**PROPOSAL TUGAS AKHIR**

**LSTM DAN BIDIRECTIONAL LSTM UNTUK MENGANALISA SENTIMEN TERHADAP UJARAN KEBENCIAN TWITTER**



Oleh:

Adrianus Charlie Hadirria Agelsadewa

195314174

**Program Studi Informatika**

**Fakultas Sains dan Teknologi**

**Universitas Sanata Dharma**

**Yogyakarta**

**2022**

**HALAMAN ABSTRAK DAN PENGESAHAN PEMBIMBING**

**ANALISIS SENTIMEN UJARAN KEBENCIAN TWITTER MENGGUNAKAN PENDEKATAN LSTM DAN BIDIRECTIONAL LSTM**

# Abstrak

|  |
| --- |
| Isi bagian ini dengan abstrak dari proposal tugas akhir anda.  Abstrak akan berisi paparan singkat mengenai latar belakang persoalan yang diangkat, rumusan masalah, tujuan penelitian, rencana data yang akan digunakan, dan rencana langkah penelitian. |

**Disetujui pada tanggal : <isikan tanggal pengesahan>**

**oleh:**

**Pembimbing Tugas Akhir**

**<tanda tangan calon dosen pembimbing>**

**<Nama Dosen>**

# DAFTAR ISI

[Abstrak 2](#_Toc129457449)

[DAFTAR ISI 4](#_Toc129457450)

[DAFTAR TABEL 6](#_Toc129457451)

[DAFTAR GAMBAR 7](#_Toc129457452)

[BAB I 10](#_Toc129457453)

[I.1. Latar Belakang 10](#_Toc129457454)

[I.2. Rumusan Masalah 13](#_Toc129457455)

[I.3. Tujuan 13](#_Toc129457456)

[I.4. Manfaat Penelitian 13](#_Toc129457457)

[I.5. Batasan Masalah 13](#_Toc129457458)

[I.6. Sistematika Penulisan 14](#_Toc129457459)

[BAB II 16](#_Toc129457460)

[II.1. Tinjauan Pustaka 16](#_Toc129457463)

[II.2. Landasan Teori 20](#_Toc129457466)

[II.2.1 Analisis sentimen 20](#_Toc129457467)

[II.2.2 Jaringan Syaraf Tiruan 21](#_Toc129457468)

[II.2.3 *Deep learning* 25](#_Toc129457469)

[II.2.4 Long Short-Term Memory (LSTM) 37](#_Toc129457470)

[II.2.5 Bidirectional Long Sshort-Term Memory (LSTM) 40](#_Toc129457471)

[II.2.6 Preprocessing 41](#_Toc129457472)

[BAB III 43](#_Toc129457473)

[III.1. Tahapan Penelitian 43](#_Toc129457475)

[III.2. Pengumpulan Data 43](#_Toc129457476)

[III.3. Preprocessing 45](#_Toc129457477)

[III.3.1 Seleksi Data 45](#_Toc129457478)

[III.3.2 *Lowercasing* 46](#_Toc129457479)

[III.3.3 Pembersihan Karakter *Non-alphanumeric* 47](#_Toc129457480)

[III.3.4 Pembersihan Karakter Khusus 48](#_Toc129457481)

[III.3.5 Stop Word Removal 49](#_Toc129457482)

[III.3.6 Lemmatization 51](#_Toc129457483)

[III.4. Labelling Data 51](#_Toc129457484)

[III.5. Tokenizing 53](#_Toc129457485)

[III.6. Pemisahan Data 55](#_Toc129457486)

[III.7. Perancangan Model 56](#_Toc129457487)

[III.7.1 Embedding Layer 56](#_Toc129457488)

[III.7.2 Dense Layer 57](#_Toc129457489)

[III.7.3 LSTM Layer 59](#_Toc129457490)

[III.7.4 Bidirectional LSTM Layer 64](#_Toc129457491)

[III.8. Evaluasi Model 64](#_Toc129457492)

[III.9. Skenario Pengujian 65](#_Toc129457493)

[III.9.1 *Layer* Model 66](#_Toc129457494)

[III.9.2 Fungsi aktifasi 67](#_Toc129457495)

[III.9.3 Optimizer 67](#_Toc129457496)

[III.9.4 Fungsi loss 67](#_Toc129457497)

[III.9.5 Skenario Pengujian Model Sederhana 67](#_Toc129457498)

[III.9.6 Skenario Pengujian Model Kompleks 68](#_Toc129457499)

[BAB IV 70](#_Toc129457500)

[IV.1. Implementasi Perangkat Lunak 70](#_Toc129457502)

[IV.1.1 Pengumpulan Data 70](#_Toc129457503)

[IV.1.2 Preprocessing Data 72](#_Toc129457504)

[IV.1.3 Labelling Data 79](#_Toc129457505)

[IV.1.4 Tokenizing Data 81](#_Toc129457506)

[IV.1.5 Pemisahan Data 84](#_Toc129457507)

[IV.1.6 Modelling 85](#_Toc129457508)

[IV.1.7 Evaluasi 87](#_Toc129457509)

[IV.2. Analisa dan Pembahasan 91](#_Toc129457510)

[IV.2.1 Pengujian Model Sederhana 91](#_Toc129457511)

[IV.2.2 Pengujian Model Kompleks 97](#_Toc129457512)

[BAB V 105](#_Toc129457513)

[V.1. Kesimpulan 105](#_Toc129457515)

[V.2. Saran 106](#_Toc129457516)

[Daftar Pustaka 108](#_Toc129457517)

[References 108](#_Toc129457518)

[Jadwal Kerja 110](#_Toc129457519)

# DAFTAR TABEL

[Tabel 2. 1 Tinjauan Pustaka Penelitian. 20](#_Toc129457307)

[Tabel 3. 1 Sample dataset 44](#_Toc129432810)

[Tabel 3. 2 Contoh data dengan kata “RT”. 45](#_Toc129432811)

[Tabel 3. 3 Contoh data dalam proses *lowercasing* 46](#_Toc129432812)

[Tabel 3. 4 Contoh data pada proses Pembersihan *Non-alphanumeric.* 47](#_Toc129432813)

[Tabel 3. 5 Contoh data dalam proses pembersihan karakter 48](#_Toc129432814)

[Tabel 3. 6 Contoh data dalam *stopword* removal. 49](#_Toc129432815)

[Tabel 3. 7 Contoh data dalam proses *stemming*. 50](#_Toc129432816)

[Tabel 3. 8 Contoh data dalam proses *lemmatization.* 50](#_Toc129432817)

[Tabel 3. 9 Labeling Data menggunakan Vader Analisis sentimen 52](#_Toc129432818)

[Tabel 3. 10 Contoh data dalam proses *tokenizing.* 54](#_Toc129432819)

[Tabel 3. 11 Label *one hot encoder*. 55](#_Toc129432820)

[Tabel 3. 12 Contoh data hasil pada proses embedding *layer* 56](#_Toc129432821)

[Tabel 3. 13 Contoh data hasil dense *layer* dengan *output* unit=1 57](#_Toc129432822)

[Tabel 3. 14 Keterangan perhitungan pada formula forget gate 59](#_Toc129432823)

[Tabel 3. 15 Keterangan perhitungan pada formula input gate 60](#_Toc129432824)

[Tabel 3. 16 Keterangan perhitungan pada formula cell state 61](#_Toc129432825)

[Tabel 3. 17 Keterangan perhitungan pada formula *output* gate 63](#_Toc129432826)

[Tabel 3. 18 Arsitektur model sederhana. 67](#_Toc129432827)

[Tabel 3. 19 Arsitektur model kompleks 68](#_Toc129432828)

[Tabel 4. 1 Komponen penting dalam tokenizing 79](#_Toc129434992)

[Tabel 4. 2 Hasil percobaan model sederhana 90](#_Toc129434993)

[Tabel 4. 3 Hasil percobaan model kompleks. 97](#_Toc129434994)

# DAFTAR GAMBAR

[Gambar 2.1 Alur Kerja Analisis sentimen (Medhat, Hassan, & Korashy, 2014) Jaringan Syaraf Tiruan(Artificial Neural Network) 21](#_Toc129457221)

[Gambar 2.2 Gambar dense *layer* (Verma, 2021) 23](#_Toc129457222)

[Gambar 2. 3 Embedding *layer* di dalam model BiLSTM (ReaserchGate) 24](#_Toc129457223)

[Gambar 2. 4 Flatten *layer* 25](#_Toc129457224)

[Gambar 2. 5 Contoh jaringan syaraf tiruan pada *deep learning*  (MathWorks, 2022) 26](#_Toc129457225)

[Gambar 2. 6 Persamaan optimizer RMSProp (Musstafa, 2021) 27](#_Toc129457226)

[Gambar 2. 7 Persamaan optimizer RMSProp (Musstafa, 2021) 28](#_Toc129457227)

[Gambar 2. 8 Persamaan optimizer adam (Musstafa, 2021) 29](#_Toc129457228)

[Gambar 2. 9 persamaan fungsi aktifasi sigmoid (Marimuthu, 2022) 30](#_Toc129457229)

[Gambar 2. 10 Grafik fungsi aktifasi sigmoid (Marimuthu, 2022) 30](#_Toc129457230)

[Gambar 2. 11 Persamaan fungsi aktifasi ReLu (Marimuthu, 2022) 31](#_Toc129457231)

[Gambar 2. 12 Grafik fungsi aktifasi ReLu (Marimuthu, 2022) 31](#_Toc129457232)

[Gambar 2. 13 Persamaan fungsi aktifasi softmax (Marimuthu, 2022) 32](#_Toc129457233)

[Gambar 2. 14 grafik fungsi aktifasi softmax (Marimuthu, 2022) 32](#_Toc129457234)

[Gambar 2. 15 Persamaan fungsi aktifasi tanh (Datta, 2020) 32](#_Toc129457235)

[Gambar 2. 16 Contoh implementasi dari Dropout 34](#_Toc129457236)

[Gambar 2. 17 Persamaan MSE (Yathish, 2022) 35](#_Toc129457237)

[Gambar 2. 18 Persamaan MAE (Yathish, 2022) 35](#_Toc129457238)

[Gambar 2. 19 Persamaan fungsi loss binary crossentropy 37](#_Toc129457239)

[Gambar 2. 20 Persamaan fungsi loss categorical crossentropy 37](#_Toc129457240)

[Gambar 2. 21 Contoh RNN (Colah, 2015) 38](#_Toc129457241)

[Gambar 2. 22 Struktur LSTM (Colah, 2015) 38](#_Toc129457242)

[Gambar 2. 23 Cell state melalui forget gate (Colah, 2015) 39](#_Toc129457243)

[Gambar 2. 24 *Output* gerbang sigmoid (Colah, 2015) 39](#_Toc129457244)

[Gambar 2. 25 Mengupdate cell state lama (Colah, 2015) 39](#_Toc129457245)

[Gambar 2. 26 Menentukan *output* terbaru dari cell state. (Colah, 2015) 40](#_Toc129457246)

[Gambar 2. 27 Bidirectional LSTM (Mungalpara, 2021) 40](#_Toc129457247)

[Gambar 3. 1 Tahapan Penelitian 43](#_Toc129457300)

[Gambar 3. 2 Cara kerja *flatten* *layer* 58](#_Toc129457301)

[Gambar 3. 3 Forget gate lstm (Colah, 2015) 59](#_Toc129457302)

[Gambar 3. 4 Input gate lstm (Colah, 2015) 60](#_Toc129457303)

[Gambar 3. 5 Cell state LSTM 62](#_Toc129457304)

[Gambar 3. 6 *Output* gate LSTM 63](#_Toc129457305)

[Gambar 3. 7 Persamaan Bidirectional LSTM (Li) 64](#_Toc129457306)

[Gambar 4. 1 Import library Python dan inisialisasi token untuk mendapat akses ke akun developer Twitter 71](#_Toc129457248)

[Gambar 4. 2 Inisialisasi API developer Twitter account 71](#_Toc129457249)

[Gambar 4. 3 Membuat fungsi untuk melakukan iterasisearching *tweet* 71](#_Toc129457250)

[Gambar 4. 4 Membuat fungsi untuk membuat dataframe data hasil scraping 72](#_Toc129457251)

[Gambar 4. 5 Searching *tweet*s menggunakan fungsi sebelumnya dan inisialisasi query untuk mencari data Twitter 72](#_Toc129457252)

[Gambar 4. 6 Implementasi membuat dataframe dan menyimpan ke file excel. 72](#_Toc129457253)

[Gambar 4. 7 Data hasil scraping Twitter 72](#_Toc129457254)

[Gambar 4. 8 Implementasi seleksi data 73](#_Toc129457255)

[Gambar 4. 9 Implementasi *lowercasing*. 74](#_Toc129457256)

[Gambar 4. 10 Hasil *lowercasing*. 74](#_Toc129457257)

[Gambar 4. 11 Implementasi Remove *Non-alphanumeric* 75](#_Toc129457258)

[Gambar 4. 12 Hasil Implementasi Remove *Non-alphanumeric* 75](#_Toc129457259)

[Gambar 4. 13 Implementasi Pembersihan karakter khusus 76](#_Toc129457260)

[Gambar 4. 14 Hasil pembersihan karakter khusus 76](#_Toc129457261)

[Gambar 4. 15 Import modul *stopword*s dari libraries nltk.corpus 77](#_Toc129457262)

[Gambar 4. 16 Implementasi penghapusan *stopword*s 77](#_Toc129457263)

[Gambar 4. 17 Hasil implementasi penghapusan *stopword*s 77](#_Toc129457264)

[Gambar 4. 18 Import dan menginisialisasi PorterStemmer 77](#_Toc129457265)

[Gambar 4. 19 Implementasi Fungsi stem dari PorterStemmer 78](#_Toc129457266)

[Gambar 4. 20 Hasil dari *stemming* data. 78](#_Toc129457267)

[Gambar 4. 21 Import dan inisialisasi WordNetLemmatizer 79](#_Toc129457268)

[Gambar 4. 22 Implementasi lemmatization 79](#_Toc129457269)

[Gambar 4. 23 Hasil implementasi lemmatization 79](#_Toc129457270)

[Gambar 4. 24 Implementasi Vader analisis sentimen 80](#_Toc129457271)

[Gambar 4. 25 Hasil labelling menggunakan Vader 81](#_Toc129457272)

[Gambar 4. 26 Implementasi menghitung panjang setiap kalimat 83](#_Toc129457273)

[Gambar 4. 27 Hasil implementasi dan visualisasinya. 83](#_Toc129457274)

[Gambar 4. 28 Import libraries untuk melakukan tokenizing. 84](#_Toc129457275)

[Gambar 4. 29 Mempersiapkan elemen komponen penting tokenizing. 84](#_Toc129457276)

[Gambar 4. 30 Contoh hasil implementasi tokenizing. 84](#_Toc129457277)

[Gambar 4. 31 Implementasi one hot encoding 85](#_Toc129457278)

[Gambar 4. 32 Implementasi penyusunan model sederhana 86](#_Toc129457279)

[Gambar 4. 33 Implementasi penyusunan model kompleks 86](#_Toc129457280)

[Gambar 4. 34 Implementasi *callback* 87](#_Toc129457281)

[Gambar 4. 35 Implementasi *training* model 87](#_Toc129457282)

[Gambar 4. 36 Implementasi fungsi evaluate oleh model keras. 88](#_Toc129457283)

[Gambar 4. 37 Implementasi Confusion Matrix. 88](#_Toc129457284)

[Gambar 4. 38 *Output* confusion matrix. 89](#_Toc129457285)

[Gambar 4. 39 Implementasi visualisasi hasil *training*. 90](#_Toc129457286)

[Gambar 4. 40 Implementasi visualisasi hasil *training* model. 90](#_Toc129457287)

[Gambar 4. 41 Grafik akurasi, validasi akurasi serta f1 score dari model nomor 4. 93](#_Toc129457288)

[Gambar 4. 42 Confusion matrix model nomor 4. 94](#_Toc129457289)

[Gambar 4. 43 Grafik akurasi, validasi akurasi serta f1 score dari model nomor 5. 94](#_Toc129457290)

[Gambar 4. 44 Confusion matrix model nomor 5. 95](#_Toc129457291)

[Gambar 4. 45 Grafik akurasi, validasi akurasi serta f1 score dari model nomor 6. 95](#_Toc129457292)

[Gambar 4. 46 Confusion matrix model nomor 6. 96](#_Toc129457293)

[Gambar 4. 47 Grafik akurasi, validasi akurasi serta f1 score dari model nomor empat. 101](#_Toc129457294)

[Gambar 4. 48 Confusion matrix model nomor empat. 101](#_Toc129457295)

[Gambar 4. 49 Grafik akurasi, validasi akurasi serta f1 score dari model nomor tujuh. 102](#_Toc129457296)

[Gambar 4. 50 Confusion matrix model nomor tujuh. 102](#_Toc129457297)

[Gambar 4. 51 Grafik akurasi, validasi akurasi serta f1 score dari model nomor delapan. 103](#_Toc129457298)

[Gambar 4. 52 Confusion matrix model nomor delapan. 103](#_Toc129457299)

# BAB I

**PENDAHULUAN**

Bagian ini akan berisi mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian dan sistematika penulisan.

## Latar Belakang

Pada zaman sekarang, informasi dan teknologi sudah berkembang pesat, tak terkecuali di Indonesia. Di Indonesia, sekarang sudah memasuki industri 4.0 dimana semuanya sudah serba digital, informasi bisa berkembang cepat melalui media digital hanya dalam hitungan detik saja. Dilansir dari (Rosyid, 2022) pengguna media sosial di Indonesia mencapai 191 Juta per Januari 2022. Hal ini membuktikan bahwa mayoritas penduduk Indonesia sudah menggunakan media sosial untuk mendapatkan serta berbagi informasi melalui media sosial.

Pengguna media sosial Twitter di Indonesia kurang lebih sudah mencapai 18,45 juta (Rizaty, 2022). Twitter merupakan jaringan informasi constant yang menghubungkan penggunanya dengan cerita, ide, pendapat dan berita terbaru apa saja yang dianggap menarik oleh banyak orang. Twitter memiliki fitur yang bernama *tweets* yang memungkinkan penggunanya untuk berbagi pendapat dan pikirannya tersebut ke platform dan pengguna Twitter. Tak hanya itu, banyak dijumpai pengguna Twitter yang mengatakan bahwa Twitter sebagai media sosial yang bebas dan tepat untuk menyampaikan pemikiran dan pendapat mereka akan suatu hal tak terkecuali ujar kebencian juga.

