3	R4	R5	Hasil
1	indah pantainya,tp sayang..pemerintah sepertiinya kurang peduli terhadap akses jalannya.Jalannya sempit,apalagi kl sisipan antar bis pasti bikin macet.Udah 5x sy berkunjung tp gak ada peningkatan akses ata pelebaran jln menuju kesana	-1	0	-1	-1	-1	-1	2	2	2	2	2	2
2	View nya bagus. Utk resto nya (resto Indrayanti) masakannya juga lumayan&harganya sesuai lah. Sayangnya ada banyak ubur2&bulu babi. Banyak yg main air tersengat hewan tersebut. Tapi rescue nya cepat.	-1	0	-1	-1	1	-1	2	2	2	2	1	2
3	Tempat hak terlalu bersih Pedagang nya bnyak yang tutup Tp view nya bagus"	-1	-1	-1	1	-1	-1	3	3	3	1	3	3

4	Pantainya bersih batu karangnya terlihat airnya jernih.. Sayang tempat bilas kurang banyak dan tempat ibadah kurang luas	1	0	-1	-1	-1	-1	3	3	4	4	4	4
5	Puas sekali...huhuhuhmmmm	1	1	1	1	1	1	1	1	1	1	1	1
6	waktu saya kecil saya pernah ke sana dan sangat indah momen itu tidak pernah saya lupakan sampai kelas 3 waktu saya kesana saya masih teka l ovyu yu pantai indrayanti	1	1	1	1	1	1	1	1	1	1	1	1
7	ombaknya besar kadang berbahaya	-1	-1	0	-1	-1	-1	2	2	2	2	2	2
8	jauh banget lokasinya dan macet kadang	-1	-1	-1	-1	-1	-1	2	2	2	2	2	2
9	Pantai yang indah dengan ombak yang bergelombang tinggi , tapi jangan coba cobaain ketengah karena ini lait kidul yang terkenal dengan ombaknya yang besar. Perjalanan lumayan lama dari jogjakarta	0	-1	-1	1	-1	-1	2	2	2	1	2	2
10	Nggak menyesal aku satang kemari, pantainya bagus bangettt	1	1	1	1	1	1	1	1	1	1	1	1
11	sudah rame banyak yang meninggalkan sampah di mana mana	-1	-1	-1	-1	-1	-1	3	3	3	3	3	3
12	karena rame jadi pemandangannya biasa aja	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1
13	pantai yang indah. letaknya berdekatan dengan pantai lainya yang tak kalah cantik.jika kita kesini pas lagi surut kita bisa melihat berbagai ikan kecil didaerah karang	1	1	1	1	1	1	1	1	1	1	1	1
14	Pemandangan dari atas tebingnya sungguh bagus, naik ke atas hanya membayar 2k per orang	1	1	1	1	1	1	1	1	1	1	1	1
15	kotor	-1	-1	-1	-1	-1	-1	3	3	3	3	3	3
16	Bagus , Pantai nya pasir putih , Agak rame"	1	1	1	1	1	1	1	1	1	1	1	1
17	kadang kotor pas hari weekend rame banget soalnya	-1	-1	-1	-1	-1	-1	3	3	3	3	3	3