Saat ini, banyak sekali ditemukan *tweet*s di media sosial Twitter yang mengandung ujar kebencian salah satunya oleh pengguna Twitter di Indonesia, khususnya yang membahas tentang fenomena yang terjadi yang menjadi trending topik di beberapa sosial media akhir-akhir ini yaitu artis Selenna Gomez. Media sosial di seluruh dunia sempat diramaikan oleh fenomena yang terjadi antara artis internasional Selena Gomez dengan Kylie Jenner dan Hailey Bieber dimana fans mereka saling serang dan menyebarkan ujaran kebencian di kedua belah pihak yang berpotensi dapat menggiring opini public akan orang lain dan dapat berpotensi menyebabkan perpecahan di media sosial. Ujaran kebencian *(hate speech)* merupakan tindakan komunikasi yang dilakukan oleh individu atau kelompok tertentu dalam bentuk provokasi, hasutan, hinaan, penistaan, pencemaran nama baik, serta penyebaran berita bohong dalam aspek seperti ras, warna kulit, gender, etnis, cacat fisik, orientasi seksual, kewarganeraaan, agama, dan lain-lain (Permatasari & Subyantoro, 2020). Penyebab dari ujar kebencian adalah kesalahpahaman dan emosi dalam menerima suatu informasi, tidak sependapat dengan orang lain, dan adanya kebencian pribadi terhadap orang lain. Ujar kebencian bisa menyebabkan seseorang menjadi tidak nyaman, bahkan sampai bisa menimbulkan perpecahan politik. Selain itu, ujar kebencian juga dapat membuat sebuah lingkungan di masyarakat terbagi menjadi beberapa kelompok. Untuk itu, sebagai pengguna sosial media, khususnya Twitter, perlu adanya sebuah urgensi untuk mengetahui *tweet* yang mengandung unsur ujar kebencian atau tidak.

Terdapat salah satu metode yang dapat untuk mendeteksi ujar kebencian yaitu menggunakan sentimen analisis dengan teknik *deep learning* dan algoritma LSTM. Analisis sentimen merupakan suatu teknik *natural language processing* yang digunakan untuk menentukan status suatu data, entah itu mengandung muatan positif, netral, atau justru negatif (algorit.ma, 2022). Analisis sentimen memungkinkan kita untuk memprediksi sebuah *tweet* yang mengandung ujar kebencian berdasarkan suatu kata yang terkandung dalam satu kalimat. *Deep learning* merupakan teknik *machine learning* yang mengajarkan komputer untuk melakukan apa yang terjadi secara alami pada manusia, yaitu dengan belajar dengan memberi contoh. *Deep learning* bekerja dengan cara mempelajari kebiasaan/pola suatu data tanpa melakukan ekstraksi fitur terhadap data tersebut. *Deep learning* biasanya digunakan untuk klasifikasi suatu data teks, gambar, suara dan lain-lain. *Deep learning* dilatih dengan sekumpulan data yang cukup besar dan berlabel dengan arsitektur jaringan syaraf yang berisi banyak lapisan. Bidirectional LSTM atau bidirectional long shortterm memory merupakan metode dalam *deep learning* yang merupakan salah satu modifikasi dari RNN (Recurent Neural Network) yang mampu mengingat kumpulan informasi yang telah disimpan dalam jangka panjang sekaligus menghapus informasi yang sudah tidak relevan (algorit.ma, 2022). LSTM lebih efisien dalam memproses, memprediksi, sekaligus mengklasifikasikan data berdasarkan urutan waktu tertentu. Bidirectional LSTM bekerja secara dua arah, yaitu dari masa lampau ke masa depan dan dari masa depan ke masa lampau dalam menyimpan suatu informasi. Dengan metode Bidirectional LSTM ini, akan memprediksi sebuah *tweet* yang mengandung ujar kebencian dengan lebih akurat dan lebih optimal, khususnya dalam data teks.

## Rumusan Masalah

Tweet selalu mengandung sebuah kecenderungan bisa cenderung positif, negative maupun netral. Dari hal tersebut, bagaimana cara mengenali kecenderungan sebuah tweets , dan apakah pendekatan machine learning khususnya penggunaan LSTM dan Bidirectional LSTM dapat melakukan hal tersebut?

## Tujuan

Dari rumusan masalah terebut, ditentukan tujuan penelitian sebagai berikut:

1. Melakukan sentimen analisis ujaran kebencian *tweet*s pengguna Twitter terhadap fenomena selebriti internasional Selena Gomez menggunakan algoritma LSTM dan Bidirectional LSTM.
2. Menghitung akurasi optimal dari algoritma LSTM dan Bidirectional LSTM.
3. Mengetahui parameter *deep learning* yang mampu memberikan hasil paling optimal.

## Manfaat Penelitian

Dapat membuat sebuah model *deep learning* menggunakan algoritma LSTM dan Bidirectional LSTM yang dapat mendeteksi/mengenali ujaran kebencian di Twitter.

## Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Data yang digunakan adalah *tweet*s yang membahas tentang kejadian fenomenal Selena Gomez.
2. Dataset yang digunakan berbahasa Inggris
3. Dataset merupakan *tweet*s yang membahas tentang kejadian fenomenal Selena Gomez.

## Sistematika Penulisan

1. BAB I PENDAHULUAN

Diuraikan latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika penulisan penelitian ini.

1. BAB II LANDASAN TEORI

Teori-teori yang relevan dan mendukung penelitian ini yang berhubungan dengan *deep learning* khususnya metode Bidirectional LSTM*,* analisis sentimen.

1. BAB III METODOLOGI PENELITIAN

Berisi bagaimana pengolahan data untuk penelitian dan rencana/Langkah-langkah yang akan ditempuh dalam melakukan penelitian ini. Selain itu, akan dijelaskan proses implementasi dari metode *deep learning* metode Bidirectional LSTM untuk dapat melakukan analisis sentimen serta memaparkan arsitektur model yang akan digunakan untuk penelitian ini.

1. BAB IV IMPLEMENTASI DAN ANALISA HASIL

Implementasi serta hasil dari penelitian ini. Setelah itu, akan melakukan pembahasan dan analisa terhadap hasil pengujian-pengujian yang telah dilakukan.

1. BAB V PENUTUP

Menyimpulkan percobaan-percobaan dalam penelitian yang telah dilakukan dan juga akan diuraikan saran dari penulis untuk pengembangan dari penelitian ini.

1. BAB VI

Berisi tentang referensi yang digunakan dalam penelitian ini.

# BAB II

**TINJAUAN PUSTAKA DAN LANDASAN TEORI**

Pada Bab II ini, akan diuraikan tinjauan Pustaka dari beberapa referensi yang terkait dengan penelitian ini serta teori-teori yang mendukung dan relevan dengan penelitian ini, seperti analisis sentimen, jaringan syaraf tiruan, *deep learning*, LSTM, Bidirectional LSTM.



## Tinjauan Pustaka

Mengklasifikasi ujaran kebencian di Twitter merupakan hal yang penting agar tidak mudah tergiring oleh opini kebencian oleh oknum-oknum yang menyebarkan ujaran tersebut demi sebuah kepentingan. Oleh karena itu, dibutuhkan sebuah keahlian khusus untuk melakukan analisis sentimen/klasifikasi ujaran kebencian. Beberapa penelitian tentang analisis sentimen telah dilakukan terkait analisis sentimen Twitter. Penelitian oleh (Ahmad, Ali, & Ahtab, 2017) yang membahas tentang analisis sentimen *tweet*s menggunakan support vector machine (SVM) mendapatkan akurasi 85%. Selanjutnya terdapat juga penelitian oleh (Fauzi, 2018) yang membahas tentang penggunaan random forest untuk analisis sentimen Bahasa Indonesia mendapatkan skor OOB/akurasi sebesar 82,9%. Kemudian, terdapat penelitian oleh (Fitri, Andreswari, & Hasibuan, 2019) yang membahas tentang analisis sentimen Twitter menggunakan algoritma Naïve Bayes, Decision Tree, dan Random Forest yang memberikan masing-masing akurasi 83.43%, 82.91%, dan 82.91%. Terdapat juga penelitian oleh (Jianqiang, Xiaolin, & Xuejun, 2018) membahas tentang analisis sentimen menggunakan algoritma Deep Convolutional Neural Network yang mendapatkan akurasi terbaik sebesar 87.62%. Penelitian yang dilakukan oleh (Huq, Ali, & Rahman, 2017) membahas tentang analisis sentimen menggunakan KNN dan support vector machine dengan menggunakan 5-fold *cross validation* mendapatkan akurasi terbaik 84,32% dan 77,97%. Terdapat penelitian yang dilakukan oleh (Liao, Wang, Yu, Sato, & Cheng, 2016) yang membahas tentang analisis sentimen untuk klasifikasi latar situasi pada data Twitter yang membuahkan akurasi pengembangan sebesar 74.5%. Selanjutnya, (Abdelgwad, Soliman, Taloba, & Farghaly, 2021) melakukan penelitian tentang analisis sentimen berbasis aspek Bahasa Arab menggunakan model Bidirectional GRU yang mendapatkan akurasi optimal sebesar 83.98%

Untuk mempermudah dalam melihat dan membandingkan penelitian yang sudah diuraikan di atas, dapat melihat ke tabel 2.1. Dari referensi di atas, dapat dilihat bahwa analisis sentimen untuk Twitter menggunakan berbagai macam algoritma mendapatkan akurasi paling minimum adalah 82,91% dan hal ini bisa dikatakan sudah cukup baik.

|  |  |  |  |
| --- | --- | --- | --- |
| Pengarang dan Tahun | Judul | Metode | Keterangan |
| Mohammed M.Abdelgwad, Taysir Hassan A Soliman, Ahmed I.Taloba,  Mohamed Fawzy Farghaly  (2021) | Arabic Aspect Based Analisis sentimen Using Bidirectional GRU Based Models | Bidirectional GRU | Mendapatkan akurasi optimal sebesar 83,98% |
| Veny Amilia Fitri, Rachmadita Andreswari, Muhammad Azani Hasibuan (2019) | Analisis sentimen of Social Media Twitter with Case of Anti-LGBT Campaign in Indonesia using Naïve Bayes, Decision Tree, and Random Forest Algorithm | Naïve Bayes | Mendapatkan akurasi sebesar 83,43% |
| Decision Tree | Mendapatkan akurasi sebesar 82,91% |
| Random Forest | Mendapatkan akurasi sebesar 82,91% |
| Muhammad Ali Fauzi (2018) | Random Forest Approach for Analisis sentimen in Indonesia Language | Random Forest | Melakukan nilai OOB/akurasi sebesar 82,9% |
| Zhao Jianqiang, Gui Xiaolin, Zhang Xuejun (2018) | Deep Convolution Neural Networks for Twitter Analisis sentimen | *Deep learning* CNN (Convolutional Neural Network) | Mendapatkan akurasi sebesar 87,62% |
| Mohammad Rezwanul Huq, Ahmad Ali, Anika Rahman (2017) | Analisis sentimen on Twitter Data using KNN and SVM | KNN Classifier | Mendapatkan akurasi terbaik 84,32% dengan 5-fold *cross validation* |
| Support Vector Machine (SVM) | Mendapatkan akurasi terbaik 77,97% dengan 5-fold *cross validation* |
| Munir Ahmad, Shabib Aftab, Iftikhar Ali (2017) | Analisis sentimen of *Tweet*s using SVM | Support Vector Machine (SVM) | Mendapatkan akurasi sebesar 85% |
| Shiyang Liaoa, Junbo Wang, Ruiyun Yua  ,Koichi Satob  ,Zixue Cheng | CNN for situations understanding based on  analisis sentimen of Twitter data | CNN | Mendapatkan akurasi tahap pengembangan sebesar 74,5% |

Tabel 2. 1 Tinjauan Pustaka Penelitian.



Menurut (Singhal & Bhattacharyya, 2016) *deep learning* mempunyai potensi untuk mengatasi kekurangan pada *machine learning*, khususnya support vector machine dalam melakukan analisis sentimen dengan memberikan beberapa potensi keuntungan diantaranya adalah kemampuan untuk beradaptasi dengan variasi tugas dengan perubahan yang sangat kecil dalam sistem itu , memungkinkan pembelajaran representasi yang baik dan tidak memerlukan *feature extraction* tetapi menggunakan *words embedding* sebagai input yang menyimpan informasi.

Berangkat dari hal ini, saya ingin melakukan penelitian analisis sentimen ujaran kebencian Twitter di Indonesia menggunakan *deep learning* dengan Teknik LSTM dan Bidirectional LSTM.

## Landasan Teori

### Analisis sentimen

Analisis sentimen adalah studi berbasis komputer tentang pendapat, keyakinan, dan emosi orang tentang entitas tertentu. Entitas memiliki kemampuan untuk mempengaruhi orang, gerakan, atau objek (Medhat, Hassan, & Korashy, 2014) . Analisis sentimen dengan kata lain digunakan untuk mengetahui keadaan suatu data, baik yang mengandung sentimen positif, netral, maupun negatif. Dimungkinkan juga untuk menggunakan analisis sentimen sebagai klasifikasi. Analisis sentimen bekerja menggunakan parameter berikut:

Diagram

Description automatically generated

Gambar 2.1 Alur Kerja Analisis sentimen (Medhat, Hassan, & Korashy, 2014) Jaringan Syaraf Tiruan(Artificial Neural Network)

### Jaringan Syaraf Tiruan

Jaringan saraf tiruan terinspirasi oleh model awal pemrosesan sensorik oleh otak. Jaringan saraf tiruan dapat dibuat dengan mensimulasikan jaringan memodelkan neuron di komputer. Dengan menerapkan algoritma yang meniru proses nyata neuron, dapat membuat jaringan 'belajar' untuk memecahkan banyak jenis masalah (A)

Jaringan saraf tiruan (JST) merupakan jaringan yang terdiri dari unit pemrosesan kecil yang dimodelkan seperti sistem saraf manusia. JST adalah sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal dan internal yang mengalir melalui jaringan. Karena sifatnya yang adaptif, JST sering disebut sebagai jaringan adaptif. Sederhananya, JST adalah alat pemodelan data statistik non-linear. JST dapat digunakan untuk memodelkan hubungan yang kompleks antara input dan *output* untuk menemukan pola dalam data. Menurut teorema yang disebut "teorema estimasi universal", JST dengan setidaknya satu lapisan tersembunyi dengan fungsi aktivasi nonlinier dapat memodelkan setiap fungsi terukur boreal dari satu dimensi ke dimensi lain.

Model pada JST pada dasarnya merupakan fungsi model matematika yang mendefinisikan fungsi {\displaystyle f:X\rightarrow Y}dari X ke Y. Istilah "jaringan" pada JST merujuk pada interkoneksi dari beberapa *neuron* yang diletakkan pada lapisan yang berbeda. Secara umum, lapisan pada JST dibagi menjadi tiga bagian:

* Lapis masukan (*input layer*) terdiri dari *neuron* yang menerima data masukan dari variabel X. Semua *neuron* pada lapis ini dapat terhubung ke *neuron* pada lapisan tersembunyi atau langsung ke lapisan luaran jika jaringan tidak menggunakan lapisan tersembunyi.
* Lapisan tersembunyi (*hidden layer*) terdiri dari *neuron* yang menerima data dari lapisan masukan.
* Lapisan luaran (*output layer*) terdiri dari *neuron* yang menerima data dari lapisan tersembunyi atau langsung dari lapisan masukan yang nilai luarannya melambangkan hasil kalkulasi dari X menjadi nilai Y.

#### Dense *Layer*

Dense *layer* merupakan *layer* yang simpel dimana setiap neuron menerima input dari *layer* sebelumnya ataupun bekerja sebagai input *layer* (Dumane). *Layer* ini akan melakukan operasi perkalian vector. Hasil dari setiap neuron dari lapisan sebelumnya menuju ke setiap neuron tunggal dari lapisan padat. Dapat dikatakan bahwa jika lapisan sebelumnya mengeluarkan matriks (M x N) dengan menggabungkan hasil dari setiap neuron, keluaran ini melewati lapisan padat di mana jumlah neuron pada lapisan padat harus N.

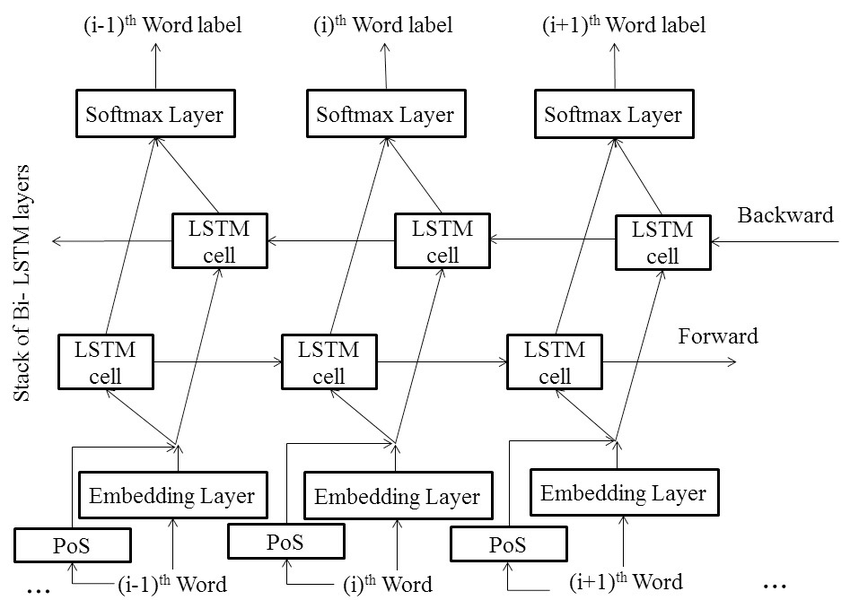
Diagram

Description automatically generated

Gambar 2.2 Gambar dense *layer* (Verma, 2021)

#### Embedding *Layer*

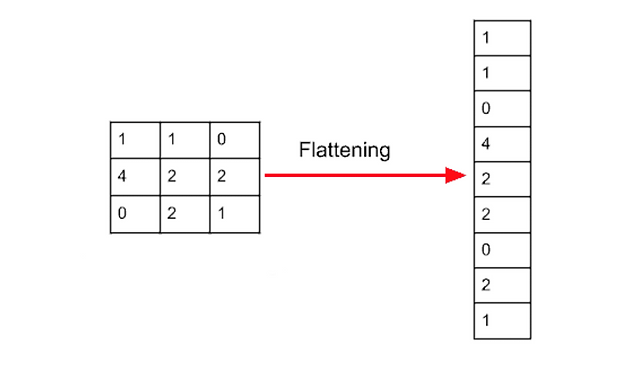
Embedding *layer* merupakan *layer* dalam *deep learning* yang merepresentasikan kosa kata dalam sebuah dokumen yang paling populer yang direpresentasikan dalam sebuah vektor (Karani, 2018). Setelah itu, word embedding dapat menangkap konteks, kesamaan semantik dan sintaksis dalam suatu dokumen.



Gambar 2. 3 Embedding *layer* di dalam model BiLSTM (ReaserchGate)

#### Flatten *Layer*

Flatten *layer* berfungsi untuk melakukan transformasi neurons dari multi dimensi menjadi dimensi tunggal. Dalam penelitian ini, flatten *layer* digunakan untuk melakukan transformasi *output* dari *layer* Bidirectional/Embedding *layer* yang tergolong multi dimensi (3 dimensi) menjadi dimensi tunggal agar bisa diterima oleh hidden *layer* dense.



Gambar 2. 4 Flatten *layer*

### *Deep learning*

Menurut (LeCun, Bengio, & Hinton, 2015), *deep learning* memungkinkan pemodelan komputasi yang memungkinkan beberapa lapisan/*layer* pemrosesan untuk mempelajari representasi data dengan berbagai tingkat abstraksi. Metode ini dapat menemukan struktur rumit dalam kumpulan data besar dengan menggunakan algoritma *backpropagation* untuk menunjukkan bagaimana mesin harus mengubah parameter internalnya yang digunakan untuk menghitung representasi di setiap lapisan dari representasi di lapisan sebelumnya.

Selain itu, menurut (MathWorks, 2022) *deep learning* adalah cara menggunakan komputer untuk melakukan tugas-tugas yang biasanya dilakukan oleh manusia atau dapat digambarkan seperti yang dilakukan dengan metode tertentu. Model komputer digunakan dalam *deep learning* untuk melakukan tes klasifikasi video, audio atau teks. Tujuan model pembelajaran adalah yang terbaru, meskipun mengurangi aktivitas manusia. Model ini didasarkan pada sejumlah besar data berlabel dan struktur jaringan *multilayer*. Istilah "deep" biasanya digunakan untuk sisa jaringan yang bersangkutan. Jaringan syaraf tradisional terdiri dari dua sampai tiga lapisan, sedangkan jaringan syaraf dengan 150 lapisan dapat ditemukan di *deep learning*. Di sisi lain, model *deep learning* menggunakan sejumlah besar data untuk menamainya dan juga menggunakan analisis jaringan saraf yang berasal dari data untuk membuat fungsi yang berbeda. Berikut adalah contoh *deep learning* dalam jaringan syaraf tiruan:Diagram

Description automatically generated

Gambar 2. 5 Contoh jaringan syaraf tiruan pada *deep learning*  (MathWorks, 2022)

Di dalam *deep learning* , terdapat beberapa komponen pendukung seperti optimizer, fungsi aktifasi, regularization dan fungsi loss.

#### Optimizer

Optimizer adalah algoritma atau metode yang digunakan untuk mengubah atribut jaringan saraf, seperti bobot tingkat pembelajaran, untuk mengurangi loss (Musstafa, 2021). Algoritma atau strategi optimizer bertanggung jawab untuk mengurangi loss dan mencapai hasil yang paling akurat.

##### RMSProp

RMSprop menggunakan sinyal gradien yang menyesuaikan ukuran langkah untuk setiap bobot secara terpisah (Musstafa, 2021). Dalam algoritma ini, dua gradien pertama dibandingkan untuk tanda. Jika mereka memiliki tanda yang sama, pergi ke arah yang benar dan karena itu tingkatkan ukuran langkah dengan sebagian kecil. Di sisi lain, jika tandanya dibalik, maka harus mengurangi ukuran langkah dan kemudian membatasi ukuran langkah dan melakukan pembaruan bobot. Masalah dengan RMSProp adalah tidak bekerja dengan baik dengan kumpulan data besar atau selama pembaruan mini-lokal. Propagator RMS juga dapat dianggap sebagai peningkatan dari pengoptimal AdaGrad karena meminimalkan tingkat pembelajaran yang menurun secara monoton. Algoritma ini terutama berfokus pada mempercepat proses optimasi dengan mengurangi jumlah evaluasi fungsi untuk mencapai minimum lokal. Algoritma mengkuadratkan rata-rata pergerakan gradien untuk setiap bobot dan membagi gradien dengan akar kuadrat dari kuadrat rata-rata.

RMS Prop(Root Mean Square) Deep Learning Optimizer

Gambar 2. 6 Persamaan optimizer RMSProp (Musstafa, 2021)

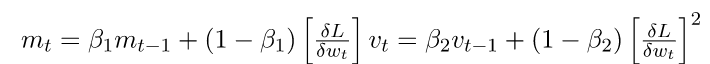
di mana gamma adalah faktor pelupa. Bobot diperbarui dengan rumus di bawah ini

RMS Prop(Root Mean Square) Deep Learning Optimizer

Gambar 2. 7 Persamaan optimizer RMSProp (Musstafa, 2021)

Sederhananya, jika ada parameter yang menyebabkan fungsi cost banyak berosilasi, itu akan menghukum memperbarui parameter itu. Algoritma ini memiliki beberapa keunggulan dibandingkan versi sebelumnya dari algoritma gradient descent. Algoritma konvergen dengan cepat dan membutuhkan lebih sedikit penyetelan daripada algoritma penurunan gradien dan variannya. Masalah dengan RMS Prop adalah kecepatan pembelajaran harus diatur secara manual dan nilai yang disarankan tidak berfungsi untuk setiap aplikasi.

##### Adam

Algoritma optimizer ini merupakan perpanjangan lebih lanjut dari penurunan gradien stokastik untuk memperbarui bobot jaringan selama pelatihan. Pengoptimal Adam memperbarui pembelajaran setiap bobot jaringan secara terpisah (Musstafa, 2021). Alih-alih menyesuaikan learning rate berdasarkan momen pertama (mean) seperti RMS Prop, Adamis juga menggunakan momen gradien kedua. Algoritma ini mudah diimplementasikan, memiliki runtime yang lebih cepat, kebutuhan memori yang lebih rendah, dan memerlukan konfigurasi yang lebih sedikit daripada algoritma pengoptimalan lainnya.

Gambar 2. 8 Persamaan optimizer adam (Musstafa, 2021)

Rumus di atas merupakan cara kerja pengoptimal adam. Di sini B1 dan B2 mewakili laju peluruhan rata-rata gradien.

Kelemahan dari optimizer ini biasanya berfokus pada waktu komputasi yang lebih cepat, sedangkan algoritma seperti penurunan gradien stokastik fokus pada titik data. Oleh karena itu, algoritma seperti SGD menggeneralisasi data dengan lebih baik dengan mengorbankan kecepatan komputer yang rendah. Oleh karena itu, algoritma optimasi dapat dipilih sesuai dengan kebutuhan dan tipe data.

#### Fungsi aktifasi

(Marimuthu, 2022) mengatakan bahwa fungsi aktifasi bertanggung jawab atas apakah neuron dalam jaringan saraf *deep learning* harus diaktifkan atau tidak. Artinya, ia menggunakan beberapa operasi matematika sederhana untuk menentukan apakah neuron input jaringan relevan atau tidak relevan dengan proses prediksi. Tujuan dari fungsi aktivasi adalah kemampuan untuk memasukkan nonlinier ke dalam jaringan saraf dan menghasilkan *output* dari sekumpulan nilai input yang dimasukkan ke dalam lapisan.

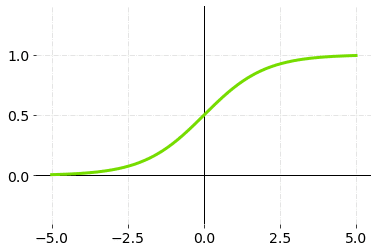
##### Sigmoid

Fungsi aktifasi sigmoid menerima nilai input dan mengembalikan 0 atau 1 (Marimuthu, 2022). Sigmoid sering digunakan untuk pemrosesan klasifikasi biner. Sigmoid memberikan probabilitas bahwa kelas tertentu ada. Ketika sigmoid diwakili dalam matematika, persamaan itu adalah:

Diagram, schematic

Description automatically generated

Gambar 2. 9 persamaan fungsi aktifasi sigmoid (Marimuthu, 2022)



Gambar 2. 10 Grafik fungsi aktifasi sigmoid (Marimuthu, 2022)

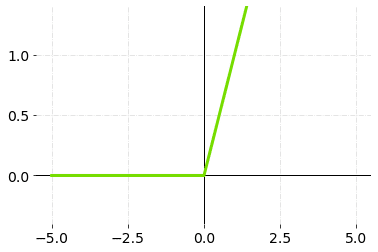
##### Relu

ReLU adalah singkatan dari Rectified Linear Unit dan merupakan salah satu fungsi aktifasi yang paling umum digunakan dalam aplikasi. Re-Lu memecahkan masalah gradien yang hilang karena fungsi re-lu memiliki gradien maksimum 1 (Marimuthu, 2022). Ini juga memecahkan masalah saturasi neuron karena kemiringan fungsi ReLU tidak pernah nol. ReLU berkisar dari 0 hingga tak terhingga. Ketika direpresentasikan dalam matematika, Re-Lu memiliki persamaan ini:

Text

Description automatically generated

Gambar 2. 11 Persamaan fungsi aktifasi ReLu (Marimuthu, 2022)



Gambar 2. 12 Grafik fungsi aktifasi ReLu (Marimuthu, 2022)

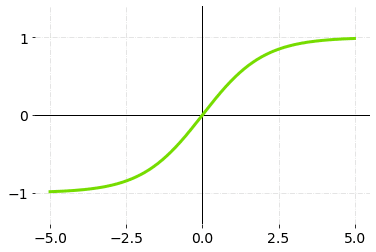
##### Softmax

Kombinasi dari banyak sigmoid disebut fungsi Softmax. Ini menentukan probabilitas relatif. Serupa dengan fungsi aktifasi sigmoid, fungsi Softmax mengembalikan probabilitas setiap kelas/label. Dalam klasifikasi multi-kelas, fungsi aktifasi softmax paling sering digunakan untuk lapisan terakhir dari jaringan saraf (Marimuthu, 2022). Fungsi softmax memberikan probabilitas kelas saat ini relatif terhadap yang lain. Ini berarti bahwa dia juga mempertimbangkan kemungkinan kelas lain. Ketika direpresentasikan dalam matematika, softmax adalah persamaan seperti ini:

Text

Description automatically generated

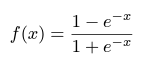
Gambar 2. 13 Persamaan fungsi aktifasi softmax (Marimuthu, 2022)



Gambar 2. 14 grafik fungsi aktifasi softmax (Marimuthu, 2022)

##### TanH

Fungsi hiperbolik tangen atau tanh cocok digunakan untuk jaringan syaraf tiruan *multi*-*layer* (Datta, 2020). Fungsi tanh didefinisikan sebagai:



Gambar 2. 15 Persamaan fungsi aktifasi tanh (Datta, 2020)

Fungsi aktifasi tanh merupakan modifikasi dari fungsi aktifasi sigmoid karena persamaan tanh didefinisikan sebagai berikut

Gambar 2.15. persamaan fungsi aktifasi tanh

Tanh merupakan fungsi kontinu dimana *outputnya* berkisar dari negatif satu sampai dengan satu. Dari hal itu, *outputnya* bisa dikatakan negatif, nol, ataupun positif. Oleh karena itu, fungsi ini terpusat nol dan menyelesaikan masalah 'fungsi aktivasi bukan berpusat nol' dari fungsi sigmoid.

#### Regularization

Regularisasi adalah serangkaian teknik yang mengurangi kompleksitas model jaringan saraf selama pelatihan untuk menghindari overfitting. Salah satu metode regularisasi yang umum digunakan adalah Dropout. Istilah "dropout" mengacu pada putusnya node (input dan lapisan tersembunyi) dalam jaringan saraf. Semua koneksi maju dan mundur dengan node yang dijatuhkan dihapus sementara, sehingga menciptakan arsitektur jaringan baru dari jaringan induk (Yadav).

Dropout berarti selama proses *training* dengan beberapa probabilitas P neuron dari jaringan syaraf dimatikan.

A picture containing watch

Description automatically generated

Gambar 2. 16 Contoh implementasi dari Dropout

Contoh dapat diambil dari gambar di atas, di mana di sebelah kiri adalah jaringan saraf feedforward tanpa Dropout. Menggunakan penurunan dengan probabilitas P = 0,5 selama proses pelatihan, secara acak mematikan neuron dan membentuk jaringan saraf seperti yang ditunjukkan di sebelah kanan. Ini berarti bahwa setengah dari neuron mati dan tidak dihitung, yang membuat jaringan saraf lebih sederhana dan tidak terlalu kompleks, yang mengurangi terjadinya over-tuning. Penonaktifan neuron dengan probabilitas tertentu P diterapkan pada setiap langkah maju dan pembaruan bobot.

#### Fungsi loss

Fungsi loss adalah fungsi yang membandingkan target dan nilai keluaran yang diprediksi; mengukur seberapa baik jaringan saraf memodelkan data pelatihan. Selama pelatihan, tujuannya adalah untuk meminimalkan loss antara kekuatan yang diprediksi dan target (Yathish, 2022). Ada dua jenis fungsi loss dalam jaringan syaraf tiruan, yaitu fungsi regresi dan fungsi kerugian klasifikasi (Yathish, 2022)

##### Regression Fungsi loss

Saat digunakan dalam jaringan saraf regresi, model memprediksi nilai keluaran yang benar dengan nilai masukan (bukan judul yang dipilih sebelumnya, misalnya, mean squared error (MSE) dan mean absolute error (MAE) (Yathish, 2022).

1. Mean Squared Error (MSE)

Text

Description automatically generated

Gambar 2. 17 Persamaan MSE (Yathish, 2022)

Fungsi ini memiliki banyak properti yang membuatnya sangat cocok untuk menghitung loss. Selisihnya kuadrat, artinya tidak masalah apakah nilai prediksi lebih besar atau lebih kecil dari nilai target; Namun, skor dengan kesalahan besar dihukum. MSE juga merupakan fungsi cembung dengan minimum global yang terdefinisi dengan baik. Ini memudahkan penggunaan pengoptimal penurunan gradien untuk menentukan nilai bobot. Namun, salah satu kelemahan dari fungsi kerugian ini adalah sangat sensitif terhadap penyimpangan, artinya jika nilai prediksi secara signifikan lebih tinggi atau lebih rendah dari nilai target, maka akan meningkatkan kerugian secara signifikan.

1. Mean Absolute Error (MAE)

Text

Description automatically generated with medium confidence

Gambar 2. 18 Persamaan MAE (Yathish, 2022)

Fungsi loss ini digunakan dalam beberapa kasus sebagai alternatif untuk MSE. MAE digunakan dalam kasus di mana ada banyak outlier dalam data pelatihan. Fungsi kerugian ini juga memiliki beberapa kelemahan; karena jarak rata-rata mendekati nol, optimasi penurunan gradien tidak akan bekerja karena turunan dari fungsi pada 0 tidak terdefinisi (yang menyebabkan kesalahan karena tidak mungkin untuk membagi dengan 0).

##### Classification Fungsi loss

Digunakan untuk klasifikasi dalam jaringan saraf, di mana jaringan saraf menghasilkan vektor probabilitas input yang dimiliki oleh beberapa kelas yang telah ditentukan, dan kemudian dapat memilih kelas dengan probabilitas kepemilikan tertinggi (Yathish, 2022). Sebagai contoh, entropi silang biner dan entropi silang kategoris dari fungsi kerugian.

1. Binary Crossentropy

Jaringan saraf klasifikasi menghasilkan vektor probabilitas dari kemungkinan bahwa input yang diberikan cocok dengan setiap kelas yang telah ditentukan, dan kemudian memilih kelas yang paling mungkin sebagai hasilnya. Dalam klasifikasi biner, hanya ada dua nilai yang mungkin, nilai aktual y adalah 0 atau 1. Untuk menentukan secara akurat kerugian antara nilai aktual dan prediksi, perlu membandingkan nilai aktual (0 atau 1) dengan probabilitas bahwa input termasuk dalam kategori itu (p(i ) ) = probabilitas, kelas itu adalah 1; 1 — p(i) = probabilitas kelas itu

Text

Description automatically generated with low confidence

Gambar 2. 19 Persamaan fungsi loss binary crossentropy

1. Categorical crossentropy

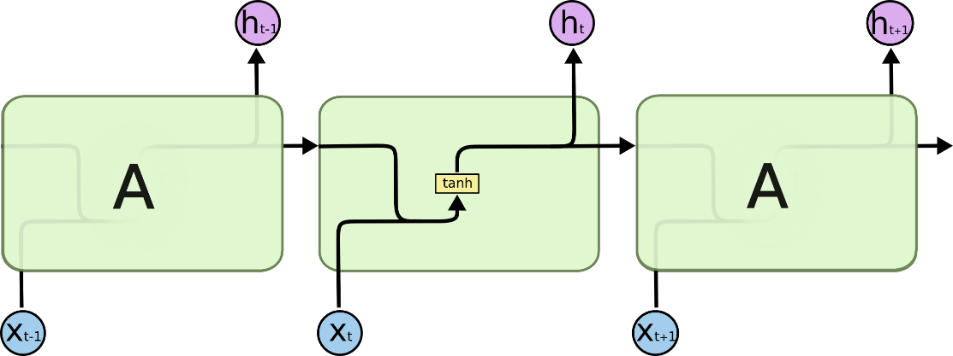
Dalam kasus di mana jumlah kategori lebih dari dua, menggunakan categorical crossentropy. Categorical crossentropy mengikuti proses yang sangat mirip dengan binary crossentropy, satu-satunya perbedaan adalah jumlah kategori yang tersedia.