18	Indah sekalee sayang kurang luas aja tmpatnya. Liat pemandangan dari atas bukit lebih kerennnn	1	0	-1	1	1	1	1	1	2	1	1	1
19	Indahnyaaaaaaaaaa....tak tertandingi. Bermain d pantai...pasti betah berlama2	1	1	1	1	1	1	1	1	1	1	1	1
20	rame, pengunjung jorok banyak sampah	1	-1	-1	-1	0	-1	1	3	3	3	3	3
21	Bagus sekali pantainya, ombaknya lumayan besar juga	1	1	1	1	0	1	1	1	1	1	2	1
22	Asik dgn pasir putih dan gunung karang,,serasa di bali	1	1	1	1	1	1	1	1	1	1	1	1
23	sudah rame jadi bikin kotor	-1	-1	-1	-1	-1	-1	3	3	3	3	3	3
24	Pantai nya yg bersih dan viewnya sangat bagus sekali.	1	1	1	1	1	1	1	1	3	3	3	3
25	Tempat wisata yang bagus banget kalo menurutku, soalnya air lautnya belum kotor kayak Parangtritis. Pasir pantainya juga masih putih dan lumayan bersih. Ombaknya pas, dan banyak spot foto yang bagus disini. Semoga bisa lebih terawat dari sampah2:)"	1	1	1	1	1	1	3	3	3	3	3	3
26	I have visited this beach once, and absolutely love the scenery. Air laut nya bersih banget, pasir nya juga putih dna bersih, tapi hati-hati akan ombak nya yang cukup menyeramkan. Sedihnya pas kesana salah spot pantai malah ke tempat sepi, dan memang lagi pasang.	1	1	0	-1	1	1	3	3	2	2	3	3
27	Tempat nya nyaman .. enak buat santai santai sma keluarga	1	1	1	1	1	1	1	4	1	4	4	4
28	Banyak tempat santai dan spot foto yg mantaap	1	1	1	1	1	1	1	4	4	4	4	4
29	Kurang aman u/anak2 krn bnyk pecahan kadang tajam dan bulu babi disekitar pantai	-1	-1	-1	-1	-1	-1	2	2	2	2	2	2
30	warga dan penjual ramah, harga makanan, minuman standart, kebersihan terjaga"	1	1	1	1	1	1	3	4	4	3	4	4

Lampiran 2. Kamus *Spelling*

kata seharusnya	kata yang salah
habis	abisss
ada	ad
adrenalin	adrenaline
air	mmair, airr
akses	access, aksws
aksesoris	asesoris
aktivitas	aktifitas
aku	akk, aq
anginnya	anginnua
apa	papa, pa
apalagi	apalgi
apik	apikkkkkk
area	areba
asik	asyik, asiiiikkk, asekk, asyikkk, uasyiiik, aseek, asikk, ayik
assalamualaikum	asalamwalaikum
atas	ats
atau	ata, ato
ayo	ayoo
background	baground
bagaimana	bgmn
bagus	bgus, buagusssss, baguss, baguuussss, baguuuuus, bagusss, buagus, bwagus, nagus, baguus, bgs, bhs, bangettt, bgt, bangettttt, bat, bangeett, bangett, bangettttttttt, baaangeeet, bangeeeet, bangeettttttttt, bnget, bangeeeeeeeet, bangen, bgtt, bangeettttttt, bgggtttt, buanget, bangeetttz, bingit, bangeeett, bangeeetttt, bgd, bgtttt, bingiittss, bangeeet, bangeet
banyak	bnyak, byk, bnyk, banyk, banyam, banyaak, banyaj
bapak	bp
barat	brt
bareng	vareng
bawa	baea, bw
beberapa	bbrp
begitu	bgtu
belakang	blkng
belum	blm, belom, lom
benar	bnr, bener
benaran	bneran
berani	brani
berapa	brpaan
berjajar	bsrjajar
berkano	berkanoe
bersama	brsm
bersantai	bersante

berselfi	berselphi
bersih	bersihhhh, beesih, bersiiiihhh, bersihhh
bertanya	brtnya
bertugas	brtugas
berwisata	berwisaya
besar	besarrrr, gede, gueede, gedhe, guuueeedeeeeeeeee, gedeeeeeeeee, besae
biasa	biasaaaaaa
bisa	busa, bs, ns, bisaaa, bsa
boleh	bleh
bosan	bosen
bro	brooo
buat	bt, bwt
bukan	bkan
bukit	bukiy
buruk	burukn
bus	bis
bye	byeeee
cakep	cakeeeppppp
camping	campingg
capek	cape
capeknya	caoeknya
cepat	cepet
cocok	coxok, cocol
coy	coyy
cukup	ckup, ckp
cuma	cman, cm, cma, cmn
dan	n, dna, lan, nan, dn
dan kawan kawan	dkk
dan lain lain	dll
dan sebagainya	dsb
dapat	dpt, daoat, dapet
dari	dri, dr, dair
daripada	drpd
datang	satang, dateng, dtang, dtg, dtng
datangnya	datengnya
deh	dahh, dech, dehhh
dekat	dket, dkt
dengan	dgn, dg, dengab, dgb, dngan
depan	dpn
di	d, fi
dibalik	dbalik
dih	idiiihh
dijalan	djln
dipakai	dipake