Diagram, text

Description automatically generated

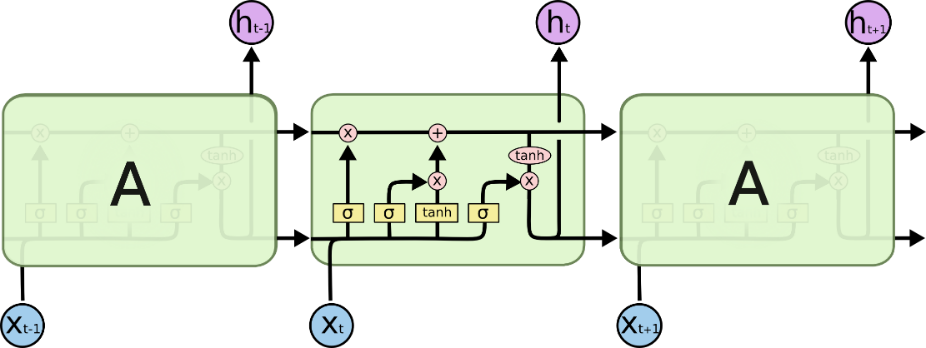
Gambar 2. 20 Persamaan fungsi loss categorical crossentropy

### Long Short-Term Memory (LSTM)

Long Short-Term Memory atau biasa dikenal dengan LSTM merupakan sebuah RNN (jaringan syaraf berulang) yang dapat mengatasi permasalahan ketergantungan jangka Panjang (Yu, Si, Hu, & Zhang, 2019). LSTM meningkatkan kapasitas mengingat sel rekuren standar (RNN) dengan mengenalkan “gate” atau gerbang ke dalam cell state. LSTM dapat mengingat informasi untuk waktu yang lama (Colah, 2015). Semua jenis RNN mengambil bentuk rantai modul jaringan saraf berulang.

Gambar 2. 21 Contoh RNN (Colah, 2015)

LSTM juga memiliki struktur seperti rantai ini, tetapi modul yang diulang memiliki struktur yang berbeda. Alih-alih satu lapisan jaringan saraf, ada empat yang berkomunikasi dengan cara yang sangat spesifik.



Gambar 2. 22 Struktur LSTM (Colah, 2015)

Kunci LSTM adalah cell state, garis horizontal yang melintasi bagian atas grafik. Keadaan sel itu bergerak langsung di sepanjang rantai dengan hanya beberapa interaksi linier kecil. Sangat mudah untuk informasi mengalir melalui tidak berubah. LSTM dapat menghapus atau menambahkan data ke keadaan sel yang dikendalikan oleh struktur yang disebut Gate. Gate adalah cara opsional untuk mentransfer informasi. Mereka terdiri dari lapisan jaringan saraf sigmoid dan operasi perkalian titik. Alur kerja dari LSTM akan berjalan seperti berikut:

Diagram

Description automatically generated

Gambar 2. 23 Cell state melalui forget gate (Colah, 2015)

Langkah pertama adalah menentukan informasi apa yang akan dibuang dari cell state yang dilakukan oleh forget gate.

Diagram

Description automatically generated

Gambar 2. 24 *Output* gerbang sigmoid (Colah, 2015)

Menentukan informasi baru yang akan disimpan dalam cell state.

Diagram

Description automatically generated with medium confidence

Gambar 2. 25 Mengupdate cell state lama (Colah, 2015)

Melakukan pembaharuan informasi terhadap cell state lama dengan cell state yang baru, atau bisa dikatakan melakukan kolaborasi informasi antara informasi di cell state lama dengan informasi yang ada di cell state yang terbaru

A picture containing text, clock

Description automatically generated

Gambar 2. 26 Menentukan *output* terbaru dari cell state. (Colah, 2015)

### Bidirectional Long Sshort-Term Memory (LSTM)

Bidirectional LSTM atau LSTM dua arah adalah perpanjangan dari LSTM tradisional yang dapat meningkatkan kinerja model untuk masalah klasifikasi urutan. Untuk masalah di mana semua langkah waktu dari urutan input tersedia, LSTM dua arah melatih 2 urutan input. Yang pertama dalam urutan input apa adanya dan yang kedua dalam salinan terbalik dari urutan input. Ini dapat memberikan konteks tambahan ke jaringan dan mengarah pada pembelajaran masalah yang lebih cepat dan lebih lengkap (Mungalpara, 2021).

Diagram

Description automatically generated

Gambar 2. 27 Bidirectional LSTM (Mungalpara, 2021)

### Preprocessing

Preprocessing merupakan sebuah proses untuk membuat input untuk analisis yang diberikan kurang kompleks dengan cara yang tidak mempengaruhi interpretabilitas atau kesimpulan substantif dari model selanjutnya (Denny & Spirling, 2017). Preprocessing pada dasarnya bertujuan untuk melakukan transformasi data tanpa mengubah isi dari data tersebut agar data dapat lebih mudah digunakan. Preprocessing yang biasanya ada dalam melakukan klasifikasi teks ialah *lowercasing*, *stemming*, *stopword* *removal*, pembersihan karakter khusus, pembersihan karakter *non-alphanumeric* dan masih banyak lagi.

#### *Lowercasing*

*Lowercasing* merupakan salah satu tahap preprocessing yang sering diaplikasikan jika menghadapi data teks. Hal ini bertujuan untuk mengatasi masalah suatu karakter yang sebenernya sama, akan tetapi dapat terdeteksi beda oleh computer karena perbedaan huruf kapital dan kecil (misal Gajah dengan gajah) (Denny & Spirling, 2017).

#### *Stemming*

*Stemming* merupakan tahapan dalam teks preprocessing yang akan mereduksi sebuah kata menjadi bentuk dasar dari kata tersebut (Denny & Spirling, 2017). *Stemming* sering diartikan sebagai Teknik pengurangan kosakata. *Stemming* dapat dimisalkan dengan reduksi kata “menyapu” menjadi “sapu”.

#### *Lemmatization*

*Lemmatization* biasanya mengacu pada melakukan sesuatu dengan benar dengan menggunakan kosa kata dan analisis morfologi kata-kata, biasanya bertujuan untuk menghilangkan akhiran infleksional saja dan mengembalikan bentuk dasar atau kamus dari sebuah kata, yang dikenal sebagai lemma (Edu). Dalam arti lain, tahapan ini berfungsi untuk mengubah suatu bentuk kata menjadi bentuk dasarnya. Sebagai contoh, kata Bahasa Inggris “drove” yang merupakan bentuk ketiga akan diubah menjadi “drive”

#### *Stopword* Removal

Terdapat beberapa kata yang memiliki makna yang berarti, dan terdapat beberapa kata yang tidak memiliki kata berarti. Kata-kata yang tidak memberikan informasi/makna berarti sering disebut dengan “stop word”. *Stopword* removal berfungsi untuk menghilangkan karakter yang kurang memiliki makna (Denny & Spirling, 2017). Kata-kata yang kurang memiliki makna biasanya merupakan kata konjungsi, kata-kata fungsi seperti “itu”, “dan”, dan “dia” sebagai contoh.

# BAB III

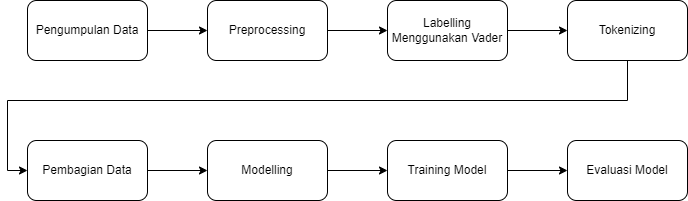
**METODE PENELITIAN**

Bagian ini menguraikan mengenai rencana langkah-langkah yang akan dilakukan dalam penelitian. Tahapan yang aka nada dalam penelitian ini diantaranya mulai dari pengumpulan data, pengolahan data, modelling, evaluasi model serta skenario pengujian yang akan dilakukan dalam penelitian ini.



## Tahapan Penelitian

Pada BAB ini, akan dijelaskan tahapan metode penelitian yang terdiri dari pengumpulan data, pengolahan data, pembuatan model *deep learning*, evaluasi model hingga skenario pengujian. Skema penelitian bisa ditinjau dalam gambar 3.1.



Gambar 3. 1 Tahapan Penelitian

## Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah hasil dari *web scrapping* di situs Twitter dengan mencari 11.646 total *tweet* yang mengandung kata “selena gomez”. Peneliti melakukan *web scrapping* untuk mengambil *tweet* yang tidak mengandung link serta mention/reply dari pengguna lain sehingga *tweet* tersebut murni sebuah *tweet* yang berisi ide/gagasan/pendapat yang ingin dituliskan oleh pengguna. Setelah itu, data disatukan menjadi sebuah dataframe/tabel yang berisi username pengunggah *tweet*, waktu dibuatnya *tweet* tersebut serta isi *tweet*/teks *tweet* tersebut. Setelah itu, dataframe disimpan dalam format excel (.xlsx) agar bisa dilihat dengan mudah dan dapat digunakan kembali ketika akan digunakan dalam penelitian. Berikut beberapa sampel dataset hasil *web scrapping* dapat dilihat pada tabel 3.1.

|  |  |  |
| --- | --- | --- |
| created at | username | teks |
| Thu Mar 02 23:59:30 +0000 2023 | darkonsun | social media is a disease and this hailey bieber selena gomez tea is too hot |
| Thu Mar 02 23:56:18 +0000 2023 | babyyyannieee | Justin Bieber &amp; Selena Gomez followed each other on Twitter 😭teenage me is crying 😭 |
| Thu Mar 02 23:55:57 +0000 2023 | likabeast101 | Selena Gomez drove by my house 🙀 |
| Thu Mar 02 23:55:36 +0000 2023 | whoismelanyway | I understand hailey Bieber because if my husbands ex was Selena Gomez I’d be worried too |
| Thu Mar 02 23:54:50 +0000 2023 | \_mayyraa | but then i always realize because SELENA GOMEZ ALWAYS LOOKS GOOD 😭🫶 |

Tabel 3. 1 Sample dataset

## Preprocessing

Tahapan ini bertujuan untuk membuat kualitas data yang akan digunakan dalam *training*/analisis menjadi baik. Dalam kasus analisis sentimen ujaran kebencian ini, untuk membuat kualitas dataset teks menjadi baik, maka perlu dilakukan beberapa tahapan preprocessing diantaranya adalah seleksi data, *lowercasing*, pembersihan karakter selain *alphanumeric*, pembersihan karakter khusus/tidak biasa, membersihkan *stopword*, *stemming*, dan lemmatization.

### Seleksi Data

Seleksi data merupakan tahapan untuk menyeleksi data agar kualitas data menjadi lebih baik dan mengurangi data yang sekiranya tidak memberikan informasi yang banyak dan berguna nantinya. Data yang akan diseleksi adalah data teks yang mengandung kata “RT”. Kata “RT” di dalam Twitter berarti seseorang melakukan *retweet* terhadap *tweet* orang lain. Dalam kata lain, *retweet* berarti juga *repost*, ataupun dalam Bahasa Indonesia berarti posting ulang. Data seperti itu perlu dihilangkan dari dataset karena termasuk ke dalam kategori duplikasi data dan akan memengaruhi kinerja model dalam *training* dan juga akurasi yang didapatkan. Berikut contoh data yang mengandung kata “RT” yang dapat dilihat pada tabel 3.2.

|  |  |  |
| --- | --- | --- |
| created at | username | teks |
| Thu Mar 02 23:59:54 +0000 2023 | izzvhatirvh | RT @armylieber100: Aren't Selena Gomez fans tired yet? it's been almost a week like move on and leave Hailey alone |
| Thu Mar 02 23:59:44 +0000 2023 | Kenzie5Peterson | RT @juliannarvivas: Im a Selena Gomez &amp; you a Hailey Bieber. |
| Thu Mar 02 23:59:44 +0000 2023 | sammii\_gv | RT @juliannarvivas: Im a Selena Gomez &amp; you a Hailey Bieber. |
| Thu Mar 02 23:59:36 +0000 2023 | \_xolesley | RT @kvnnrxa: justin bieber and selena gomez follow eachother. id kms if i was hailey |
| Thu Mar 02 23:59:34 +0000 2023 | DoncasterAlexis | RT @juliannarvivas: Im a Selena Gomez &amp; you a Hailey Bieber. |

Tabel 3. 2 Contoh data dengan kata “RT”.

### *Lowercasing*

*Lowercasing* merupakan tahapan transformasi huruf menjadi huruf kecil. Input dari tahapan ini adalah sebuah kalimat yang terdiri dari huruf besar dan kecil yang akan diproses menjadi sebuah *output* kalimat yang terdiri dari huruf kecil saja. Berikut input dan *output* setelah dilakukan *lowercasing* dapat dilihat pada tabel 3.3.

|  |  |
| --- | --- |
| **Sebelum** | **Sesudah** |
| social media is a disease and this hailey bieber selena gomez tea is too hot | social media is a disease and this hailey bieber selena gomez tea is too hot |
| Justin Bieber &amp; Selena Gomez followed each other on Twitter 😭teenage me is crying 😭 | justin bieber &amp; selena gomez followed each other on Twitter 😭teenage me is crying 😭 |
| Selena Gomez drove by my house 🙀 | selena gomez drove by my house 🙀 |

Tabel 3. 3 Contoh data dalam proses *lowercasing.*

### Pembersihan Karakter *Non-alphanumeric*

*Alphanumeric* merupakan karakter yang teridiri dari huruf dan angka. Pembersihan *non-alphanumeric* bertujuan untuk membersihkan karakter selain huruf dan angka, seperti tanda baca dan karakter khusus lainnya. Tahapan ini dilakukan agar data yang akan digunakan dalam penelitian dapat bekerja secara maksimal karena dalam membuat model *deep learning*, nantinya setiap kata akan dijadikan sebuah token yang di transformasi menjadi sebuah angka. Berikut input dan *output* untuk proses pembersihan *non-alphanumeric* karakter yang dapat dilihat pada tabel 3.4.

|  |  |
| --- | --- |
| **Sebelum** | **Sesudah** |
| social media is a disease and this hailey bieber selena gomez tea is too hot | social media is a disease and this hailey bieber selena gomez tea is too hot |
| justin bieber &amp; selena gomez followed each other on Twitter 😭teenage me is crying 😭 | justin bieber amp selena gomez followed each other on Twitter teenage me is crying |
| selena gomez drove by my house 🙀 | selena gomez drove by my house |

Tabel 3. 4 Contoh data pada proses Pembersihan *Non-alphanumeric.*

### Pembersihan Karakter Khusus

Pembersihan karakter khusus ini bertujuan untuk menghapus beberapa karakter khusus yang ada di dalam kalimat/teks, seperti kalimat yang mengandung kata “\n”, url, serta emoji. Hal ini akan membuat dataset yang nantinya akan digunakan kualitasnya menjadi baik karena kata-kata khusus tersebut yang berasal dari media sosial Twitter cenderung tercantum di dalam kalimat/dataset tersebut sehingga akan membuat model *machine learning* kurang baik dalam melakukan klasifikasi. Berikut contoh input dan *output* untuk pembersihan karakter khusus yang dapat dilihat pada tabel 3.7.

|  |  |
| --- | --- |
| **Sebelum** | **Sesudah** |
| social media is a disease and this hailey bieber selena gomez tea is too hot | social media is a disease and this hailey bieber selena gomez tea is too hot |
| justin bieber amp selena gomez followed each other on Twitter teenage me is crying | justin bieber amp selena gomez followed each other on Twitter teenage me is crying |
| selena gomez drove by my house | selena gomez drove by my house |

Tabel 3. 5 Contoh data dalam proses pembersihan karakter

### Stop Word Removal

Tahapan pembersihan *stopword* berfungsi untuk menghilangkan kata umum (common words) yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna. Tujuan dari menghilangkan *stopword*s adalah mengurangi jumlah kata dalam sebuah dokumen yang akan berpengaruh terhadap kecepatan dan performa model. *Stopword* yang akan dijadikan sebagai acuan didapatkan dari libraries python nltk. Berikut beberapa contoh data sebelum dan sesudah dilakukannya tahapan pembersihan *stopword* yang dapat dilihat pada tabel 3.6

|  |  |
| --- | --- |
| **Sebelum** | **Sesudah** |
| social media is a disease and this hailey bieber selena gomez tea is too hot | social media disease hailey bieber selena gomez tea hot |
| justin bieber amp selena gomez followed each other on Twitter teenage me is crying | justin bieber amp selena gomez followed Twitter teenage crying |
| selena gomez drove by my house | selena gomez drove house |

Tabel 3. 6 Contoh data dalam *stopword* removal.

*Stemming* merupakan teknik menguraikan bentuk suatu kata menjadi bentuk asalnya/dasarnya. Hal ini dilakukan untuk mengubah kata berimbuhan menjadi bentuk dasarnya agar mempermudah model dalam melakukan klasifikasi, mengurangi variansi serta meningkatkan kualitas dataset menjadi lebih optimal. Sebagai contoh, untuk kata membenci akan ditransformasi menjadi benci, menyukai menjadi suka dan lebih banyak lagi. Berikut beberapa contoh *stemming* yang dapat dilihat pada tabel 3.7.

|  |  |
| --- | --- |
| **Sebelum** | **Sesudah** |
| social media is a disease and this hailey bieber selena gomez tea is too hot | social media diseas hailey bieber selena gomez tea hot |
| justin bieber amp selena gomez followed each other on Twitter teenage me is crying | justin bieber amp selena gomez follow Twitter teenag cri |
| selena gomez drove by my house | selena gomez drove hous |

Tabel 3. 7 Contoh data dalam proses *stemming*.

### Lemmatization

*Lemmatization* merupakan tahapan untuk mengubah kata menjadi bentuk dasarnya. Satu-satunya perbedaan adalah lemmatization dari stemming adalah memastikan bahwa kata dasar milik bahasa. *Lemmatization* berfungsi untuk mengurangi variasi kata dengan cara mengubah kata ke bentuk dasarnya. Untuk lebih detail, dapat dilihat pada tabel 3.8.

|  |  |
| --- | --- |
| **Sebelum** | **Sesudah** |
| social media diseas hailey bieber selena gomez tea hot | social media diseas hailey bieber selena gomez tea hot |
| justin bieber amp selena gomez follow Twitter teenag cri | justin bieber amp selena gomez follow Twitter teenag cri |
| selena gomez drove hous | selena gomez drive hous |

Tabel 3. 8 Contoh data dalam proses *lemmatization.*

## Labelling Data

Pelabelan data berfungsi untuk memberikan label yang terdiri dari positif, negatif maupun netral terhadap dataset. Hal ini perlu dilakukan karena model *deep learning* yang akan dibangun membutuhkan sebuah label untuk dapat berlatih dan mengklasifikasikan ujaran kebencian Twitter. Untuk melakukan pelabelan terhadap dataset, menggunakan Vader Lexicon. Vader (Valence Aware Dictionary and Sentimen Reasoner) Lexicon merupakan sebuah tools/libraries yang berfungsi untuk melakukan analisis sentimen berbasis aturan yang secara khusus disesuaikan dengan sentimen yang diungkapkan di media sosial (MIT, 2014). Tools ini bersifat *open-source* dan dibawah lisensi MIT. Vader dapat menangani beberapa kasus khusus dalam analisis sentimen, seperti memahami negasi, memahami arti emoticon/emoji, memahami arti sebuah akronim, penggunaan tanda baca konvensional menandakan peningkatan intensitas sentimen, dan masih banyak lagi.

Dengan menggunakan Vader, data teks akan diproses dan dicari nilai sentimennya. *Output* yang dihasilkan oleh Vader merupakan sebuah probabilitas dari 3 kemungkinan yaitu positif, negatif dan netral. Dari probabilitas tersebut, untuk melakukan labelling terhadap suatu teks di dalam dataset, maka diambil nilai compound dari ketiga probabilitas tersebut. Nilai compound adalah metric yang menghitung semua peringkat leksikon yang telah dinormalisasi antara -1 (paling negatif) sampai dengan 1(paling positif). Untuk penelitian ini sentimen positif nilai compound nya dalam rentang lebih dari sama dengan 0,05, sentimen netral diantara -0,05 sampai dengan 0,05 serta untuk sentimen negatif kurang dari sama dengan -0,05. Setelah menemukan nilai compound dan melakukan pelabelan data berdasarkan sentimen hasil nilai compound, maka dataset hasil pelabelan siap digunakan untuk diproses ditahap selanjutnya yaitu tahap preprocessing. Berikut contoh *output* dari Vader analisis sentimen.

|  |  |  |
| --- | --- | --- |
| **Teks** | **Vader *Output*** | **Label** |
| Bjorka is fun | {'neg': 0.0, 'neu': 0.694, 'pos': 0.306, 'compound': 0.296} | Positif |
| Bjorka please leak the secret document of suspicious plot funds | {'neg': 0.189, 'neu': 0.775, 'pos': 0.036, 'compound': -0.6728} | Negatif |
| Talk to Bjorka Drinking Coffee Together | {'neg': 0.0, 'neu': 0.954, 'pos': 0.046, 'compound': 0.0387} | Netral |

Tabel 3. 9 Labeling Data menggunakan Vader Analisis sentimen

Setelah mendapatkan label, maka label yang masih berupa teks perlu dilakukan pengubahan menjadi kategori numeris. Karena terdapat tiga label, maka akan dikonversi menjadi 0, 1 dan 2. Hal ini bertujuan untuk dapat mengimplementasikan data kedalam model yang membutuhkan label numeris.

## Tokenizing

Tahapan tokenizing bertujuan untuk mengubah kata menjadi sebuah token, dan dalam penelitian ini kata akan diubah menjadi sebuah token angka. Tokenizing dapat dilakukan dengan cara berikut:

1. Mengimport library keras Tokenizer
2. Menginisialisasi tokenizer
3. Melakukan fit tokenizer dengan dataset. Hal ini bertujuan untuk mendeteksi semua kata yang ada di dalam dataset dan mengubah kata tersebut menjadi sebuah token angka.
4. Mengaplikasikan tokenizer ke setiap data yang ada di dalam data latih dan data uji dan menyimpannya sebagai sebuah sequences.
5. Memberikan padding ke setiap sequences sesuai dengan sequences terpanjang dengan menambahkan token “0” sampai ke elemen terakhir padding.

Untuk mempermudah melihat hasilnya, dapat dilihat pada tabel 3.10

|  |  |  |
| --- | --- | --- |
| Kalimat | Sequences | Padded |
| social media diseas hailey bieber selena gomez tea hot | [183, 226, 1686, 7, 6, 2, 3, 219, 285] | [ 183, 226, 1686, 7, 6, 2, 3, 219, 285, 0, 0, 0, 0, 0, 0] |
| justin bieber amp selena gomez follow Twitter teenag cri | [9, 6, 18, 2, 3, 20, 227, 799, 334] | [ 9, 6, 18, 2, 3, 20, 227, 799, 334, 0, 0, 0, 0, 0, 0] |
| selena gomez drove hous | [2, 3, 800, 335] | [2, 3, 800, 335, 0, 0, 0, 0, 0, 0, 0, 0, 0 ,0, 0] |
| social media diseas hailey bieber selena gomez tea hot | [2, 3, 800, 335] | [160, 7, 6, 388, 256, 2, 3, 503, 0, 0, 0, 0, 0, 0, 0] |

Tabel 3. 10 Contoh data dalam proses *tokenizing.*

## Pemisahan Data

Pemisahan data bertujuan untuk membagi dataset menjadi dua bagian, yaitu bagian data latih dan data uji. Hal ini dilakukan untuk memisahkan data yang akan digunakan untuk melatih model dan data yang digunakan untuk uji data tunggal/testing model. Dataset akan dibagi menjadi data latih dan data uji dengan rasio 0.9:0.1 dimana data latih sebesar 4685 dan data uji sebesar 521. Rasio tersebut diambil dengan pertimbangan data yang didapatkan sedikit, sehingga dalam tahapan *training* diperlukan data yang banyak agar model lebih mengenali banyak variasi data. Setelah dipisahkan, label pada data latih dan data uji akan dipisahkan untuk siap digunakan pada tahap melatih model.

Setelah data dipisahkan, maka label harus dilakukan transformasi menjadi bentuk *one hot encoder*. Label akan ditransformasi dari data tunggal menjadi sebuah array yang berisi 3 komponen angka. Hal ini perlu dilakukan untuk menyesuaikan penerapan *deep learning* menggunakan keras libraries. Berikut contoh penerapan *one hot encoder* yang dapat dilihat pada tabel 3.11.

|  |  |
| --- | --- |
| Label | Label *One hot encoder* |
| 0 | [1,0,0] |
| 1 | [0,1,0] |
| 2 | [0,0,1] |

Tabel 3. 11 Label *one hot encoder*.

## Perancangan Model

Model yang akan dirancang dalam penelitian ini akan terdiri dari beberapa hidden *layer* yang akan disusun menjadi sebuah sequences yang terdiri dari beberapa *layer*s. Model yang akan dirancang akan mengimplementasikan model Sequentials dari libraries keras yang berarti mengelompokkan tumpukan lapisan linier. Dalam penelitian ini, akan lebih fokus pada susunan *layer* model *deep learning* dibandingkan parameter lain karena *layer* lebih berpengaruh dalam kasus analisis sentimen.

### Embedding Layer

Embedding *layer* merupakan input *layer* atau bisa dikatakan dengan *layer* pertama dalam model *deep learning* yang akan dirancang. *Layer* ini berfungsi untuk mengelompokan kata menjadi dua kutub yang berbeda. Parameter yang digunakan dalam *layer* ini adalah sebagai berikut:

* Input\_length: Panjang input sequence
* *Output*\_dim: dimensi *output*
* Input\_dim: ukuran vocabulary

Hasil dari embbeding *layer* dapat dilihat pada tabel 3.12.

|  |  |
| --- | --- |
| Sebelum | Sesudah |
| [131, 17, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] | [[[-0.03162958 0.04269071 0.01509007 0.00124536]  [-0.04351017 0.000172 -0.03358928 0.0472861 ]  [-0.01396496 0.02855982 0.03887559 -0.01323872]  [-0.00201822 0.03641895 0.0171499 -0.02244518]  [-0.02064002 0.02616386 -0.03546425 -0.03617959]  …………………………….  [-0.02064002 0.02616386 -0.03546425 -0.03617959]]] |

Tabel 3. 12 Contoh data hasil pada proses embedding *layer*

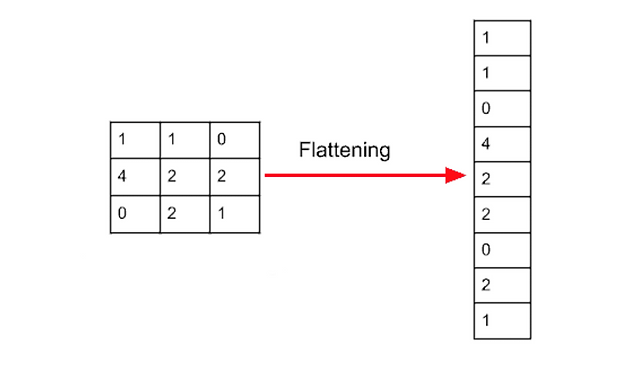
### Dense Layer

Dense *layer* akan memetakan input dari hidden *layer* sebelumnya menjadi sebuah *output* yang jumlahnya sesuai dengan jumlah unit di *layer* tersebut. Berikut contoh dari dense *layer* dari input hidden *layer* sebelumnya yang kemudian dipetakan sesuai jumlah *output* unit di *layer* tersebut.

|  |  |
| --- | --- |
| [[[-0.03162958 0.04269071 0.01509007 0.00124536]  [-0.04351017 0.000172 -0.03358928 0.0472861 ]  [-0.01396496 0.02855982 0.03887559 -0.01323872]  [-0.00201822 0.03641895 0.0171499 -0.02244518]  [-0.02064002 0.02616386 -0.03546425 -0.03617959]  …………………………….  [-0.02064002 0.02616386 -0.03546425 -0.03617959]]] | [[[-0.03966843]  [ 0.04057634]  [-0.01823187]  [ 0.05386413]  [-0.05570652]  [-0.05570652]  [-0.05570652]  ………………..  [-0.05570652]]] |

Tabel 3. 13 Contoh data hasil dense *layer* dengan *output* unit=1

Flatten *layer* berfungsi untuk melakukan transformasi neurons dari multi dimensi menjadi dimensi tunggal. Dalam penelitian ini, flatten *layer* digunakan untuk melakukan transformasi *output* dari *layer* Bidirectional/Embedding *layer* yang tergolong multi dimensi (3 dimensi) menjadi dimensi tunggal agar bisa diterima oleh hidden *layer* dense.



Gambar 3. 2 Cara kerja *flatten* *layer*

### LSTM Layer

Di dalam *layer* LSTM, terdapat beberapa Langkah perhitungan. Berikut ilustrasi pemodelan perhitungan LSTM pada suatu *layer* model *deep learning*:

Akan terdapat sebuah cell state dengan nilai , maka akan melakukan perhitungan dengan Langkah seperti di bawah ini.

* 1. Forget gate

Forget gate berfungsi untuk memilih informasi mana yang akan dilupakan/dihilangkan dari cell state.

Diagram

Description automatically generated

Gambar 3. 3 Forget gate lstm (Colah, 2015)

* Menghitung nilai bobot forget gate dikalian dengan gabungan *output* dari waktu ke t-1 dan nilai input pada waktu ke t
* Menambahkan nilai hasil kalkulasi di atas dengan nilai bias forget gate
* Menghitung nilai dari forget gate

|  |  |
| --- | --- |
|  | Nilai forget gate |
|  | Nilai bobot untuk forget gate |
|  | Nilai *output* dari waktu ke t−1 |
|  | Nilai input pada waktu ke t |
|  | Bias pada forget gate |
|  | Fungsi sigmoid |

Tabel 3. 14 Keterangan perhitungan pada formula forget gate

* 1. Input Gate

Input gate berfungsi untuk menyediakan informasi baru yang akan diteruskan ke dalam cell state.

Diagram

Description automatically generated

Gambar 3. 4 Input gate lstm (Colah, 2015)

* Menghitung nilai dari input gate
* Menghitung nilai dari

|  |  |
| --- | --- |
|  | Nilai input gate |
|  | Nilai bobot untuk input gate |
|  | Nilai *output* dari waktu ke t−1 |
|  | Nilai input pada waktu ke t |
|  | Bias pada input gate |
|  | Nilai kandidat cell state |
|  | Fungsi sigmoid |
|  | Fungsi tanh |

Tabel 3. 15 Keterangan perhitungan pada formula input gate

* 1. Cell State

Menampung hasil operasi forget gate dan input gate yang nantinya akan digunakan untuk memperbarui nilai dari cell state.

Diagram

Description automatically generated with medium confidence

Gambar 3. 5 Cell state LSTM

|  |  |
| --- | --- |
|  | Nilai forget gate |
|  | Nilai input gate |
|  | Nilai memory cell state |
|  | Nilai memory pada cell state sebelumnya |
|  | Nilai kandidat cell state |

Tabel 3. 16 Keterangan perhitungan pada formula cell state

* 1. *Output* gate

Menentukan informasi apa yang akan dijadikan sebuah *output* dari *layer* tersebut (melakukan filter terhadap cell state)

A picture containing text, clock

Description automatically generated

Gambar 3. 6 *Output* gate LSTM

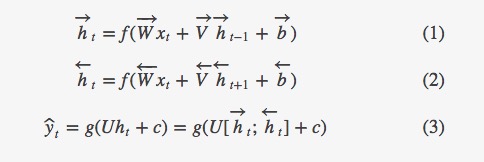
* Menghitung nilai input gate
* Menghitung nilai *output* gate

|  |  |
| --- | --- |
|  | Nilai input gate |
|  | Nilai bobot untuk *output* gate |
|  | Nilai *output* dari waktu ke t−1 |
|  | Nilai input pada waktu ke t |
|  | Bias pada *output* gate |
|  | Nilai *output* gate |
|  | Fungsi sigmoid |
|  | Fungsi tanh |

Tabel 3. 17 Keterangan perhitungan pada formula *output* gate

### Bidirectional LSTM Layer

Bidirectional LSTM berarti terdapat dua LSTM di dalam layer ini, satu bekerja secara ke depan, dan satu bekerja ke belakang. Persamaannya dapat dilihat pada gambar 3.7.



Gambar 3. 7 Persamaan Bidirectional LSTM (Li)

Persamaan 1 dan 2 mewakili arti matematis dari lapisan tersembunyi RNN dua arah. Satu-satunya perbedaan antara kedua hubungan tersebut adalah arah putarannya. Persamaan 3 menunjukkan bahwa dengan meringkas representasi kata masa lalu dan masa depan, hubungan kategori digunakan untuk memprediksi prediksi kata berikutnya.

## Evaluasi Model

Evaluasi model dilaukan dengan melakukan prediksi terhadap data uji, dan kemudian menyocokan kesamaan antara label asli dengan label hasil prediksi. Evaluasi akan ditinjau dari segi akurasi serta metric f1-score. Berikut rumus untuk f1-score yang dapat dilihat pada persamaan dibawah ini:

Keterangan :

TP = True Positif (data prediksi benar dan data actual benar)

TN = True Negatif (data yang diprediksi benar dan data actual salah)

FP = False Positif (data prediksi salah dan data actual benar)

FN = False Negatif (data prediksi salah dan data actual salah)

## Skenario Pengujian

Tahapan ini bertujuan untuk menjawab rumusan masalah yang ada pada Bab I yaitu analisis sentimen terhadap ujaran kebencian Twitter. Pada Rangkaian skenario pengujian, data akan diimplementasikan secara sama, yaitu dilakukan beberapa tahapan diantaranya adalah seleksi data, preprocessing, pelabelan data *tokenizing* data, pemisahan data.

Tahapan diatas dilakukan secara berurutan dan bertahap. Setelah data siap untuk digunakan, maka akan dilakukan beberapa skenario yang berbeda dimuali dari *modelling* sampai dengan evaluasi. Skenario untuk tahapan dari *modelling* sampai dengan evaluasi akan dibagi menjadi beberapa fokus besar diantaranya adalah skenario pengujian model sederhana dan skenario pengujian model kompleks. Yang membedakan kedua skenario ialah arsitektur model yang dibangun, yaitu model dengan arsitektur sederhana dan model dengan arsitektur kompleks.

Kedua skenario di atas akan dilatih sebanyak 20 *epochs*/iterasi. Kemudian *validation\_split* sebesar 0.1 dimana nantinya data *training* akan dibagi lagi menjadi dua, yaitu dimana 0.1 data latih akan digunakan untuk validasi data pada saat tahap *training*. Model yang dibangun juga menggunakan sebuah *callback* fungsi yang memungkinkan menghentikan iterasi pada saat proses *training* sesuai dengan *callback* yang dipasang. Untuk *callback* yang digunakan dalam model ini ialah *early stopping*. *Early stopping* memungkinkan iterasi dihentikan secara langsung sesuai dengan nilai *patience* yang dipasang ketika model yang dilatih tidak menunjukan sebuah peningkatan. *Patience* merupakan sebuah parameter untuk menentukan kapan iterasi harus dihentikan, sebagai contoh jika *patience* dipasang dengan nilai satu, maka akan menghentikan iterasi pada iterasi n+1.

Setelah dilakukan tahapan modelling dan *training*, maka akan dilanjutkan dengan tahapan evaluasi yaitu dengan cara melakukan *testing* ataupun uji data tinggal terhadap model yang ada. Model dievaluasi menggunakan nilai akurasi, loss serta f1 score. Setelah itu, akan divisualisasikan menggunakan grafik yang memuat grafik akurasi, loss dan f1 score dari model serta confusion matrix dari hasil evaluasi model.

### *Layer* Model

*Layer* model yang akan disusun terdiri dari beberapa kombinasi *layer* diantaranya embedding *layer*, lstm *layer*, bidirectional lstm *layer*, dense *layer* serta flatten *layer*.

### Fungsi aktifasi

Fungsi aktifasi yang akan di dalam *layer* model terdiri dari beberapa susunan dan ketentuan. Untuk fungsi aktifasi tanh akan digunakan di dalam *layer* lstm dan bidirectional lstm. Untuk fungsi aktifasi relu akan digunakan di dalam *layer* dense. Untuk fungsi aktifasi softmax, akan digunakan pada *output* *layer* yaitu pada *layer* dense terakhir dalam model

### Optimizer

Optimizer yang akan digunakan dalam beberapa scenario pengujian diantaranya adalah rmsprop serta adam.

### Fungsi loss

Fungsi loss yang akan digunakan dalam model ini terdiri dari dua, yaitu categorical crossentropy karena baik dalam mengatasi klasifikasi berkategori serta mean squared error karena nilai selisih error dikuadratkan sehingga cocok untuk memberi hukuman/terhadap suatu nilai agar pembelajaran berjalan lebih baik.

### Skenario Pengujian Model Sederhana

Arsitektur model yang akan dibangun pada skenario ini adalah sebagai berikut:

|  |  |  |  |
| --- | --- | --- | --- |
| No | *Layer*/Aktifasi | Optimizer | Fungsi loss |
| 1 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense (3/softmax) | Adam | Categorical |
| 2 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense (3/softmax) | Adam | MSE |
| 3 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense(3/softmax) | RMSProp | Categorical |
| 4 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense(3/softmax) | RMSProp | MSE |
| 5 | * Embedding (16) * BiLSTM(64/tanh) * Flatten * Dense(3/softmax) | Adam | Categorical |
| 6 | * Embedding (16) * BiLSTM(64/tanh) * Flatten * Dense(3/softmax) | Adam | MSE |
| 7 | * Embedding (16) * BiLSTM(64/tanh) * Flatten * Dense(3/softmax) | RMSProp | Categorical |
| 8 | * Embedding (16) * BiLSTM(64/tanh) * Flatten * Dense(3/softmax) | RMSProp | MSE |

Tabel 3. 18 Arsitektur model sederhana.