diperbolehkan	perbolehkab
dipinggir	dpinggir
disana	dsn
disekitar	dsekitar
disini	dsini, dsni
ditanya	ditanyaaa
ditempatnya	dtmptnya
doang	doank
drini	ndrini
dulu	duluuuu
eksotis	exsotis, exotis, exotic, eksotik
eksplor	explore
ekspos	exposs
ekstra	extra
ekstrim	ektrem, extreme, extrime, exstreme
emang	emg
enak	enk, ena, ueeenak, wuennakkj, wenakk
fasilitas	fasikitas
favorit	fave, favourite, fav
foto	fto, photo, poto
fresh	fress
gaes	gaeess, gaiiis, gais, gaess
gazebo	gasebo
gebetan	gebetann
gila	gilaa, gilaaaa
ginian	gnian
gitu	gt, gtu
gokil	goks
good	guddd
gue	gw
gunung	gn
gunung kidul	gnkidul, gunkid
habis	habisss, abiz, abiss
hafal	hapal
halo	hlo
hanya	hnya, hny
harga	hrga
harus	hrs
hati	ati
hijau	ijo
hijaunya	ijonya
hitam	item
hits	hitz
hobi	hobby, hobbh

holiday	holyeay
horor	horror
ih	ihhhh
indah	indahhhhhhhhhh, endah, indahh, indaah, indaaaaah, iiiinnddaaahh, imdah, indaahhhhh, indaahhh, indaaahhhh, indahhhhh, indahhh
indahnya	indahnyaaaaaaaaaa, indahnyaaaaaaa
ingin	pengen, pengin
ini	nie
instagramable	instanable
instagramer	instaramer
isu	issu, issue
iya	yak
jadi	jdi, jd, jadii
jalan	jln, jlan
jalanan	jalannan
jalannya	jalanya
jalur	jalud
jam	jm
jangan	jgn, jgan, jngn
jauh	juaaauuuhh
jelas	jelaz
jernih	jernihh, jermih
jos	jooossss, jossss, joss, jozz, josttt
juara	juarakk, juaraaa
juga	jg, jga, jug, juaga
kalau	kl, klo, kalo, klu, klau
kali	kli
kalian	klyn
kamar	kamat
karangnya	karangy
karena	krn, karna, krna, cz
kawasan	kwasan
kayaknya	kyknya, kyaknya
ke	k
kebangetan	kebangetannn
kebanyakan	kbnykn, kbnyakn
kece	ketjehhh, kecekk, keceh
kecil	jecil
kedepan	kdpn
keindahan	keimdahan
kelaparan	kelaperan
kelihatan	kliatan
keluarga	kluarga, kelg, klrga, keluaga
kemari	kmari

kemarin	kmrn, kmaren, kemaren
kemping	cemping, camping, camp
kenapa	knpa, knapa
kencang	kencengg
kepantai	keoantai
keren	kerennnn, kereennnn, kerennn, kereeeen, kereen
kesana	kesanaaa, ksana
kesel	kzl
kesini	ksni
kesini	kesinih, ksini, kesni
khawatir	kawatir, kwatir, kuatir
kita	qta, qt
kok	kokk, koq, kq
kualitasnya	kwalitasnya
kurang	krg
lagi	lg, lgi, lagiii
lah	laaah, lach, lha, laahhh
langsung	lgsg, lngsung
laut	lait
lebih	lbh
lelah	capek
lengkap	lengap, lengkapppp
lewat	lwt
lewatin	lwatin
lihat	liat
loh	lho, lohhh
lokasinya	lokasinta
luar	ruar, luarrrrrrrr, luaar
luas	luaass
lumayan	mayan, lmyn, lmyan
lunas	lunasss
lurus	luruuss
main	maen
mainan	maenan
mainnya	mainny
makan	mkn
makanan	mkanan
maksimal	maximal
malam	mlem, mlm
mampir	mmpir, mampiir
mana	mna
mandi	mandy
mangrovenya	mangovenya
manja	muanjah