### Skenario Pengujian Model Kompleks

Arsitektur model yang akan dibangun pada skenario ini adalah sebagai berikut:

|  |  |  |  |
| --- | --- | --- | --- |
| No | *Layer*/Aktifasi | Optimizer | Fungsi loss |
| 1 | * Embedding (128) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | Adam | Categorical |
| 2 | * Embedding (128) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | Adam | MSE |
| 3 | * Embedding (128) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | RMSProp | Categorical |
| 4 | * Embedding (128) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | RMSProp | MSE |
| 5 | * Embedding(128) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | Adam | Categorical |
| 6 | * Embedding(128) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | Adam | MSE |
| 7 | * Embedding(128) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | RMSProp | Categorical |
| 8 | * Embedding(128) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | RMSProp | MSE |

Tabel 3. 19 Arsitektur model kompleks

# BAB IV

**IMPLEMENTASI DAN HASIL PENELITIAN**



## Implementasi Perangkat Lunak

Pada bagian ini akan diuraikan implementasi dari metode yang ada pada Bab III. Implementasi terdiri dari beberapa tahapan yaitu pengumpulan data, *preprocessing* data, *labelling* data, *tokenizing* data, pemisahan data, *modelling* serta evaluasi.

### Pengumpulan Data

Tahapan ini merupakan tahapan untuk mendapatkan data *tweet* yang terdapat di platform Twitter dengan cara scrapping laman Twitter menggunakan library Python tweepy dan membutuhkan akun serta akses developer Twitter. Data didapatkan dengan membuat sebuah script Python bernama “data.ipynb”. Data yang dicari diantaranya adalah data *tweet* dibuat, data username, serta teks dari *tweet* tersebut. Setelah didapatkan, maka data *tweet* tersebut akan disimpan dalam sebuah dataframe dan kemudian diexport menjadi sebuah file excel. Saat implementasi, didapatkan data sebanyak 11.646 sample data dari Twitter dengan bahasa inggris dan tidak disertakan link di dalam *tweet*nya. Listing program dalam mengumpulkan data dapat dilihat pada gambar 4.1 sampai 4.6.

Text

Description automatically generated

Gambar 4. 1 Import library Python dan inisialisasi token untuk mendapat akses ke akun developer Twitter

Text

Description automatically generated

Gambar 4. 2 Inisialisasi API developer Twitter account

Text

Description automatically generated

Gambar 4. 3 Membuat fungsi untuk melakukan iterasisearching *tweet*

Text

Description automatically generated

Gambar 4. 4 Membuat fungsi untuk membuat dataframe data hasil scraping

A screenshot of a computer

Description automatically generated with medium confidence

Gambar 4. 5 Searching *tweet*s menggunakan fungsi sebelumnya dan inisialisasi query untuk mencari data Twitter



Gambar 4. 6 Implementasi membuat dataframe dan menyimpan ke file excel.

Graphical user interface, text

Description automatically generated

Gambar 4. 7 Data hasil scraping Twitter

### Preprocessing Data

Pada tahap ini, akan dijelaskan secara rinci bagaimana tahapan preprocessing data yang ada agar bisa digunakan untuk membangun model *deep learning* analisis sentimen ujaran kebencian Twitter. Preprocessing yang akan dilakukan diantaranya adalah seleksi data, *lowercasing*, pembersihan karakter selain *alphanumeric*, pembersihan karakter khusus/tidak biasa, membersihkan *stopword*, *stemming*, dan lemmatization.

#### Seleksi Data

Karena data yang didapatkan adalah data teks dari Twitter, maka aka nada banyak peluang bahwa data tersebut merupakan data re*tweet*, dimana user dapat memposting ulang *tweet* orang lain, dan hal ini akan menimbulkan banyak duplikasi data. Oleh karena itu, dilakukan seleksi data dengan mengambil data yang tidak mengandung kata “RT” di dalam teks tersebut. Untuk implementasi hal tersebut, dapat dilihat pada tabel 4.8.

Text

Description automatically generated

Gambar 4. 8 Implementasi seleksi data

Setelah dilakukan implementasi, didapatkan data hasil seleksi sebanyak 5206 kalimat.

#### *Lowercasing*

Pada tahapan ini akan diimplementasikan mengubah karakter huruf yang ada di dalam dataset menjadi bentuk huruf kecil. Untuk melakukan implementasi ini, memanggil fungsi “lower()” dari modul String pada python yang dapat dilihat implementasinya pada gambar 4.9 dan untuk hasilnya dapat dilihat pada gambar 4.10.

A picture containing text

Description automatically generated

Gambar 4. 9 Implementasi *lowercasing*.

Text

Description automatically generated

Gambar 4. 10 Hasil *lowercasing*.

#### Pembersihan Karakter *Non-alphanumeric*

Pembersihan *Non-alphanumeric* bertujuan untuk membersihkan karakter selain huruf dan angka, seperti tanda baca dan karakter khusus lainnya. Tahapan ini dilakukan dengan menggunakan libraries re (regular expression) dengan fungsi sub yang berfungsi untuk mengganti kata yang sesuai dengan pola yang ada, yaitu pola karakter selain *alphanumeric* dan kemudian diganti dengan whitespace. Setelah itu, digabungkan menjadi satu kalimat lagi dengan memfilter apakah kata yang digabungkan merupakan digit atau bukan yang dapat dilihat implementasinya pada gambar 4.11 dan untuk hasilnya dapat dilihat pada gambar 4.12.

Text

Description automatically generated

Gambar 4. 11 Implementasi Remove *Non-alphanumeric*

Text

Description automatically generated

Gambar 4. 12 Hasil Implementasi Remove *Non-alphanumeric*

#### Pembersihan Karakter Khusus

Pembersihan karakter khusus ini bertujuan untuk menghapus beberapa karakter khusus yang ada di dalam kalimat/teks, seperti kalimat yang mengandung kata “\n”, url, serta emoji. Untuk implementasi, menggunakan libraries re (regular expression) dengan fungsi sub untuk mengganti kata denga pola yang sesuai dengan pola yang dicari digantikan dengan string kosongdan dapat dilihat implementasinya pada gambar 4.13 dan untuk hasilnya dapat dilihat pada gambar 4.14.

Text

Description automatically generated

Gambar 4. 13 Implementasi Pembersihan karakter khusus

Text

Description automatically generated

Gambar 4. 14 Hasil pembersihan karakter khusus

#### *Stopword* Removal

Tahapan pembersihan *stopword* berfungsi untuk menghilangkan kata umum (common words) yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna. Tahapan ini diawali dengan mengimport modul *stopword*s dari libraries nltk.corpus. Setelah itu, mengambil set *stopword* inggris dari modul *stopword*s. Setelah itu, melakukan iterasi terhadap *stopword* dan melakukan seleksi tiap kata yang ada di dalam kalimat. Setelah iterasi selesai dilakukan, maka akan mengembalikan teks tanpa *stopword*s. Untuk implementasi, dapat dilihat pada gambar 4.15 sampai dengan gambar 4.17.



Gambar 4. 15 Import modul *stopword*s dari libraries nltk.corpusText

Description automatically generated

Gambar 4. 16 Implementasi penghapusan *stopword*s

Text

Description automatically generated

Gambar 4. 17 Hasil implementasi penghapusan *stopword*s

*Stemming* merupakan teknik menguraikan bentuk suatu kata menjadi bentuk asalnya/dasarnya. Untuk melakukan tahapan ini, diperlukan import modul PorterStemmer dari libraries nltk.stem.porter. Setelah itu, menginisialisasi PorterStemmer. Setelah diinisialisasi, maka memanggil fungsi stem dari PorterStemmer untuk setiap kata yang ada di dalam kalimat/data. Untuk detail implementasi dapat dilihat pada gambar 4.18 sampai dengan 4.20.

Text

Description automatically generated

Gambar 4. 18 Import dan menginisialisasi PorterStemmer

Text

Description automatically generated

Gambar 4. 19 Implementasi Fungsi stem dari PorterStemmer

Text

Description automatically generated

Gambar 4. 20 Hasil dari *stemming* data.

#### Lemmatization

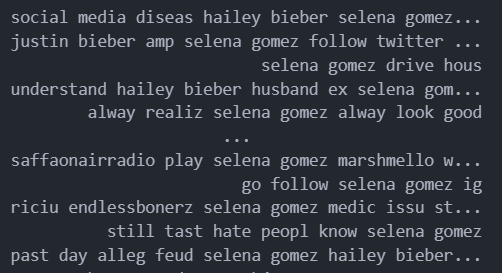
Seperti *stemming*, lemmatization juga mengubah kata menjadi bentuk dasarnya. Satu-satunya perbedaan adalah lemmatization memastikan bahwa kata dasar milik bahasa. Untuk melakukan tahapan ini, diawali dengan melakukan import modul WordLemmatizer dari libraries nltk.stem dan kemudian menginisialisasi WordNetLemmatizer. Setelah diinisialisasi, maka memanggil fungsi lemmatize dari WordNetLemmatizer untuk setiap kata yang ada di dalam kalimat/data. Untuk detail implementasi dapat dilihat pada gambar 4.21 sampai dengan 4.23.



Gambar 4. 21 Import dan inisialisasi WordNetLemmatizer

Text

Description automatically generated

Gambar 4. 22 Implementasi lemmatization

Gambar 4. 23 Hasil implementasi lemmatization

### Labelling Data

Proses labelling data merupakan tahapan memberikan label terhadap data yang dimiliki. Hal ini diperlukan karena untuk melakukan klasifikasi ujaran kebencian, diperlukan sebuah label untuk dapat melakukan *training* model *deep learning* nantinya. Label yang akan dibentuk diantaranya adalah positif, negatif dan netral karena sentimen biasanya terdiri dari tiga label tersebut. Untuk melakukan labelling, digunakan libraries python untuk labelling analisis sentimen yaitu Vader. Cara kerja Vader yaitu setiap data teks yang ada di dalam dataset akan diimplementasikan ke dalam fungsi polarity\_scores dari SentimenIntensityAnalyzer milik Vader dan kemudian akan mengembalikan berupa sebuah data bertipe data dictionary Python yang di dalamnya terdiri skor dari hasil polarity\_score terhadap masing-masing teks. Setelah didapatkan nilai tersebut, kemudian dikelompokkan/diberikan label pada niali “compound” yang ada di dalam data dictionary. Jika compound >=0.05 maka label yang diberikan adalah positif, jika diantara -0.05 dan 0.05 maka label yang diberikan netral, selain itu diberikan label negatif. Untuk implementasi dan hasilnya dapat dilihat pada gambar 4.24 dan gambar 4.25.



Gambar 4. 24 Implementasi Vader analisis sentimen

Text

Description automatically generated

Gambar 4. 25 Hasil labelling menggunakan Vader

Setelah mendapatkan label data, kemudian label dipindahkan ke data yang sudah dilakukan preprocessing karena data yang diberikan label merupakan data yang murni, hal ini dilakukan agar ketika dilakukan penerjemahan dan labelling, tidak ada makna yang tereduksi. Setelah dijadikan satu antara data hasil preprocessing dan juga labelnya, maka melakukan transformasi label menjadi kategorikal angka dengan label 0,1 dan 2. Hal itu dilakukan untuk mempermudah dalam melakukan pemodelan data pada saat melakukan pemnbuatan model *deep learning*.

### Tokenizing Data

Tahapan ini akan mengubah data teks menjadi sebuah token yang merepresentasikan sebuah kata. Dalam tahapan ini, ada beberapa komponen penting yang harus diperhatikan yang dirangkum dalam tabel 4.1 sebagai berikut:

|  |  |
| --- | --- |
| Elemen | Keterangan |
| Vocab Size | Ukuran kata yang akan diubah menjadi index kata/kamus kata yang mewakili semua kata yang ada di dalam dataset |
| max\_len | Ukuran Panjang maksimal dari sebuah sequences/array yang menampung kumpulin token/index kata yang mewakili kata asli |
| Truncating type | Cara untuk memotong token dalam sequences/array yang ukurannya melebihi max\_len. |
| Padding type | Cara untuk memenuhi/menggenapi sequences yang ukurannya kurang dari max\_len agar mempunyai ukuran yang sama. |
| Oov token | Oov adalah singkatan dari Out of Vocabulary Token, yang berarti komponen yang akan menggantikan kata yang muncul di dalam dataset yang tidak terepresentasikan oleh word index/kamus kata. |

Tabel 4. 1 Komponen penting dalam tokenizing

Pada tahapan ini, vocab size diset dengan nilai 6995, atau bisa dikatakan membuat kamus index dengan ukuran 6995 yang setara dengan Jumlah kata unik dalam dataset. Hal ini bearti semua kata di dalam dataset sudah diwakilkan oleh sebuah token yang disimpan dalam word index. Setelah mendapatkan banyaknya jumlah variasi kata, maka selanjutnya diperlukan mengetahui variasi panjang kalimat di dalam dataset tersebut. Untuk itu, dapat dilakukan dengan melakukan iterasi terhadap kalimat yang ada di dalam dataset dan kemudian menghitung panjang setiap kalimat. Untuk implementasi, dapat dilihat pada gambar 4.26 dan gambar 4.27.

Text

Description automatically generated

Gambar 4. 26 Implementasi menghitung panjang setiap kalimat

A picture containing chart

Description automatically generated

Gambar 4. 27 Hasil implementasi dan visualisasinya.

Pada gambar 4.27, dapat dilihat bahwa lebih dari 75% data memiliki panjang kalimat 15 kata, dan oleh karena itu, akan lebih baik jika data panjang sequences/token diset menjadi 15 token. Setelah itu, truncating type yang digunakan ialah “post” yang berarti jika terdapat sebuah sequences/array yang memiliki Panjang lebih dari 15, maka yang akan dipotong/tidak diambil tokennya adalah bagian setelah token ke-15. Padding type pada tahapan ini menggunakan padding “post” yang berarti jika terdapat sebuah sequences/array yang panjangnya kurang dari 15, maka akan otomatis diisi oleh “0” setelah token terakhir yang ada di dalam sequences sampai dengan 100. Dan yang terakhir, oov token dari data ini menggunakan “<OOV>” yang berarti jika terdapat suatu kata yang tidak teridentifikasi di dalam index kata, maka akan digantikan oleh oov token yang telah dipasang. Untuk mengetahui lebih detail, dapat melihat lebih detail pada gambar 4.28 sampai dengan 4.30.

A screenshot of a computer

Description automatically generated with medium confidence

Gambar 4. 28 Import libraries untuk melakukan tokenizing.

Text

Description automatically generated

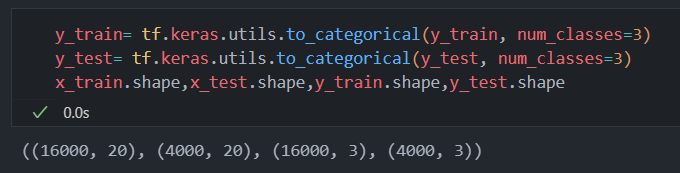
Gambar 4. 29 Mempersiapkan elemen komponen penting tokenizing.



Gambar 4. 30 Contoh hasil implementasi tokenizing.

### Pemisahan Data

Pada tahapan ini akan dilakukan pemisahan dataset menjadi dua bagian, yaitu data untuk dilatih dan data untuk menguji model yang telah dilatih. Dua puluh ribu data teks *tweet* akan dibagi dengan rasio 90% untuk *training* (4685sampel data) dan 10% untuk testing (521sampel data). Pada tahapan ini menggunakan libraries train\_test\_split dari libraries sklearn yang nantinya akan otomatis membagi data dan label sesuai dengan rasio yang diinginkan. Setelah itu karena model membutuhkan input label berupa *one hot encode* label, maka label harus ditransformasi dari bentuk numeris menjadi array/list berisi tiga angka numeris yang merespresentasikan tiap label, dimana angka yang bernilai 1 di indeks label merupakan label tersebut. Untuk melakukan itu, menggunakan libraries dari keras yaitu fungsi to\_categorical, dengan class yang diinginkan yaitu 3. Untuk detail implementasi, dapat dilihat pada gambar 4.31.



Gambar 4. 31 Implementasi one hot encoding

Setelah proses tersebut, maka data siap digunakan dalam tahap *training*.

### Modelling

Dalam penelitian ini, ada beberapa model yang akan dirancang sesuai dengan skenario yang ada. Untuk membuat model *deep learning*, perlu melakukan import libraries tensorflow yaitu diantaranya. Sequentials, Embedding, Bidirectional, LSTM, Flatten, Dense, Adam, dan RMSProp dari modul keras. Setelah melakukan import, maka membuat sebuah fungsi untuk membangun model sesuai dengan skenario yang diinginkan. Untuk implementasi model, dapat dilihat pada gambar 4.32.

Text

Description automatically generated

Gambar 4. 32 Implementasi penyusunan model sederhana



Gambar 4. 33 Implementasi penyusunan model kompleks

Setelah model dibuat, maka dapat melakukan tahapan *training* model dimana model akan dilatih selama 20 *epochs*, dan rasio validasi data *training* sebesar 0.2. Dalam *training*, diinisialisasi sebuah *callback*s yang berfungsi untuk menghentikan *training* ketika matrix yang menjadi acuan, baik akurasi maupun loss tidak menunjukan peningkatan signifikan. *Callback* yang digunakan yaitu EarlyStopping dengan patience sebesar 5, yang berarti *training* akan dihentikan ketika lima epoch setelah matrix acuan terbaik, dan akan mengembalikan bobot model terbaik jika nilai restore\_best\_weight bernilai True. Listing program yang dapat dilihat pada gambar 4.34 dan 4.35.

Text

Description automatically generated

Gambar 4. 34 Implementasi *callback*



Gambar 4. 35 Implementasi *training* model

### Evaluasi

Evaluasi model dilihat dari hasil akurasi akurasi, loss dan nilai f1\_m dari data *training* dan validasi serta hasil evaluasi model dengan memanggil fungsi evaluate dari model keras. Selain itu, evaluasi dapat ditinjau jujga melalui confusion matrix yang dapat diperoleh dengan menggunakan fungsi confusion\_matrix dari libraries sklearn, setelah itu untuk menampilkan visualisasinya dapat menggunakan ConfusionMatrixDisplay. Untuk mendapatkan confusion matrix, maka model harus melakukan prediksi data terlebih dahulu, setelah itu didapatkanlah label hasil prediksi. Hasil prediksi dan juga data testing kemudian dicocokan menggunakan confusion matrix. Akan tetapi, karena keduanya masih berupa array/list untuk setiap datanya, maka perlu dilakukan transformasi menggunakan fungsi argmax dari libraries numpy untuk mengambil kelas yang memiliki nilai probabilitas prediksi paling besar. Setelah mendapatkan confusion matrixnya, maka dapat melakukan visualisasi menggunakan ConfusionMatrix Display. Untuk implementasi program, dapat dilihat pada gambar 4.36 sampai dengan gambar 4.38.

Graphical user interface, text, application

Description automatically generated

Gambar 4. 36 Implementasi fungsi evaluate oleh model keras.

Text

Description automatically generated

Gambar 4. 37 Implementasi Confusion Matrix.

Chart, treemap chart

Description automatically generated

Gambar 4. 38 *Output* confusion matrix.

Setelah itu, untuk dapat menganalisa lebih detail terkait hasil *training* model, maka dibuatlah sebuah fungsi untuk melakukan visualisasi hasil *training* yang terdiri dari elemen akurasi, loss dan f1 score terhadap iterasi *epochs*. Untuk implementasi, dapat dilihat pada gambar 4.39 dan gambar 4.40.

Text

Description automatically generated

Gambar 4. 39 Implementasi visualisasi hasil *training*.

A picture containing text

Description automatically generated

Gambar 4. 40 Implementasi visualisasi hasil *training* model.

## Analisa dan Pembahasan

Pada bagian ini, akan membahas tentang hasil *training* dan juga Analisa hasil *training* yang terjadi di setiap skenario. Skenario dibagi menjadi beberapa focus, yaitu model dengan *layer* LSTM dan model dengan *layer* BiLSTM.

### Pengujian Model Sederhana

Dalam tahap pengujian ini, akan dilakukan secara bertahap yaitu dimulai dengan membandingkan arsitektur deep learning, activation function, optimizer dan juga loss function. Model akan dibangun dengan data yang sama, epoch sebanyak 20, *validation\_split* sebesar 0.1, dan menggunakan EarlyStopping *callback*.

#### Pengujian Terhadap Arsitektur Model

Pada bagian ini, akan membandingkan antara arsitektur model dengan layer LSTM dengan model dengan layer BiLSTM. Model yang akan dibagun sederhana, dengan komposisi input layer berupa Embedding layer, hidden layer berupa satu layer LSTM ataupun BiLSTM, dan output layer berupa Dense. Optimizer yang akan digunakan ialah Adam, activation function untuk hidden layer tanh dan untuk output layer adalah softmax, serta loss function yang digunakan ialah categorical. Setelah dilakukan percobaan, didapatkan data hasil penelitian yang dapat dilihat pada tabel 4.2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | *Layer*/Aktifasi | Optimizer | Fungsi loss | Early stopping | Loss | Accuracy | F1 Score |
| 1 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense (3/softmax) | Adam | Categorical | 9 | 0.5386 | 0.8157 | 0.8007 |
| 2 | * Embedding (16) * BiLSTM(64/tanh) * Flatten * Dense(3/softmax) | Adam | Categorical | 8 | 0.4712 | 0.8522 | 0.8427 |

Tabel 4. 2 Hasil training perbandingan arsitektur layer model.

Setelah dilakukan training terhadap dua model dengan parameter yang sama, dapat dilihat bahwa model dengan arsitektur layer BiLSTM lebih unggul di semua aspek, baik akurasi, nilai loss dan juga nilai f1 score dibandingkan dengan model LSTM. Kedua model ditraining dengan epoch yang sama, akan tetapi model BiLSTM berhenti terlebih dahulu di epochs ke delapan dan menghasilkan akurasi yang lebih tinggi juga dibandingkan dengan model LSTM. Hal ini disebabkan karena BiLSTM tidak hanya mengatasi masalah ketergantungan jangka panjang seperti LSTM pada umumnya, akan tetapi mengatasi masalah konteks terhadap suatu teks (Zhou, Long, & Ou). Sebagai contoh, sebuah kata yang sama dengan konteks yang berbeda yaitu kata “I like apple fruit” dan “I like apple product”. BiLSTM memungkinkan untuk mengenali konteks “apple” di dalam kalimat tersebut karena model ini bekerja secara dua arah, yaitu *forward* dan *backward* atau dengan kata lain berjalan secara dua arah. Dalam fenomena Selena Gomez ini, yang telah menyebar luas di Twitter, tentunya muncul berbagai macam konteks tweet yang dikirimkan oleh berbagai macam pengguna dari berbagai macam negara dengan latar belakang yang berbeda-beda pula, dan tentunya layer BiLSTM sangatlah cocok dan terbukti lebih baik untuk mengklasifikasikan ujaran kebencian.

Chart, line chart

Description automatically generated

Gambar 4. 41 Akurasi dan validasi akurasi dari training model LSTM dan BiLSTM

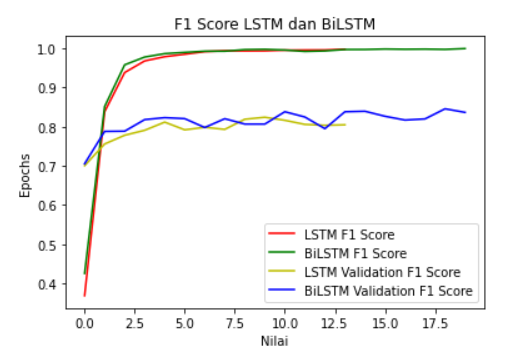
Jika dilihat dari grafik akurasi dan validasi akurasi di atas, kedua model masih mengalami overfitting dimana model kurang bisa mengenali data dengan variasi baru dibandingkan dengan data yang telah digunakan saat training model. Walaupun demikian, model BiLSTM terlihat lebih baik dari segi akurasi, dan validasi akurasi, serta secara proses training, model ini lebih baik karena tidak mengalami EarlyStoping sampai dengan epochs terakhir yang menandakan terjadi peningkatan akurasi terhadap validasi akurasi setiap lima epochs sekali.

Chart, line chart

Description automatically generated

Gambar 4. 42 Loss dan validasi loss dari training model LSTM dan BiLSTM

Dari grafik di atas, kedua model mendapatkan nilai loss yang hampir sama dan relatif cukup rendah, sedangkan untuk nilai validasi lossnya model BiLSTM lebih rendah dibandingkan dengan model LSTM.



Gambar 4. 43 F1 score dan validasi f1 score dari training model LSTM dan BiLSTM

Dari grafik di atas, terlihat bahwa nilai f1 score dan validasi f1 score hampir sama dengan nilai akurasi, dan model termasuk dalam kategori overfitting karena nilai antara fi score dengan validasi f1 score terpaut jauh.

Dari percobaan arsitektur model ini antara model dengan layer LSTM dengan BiLSTM dapat dilihat bahwa BiLSTM unggul di akurasi, loss dan juga f1 score dibandingkan dengan model LSTM, dan hal ini dikarenakan BiLSTM bekerja secara dua arah sehingga dapat lebih baik mengenali konteks sebuah kalimat.

#### Pengujian Terhadap Activation Function

Pada bagian ini akan membandingkan antara fungsi aktifasi pada setiap layer yang ada pada model yang telah dirancang. Fokus fungsi aktivasi akan dibagi menjadi dua bagian, yaitu *hidden* layer dan *output* layer. Fungsi aktifasi yang akan digunakan pada *hidden* layer adalah relu, tanh dan sigmoid, untuk *output* layer akan menggunakan sigmoid dan softmax. Pemilihan kombinasi di atas berdasarkan model yang pada umumnya digunakan sesuai dengan kondisi yang ada. Model diuji menggunakan parameter yang sama, yaitu arsitektur BiLSTM, untuk fungsi loss menggunakan categorical dan untuk optimizer menggunakan adam serta dilatih selama 20 epochs menggunakan *callback* EarlyStoping.

Setelah dilakukan percobaan, didapatkan hasil untuk pengujian fungsi aktivasi yang dapat dilihat pada tabel 4.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Kombinasi Aktifasi | Optimizer | Fungsi loss | Early stopping | Loss | Accuracy | F1 Score |
| 1 | * Relu (hidden) * Sigmoid (output) | Adam | Categorical | 8 | 0.5686 | 0.8330 | 0.8228 |
| 2 | * Relu (hidden) * Softmax (output) | Adam | Categorical | 11 | 0.6973 | 0.8522 | 0.8591 |
| 3 | * Sigmoid (hidden) * Sigmooid (output) | Adam | Categorical | 16 | 0.7553 | 0.8311 | 0.7964 |
| 4 | * Sigmoid (hidden) * Softmax (output) | Adam | Categorical | 12 | 0.5828 | 0.8369 | 0.8394 |
| 5 | * Tanh (hidden) * Sigmoid (output) | Adam | Categorical | 8 | 0.5911 | 0.8349 | 0.8233 |
| 6 | * Tanh (hidden) * Softmax (output) | Adam | Categorical | 9 | 0.7797 | 0.8138 | 0.8134 |

Tabel 4. 3 Hasil training perbandingan fungsi aktifasi menggunakan layer BiLSTM

Dari tabel di atas, dapat terlihat bahwa aktifasi yang dapat memberikan akurasi yang optimal adalah relu untuk *hidden* layer, dan softmax untuk *output* layer. Untuk hidden layer, relu mampu memberikan performa yang optimal ketika dikombinasikan dengan softmax sebagai output layer. Hal ini dikarenakan relu dapat mempercepat konvergensi *stochastic gradient descent* dibandingkan dengan sigmoid dan tanh (Krizhevsky, Sutskever, & Hinton) dimana gradien yang dimiliki maksimum satu dan nilai dari relu berkisar dari nol sampai tak hingga (Marimuthu, 2022).

Salah satu penyebab model dnegan fungsi aktivasi softmax lebih baik dibandingkan dengan sigmoid ketika digunakan sebagai output layer adalah softmax lebih baik dalam melakukan klasifikasi multi kelas/ klasifikasi dengan kelas lebih dari dua (Marimuthu, 2022) dibandingkan dengan sigmoid yang lebih baik menangani masalah klasifikasi biner (Marimuthu, 2022). Untuk melihat lebih detail perbandingan hasil training dari semua model yang ada, dapat dilihat pada gambar 4.44 sampai dengan gambar 4.49.

Chart

Description automatically generated

Gambar 4. 44 Grafik perbandingan akurasi model berdasarkan fungsi aktivasi

Chart, line chart

Description automatically generated

Gambar 4. 45 Grafik perbandingan validasi akurasi model berdasarkan fungsi aktivasi

Chart

Description automatically generated

Gambar 4. 46 Grafik perbandingan loss model berdasarkan fungsi aktivasi

Chart, line chart

Description automatically generated

Gambar 4. 47 Grafik perbandingan validasi loss model berdasarkan fungsi aktivasi

Chart, line chart

Description automatically generated

Gambar 4. 48 Grafik perbandingan f1 score model berdasarkan fungsi aktifasi

Chart, line chart

Description automatically generated

Gambar 4. 49 Grafik perbandingan validasi f1 score model berdasarkan fungsi aktifasi

Dari grafik akurasi, loss serta f1 score dari semua model, terlihat semua model hampir memiliki nilai yang sama, yang membedakan ialah pada iterasi ke berapa model tersebut berhenti dikarenakan Early Stopping. Sedangkan di bagian validasi akurasi, validasi loss serta validasi f1 score, terlihat beberapa perbedaan yang cukup berbeda satu sama lain. Model dengan hidden layer simoid lebih sedikit ditemukan sebuah *noise* dan nilai validasi akurasi, validasi loss dan validasi f1 score naik perlahan, berbeda dengan model yang lain. Perbedaan ini dikarenakan kombinasi fungsi aktifasi yang ada di dalam model.

Untuk percobaan ini, fungsi aktifasi yang paling optimal untuk digunakan ialah fungsi aktifasi relu untuk di hidden layer, dan fungsi aktifasi softmax untuk di output layer.

#### Pengujian Terhadap Optimizer

Pada bagian ini, akan membandingkan antara arsitektur model dengan layer LSTM dengan model dengan layer BiLSTM. Model yang akan dibagun sederhana, dengan komposisi input layer berupa Embedding layer, hidden layer berupa satu layer LSTM ataupun BiLSTM, dan output layer berupa Dense. Optimizer yang akan digunakan ialah Adam, activation function untuk hidden layer tanh dan untuk output layer adalah softmax, serta loss function yang digunakan ialah categorical. Setelah dilakukan percobaan, didapatkan data hasil penelitian yang dapat dilihat pada tabel 4.2.

Model sederhana yang disusun terdiri dari beberapa komponen penting diantaranya adalah *layer* dan fungsinya aktifasinya, optimizer, fungsi loss, learning rate dan epoch. *Layer* yang akan disusun terdiri dari Embedding *layer*, LSTM *layer*, BiLSTM *layer*, Flatten *layer* dan Dense *layer*. Optimizer yang akan digunakan diantaranya adalah Adam dan RMSProp. Fungsi aktifasi yang akan digunakan untuk tiap *layer* diantaranya ialah untuk hidden *layer* berupa relu, dan untuk *output* *layer* berupa softmax. Fungsi loss yang digunakan diataranya adalah categorical\_crossentropy dan mse (mean squared error). Model yang akan dilatih menggunakan sebuah *callback*s EarlyStopping dengan nilai patience lima yang berarti jika tidak ada peningkatan matrix yang diinginkan selama lima kali iterasi, maka *training* akan diberhentikan. Setelah itu, dilakukan *training* ke beberapa model yang telah disusun yang dapat dilihat pada tabel 4.2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | *Layer*/Aktifasi | Optimizer | Fungsi loss | Early stopping | Loss | Accuracy | F1 Score |
| 1 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense (3/softmax) | Adam | Categorical | 9 | 0.5386 | 0.8157 | 0.8007 |
| 2 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense (3/softmax) | Adam | MSE | 20 | 0.0863 | 0.8407 | 0.83 |
| 3 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense(3/softmax) | RMSProp | Categorical | 13 | 0.4791 | 0.8369 | 0.8277 |
| 4 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense(3/softmax) | RMSProp | MSE | 14 | 0.0792 | 0.8541 | 0.8528 |
| 5 | * Embedding (16) * BiLSTM(64/tanh) * Flatten * Dense(3/softmax) | Adam | Categorical | 8 | 0.4712 | 0.8522 | 0.8427 |
| 6 | * Embedding (16) * BiLSTM(64/tanh) * Flatten * Dense(3/softmax) | Adam | MSE | 17 | 0.0755 | 0.8637 | 0.8676 |
| 7 | * Embedding (16) * BiLSTM(64/tanh) * Flatten * Dense(3/softmax) | RMSProp | Categorical | 7 | 0.5704 | 0.8407 | 0.8334 |
| 8 | * Embedding (16) * BiLSTM(64/tanh) * Flatten * Dense(3/softmax) | RMSProp | MSE | 11 | 0.0793 | 0.8464 | 0.8468 |

Tabel 4. 4 Hasil percobaan model sederhana

Hasil percobaan diatas merupakan hasil *training* kombinasi antara arsitektur *layer*, optimizer serta fungsi loss yang berbeda. Dari percobaan yang telah dilakukan, didapatkan 8 kombinasi model beserta hasil *training*nya. Dari tabel diatas, terdapat tiga kombinasi model dengan akurasi dan nilai f1 score terbaik, yaitu model nomor empat, lima dan enam dimana urutan terbaiknya yaitu nomor enam, empat dan kemudian lima.

Perbedaan yang ada antara model nomor enam dan nomor empat ialah perbedaan penggunaan *layer* dan juga optimizer dimana penggunaan model BiLSTM dengan optimizer Adam lebih unggul dibandingkan dengan penggunaan model LSTM dengan optimizer RMSProp. Perbedaan antara model nomor lima dengan nomor enam terletak pada fungsi lossnya dimana mse lebih unggul dan lebih baik dibandingkan categorical pada model ini dengan selisih loss sebesar 0.397. Untuk melihat lebih dalam dalam proses *training* model, dapat melihat beberapa grafik hasil *training* dan juga nilai confusion matrix tiap model.

Chart, line chart, histogram

Description automatically generated

Gambar 4. 50 Grafik akurasi, validasi akurasi serta f1 score dari model nomor 4.

Chart, treemap chart

Description automatically generated

Gambar 4. 51 Confusion matrix model nomor 4.

Graphical user interface, chart, line chart

Description automatically generated

Gambar 4. 52 Grafik akurasi, validasi akurasi serta f1 score dari model nomor 5.

Chart, treemap chart

Description automatically generated

Gambar 4. 53 Confusion matrix model nomor 5.

Graphical user interface, chart, line chart, histogram

Description automatically generated

Gambar 4. 54 Grafik akurasi, validasi akurasi serta f1 score dari model nomor 6.

Chart, treemap chart

Description automatically generated

Gambar 4. 55 Confusion matrix model nomor 6.

Jika dilihat dari grafik yang ada, dapat terlihat bahwa semua model mengalami overfitting yang jaraknya antara akurasi dan validasi akurasi hampir sama semua. Yang dapat dilihat dari ketiga grafik diatas ialah, seberapa “noisy” grafik yang ada. Pola *training* dari model nomor empat dan lima hampir sama dan lebih cenderung memiliki “noisy” naik dan turun dibandingkan dengan model nomor lima yang cenderung landau dan statis. Hal tersebut disebabkan oleh fungsi loss yang dimiliki oleh model nomor empat dan nomor lima adalah mse.

Selanjutnya, jika ditinjau dari confusion matrix, ketiga model tersebut memiliki selisih yang tidak jauh di dalam nilai setiap true positif tiap label. True positif berarti data yang ditebak memanglah benar adanya. Untuk label 0 ataupun netral, model yang paling banyak memiliki nilai true positif adalah model nomor enam. Untuk label 1 atau positif, model yang dapat menebak lebih baik dibandingkan model lain ialah model nomor empat. Dan untuk label 2 ataupun label negatif, model yang unggul ialah model nomor empat dan enam yang memiliki jumlah true positif yang sama yaitu 92. Dalam hal ini, dapat dikatakan bahwa model nomor enam adalah model terbaik dalam scenario pengujian sederhana ini, yaitu model dengan *layer* BiLSTM dengan optimizer Adam dan fungsi loss mse.