mantap	mantappp, mantaff, mantapppp, mantep, mantab, muantap, mantaps, mantaabb
mantul	mantullll, mantullllll, mantulll, mantuuullll, mantullllllllll
mantullllll	puasss
masa	masaa
masih	msh, msih, masik, mash
matahari	mahahari
matang	mateng
mau	mo, mw
melelahkan	mrlelahkan
melewati	mlewati
melihat	mwlihat
memang	mmng
membayar	mbyar
membosankan	mbosankan
menarik	menrik
mending	mnding
menikmati	menimmati, menimkati, mnikmati
menu	meu
menuju	mnju
menurut	memurut
menyesal	nyesel, nyesell
motor	mtr
mudah	mudaahh, mdh
mulus	mulusss
mungkin	mgkn, mngkn
murah	murrah
murah meriah	murmer
mushollah	musholla, mushola, musala, mushalla, musola
mushollahnya	musholanya
naik	naim
namun	namum
nan	nun
negeri	negri
ngecamp	kemping, ngecame, ngrcamp, ngcamp
ngehits	ngehitz, ngehitzz
ngelihat	ngeliat
ngeri	ngeriii
ngojek	ngojekk
nih	nieh
nikmatnya	nikmatnyaaaaaa
nongkrong	nongkii
nya	ny
nyaman	nyamann

nyebur	nyeburr
nyelam	nyelem
oh	ohh
oke	ok, okee
ombak	omba, ombangk, ombah
ombaknya	ombknya, ombsknya, ombakny
orang	org
padahal	pdhl
pagi	pgie
paginya	pagine
pakai	pake
pakaian	pakain
panas	panassss, panaasss
panjang	panjangggggggg
pantai	oantai, pntai, pante, pantaii, pantao
pantainya	pantainyaaa, psntainya, pantaiinyya, oabtainya, pantene, pantsainya
parah	paraaaaaaaah
parkir	prkir
parkirnya	parikirnya
pas	pass
pasang	pasangg
pemandangan	pemndngan, pemangangan, pemndangan, pemandangaa
pemandangannya	pemandangannx, pmandangannya
penasaran	penasarn
pengen	pwngen
penginapannya	peninapannya
pentol	penthol
pergi	prgi
perjalanan	oerjalanan
pernah	prnh
pilihan	pillihan
pinggir	punggir, pinghir, pinggr
plang	plank
plus	ples
pokoknya	pkknya, pokokna, pokokknyya
positif	positiv
puas	puaaasss
pulang	plang
punya	py
ramai	rame, ramaii, ramee, rane
rapi	rapih
rasa	rsa
refresh	ngerifres
rekommended	rekokended

relatif	relative
reques	req
review	revew
ribu	rbu
rileks	rilex
rumah	rm, rmh
rute	route
saja	aja, j, sj
sama	sma, sm, ama
sambil	smbl
sampah	smpah
sampahnya	smphnya
sampai	mpe, nyampek, smpe, smpai, sampe, nyampe, smp
sangat	sgt, sngt, sangatt, snangat
santai	sante
saudara	sodara
saya	sy, sya
sayang	syayang, syg, sanyang, sayag
sayangnya	sygnya
sebagai	sbg
sebagainya	sebgainya
sebelahan	sblhan
sebelahnya	sblhny
sebelum	sblm
sebenarnya	sebenernya, sbnrnya
secara	scr, scra
segar	syeger, seger, segerrrr
segerin	segering
segitu	segotu
sejuk	sejukkk
sekali	sekalee, sekalii, skali, sekallee, srkali, skli, sekaliii, syekaliii, syekaaali
sekalian	skalian
sekarang	skrg, skg
selain	slain
selalu	sll
selfi	selfiria, selfie, selphi, selvi, selpi
semoga	smg, smoga
sempatkan	sempatka
semuanya	smuanya
sensasi	sensari
sepatu	spatu
seperti	spt
sepertinya	sepertiinya, seperrinya
sepi	sepiiii, sepiii

sepoi	sepoy
seramai	serame
sering	srg
seru	seruuu
setelah	stlh
sewa	swa
sih	seh
sip	siip, siippp, sipp, sjiippp
snorkeling	snowrkling, snorkling, senorkling, snorekeling, snornkling
snorkelingnya	snorkellingnya, snorekelingnya
spontan	apontan
stres	stress, strezz
subhanallah	subhanlloh, subahalloh
sudah	sdh, udah, sudh, udh, udahh, sdah
sungguh	sunguhh
tahun	th
takut	tkut
tambah	tmbah, tambh
tangga	tngga
tanya	tnya
tapi	tp, tapii, tpi
teduh	teduhhhh
teman	temsss
temanku	tmenku
tempat	tmpt, tpt
tempatnya	tmpatnya
tempatnya	tempaty, temate, tempate
tenang	tenamg
tepatnya	tpatnya
terbaik	terbaikkkk, terbaiks
terbayarnya	teebayarnya
terima	trima
terimakasih	trmksh
terlalu	trll
termasuk	trmasuk
tersedia	trsedia
tertentu	tertwntu
terus	trs, trus
tetanggaan	tetanggan
tetap	tetepp, ttp, tetep, ttep
tiba tiba	tbrb
tidak	gak, hak, ga, nggak, g, ngga, blom, nggk, belum, tak, tdk, kaga, tdak, ngak, ndak, enggk, kagak, gk, gag, gx, enggak, gax, gg
tidak apa	gpp

tidak bisa	gabisa
tiket	ticket
timang	timangggg
tsunami	stunami
untuk	utk, untk, umtuk, unt, hntuk, tuk, untul
voli	voly, volley
waduh	waduohhhhh
wah	waaahhhhh
waktu	wkt
walaupun	wlwpn
warung	warungggg
wisata	wista
wow	waaowww, wew
ya	yaa, yew, yakk, yaaah, ea
yahud	yaaahuuud
yang	yg, ug, yanh, yv
yuk	yuukk, kuy, yuuuk, yuuks, yux, yuksssssssssss

Lampiran 3. Kamus *Negation Word Conversion*

'terlupakan':'ingat','jauh':'dekat','disarankan':'dilarang','salah':'benar','boleh':'dilarang','begitu':'begini','mahal':'murah','karuan':'berantakan','rekomendasi':'jangan','ramah':'sopan','siasia':'berguna','besar':'kecil','nyaman':'risih','memadai':'kurang','kalah':'menang','nyesel':'puas','jelas':'samar','terurus':'terlantar','enak':'tidakenak','suka':'duka','aman':'bahaya','capek':'kuat','nyangka':'terkejut','cocok':'aneh','Kekurangannya':'kelebihannya','banyak':'sedikit','serame':'sepi','luas':'sempit','berani':'takut','kecewa':'puas','fit':'sakit','macet':'lancar','rugi':'untung','bersentuhan':'berjauhan','bosen':'senang','peduli':'acuh','bayar':'gratis','dirawat':'dibiarkan','seneng':'sedih','rata':'kasar','hepi':'sedih','fresh':'layu','bingung':'faham','cukup':'kurang','takutnya':'beraninya','kesampaian':'gagal','membayar':'gratis','pergi':'pulang','lebih':'kurang','panas':'dingin','terlihat':'kasat','reomended':'jangan','ramai':'sepi','terawat':'dibiarkan','kepanasan':'kedinginan','murah':'mahal','tajam':'halus','berfungsi':'rusak','berubah':'tetap','sebanyak':'sedikit','masuk':'keluar','rekomen':'jangan','kedinginan':'kepanasan','berbahaya':'aman','dapat':'susah','kelihatan':'kasat','kepenuhan':'longgar','kumuh':'bersih','tersedia':'kosong','kuat':'lemah','dirusak':'dirawat','membahayakan':'aman','ingin':'enggan','bagus':'jelek','jelek':'bagus','diperbolehkan':'dilarang','tertata':'berantakan','kecewadan':'puas','tenang':'gelisah','langsung':'tunda','keras':'lunak','banyaknya':'sedikitnya','berisik':'tenang','naik':'turun','sesuai':'beda','kotor':'bersih','khawatir':'yakin','beraspal':'berbatu','masalah':'aman','membuang':'menyimpan','buka':'tutup','seru':'bosan','mendung':'terang','dianjurkan':'jangan','mengecewakan':'memuaskan','mengurangi':'menambah','takut':'berani','kuatir':'aman','sulit':'gampang','asing':'familiar','dimaksimalkan':'minimal','sebagus':'jelek','menakutkan':'menyenangkan','tinggi':'pendek','jernih':'keruh','berhatihati':'ceroboh','berlubang':'halus','lama':'sebentar','kawatir':'aman','bersih':'kotor','keurus':'dibiarkan','memperburuk':'memperindah','berakal':'gila','sebersih':'kotor','dimahalin':'murah','terluka':'aman','dekat':'jauh','recommend':'jangan','pelit':'dermawan','terpakai':'terbuang','direkomendasikan':'jangan','rame':'sepi','kaget':'tenang','mahalmantap':'murah','terbatas':'luas','mengecewakanðŸ˜\x8d':'puas','dibersihkan':'kotor','bahaya':'aman','asik':'bosan','mulus':'kasar','pasang':'surut'

Lampiran 4. Kamus *Stopword Removal*

'tidak','a','ada','adalah','adanya','adapun','agak','agaknya','agar','akan','akankah','akhir','akhiri','akhirnya','aku','akulah','amat','amatlah','anda','andalah','antar','antara','antaranya','apa','apaan','apabila','apakah','apalagi','apatah','arti','artinya','asal','asalkan','atas','atau','ataukah','ataupun','awal','awalnya','b','bagai','bagaikan','bagaimana','bagaimanakah','bagaimanapun','bagainamakah','bagi','bagian','bahkan','bahwa','bahwasannya','bahwasanya','baiklah','bakal','bakalan','balik','banyak','bapak','baru','bawah','beberapa','begini','beginian','beginikah','beginilah','begitu','begitukah','begitulah','begitupun','bekerja','belakang','belakangan','belumlah','benar','benarkah','benarlah','berada','berakhir','berakhirlah','berakhirnya','berapa','berapakah','berapalah','berapapun','berarti','berawal','berbagai','berdatangan','beri','berikan','berikut','berikutnya','berjumlah','berkali-kali','berkata','berkehendak','berkeinginan','berkenaan','berlainan','berlalu','berlangsung','berlebihan','bermacam','bermacammacam','bermaksud','bermula','bersama','bersama-sama','bersiap','bersiap-siap','bertanya','bertanya-tanya','berturut','berturut-turut','bertutur','berujar','berupa',besar','betul','betulkah','biasa','biasanya','bila','bilakah','bisa','bisakah','boleh','bolehkah','bolehlah','buat','bukan','bukankah','bukanlah','bukannya','bulan','bung','c','cara','caranya','cukup','cukupkah','cukuplah','cuma','d','dahulu','dalam','dan','dapat','dari','daripada','datang','demi','demikian','demikianlah','dengan','depan','di','dia','diakhiri','diakhirinya','dialah','diantara','diantaranya','diberi','diberikan','diberikannya','dibuat','dibuatnya','didapat','didatangkan','digunakan','diibaratkan','diibaratkannya','diingat','diingatkan','diinginkan','dijawab','dijelaskan','dijelaskannya','dikarenakan','dikatakan','dikatakannya','dikerjakan','diketahui','diketahuinya','dikira','dilakukan','dilalui','dilihat','dimaksud','dimaksudkan','dimaksudkannya','dimaksudnya','diminta','dimintai','dimisalkan','dimulai','dimulailah','dimulainya','dimungkinkan','dini','dipastikan','diperbuat','diperbuatnya','dipergunakan','diperkirakan','diperlihatkan','diperlukan','diperlukannya','dipersoalkan','dipertanyakan','dipunyai','diri','dirinya','disampaikan','disebut','disebutkan','disebutkannya','disini','disinilah','ditambahkan','ditandaskan','ditanya','ditanyai','ditanyakan','ditegaskan','ditujukan','ditunjuk','ditunjuki','ditunjukkan','ditunjukkannya','ditunjuknya','dituturkan','dituturkannya','diucapkan','diucapkannya','diungkapkan','dong','dua','dulu','e','empat','enggak','enggaknya','entah','entahlah','f','g','guna','gunakan','h','hadap','hai','hal','halo','hallo','hampir','hanya','hanyalah','hari','harus','haruslah','harusnya','helo','hello','hendak','hendaklah','hendaknya','hingga','i','ia','ialah','ibarat','ibaratkan','ibaratnya','ibu','ikut','ingat','ingat-ingat','ingin','inginkah','inginkan','ini','inikah','inilah','itu','itukah','itulah','j','jadi','jadilah','jadinya','jawab','jawaban','jawabnya','jelas','jelaskan','jelaslah','jelasnya','jika','jikalau','juga','jumlah','jumlahnya','justru','k','kadar','kala','kalau','kalaulah','kalaupun','kali','kalian','kami','kamilah','kamu','kamulah','kan','kapan','kapankah','kapanpun','karena','karenanya','kasus','kata','katakan','katakanlah','katanya','ke','keadaan','kebetulan','kecil','kedua','keduanya','keinginan','kelamaan','kelihatan','kelihatannya','kelima','keluar','kembali','kemudian','kemungkinan','kemungkinannya','kena','kenapa','kepada','kepadanya','kerja','kesampaian','keseluruhan','keseluruhannya','keterlaluan','ketika','khusus','khususnya','kini','kinilah','kira','kira-kira','kiranya','kita','kitalah','kok','l','lagi','lagian','lah','lain','lainnya','lak
u','lalu','lama','lamanya','langsung','lanjut','lanjutnya','lebih','lewat','lihat','lima','luar','m','macam','maka','makanya','makin','maksud','malah','malahan','mampu','mampukah','mana','manakala','manalagi','masa','masalah','masalahnya','masih','masihkah','masing','masing-masing','masuk','mata','mau','maupun','melainkan','melakukan','melalui','melihat','melihatnya','memang','memastikan','memberi','memberikan','membuat','memerlukan','memihak','meminta','memintakan','memisalkan','memperbuat','mempergunakan','memperkirakan','memperlihatkan','mempersiapkan','mempersoalkan','mempertanyakan','mempunyai','memulai','memungkinkan','menaiki','menambahkan','menandaskan','menanti','menanti-nanti','menantikan','menanya','menanyai','menanyakan','mendapat','mendapatkan','mendatang','mendatangi','mendatangkan','menegaskan','mengakhiri','mengapa','mengatakan','mengatakannya','mengenai','mengerjakan','mengetahui','menggunakan','menghendaki','mengibaratkan','mengibaratkannya','mengingat','mengingatkan','menginginkan','mengira','mengucapkan','mengucapkannya','mengungkapkan','menjadi','menjawab','menjelaskan','menuju','menunjuk','menunjuki','menunjukkan','menunjuknya','menurut','menuturkan','menyampaikan','menyangkut','menyatakan','menyebutkan','menyeluruh','menyiapkan','merasa','mereka','merekalah','merupakan','meski','meskipun','meyakini','meyakinkan','minta','mirip','misal','misalkan','misalnya','mohon','mula','mulai','mulailah','mulanya','mungkin','mungkinkah','n','nah','naik','namun','nanti','nantinya','nya','nyaris','nyata','nyatanya','o','oleh','olehnya','orang','p','pada','padahal','padanya','pak','paling','panjang','pantas','para','pasti','pastilah','penting','pentingnya','per','percuma','perlu','perlukah','perlunya','pernah','persoalan','pertama','pertama-tama','pertanyaan','pertanyakan','pihak','pihaknya','pukul','pula','pun','punya

,'q','r','rasa','rasanya','rupa','rupanya','s','saat','saatnya','saja','sajalah','salam','saling','sama','sama-sama','sambil','sampai','sampai-sampai','sampaikan','sana','sangat','sangatlah','sangkut','satu','saya','sayalah','se','sebab','sebabnya','sebagai','sebagaimana','sebagainya','sebagian','sebaik','sebaik-baiknya','sebaiknya','sebaliknya','sebanyak','sebegini','sebegitu','sebelum','sebelumnya','sebenarnya','seberapa','sebesar','sebetulnya','sebisanya','sebuah','sebut','sebutlah','sebutnya','secara','secukupnya','sedang','sedangkan','sedemikian','sedikit','sedikitnya','seenaknya','segala','segalanya','segera','seharusnya','sehingga','seingat','sejak','sejenak','sejumlah','sekadar','sekadarnya','sekali','sekali-kali','sekalian','sekaligus','sekalipun','sekarang','sekaranglah','sekecil',seketika','sekiranya','sekitar','sekitarnya','sekurang-kurangnya','sekurangnya','sela','selain','selaku','selalu','selama','selama-lamanya','selamanya','selanjutnya','seluruh','seluruhnya','semacam','semakin','semampu','semampunya','semasa','semasih','semata','semata-mata','semaunya','sementara','semisal','semisalnya','sempat','semua','semuanya','semula','sendiri','sendirian','sendirinya','seolah','seolah-olah','seorang','sepanjang','sepantasnya','sepantasnyalah','seperlunya','seperti','sepertinya','sepihak','sering','seringnya','serta','serupa','sesaat','sesama','sesampai','sesegera','sesekali','seseorang','sesuatu','sesuatunya','sesudah','sesudahnya','setelah','setempat','setengah','seterusnya','setiap','setiba','setibanya','setidak-tidaknya','setidaknya','setinggi','seusai','sewaktu','siap','siapa','siapakah','siapapun','sini','sinilah','soal','soalnya','suatu','sudah','sudahkah','sudahlah','supaya','t','tadi','tadinya','tahu','tak','tambah','tambahnya','tampak','tampaknya','tandas','tandasnya','’tanpa','tanya','tanyakan','tanyanya','tapi','tegas','tegasnya','telah','tempat','tentang','tentu','tentulah','tentunya','tepat','terakhir','terasa','terbanyak','terdahulu','terdapat','terdiri','terhadap','terhadapnya','teringat','teringat-ingat','terjadi','terjadilah','terjadinya','terkira','terlalu','terlebih','terlihat','termasuk','ternyata','tersampaikan','tersebut','tersebutlah','tertentu','tertuju','terus','terutama','tetap','tetapi','tiap','tiba','tiba-tiba','tidakkah','tidaklah','tiga','toh','tuju','tunjuk','turut','tutur','tuturnya','u','ucap','ucapnya','ujar','ujarnya','umumnya','ungkap','ungkapnya','untuk','usah','usai','v','w','waduh','wah','wahai','waktunya','walau','walaupun','wong','x','y','ya','yaitu','yakin','yakni','yang','z'