### Pengujian Model Kompleks

Model kompleks yang akan disusun tidaklah jauh berbeda dengan skenario sederhana yang ada, yang membedakan adalah komplekksitas *layer* model yang ada. Pada pengujian ini, *layer* akan ditambahkan dengan beberapa kombinasi tambahan dengan menggunakan kombinasi Dropout *layer* dan juga tambahan Dense *layer* di setiap model LSTM dan BiLSTM yang ada. Model kompleks dilatih selama 20 *epochs* dengan menggunakan *callback* EarlyStopping dengan nilai patience sebesar lima. Optimizer yang digunakan adalah Adam dan RMSProp, sedangkan untuk fungsi loss yang digunakan ialan categorical dan mse. Untuk fungsi aktifasi di dalam hidden *layer* Dense yang baru, menggunakan relu di setiap *layer*nya. Untuk hasil *training* model kompleks dapat dilihat pada tabel 4.3.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | *Layer*/Aktifasi | Optimizer | Fungsi loss | Early stopping | Loss | Accuracy | F1 Score |
| 1 | * Embedding (128) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | Adam | Categorical | 8 | 0.4796 | 0.8503 | 0.8339 |
| 2 | * Embedding (128) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | Adam | MSE | 12 | 0.0728 | 0.8695 | 0.867 |
| 3 | * Embedding (128) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | RMSProp | Categorical | 11 | 0.4796 | 0.8772 | 0.8682 |
| 4 | * Embedding (128) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * LSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | RMSProp | MSE | 15 | 0.0709 | 0.8772 | 0.8731 |
| 5 | * Embedding(128) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | Adam | Categorical | 11 | 0.7264 | 0.8522 | 0.8408 |
| 6 | * Embedding(128) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | Adam | MSE | 17 | 0.0804 | 0.858 | 0.8502 |
| 7 | * Embedding(128) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | RMSProp | Categorical | 8 | 0.5509 | 0.8906 | 0.8914 |
| 8 | * Embedding(128) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * BiLSTM(128/tanh,DO(0.5)) * Flatten * Dense(512/relu) * DO(0.5) * Dense(3/softmax) | RMSProp | MSE | 10 | 0.0396 | 0.9021 | 0.8941 |

Tabel 4. 5 Hasil percobaan model kompleks.

Hasil percobaan di atas merupakan hasil *training* dengan scenario kompleks. Dapat dilihat bahwa hasil *training* baik dari segi akurasi, losss serta f1 score tidaklah terpaut telalu jauh akan tetapi skenario ini dapat menaikan akurasi dan juga f1 score serta menurunkan nilai loss yang ada daripada skenario sebelumnya. Dari delapan kombinasi model yang ada, diambil tiga kombinasi terbaik dan didapatkan model nomor empat, tujuh dan delapan menjadi model terbaik diantara model lainnya di skenario pengujian kompleks ini.

Yang membedakan antara model nomor empat dan nomor tujuh ialah arsitektur *layer* model dan juga fungsi loss yang digunakan. Model nomor empat menggunakan arsitektur dengan LSTM dan fungsi loss categorical sedangkan model nomor tujuh menggunakan arsitektur BiLSTM dan fungsi loss mse. Walaupun akurasi dan f1 score dari model nomor tujuh lebih besar dengan selisih 0,0134 dan 0,0183, akan tetapi nilai loss dari model nomor empat lebih kecil dibandingkan dengan model nomor empat.

Model nomor tujuh dengan nomor delapan hampir sama, yang membedakan hanyalah fungsi loss yang digunakan yaitu categorical untuk model nomor tujuh dan mse untuk model nomor delapan. Model nomor delapan lebih unggul dalam akurasi dengan terpaut selisih sebesar 0.0115 dan untuk loss terpaut lumayan jauh sebesar 0,5113.

Model nomor empat dan nomor tujuh memiliki kombinasi optimizer dan fungsi loss yang sama. Yang membedakan ialah arsitektur *layer* model yang ada. Untuk akurasi model terpaut sebesar 0.024 untuk keunggulan model nomor delapan, loss terpaut 0.0313 lebih unggul model nomor delapan dan untuk f1 score terpaut sebesar 0,0027 untuk keunggulan model nomor delapan.

Dari hasil percobaan skenario pengujian kompleks di atas, dapat dilihat bahwa optimizer terbaik untuk skenario ini ialah rmsprop. Fungsi loss terbaik ialah mse karena dapat menekan nilai loss sangatlah kecil dibandingkan dengan categorical. Dan untuk arsitektur terbaik dalam skenario ini ialah model dengan BiLSTM *layer* dengan hasil percobaan dua teratas. Untuk melihat lebih dalam dalam proses *training* model, dapat melihat beberapa grafik hasil *training* dan juga nilai confusion matrix tiap model.

Graphical user interface, chart, line chart, histogram

Description automatically generated

Gambar 4. 56 Grafik akurasi, validasi akurasi serta f1 score dari model nomor empat.

Chart, treemap chart

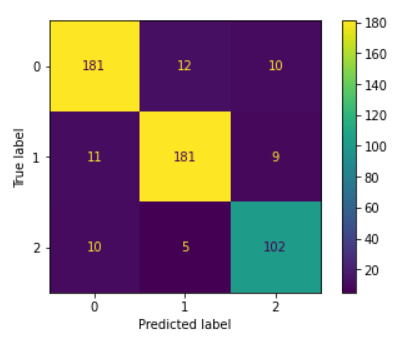
Description automatically generated

Gambar 4. 57 Confusion matrix model nomor empat.

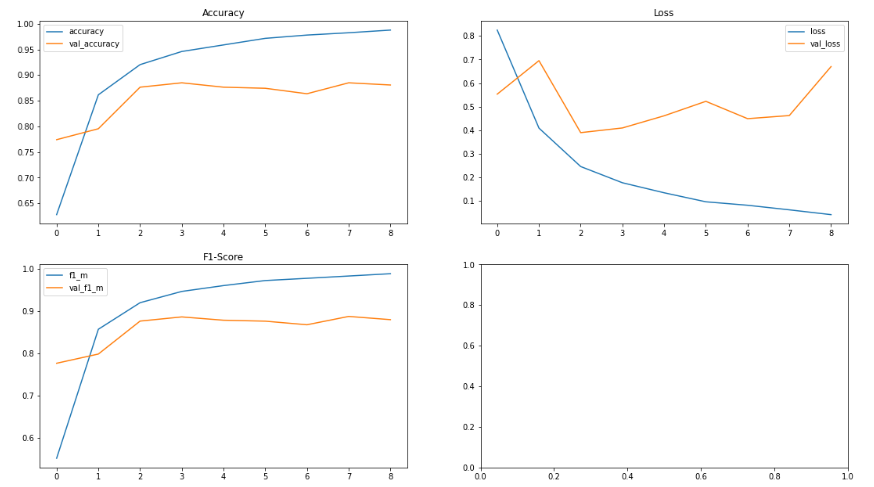
Chart, line chart

Description automatically generated

Gambar 4. 58 Grafik akurasi, validasi akurasi serta f1 score dari model nomor tujuh.



Gambar 4. 59 Confusion matrix model nomor tujuh.



Gambar 4. 60 Grafik akurasi, validasi akurasi serta f1 score dari model nomor delapan.

Chart, treemap chart

Description automatically generated

Gambar 4. 61 Confusion matrix model nomor delapan.

Jika dilihat dari beberapa grafik yang ada, terlihat bahwa semua model masih mengalami overfitting yang tidak jauh berbeda dari skenario pengujian simpel. Pola *training* dari model nomor empat lebih “noisy” dibandingkan dengan model yang lain. Dan untuk model nomor delapan adalah model dengan pola *training* yang paling tidak noisy dibandingkan dengan model yang lain.

Jika ditinjau dari confusion matrix, ketiga model tersebut memiliki selisih yang tidak jauh di dalam nilai setiap true positif tiap label. Untuk label 0 ataupun netral sebanyak 188 dan label 1 ataupun positif sebanyak 183, model yang paling banyak memiliki nilai true positif adalah model nomor delapan. Dan untuk label 2 ataupun label negatif, model yang unggul ialah model nomor tujuh kemudian diikuti oleh model nomor empat dan delapan dengan jumlah yang sama yaitu 99.

Dalam hal ini, dapat dikatakan bahwa model nomor delapan adalah model terbaik dalam scenario pengujian sederhana ini, yaitu model dengan *layer* BiLSTM dengan optimizer RMSProp dan fungsi loss mse.

# BAB V

**PENUTUP**



## Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan yaitu analisis sentimen ujaran kebencian Twitter dengan topik “Selena Gomez” menggunakan LSTM dan Bidirectional LSTM diperoleh beberapa kesimpulan sebagai berikut:

1. Metode *deep learning* dengan arsitektur LSTM dan BiLSTM dapat melakukan analisis sentimen ujaran kebencian di Twitter.
2. Setelah diuji menggunakan dua skenario, dapat diketahui bahwa model dengan layer BiLSTM lebih unggul terhadap model dengan layer LSTM saja di skenario simple dengan akurasi 86,37% dan skenario kompleks dengan akurasi 90,21%.
3. Dari penelitian yang telah dilakukan, maka ditemukanlah parameter *deep learning* yang paling optimal untuk kasus ini yaitu:
   1. Fungsi loss menggunakan mean squared error karena dapat menekan nilai loss sampai dengan nilai paling minimum dan menghasilkan beberapa model terbaik yang dihasilkan oleh beberapa percobaan dalam penelitian ini.
   2. Untuk optimizer , optimizer adam dan rmsprop tidaklah jauh berbeda dalam memberikan efek terhadap model. Akan tetapi, setelah dilakukan dua skenario dan diambil tiga model terbaik di setiap skenario, maka di dapatkan optimizer adam optimal pada saat dilakukan skenario pengujian sederhana dengan perbandingan 2:1 terhadap rmsprop. Akan tetapi, pada saat memasuki model kompleks, optimizer adam tidak masuk dalam tiga besar model terbaik dalam skenario pengujian ini dan terlihat bahwa optimizer rmsprop lebih optimal menangani model kompleks. Diambil kesimpulan bahwa optimizer rmsprop lebih unggul dalam kedua skenario, walaupun pada saat skenario pengujian sederhana tidaklah terlalu unggul terhadap optimizer adam.
   3. Callbacks Early Stopping sangatlah membantu dalam *training* model. Hal ini dikarenakan ketika melakukan *training* sebanyak 20 epochs, ada suatu titik dimana ketika model dilatih dan mengalami akurasi yang menurun secara signifikan dan konsisten, maka callbacks ini dapat mengembalikan model dengan bobot terbaiknya sehingga akurasi model tidaklah turun.

## Saran

Berdasarkan hasil penelitian yang telah dilakukan yaitu analisis sentimen ujaran kebencian Twitter dengan topik “Selena Gomez” menggunakan LSTM dan Bidirectional LSTM, peneliti memberikan beberapa saran sebagai berikut:

1. Dataset yang digunakan lebih banyak lagi jumlahnya agar model dibangun jauh lebih baik lagi karena mengenali lebih banyak variasi data yang ada.
2. Mencari dataset yang lebih variatif. Jika menggunakan data Twitter, diusahakan menghindari data *tweet* yang merupakan re*tweet*.
3. Dilakukan penelitian lain yang membahas topik dan data yang sama, namun dengan metode yang berbeda.
4. Menggunakan *feature extraction* yang berbeda seperti TF-IDF agar mendapatkan model yang lebih optimal.

# Daftar Pustaka

# References

A, K. (n.d.). What are artificial neural network? *2008*. Retrieved from https://www.nature.com/articles/nbt1386

Abdelgwad, M. M., Soliman, T. H., Taloba, A. I., & Farghaly, M. F. (2021). Arabic aspect based sentiment analysis using bidirectional GRU based models. Retrieved from https://www.sciencedirect.com/science/article/pii/S1319157821002482

Ahmad, M., Ali, I., & Ahtab, S. (2017). *Sentiment Analysis of Tweets using SVM*. Retrieved from https://www.researchgate.net/profile/Shabib-Aftab-2/publication/321084834\_Sentiment\_Analysis\_of\_Tweets\_using\_SVM/links/5a1497b90f7e9b925cd514b0/Sentiment-Analysis-of-Tweets-using-SVM.pdf

algorit.ma. (2022). Apa Itu Long Short Term Memory Network (LSTM)? Retrieved from https://algorit.ma/blog/lstm-network-adalah-2022/

algorit.ma. (2022). Apa Itu Sentiment Analysis. Retrieved from https://algorit.ma/blog/sentiment-analysis-adalah-2022/

Colah. (2015). Understanding LSTM Networks. Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Datta, L. (2020). A Survey on Activation Functions and their relation with Xavier and He Normal Initialization.

Denny, M. J., & Spirling, A. (2017). Text Preprocessing For Unsupervised Learning: Why It. Retrieved from https://www.cambridge.org/core/journals/political-analysis/article/abs/text-preprocessing-for-unsupervised-learning-why-it-matters-when-it-misleads-and-what-to-do-about-it/AA7D4DE0AA6AB208502515AE3EC6989E

Dumane, G. (n.d.). Introduction to Convolutional Neural Network (CNN) using Tensorflow. *2020*. Retrieved from https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83#:~:text=Dense%20Layer%20is%20simple%20layer,multiple%20number%20of%20such%20neurons.

Edu, S. U. (n.d.). Stemming and lemmatization. Retrieved from https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html#:~:text=Lemmatization%20usually%20refers%20to%20doing,is%20known%20as%20the%20lemma%20.

Fauzi, M. A. (2018). *Random Forest Approach for Sentiment Analysis in Indonesian Language*. Retrieved from https://www.researchgate.net/profile/Muhammad-Fauzi-6/publication/327060733\_Random\_Forest\_Approach\_for\_Sentiment\_Analysis\_in\_Indonesian\_Language/links/5d305ff3458515c11c39adfd/Random-Forest-Approach-for-Sentiment-Analysis-in-Indonesian-Language.pdf

Fitri, V. A., Andreswari, R., & Hasibuan, M. A. (2019). *Sentiment Analysis of Social Media Twitter with Case of Anti-LGBT Campaign in Indonesia using Naïve Bayes, Decision Tree, and Random Forest Algorithm*. Retrieved from https://www.researchgate.net/profile/Muhammad-Fauzi-6/publication/327060733\_Random\_Forest\_Approach\_for\_Sentiment\_Analysis\_in\_Indonesian\_Language/links/5d305ff3458515c11c39adfd/Random-Forest-Approach-for-Sentiment-Analysis-in-Indonesian-Language.pdf

Huq, M. R., Ali, A., & Rahman, A. (2017). Sentiment Analysis on Twitter Data using KNN and SVM. Retrieved from https://pdfs.semanticscholar.org/05a8/78000170abcd0c6f8208080470858422e17c.pdf

Jianqiang, Z., Xiaolin, G., & Xuejun, Z. (2018). *Deep Convolution Neural Networks for Twitter Sentiment Analysi*. Retrieved from https://ieeexplore.ieee.org/abstract/document/8244338

Karani, D. (2018). Introduction to Word Embedding and Word2Vec. Retrieved from https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. Retrieved from https://www.nature.com/articles/nature14539

Li, Y. (n.d.). Bidirectional RNN. Retrieved from https://sthsf.github.io/2017/08/31/Tensorflow%E5%9F%BA%E7%A1%80%E7%9F%A5%E8%AF%86-bidirectional-rnn/index.html

Liao, S., Wang, J., Yu, R., Sato, K., & Cheng, Z. (2016). CNN for situations understanding based on. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050917312103

Marimuthu, P. (2022). How Activation Functions Work in Deep Learning. Retrieved from https://www.kdnuggets.com/2022/06/activation-functions-work-deep-learning.html

MathWorks. (2022). What Is Deep Learning? Retrieved from https://www.mathworks.com/discovery/deep-learning.html#:~:text=Deep%20learning%20is%20a%20machine,a%20pedestrian%20from%20a%20lamppost

Medhat, W., Hassan, A., & Korashy, H. (2014). *Sentiment analysis algorithms and applications: A survey*. Retrieved from https://www.sciencedirect.com/science/article/pii/S2090447914000550

MIT. (2014). VADER-Sentiment-Analysis. Retrieved from https://github.com/cjhutto/vaderSentiment

Mungalpara, J. (2021). What does it mean by Bidirectional LSTM? Retrieved from https://medium.com/analytics-vidhya/what-does-it-mean-by-bidirectional-lstm-63d6838e34d9

Musstafa. (2021). Optimizers in Deep Learning. Retrieved from https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0

Permatasari, D. I., & Subyantoro. (2020). UJARAN KEBENCIAN FACEBOOK TAHUN 2017-2019. Retrieved from https://journal.unnes.ac.id/sju/index.php/jsi/article/view/33020

ReaserchGate. (n.d.). Gambar Embedding Layer BiLSTM. Retrieved from https://www.researchgate.net/publication/340716994/figure/fig3/AS:881352496918530@1587142218942/Network-Architecture-using-Bi-LSTM-model-with-input-being-Embedding-layer-and-PoS-vector.ppm

Rizaty, M. A. (2022). Pengguna Twitter di Indonesia Capai 18,45 Juta pada 2022. Retrieved from https://dataindonesia.id/digital/detail/pengguna-twitter-di-indonesia-capai-1845-juta-pada-2022

Rosyid. (2022). Hasil Survei Mengungkapkan Media Sosial Paling Digemari di Indonesia. Retrieved from https://www.gatra.com/news-548811-nasional-hasil-survei-mengungkapkan-media-sosial-paling-digemari-di-indonesia-.html#:~:text=Laporan%20We%20Are%20Social%20mengungkapkan,2021%20sebesar%20170%20juta%20orang.

Sastrawi. (2017). Sastrawi. Retrieved from https://github.com/sastrawi/sastrawi

Singhal, P., & Bhattacharyya, P. (2016). Sentiment Analysis and Deep Learning: A Survey. Retrieved from https://www.cfilt.iitb.ac.in/resources/surveys/sentiment-deeplearning-2016-prerna.pdf

Verma, Y. (2021). A Complete Understanding of Dense Layers in Neural Networks. Retrieved from https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/

Wikipedia. (2022). Ucapan Kebencian. Retrieved from https://id.wikipedia.org/wiki/Ucapan\_kebencian

Wikipedia. (n.d.). Jaringan Syaraf Tiruan. *2022*. Retrieved from https://id.wikipedia.org/wiki/Jaringan\_saraf\_tiruan

Yadav, H. (n.d.). Dropout in Neural Networks. *2022*. Retrieved from https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9

Yathish, V. (2022). Loss Functions and Their Use In Neural Networks. Retrieved from https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Retrieved from https://direct.mit.edu/neco/article-abstract/31/7/1235/8500/A-Review-of-Recurrent-Neural-Networks-LSTM-Cells

# Jadwal Kerja

Isi jadwal kerja dari minggu ke minggu sesuai dengan contoh berikut ini.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | Nama Kegiatan | Waktu / Tahun | | | | | | | | | | | |
| Nama Bulan | | | | Nama Bulan | | | | Nama Bulan dst | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |