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

**SKRIPSI**

Diajukan untuk Memenuhi Salah Satu Syarat Memeroleh Gelar Sarjana Pendidikan Program Studi Informatika



Oleh:

Adrianus Charlie Hadirria Agelsadewa

NIM : 195314174

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS SANATA DHARMA**

**YOGYAKARTA**

**2023**

**LSTM AND BIDIRECTIONAL LSTM FOR ANALYZING SENTIMENT TOWARDS HATE SPEECH ON TWITTER**

**A Thesis**

Presented as Partial Fulfillment of the Requirements to Obtain Computer Science Degree in Department of Informatics



By:

Adrianus Charlie Hadirria Agelsadewa

Student ID: 195314174

**INFORMATICS STUDY PROGRAM**

**FACULTY OF SCIENCE AND TECHNOLOGY**

**UNIVERSITAS SANATA DHARMA**

**YOGYAKARTA**

**2023**

# HALAMAN PERSEMBAHAN

*“Jika mimpimu belum ditertawakan orang lain, maka mimpimu terlalu kecil!”*

– Monkey D. Luffy

# ABSTRAK

|  |
| --- |
| Teknologi informasi telah berkembang pesat di Indonesia yang memasuki era industri 4.0. Namun, terdapat peningkatan penggunaan ujaran kebencian di media sosial yang dapat memicu perpecahan di masyarakat. Salah satu metode untuk mendeteksi tweet yang mengandung ujaran kebencian adalah sentimen analisis dengan teknik *deep learning* algoritma LSTM dan Bi-LSTM yang memiliki kemampuan untuk mengingat informasi dalam jangka panjang dan menghapus informasi yang tidak relevan. Penting bagi pengguna media sosial untuk menghindari penggunaan ujaran kebencian dan memastikan tweet mereka tidak mengandung unsur ujaran kebencian dengan adanya metode ini. Sentimen analisis pada penelitian ini menggunakan data Twitter “Selena Gomez” dan didapatkan data sebanyak 11.646. Penelitian berfokus pada arsitektur model antara LSTM dan Bi-LSTM serta pengoptimalan beberapa parameter diantaranya ialah fungsi aktivasi, dan optimizer. Didapatkan model dengan performa terbaik dan optimal yaitu dengan akurasi 89.64% dengan arsitektur Bi-LSTM kompleks, fungsi aktivasi relu untuk hidden layer dan softmax untuk output layer, optimizer Adam serta fungsi loss *categorical crossentropy*. |

# ABSTRACT

|  |
| --- |
| With the rapid advancement of information technology in Indonesia, particularly in the era of Industry 4.0, there has been a notable increase in the usage of hate speech on social media platforms, which can potentially lead to social division. To address this issue, sentiment analysis techniques employing deep learning algorithms, such as LSTM and Bi-LSTM, have been widely explored for the detection of hate speech in tweets. These algorithms possess the capability to retain long-term information while discarding irrelevant data. The application of such methods is crucial for social media users to avoid hate speech and ensure their tweets do not contain any offensive content. In this research, sentiment analysis is performed on a dataset comprising 11,646 tweets related to the Twitter account of "Selena Gomez." The fokus of the study lies in comparing the architectural models of LSTM and Bi-LSTM while optimizing various parameters including activation functions and optimizers. The results demonstrate that the Bi-LSTM architecture with a complex structure, employing relu as the activation function for hidden layers and softmax for the output layer, Adam as the optimizer, and categorical cross-entropy as the loss function, achieves the best and optimal performance with an accuracy of 89.64%.Top of Form |

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

**oleh:**

**Pembimbing Tugas Akhir**

**Drs. Hari Suparwito, S.J., M. App.IT**

# KATA PENGANTAR

Puji dan Syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat dan karunianya, penulis dapat menyelesaikan tugas akhir ini yang berjudul “**LSTM Dan Bidirectional LSTM Untuk Menganalisa Sentimen Terhadap Ujaran Kebencian Twitter”**

Dalam penulisan tugas akhir ini, penulis juga ingin mengungkapkan rasa terima kasih atas dukungan, motivasi, ucapan doa dan segala pertolongan kepada:

1. Tuhan Yang Maha Esa yang senantiasa memberikan saya berkat, lindungan dan bantuannya dalam saya mengerjakan tugas akhir ini
2. Romo Drs. Hari Suparwito, S.J., M. App.IT
3. Ibuk yang selalu mendukung dan mendoakan saya di setiap waktu kapanpun itu
4. Orang tua dan saudara dari penulis yang selalu memberikan dukungan kepada penulis
5. Laurensia Simanihuruk yang selalu ada, mendukung dan membantu penulis dalam suka maupun duka dalam proses pengerjaan penelitian.
6. Sobat rekan “mitigation” yang selalu menghibur penulis dan menemani penulis Menyusun penelitian
7. Seluruh pihak yang mendukung penulis dalam menyelesaikan tugas akhir ini

Semoga penelitian yang dilakukan ini bermanfaat bagi sesama, terutama orang yang tertarik dengan dunia informatika seperti machine learning.

# DAFTAR ISI

[HALAMAN PERSEMBAHAN i](#_Toc137574200)

[ABSTRAK ii](#_Toc137574201)

[ABSTRACT iii](#_Toc137574202)

[KATA PENGANTAR v](#_Toc137574203)

[DAFTAR TABEL ix](#_Toc137574204)

[DAFTAR GAMBAR x](#_Toc137574205)

[BAB I 1](#_Toc137574206)

[I.1. Latar Belakang 1](#_Toc137574207)

[I.2. Rumusan Masalah 3](#_Toc137574208)

[I.3. Tujuan 4](#_Toc137574209)

[I.4. Manfaat Penelitian 4](#_Toc137574210)

[I.5. Batasan Masalah 4](#_Toc137574211)

[I.6. Sistematika Penulisan 5](#_Toc137574212)

[BAB II 7](#_Toc137574213)

[II.1. Tinjauan Pustaka 7](#_Toc137574216)

[II.2. Landasan Teori 11](#_Toc137574219)

[II.2.1 Analisis sentimen 11](#_Toc137574220)

[II.2.2 Jaringan Syaraf Tiruan 12](#_Toc137574221)

[II.2.3 *Deep learning* 16](#_Toc137574222)

[II.2.4 Long Short-Term Memory (LSTM) 28](#_Toc137574223)

[II.2.5 Bidirectional Long Sshort-Term Memory (LSTM) 31](#_Toc137574224)

[II.2.6 Preprocessing 32](#_Toc137574225)

[BAB III 34](#_Toc137574226)

[III.1. Tahapan Penelitian 34](#_Toc137574228)

[III.2. Pengumpulan Data 34](#_Toc137574229)

[III.3. Preprocessing 36](#_Toc137574230)

[III.3.1 Seleksi Data 37](#_Toc137574231)

[III.3.2 *Lowercasing* 39](#_Toc137574232)

[III.3.3 Pembersihan Karakter Khusus 39](#_Toc137574233)

[III.3.4 Pembersihan Karakter *Non-alphanumeric* 40](#_Toc137574234)

[III.3.5 Stop Word Removal 41](#_Toc137574235)

[III.3.6 Lemmatization 42](#_Toc137574236)

[III.4. Labelling Data 43](#_Toc137574237)

[III.5. Tokenizing 45](#_Toc137574238)

[III.6. Pembagian Data 46](#_Toc137574239)

[III.7. Perancangan Model 47](#_Toc137574240)

[III.7.1 Embedding *Layer* 47](#_Toc137574241)

[III.7.2 Dense *Layer* 48](#_Toc137574242)

[III.7.3 LSTM *Layer* 50](#_Toc137574243)

[III.7.4 Bidirectional LSTM *Layer* 55](#_Toc137574244)

[III.8. Evaluasi Model 56](#_Toc137574245)

[III.9. Skenario Pengujian 57](#_Toc137574246)

[III.9.1 Skenario Pengujian Model Sederhana 58](#_Toc137574247)

[III.9.2 Skenario Pengujian Model Kompleks 60](#_Toc137574248)

[BAB IV 63](#_Toc137574249)

[IV.1. Implementasi Perangkat Lunak 63](#_Toc137574251)

[IV.1.1 Pengumpulan Data 63](#_Toc137574252)

[IV.1.2 Preprocessing Data 65](#_Toc137574253)

[IV.1.3 Labelling Data 70](#_Toc137574254)

[IV.1.4 Tokenizing Data 73](#_Toc137574255)

[IV.1.5 Pembagian Data 76](#_Toc137574256)

[IV.1.6 Modelling 77](#_Toc137574257)

[IV.1.7 Evaluasi 81](#_Toc137574258)

[IV.2. Analisa dan Pembahasan 85](#_Toc137574259)

[IV.2.1 Pengujian Model Sederhana 85](#_Toc137574260)

[IV.2.2 Pengujian Model Kompleks 100](#_Toc137574261)

[BAB V 110](#_Toc137574262)

[V.1. Kesimpulan 110](#_Toc137574264)

[V.2. Saran 112](#_Toc137574265)

[DAFTAR PUSTAKA 113](#_Toc137574266)

[Daftar Istilah / Glosarium 116](#_Toc137574267)

[Jadwal Kerja 125](#_Toc137574268)

# DAFTAR TABEL

[Tabel 2. 1 Tinjauan Pustaka Penelitian. 11](#_Toc137497809)

[Tabel 3. 1 Sample dataset 36](#_Toc137497788)

[Tabel 3. 2 Contoh data dengan kata “RT”. 39](#_Toc137497789)

[Tabel 3. 3 Contoh data dalam proses *lowercasing.* 39](#_Toc137497790)

[Tabel 3. 4 Contoh data dalam proses pembersihan karakter 40](#_Toc137497791)

[Tabel 3. 5 Contoh data pada proses Pembersihan *Non-alphanumeric.* 41](#_Toc137497792)

[Tabel 3. 6 Contoh data dalam *stopword* removal. 42](#_Toc137497793)

[Tabel 3. 7 Contoh data dalam proses *lemmatization.* 43](#_Toc137497794)

[Tabel 3. 8 Labeling Data menggunakan Vader Analisis sentimen 45](#_Toc137497795)

[Tabel 3. 9 Contoh data dalam proses *tokenizing.* 46](#_Toc137497796)

[Tabel 3. 10 Label *one hot encoder*. 47](#_Toc137497797)

[Tabel 3. 11 Contoh data hasil pada proses embedding *layer* 49](#_Toc137497798)

[Tabel 3. 12 Contoh data hasil dense *layer* dengan *output* unit=1 49](#_Toc137497799)

[Tabel 3. 13 Keterangan perhitungan pada formula forget *gate* 51](#_Toc137497800)

[Tabel 3. 14 Keterangan perhitungan pada formula input *gate* 53](#_Toc137497801)

[Tabel 3. 15 Keterangan perhitungan pada formula *cell state* 54](#_Toc137497802)

[Tabel 3. 16 Keterangan perhitungan pada formula *output* *gate* 55](#_Toc137497803)

[Tabel 3. 17 Arsitektur model skenario pengujian sederhana 59](#_Toc137497804)

[Tabel 3. 18 Susunan model berdasarkan fungsi aktivasi yang akan diujikan 60](#_Toc137497805)

[Tabel 3. 19 Susunan model berdasarkan optimizer yang akan diujikan 60](#_Toc137497806)

[Tabel 3. 20 Arsitektur model kompleks berdasarkan *layer* model. 61](#_Toc137497807)

[Tabel 3. 21 Arsitektur model berdasarkan optimizer 62](#_Toc137497808)

[Tabel 4. 1 Komponen penting dalam tokenizing 73](#_Toc136703754)

[Tabel 4. 2 Hasil evaluasi training perbandingan arsitektur *layer* model. 86](#_Toc136703755)

[Tabel 4. 3 Hasil evaluasi training perbandingan fungsi aktifasi menggunakan *layer* Bi-LSTM 92](#_Toc136703756)

[Tabel 4. 4 Hasil evaluasi training model terhadap optimizer 97](#_Toc136703757)

[Tabel 4. 6 Hasil training model terhadap model kompleks 101](#_Toc136703758)

[Tabel 4. 7 Hasil training model kompleks terhadap optimizer 106](#_Toc136703759)

# DAFTAR GAMBAR

[Gambar 2.1 Alur Kerja Analisis sentimen (Medhat, Hassan, & Korashy, 2014) 12](#_Toc137231310)

[Gambar 2.2 Gambar dense *layer* (Verma, 2021) 14](#_Toc137231311)

[Gambar 2. 3 Embedding *layer* di dalam model Bi-LSTM (Reaserch*Gate*) 15](#_Toc137231312)

[Gambar 2. 4 Flatten *layer* 16](#_Toc137231313)

[Gambar 2. 5 Contoh jaringan syaraf tiruan pada *deep learning*  (MathWorks, 2022) 17](#_Toc137231314)

[Gambar 2. 6 Persamaan optimizer RMSPROP (Musstafa, 2021) 18](#_Toc137231315)

[Gambar 2. 7 Persamaan optimizer RMSPROP (Musstafa, 2021) 19](#_Toc137231316)

[Gambar 2. 8 Persamaan optimizer Adam (Musstafa, 2021) 20](#_Toc137231317)

[Gambar 2. 9 persamaan fungsi aktifasi sigmoid (Marimuthu, 2022) 21](#_Toc137231318)

[Gambar 2. 10 Grafik fungsi aktifasi sigmoid (Marimuthu, 2022) 21](#_Toc137231319)

[Gambar 2. 11 Persamaan fungsi aktifasi ReLu (Marimuthu, 2022) 22](#_Toc137231320)

[Gambar 2. 12 Grafik fungsi aktifasi ReLu (Marimuthu, 2022) 22](#_Toc137231321)

[Gambar 2. 13 Persamaan fungsi aktifasi softmax (Marimuthu, 2022) 23](#_Toc137231322)

[Gambar 2. 14 grafik fungsi aktifasi softmax (Marimuthu, 2022) 23](#_Toc137231323)

[Gambar 2. 15 Persamaan fungsi aktifasi tanh (Datta, 2020) 23](#_Toc137231324)

[Gambar 2. 16 Contoh implementasi dari *Dropout* 25](#_Toc137231325)

[Gambar 2. 17 Persamaan MSE (Yathish, 2022) 26](#_Toc137231326)

[Gambar 2. 18 Persamaan MAE (Yathish, 2022) 26](#_Toc137231327)

[Gambar 2. 19 Persamaan fungsi loss binary crossentropy 28](#_Toc137231328)

[Gambar 2. 20 Persamaan fungsi loss categorical crossentropy 28](#_Toc137231329)

[Gambar 2. 21 Contoh RNN (Colah, 2015) 29](#_Toc137231330)

[Gambar 2. 22 Struktur LSTM (Colah, 2015) 29](#_Toc137231331)

[Gambar 2. 23 *Cell state* melalui forget *gate* (Colah, 2015) 30](#_Toc137231332)

[Gambar 2. 24 *Output* gerbang sigmoid (Colah, 2015) 30](#_Toc137231333)

[Gambar 2. 25 Mengupdate *cell state* lama (Colah, 2015) 30](#_Toc137231334)

[Gambar 2. 26 Menentukan *output* terbaru dari *cell state*. (Colah, 2015) 31](#_Toc137231335)

[Gambar 2. 27 Bidirectional LSTM (Mungalpara, 2021) 31](#_Toc137231336)

[Gambar 3. 1 Tahapan Penelitian 34](#_Toc137067677)

[Gambar 3. 2 Cara kerja *flatten* *layer* 49](#_Toc137067678)

[Gambar 3. 3 Forget *gate* lstm (Colah, 2015) 49](#_Toc137067679)

[Gambar 3. 4 Input *gate* lstm (Colah, 2015) 51](#_Toc137067680)

[Gambar 3. 5 *Cell state* LSTM 52](#_Toc137067681)

[Gambar 3. 6 *Output* *gate* LSTM 53](#_Toc137067682)

[Gambar 3. 7 Persamaan Bidirectional LSTM (Li) 54](#_Toc137067683)

[Gambar 4. 1 Import library Python dan inisialisasi token untuk mendapat akses ke akun developer Twitter. 63](#_Toc136703907)

[Gambar 4. 2 Inisialisasi API developer Twitter account 64](#_Toc136703908)

[Gambar 4. 3 Membuat fungsi untuk melakukan iterasisearching *tweet* 64](#_Toc136703909)

[Gambar 4. 4 Membuat fungsi untuk membuat dataframe data hasil scraping 64](#_Toc136703910)

[Gambar 4. 5 Searching *tweet*s menggunakan fungsi sebelumnya dan inisialisasi query untuk mencari data Twitter 64](#_Toc136703911)

[Gambar 4. 6 Implementasi membuat dataframe dan menyimpan ke file excel. 64](#_Toc136703912)

[Gambar 4. 7 Data hasil scraping Twitter. 65](#_Toc136703913)

[Gambar 4. 8 Implementasi seleksi data 66](#_Toc136703914)

[Gambar 4. 9 Implementasi *lowercasing*. 66](#_Toc136703915)

[Gambar 4. 10 Hasil *lowercasing*. 66](#_Toc136703916)

[Gambar 4. 11 Implementasi Pembersihan karakter khusus 67](#_Toc136703917)

[Gambar 4. 12 Hasil pembersihan karakter khusus 67](#_Toc136703918)

[Gambar 4. 13 Implementasi Remove *Non-alphanumeric.* 68](#_Toc136703919)

[Gambar 4. 14 Hasil Implementasi Remove *Non-alphanumeric* 68](#_Toc136703920)

[Gambar 4. 15 Import modul *stopwords* dari libraries nltk.corpus 69](#_Toc136703921)

[Gambar 4. 16 Implementasi penghapusan *stopwords* 69](#_Toc136703922)

[Gambar 4. 17 Hasil implementasi penghapusan *stopwords* 69](#_Toc136703923)

[Gambar 4. 18 Import dan inisialisasi WordNetLemmatizer 70](#_Toc136703924)

[Gambar 4. 19 Implementasi lemmatization 70](#_Toc136703925)

[Gambar 4. 20 Hasil implementasi lemmatization 70](#_Toc136703926)

[Gambar 4. 21 Implementasi Vader analisis sentimen 72](#_Toc136703927)

[Gambar 4. 22 Hasil labelling menggunakan Vader 72](#_Toc136703928)

[Gambar 4. 23 Implementasi menghitung panjang setiap kalimat 74](#_Toc136703929)

[Gambar 4. 24 Hasil implementasi dan visualisasinya. 75](#_Toc136703930)

[Gambar 4. 25 Import libraries untuk melakukan tokenizing. 76](#_Toc136703931)

[Gambar 4. 26 Mempersiapkan elemen komponen penting tokenizing. 76](#_Toc136703932)

[Gambar 4. 27 Contoh hasil implementasi tokenizing. 76](#_Toc136703933)

[Gambar 4. 28 Implementasi one hot encoding 77](#_Toc136703934)

[Gambar 4. 29 Implementasi penyusunan model sederhana berdasarkan arsitektur 78](#_Toc136703935)

[Gambar 4. 30 Implementasi penyusunan model sederhana berdasarkan fungsi aktivasi 78](#_Toc136703936)

[Gambar 4. 31 Implementasi training hasil penyusunan model sederhana berdasarkan arsitektur model 78](#_Toc136703937)

[Gambar 4. 32 Implementasi training hasil penyusunan model sederhana berdasarkan fungsi aktivasi 78](#_Toc136703938)

[Gambar 4. 33 Implementasi penyusunan model sederhana berdasarkan optimizer 79](#_Toc136703939)

[Gambar 4. 34 Implementasi training hasil penyusunan model sederhana berdasarkan optimizer 79](#_Toc136703940)

[Gambar 4. 35 Implementasi penyusunan model kompleks berdasarkan arsitektur model 79](#_Toc136703941)

[Gambar 4. 36 Implementasi training hasil penyusunan model kompleks berdasarkan arsitektur model 80](#_Toc136703942)

[Gambar 4. 37 Implementasi penyusunan model kompleks berdasarkan optimizer 80](#_Toc136703943)

[Gambar 4. 38 Implementasi training hasil penyusunan model kompleks berdasarkan arsitektur optimizer 80](#_Toc136703944)

[Gambar 4. 39 Implementasi *callback* 81](#_Toc136703945)

[Gambar 4. 40 Implementasi dan hasil dari evaluasi model berdasarkan arsitektur model 81](#_Toc136703946)

[Gambar 4. 41 Implementasi dan hasil dari evaluasi model berdasarkan fungsi aktivasi 82](#_Toc136703947)

[Gambar 4. 42 Implementasi dan hasil dari evaluasi model berdasarkan optimizer 82](#_Toc136703948)

[Gambar 4. 44 Implementasi grafik hasil training model berdasarkan arsitektur model 83](#_Toc136703949)

[Gambar 4. 45 Implementasi grafik akurasi hasil training model berdasarkan fungsi aktivasi 83](#_Toc136703950)

[Gambar 4. 46 Implementasi grafik loss hasil training model berdasarkan fungsi aktivasi 84](#_Toc136703951)

[Gambar 4. 47 Implementasi grafik f1-score hasil training model berdasarkan fungsi aktivasi 84](#_Toc136703952)

[Gambar 4. 48 Implementasi grafik hasil training model berdasarkan optimizer 85](#_Toc136703953)

[Gambar 4. 49 Akurasi dan validasi akurasi dari training model LSTM dan Bi-LSTM 88](#_Toc136703954)

[Gambar 4. 50 Loss dan validasi loss dari training model LSTM dan Bi-LSTM 89](#_Toc136703955)

[Gambar 4. 51 F1-score dan validasi f1-score dari training model LSTM dan Bi-LSTM 90](#_Toc136703956)

[Gambar 4. 52 Grafik perbandingan akurasi model berdasarkan fungsi aktivasi 93](#_Toc136703957)

[Gambar 4. 53 Grafik perbandingan validasi akurasi model berdasarkan fungsi aktivasi 93](#_Toc136703958)

[Gambar 4. 54 Grafik perbandingan loss model berdasarkan fungsi aktivasi 94](#_Toc136703959)

[Gambar 4. 55 Grafik perbandingan validasi loss model berdasarkan fungsi aktivasi 94](#_Toc136703960)

[Gambar 4. 56 Grafik perbandingan f1-score model berdasarkan fungsi aktifasi 95](#_Toc136703961)

[Gambar 4. 57 Grafik perbandingan validasi f1-score model berdasarkan fungsi aktifasi 95](#_Toc136703962)

[Gambar 4. 58 Grafik akurasi dan validasi akurasi dari model berdasarkan optimizer 97](#_Toc136703963)

[Gambar 4. 59 Grafik loss dan validasi loss dari model berdasarkan optimizer 98](#_Toc136703964)

[Gambar 4. 60 Grafik f1-score dan validasi f1-score dari model berdasarkan optimizer 99](#_Toc136703965)

[Gambar 4. 64 Grafik akurasi dan validasi akurasi hasil training model terhadap model kompleks 102](#_Toc136703966)

[Gambar 4. 65 Grafik loss dan validasi loss hasill training model terhadap model kompleks 103](#_Toc136703967)

[Gambar 4. 66 Grafik f1-score dan validasi f1-score hasil training terhadap model kompleks 104](#_Toc136703968)

[Gambar 4. 67 Hasil akurasi training model kompleks berdasarkan optimizer 106](#_Toc136703969)

[Gambar 4. 68 Hasil loss training model kompleks terhadap optimizer 107](#_Toc136703970)

[Gambar 4. 69 Hasil f1-score training model kompleks terhadap optimizer 108](#_Toc136703971)

# 

# 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. Pengguna media sosial Twitter di Indonesia kurang lebih sudah mencapai 18,45 juta (Rizaty, 2022). Twitter merupakan jaringan informasi konstan 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.

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 publik akan orang lain dan dapat berpotensi menyebabkan perpecahan di media sosial. Selena Gomez merupakan seorang penyanyi, aktris, pebisnis, produser film, dan seorang *influencer* di salah satu organisasi internasional (UNICEF) tingkat internasional. Pemilihan topik dan kata kunci “Selena Gomez” menjadi salah satu hal penting karena Selena Gomez memiliki pengaruh yang besar terutama bagi remaja dan orang dewasa, ia merupakan wanita dengan akun Instagram wanita dengan pengikut terbanyak di dunia, serta dirinya yang menjadi trending topik di Twitter dalam satu bulan terakhir dimana hampir satu dunia membicarakan namanya.

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. 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 algoritma LSTM dan Bidirectional 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. Bidirectional LSTM atau bidirectional long short-term 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 bagian ini, akan diuraikan tinjauan pustaka dari beberapa referensi yang terkait dengan penelitian ini serta beberapa teori yang mendukung dan relevan, 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

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.

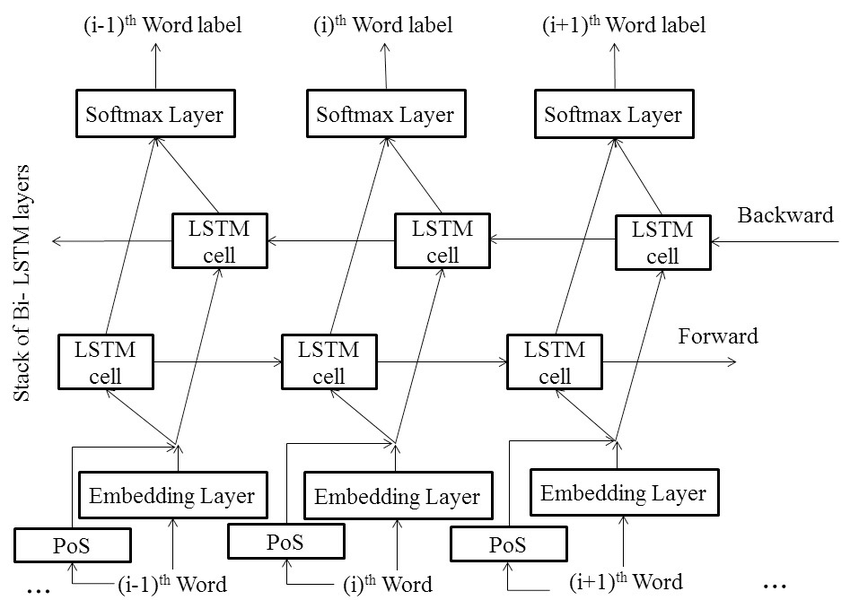
Diagram

Description automatically generated

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

#### Embedding *Layer*

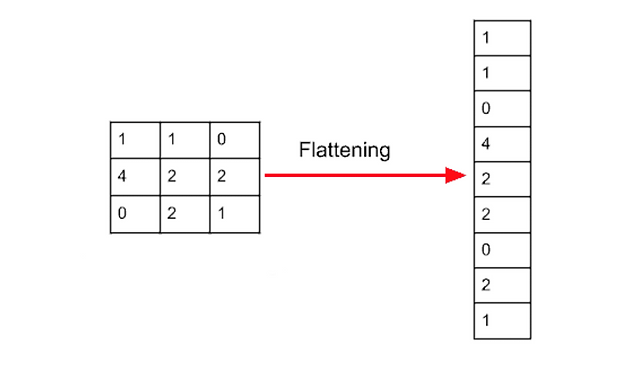
Embedding *layer* merupakan *layer* dalam *deep learning* yang merepresentasikan kosa kata dalam sebuah dokumen yang paling populer yang direpresentasikan dalam sebuah vektor (Karani, 2018). Setelah itu, word embedding dapat menangkap konteks, kesamaan semantik dan sintaksis dalam suatu dokumen.



Gambar 2. 3 Embedding *layer* di dalam model Bi-LSTM (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)

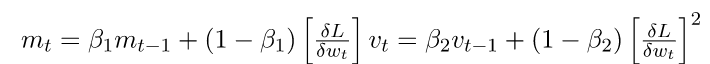
di mana gamma adalah faktor pelupa. Bobot diperbarui dengan rumus di bawah ini

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, Adam 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.

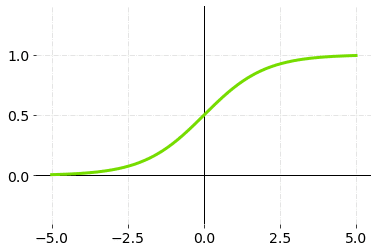
##### Sigmoid

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)

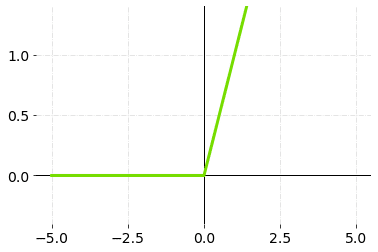
##### Relu

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 relu 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)

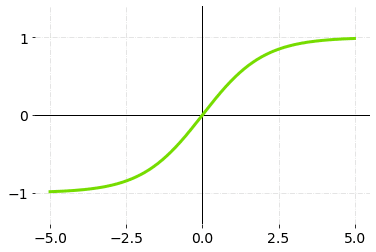
##### Softmax

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

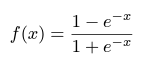
Gambar 2. 13 Persamaan fungsi aktifasi softmax (Marimuthu, 2022)



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

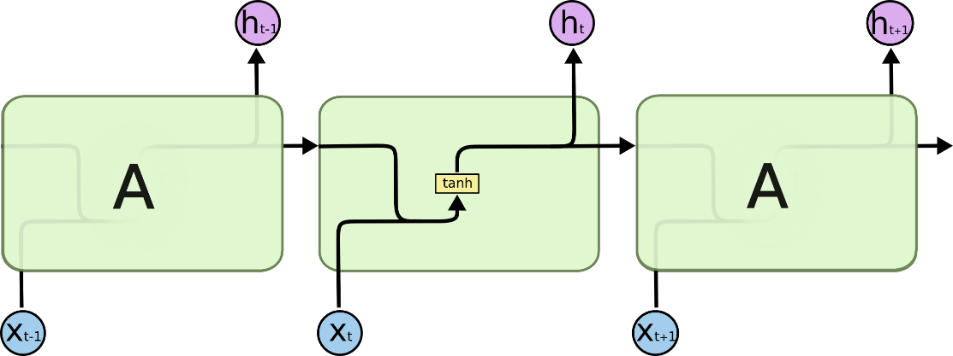
Dalam kasus di mana jumlah kategori lebih dari dua, menggunakan categorical crossentropy. Categorical crossentropy mengikuti proses yang sangat mirip dengan binary crossentropy, satu-satunya perbedaan adalah jumlah kategori yang tersedia.

Diagram, text

Description automatically generated

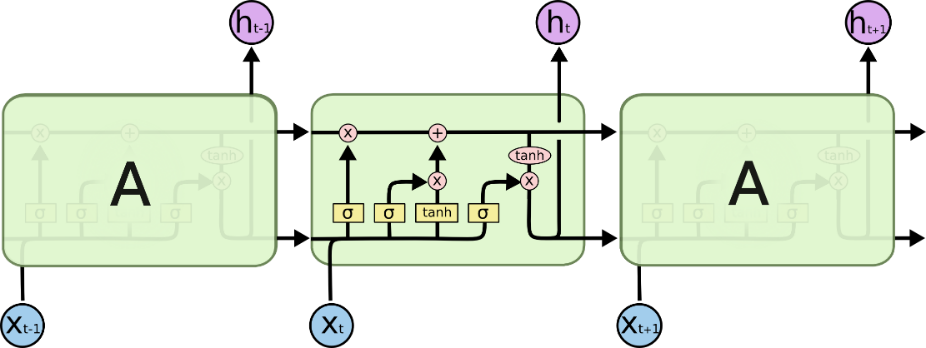
Gambar 2. 20 Persamaan fungsi loss categorical crossentropy

### 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

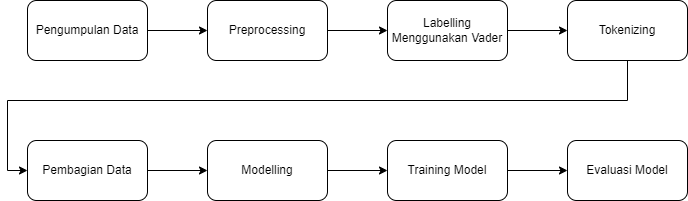
**METODE PENELITIAN**

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



## 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 khusus/tidak biasa, pembersihan karakter selain *alphanumeric*, membersihkan *stopword*, dan lemmatization.

A diagram of a flowchart

Description automatically generated with low confidence

Gambar 3. 2 Tahapan preprocessing dataset

### 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 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 3.4.

|  |  |
| --- | --- |
| **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 dalam proses pembersihan karakter

### 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 3.5.

|  |  |
| --- | --- |
| **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 pada proses Pembersihan *Non-alphanumeric.*

### 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 *stopwords* 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.

### 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 3.7.

|  |  |
| --- | --- |
| **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 house | selena gomez drive house |

Tabel 3. 7 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 sentiment yang dapat dilihat pada tabel 3.8.

|  |  |  |
| --- | --- | --- |
| **Teks** | **Vader *Output*** | **Label** |
| social media disease hailey bieber selena gomez tea hot | {'neg': 0.0, 'neu': 0.694, 'pos': 0.306, 'compound': 0.296} | Positif |
| justin bieber amp selena gomez followed twitter teenage crying | {'neg': 0.189, 'neu': 0.775, 'pos': 0.036, 'compound': -0.6728} | Negatif |
| selena gomez drive house | {'neg': 0.0, 'neu': 0.954, 'pos': 0.046, 'compound': 0.0387} | Netral |

Tabel 3. 8 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 3.9

|  |  |  |
| --- | --- | --- |
| 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. 9 Contoh data dalam proses *tokenizing.*

## Pembagian 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 3.10.

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

Tabel 3. 10 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.

### 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 3.11.

|  |  |
| --- | --- |
| 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. 11 Contoh data hasil pada proses embedding *layer*

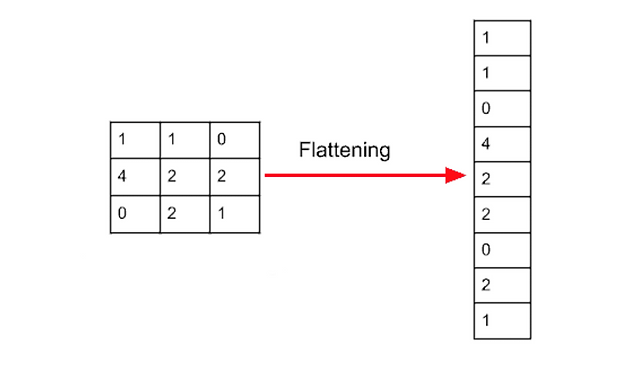
### Dense *Layer*

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. 12 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. 3 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*:

Sebagai contoh, akan terdapat sebuah *cell state* dengan nilai dan *hidden state* pada waktu ke , 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. 4 Forget *gate* lstm (Colah, 2015)

* Menghitung nilai bobot forget *gate* dikalian dengan gabungan *hidden state* 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 hidden state pada waktu t-1 |
|  | Nilai input pada waktu ke t |
|  | Bias pada forget *gate* |
|  | Fungsi sigmoid |

Tabel 3. 13 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. 5 Input *gate* lstm (Colah, 2015)

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

|  |  |
| --- | --- |
|  | Nilai input *gate* |
|  | Nilai bobot untuk input *gate* |
|  | Nilai hidden state pada waktu t-1 |
|  | Nilai input pada waktu ke t |
|  | Bias pada input *gate* |
|  | Fungsi sigmoid |

Tabel 3. 14 Keterangan perhitungan pada formula input *gate*

* 1. *Update 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. 6 *Cell state* LSTM

* Menghitung kandidat baru

|  |  |
| --- | --- |
|  | Nilai forget *gate* |
|  | Nilai input *gate* |
|  | Nilai memory *cell state* |
|  | Nilai memory pada *cell state* sebelumnya |
|  | Nilai kandidat *cell state* |
|  | Nilai bobot untuk cell state |
|  | Nilai bias untuk cell state |

Tabel 3. 15 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. 7 *Output* *gate* LSTM

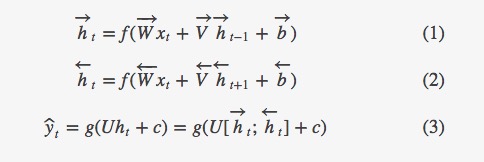
* Menghitung nilai bobot output *gate* dikalian dengan gabungan *hidden state* dari waktu ke t-1 dan nilai input pada waktu ke t
* Menambahkan nilai hasil kalkulasi di atas dengan nilai bias output *gate*
* Menghitung nilai dari output *gate*
* Update nilai *hidden state*

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

Tabel 3. 16 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. 8 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. Bidirectional LSTM akan menghitung LSTM secara maju dan secara mundur, dan kemudian akan menyatukan hasil perhitungan output di setiap LSTM.

## 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, loss 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 dimulai 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.

### Skenario Pengujian Model Sederhana

Pada skenario ini, akan diujikan model arsitektur sederhana yang akan diujikan secara bertahap untuk mencari parameter yang optimal diantaranya akan diujikan pada *layer* model, fungsi aktivasi dan optimizer model. Model akan dilatih selama 20 epoch menggunakan callback EarlyStopping dan memiliki validation split sebesar 0.1 untuk semua skenario pengujian terhadap parameter-parameter model sederhana.

#### Arsitektur Model

Model akan dilatih berdasarkan dua arsitektur model yang berbeda, yaitu model dengan LSTM *layer* dan model dengan Bi-LSTM *layer*. Untuk fungsi aktivasi, optimizer dan fungsi loss akan sama. Model akan disusun dengan arsitektur yang dapat dilihat pada tabel 3.17.

|  |  |  |  |
| --- | --- | --- | --- |
| No | *Layer*/Aktifasi | Optimizer | Fungsi loss |
| 1 | * Embedding (16) * LSTM (64/tanh) * Flatten * Dense (3/softmax) | Adam | Categorical Crossentropy |
| 2 | * Embedding (16) * Bi-LSTM(64/tanh) * Flatten * Dense(3/softmax) | Adam | Categorical Crossentropy |

Tabel 3. 17 Arsitektur model skenario pengujian sederhana

#### Fungsi Aktivasi

Model akan dilatih berdasarkan beberapa kombinasi fungsi aktivasi yang berbeda. Fungsi aktivasi dibedakan menjadi dua, yaitu fungsi aktivasi pada hidden *layer* dan fungsi aktivasi pada output *layer*. Fungsi aktivasi yang akan digunakan pada hidden *layer* diantaranya adalah tanh, sigmoid dan relu. Fungsi aktivasi yang akan digunakan pada output *layer* ialah sigmoid dan softmax. Padang pengujian ini, akan menggunakan *layer* terbaik pada pengujian sebelumnya dengan parameter yang sama dengan percobaan sebelumnya. Model akan disusun dengan arsitektur yang dapat dilihat pada tabel 3.18.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Fungsi Aktivasi | Optimizer | Fungsi loss |
| 1 | * Relu (hidden) * Sigmoid (output) | Adam | Categorical Crossentropy |
| 2 | * Relu (hidden) * Softmax (output) | Adam | Categorical Crossentropy |
| 3 | * Sigmoid (hidden) * Sigmoid (output) | Adam | Categorical Crossentropy |
| 4 | * Sigmoid (hidden) * Softmax (output) | Adam | Categorical Crossentropy |
| 5 | * Tan H (hidden) * Sigmoid (output) | Adam | Categorical Crossentropy |
| 6 | * Tan H (hidden) * Softmax (output) | Adam | Categorical Crossentropy |

Tabel 3. 18 Susunan model berdasarkan fungsi aktivasi yang akan diujikan

#### Optimizer

Model akan dilatih dengan dua variasi optimizer yang akan digunakan pada penelitian ini, yaitu optimizer Adam dan optimizer RMSPROP. Model akan disusun menggunakan arsitektur dan fungsi aktivasi model yang terbaik dari tahap percobaan sebelumnya. Model akan disusun dengan rincian yang dapat dilihat pada tabel 3.19.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Fungsi Aktivasi | Optimizer | Fungsi loss |
| 1 | * Embedding (16) * LSTM/Bi-LSTM(64/fungsi aktivasi hidden *layer*) * Flatten * Dense(3/fungsi aktivasi output *layer*) | Adam | Categorical Crossentropy |
| 2 | * Embedding (16) * LSTM/Bi-LSTM(64/fungsi aktivasi hidden *layer*) * Flatten * Dense(3/fungsi aktivasi output *layer*) | RMSPROP | Categorical Crossentropy |

Tabel 3. 19 Susunan model berdasarkan optimizer yang akan diujikan

### Skenario Pengujian Model Kompleks

Pada skenario ini, akan diujikan model dengan arsitektur kompleks. Pada tahapan ini, akan ditambahkan beberapa *layer* ke dalam model seperti penambahan LSTM/Bi-LSTM *layer*, Dense *layer* serta penggunaan *dropout* *layer* untuk mengurangi overfitting yang ada. Dalam skenario ini akan mencari kembali parameter yang optimal diantaranya adalah arsitektur model dan optimizer model. Model akan dilatih selama 20 epoch menggunakan callback EarlyStopping dan memiliki validation split sebesar 0.1 untuk semua pengujian pada model kompleks.

#### Arsitektur Model

Model akan dilatih berdasarkan dua arsitektur model yang berbeda, yaitu model dengan LSTM *layer* dan model dengan Bi-LSTM *layer*. Akan ditambahkan beberapa dense *layer* dan *dropout* *layer* pada model. Untuk fungsi aktivasi, optimizer dan fungsi loss akan mengunakan hasil terbaik dari skenario pengujian model sederhana. Untuk detail arsitektur model, dapat dilihat pada tabel 3.20.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Model | Optimizer | Fungsi loss |
| 1 | * Embedding (128) * LSTM(128/relu/*Dropout* 0.2), * LSTM(128/relu/*Dropout* 0.2), * LSTM(128/relu/*Dropout* 0.2), * Flatten * Dense(512/relu) * *Dropout* (0.5) * Dense(3/softmax) | RMSPROP | Categorical Crossentropy |
| 2 | * Embedding (128) * Bi-LSTM(128/relu/*Dropout* 0.2), * Bi-LSTM(128/relu/*Dropout* 0.2), * Bi-LSTM(128/relu/*Dropout* 0.2), * Flatten * Dense(512/relu) * *Dropout*(0.5) * Dense(3/softmax) | RMSPROP | Categorical Crossentropy |

Tabel 3. 20 Arsitektur model kompleks berdasarkan *layer* model.

#### Optimizer Model

Model akan dilatih dengan dua variasi optimizer yang akan digunakan pada penelitian ini, yaitu optimizer Adam dan optimizer RMSPROP. Model akan disusun menggunakan arsitektur dan fungsi aktivasi model yang terbaik dari tahap percobaan sebelumnya. Model akan disusun dengan rincian yang dapat dilihat pada tabel 3.21.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Model | Optimizer | Fungsi loss |
| 1 | * Embedding (128) * LSTM/Bi-LSTM(128/relu/*Dropout* 0.2), * LSTM/Bi-LSTM(128/relu/*Dropout* 0.2), * LSTM/Bi-LSTM(128/relu/*Dropout* 0.2), * Flatten * Dense(512/relu) * *Dropout* (0.5) * Dense(3/softmax) | Adam | Categorical Crossentropy |
| 2 | * Embedding (128) * LSTM/Bi-LSTM(128/relu/*Dropout* 0.2), * LSTM/Bi-LSTM(128/relu/*Dropout* 0.2), * LSTM/Bi-LSTM(128/relu/*Dropout* 0.2), * Flatten * Dense(512/relu) * *Dropout* (0.5) * Dense(3/softmax) | RMSPROP | Categorical Crossentropy |

Tabel 3. 21 Arsitektur model berdasarkan optimizer

# BAB IV

**IMPLEMENTASI DAN ANALISIS 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 4.7.

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 khusus/tidak biasa, pembersihan karakter selain *alphanumeric*, 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 *retweet*, 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 gambar 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*.

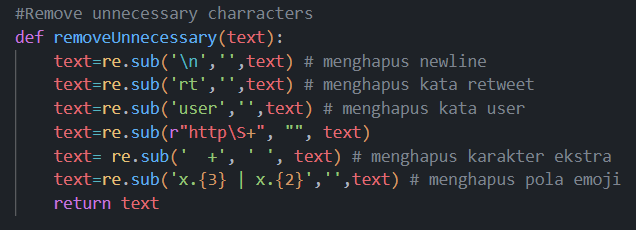
Text

Description automatically generated

Gambar 4. 10 Hasil *lowercasing*.

#### Pembersihan Karakter 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 kosong dan dapat dilihat implementasinya pada gambar 4.11 dan untuk hasilnya dapat dilihat pada gambar 4.12.



Gambar 4. 11 Implementasi Pembersihan karakter khusus

Text

Description automatically generated

Gambar 4. 12 Hasil pembersihan karakter khusus

#### 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 yang dapat dilihat implementasinya pada gambar 4.13 dan untuk hasilnya dapat dilihat pada gambar 4.14.

Text

Description automatically generated

Gambar 4. 13 Implementasi Remove *Non-alphanumeric.*

Text

Description automatically generated

Gambar 4. 14 Hasil Implementasi Remove *Non-alphanumeric*

#### *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 *stopwords* dari libraries nltk.corpus. Setelah itu, mengambil set *stopword* inggris dari modul *stopwords*. 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 *stopwords*. Untuk implementasi, dapat dilihat pada gambar 4.15 sampai dengan gambar 4.17.



Gambar 4. 15 Import modul *stopwords* dari libraries nltk.corpusText

Description automatically generated

Gambar 4. 16 Implementasi penghapusan *stopwords*

Text

Description automatically generated

Gambar 4. 17 Hasil implementasi penghapusan *stopwords*

#### Lemmatization

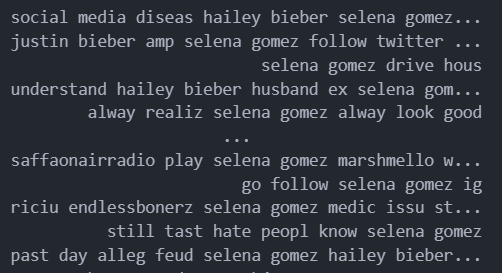
Lemmatization mengubah kata menjadi bentuk dasarnya. Lemmatization memastikan bahwa kata dasar milik bahasa. Untuk melakukan tahapan ini, diawali dengan melakukan import modul WordLemmatizer dari libraries nltk.stem dan kemudian menginisialisasi WordNetLemmatizer. Setelah diinisialisasi, maka memanggil fungsi lemmatize dari WordNetLemmatizer untuk setiap kata yang ada di dalam kalimat/data. Untuk detail implementasi dapat dilihat pada gambar 4.18 sampai dengan 4.20.



Gambar 4. 18 Import dan inisialisasi WordNetLemmatizer

Text

Description automatically generated

Gambar 4. 19 Implementasi lemmatization

Gambar 4. 20 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 4.21 dan gambar 4.22.



Gambar 4. 21 Implementasi Vader analisis sentimen

Text

Description automatically generated

Gambar 4. 22 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 4.23 dan gambar 4.24.

Text

Description automatically generated

Gambar 4. 23 Implementasi menghitung panjang setiap kalimat

A picture containing chart

Description automatically generated

Gambar 4. 24 Hasil implementasi dan visualisasinya.

Pada gambar 4.24, 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 4.25 sampai dengan 4.27.

A screenshot of a computer

Description automatically generated with medium confidence

Gambar 4. 25 Import libraries untuk melakukan tokenizing.

Text

Description automatically generated

Gambar 4. 26 Mempersiapkan elemen komponen penting tokenizing.



Gambar 4. 27 Contoh hasil implementasi tokenizing.

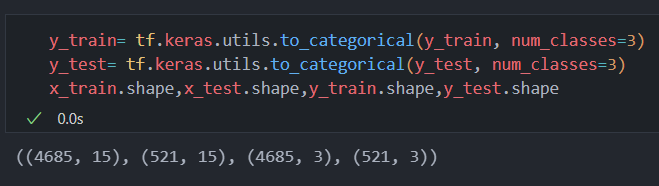
### Pembagian Data

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.



Gambar 4. 28 Implementasi pembagian data latih dan data uji

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 4.29.

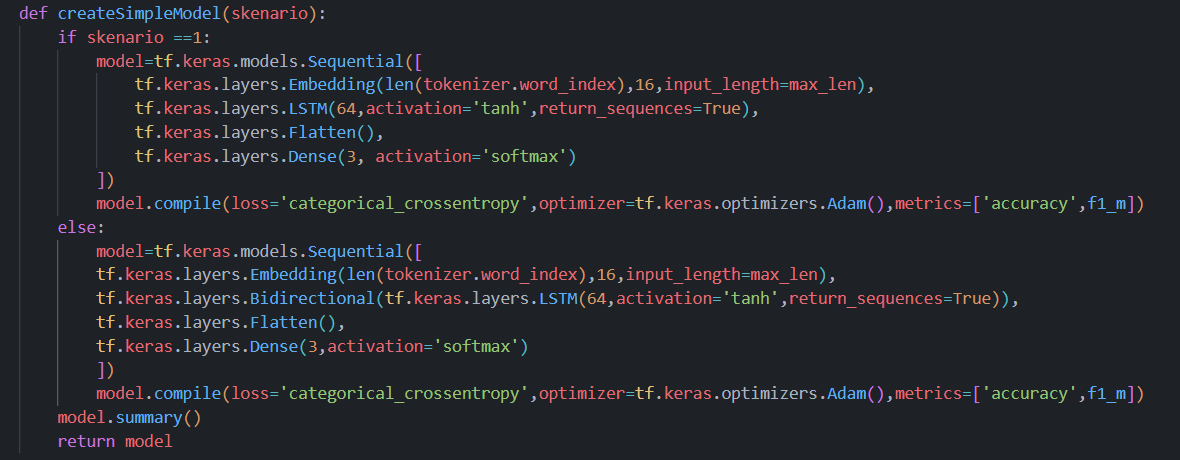


Gambar 4. 29 Implementasi one hot encoding

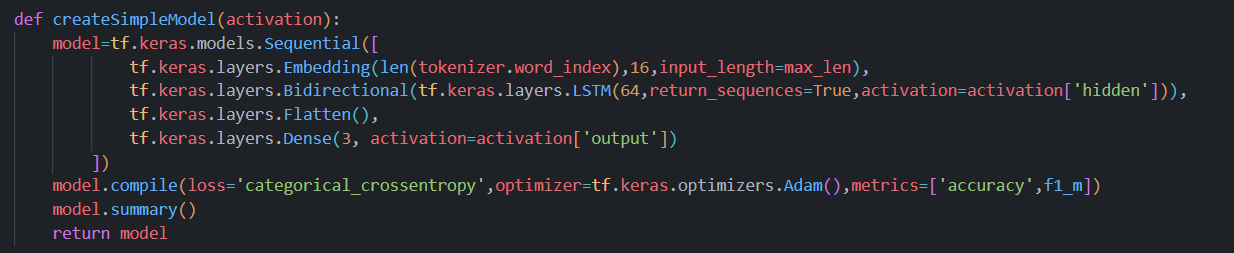
Setelah proses tersebut, maka data siap digunakan dalam tahap *training*.

### 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 4.30 sampai 4.39.



Gambar 4. 30 Implementasi penyusunan model sederhana berdasarkan arsitektur



Gambar 4. 31 Implementasi penyusunan model sederhana berdasarkan fungsi aktivasi

A picture containing text, screenshot, font

Description automatically generated

Gambar 4. 32 Implementasi training hasil penyusunan model sederhana berdasarkan arsitektur model

Text

Description automatically generated

Gambar 4. 33 Implementasi training hasil penyusunan model sederhana berdasarkan fungsi aktivasi

Text

Description automatically generated

Gambar 4. 34 Implementasi penyusunan model sederhana berdasarkan optimizer

A picture containing text, screenshot, font

Description automatically generated

Gambar 4. 35 Implementasi training hasil penyusunan model sederhana berdasarkan optimizer

A screen shot of a computer program

Description automatically generated with low confidence

Gambar 4. 36 Implementasi penyusunan model kompleks berdasarkan arsitektur model

A picture containing text, screenshot, font

Description automatically generated

Gambar 4. 37 Implementasi training hasil penyusunan model kompleks berdasarkan arsitektur model

A screen shot of a computer code

Description automatically generated with low confidence

Gambar 4. 38 Implementasi penyusunan model kompleks berdasarkan optimizer

A screen shot of a computer code

Description automatically generated with low confidence

Gambar 4. 39 Implementasi training hasil penyusunan model kompleks berdasarkan arsitektur optimizer

Setelah model dibuat, maka dapat melakukan tahapan *training* model dimana model akan dilatih selama 20 *epochs*, dan rasio validasi data *training* sebesar 0.1. 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 4.40.

Text

Description automatically generated

Gambar 4. 40 Implementasi *callback*

### Evaluasi

Evaluasi model akan dilihat dari hasil akurasi, loss dan f1-score dari model hasil training. Model akan ditinjau dari hasil evaluasi yang diimplementasian menggunakan fungsi evaluate dari model, setelah itu akan ditinjau dari segi grafik training model.

Model akan dievaluasi menggunakan data test yang telah dipisahkan dari dataset awal. Data test berukuran 10% dari dataset keseluruhan. Data test yang sudah dipisahkan labelnya, kemudian digunakan untuk tahapan evaluasi. Untuk detail implementasi, dapat dilihat pada gambar 4.41 sampai dengan gambar 4.43.

A screen shot of a computer

Description automatically generated with medium confidence

Gambar 4. 41 Implementasi dan hasil dari evaluasi model berdasarkan arsitektur model

A screen shot of a computer

Description automatically generated with medium confidence

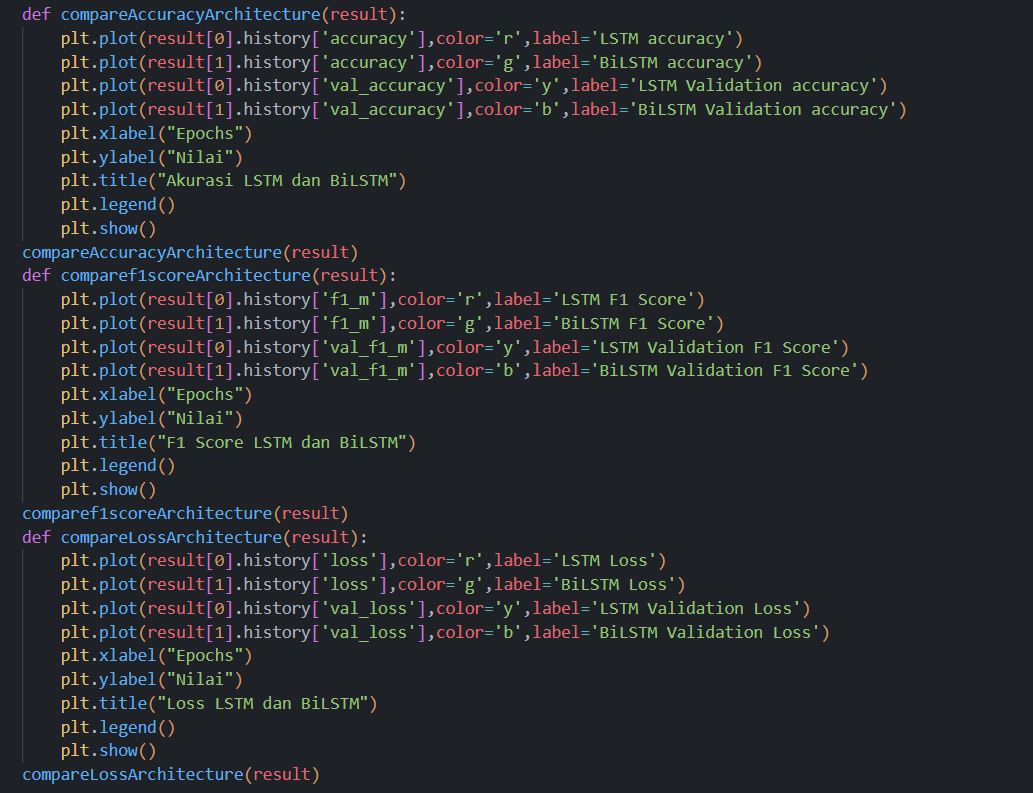
Gambar 4. 42 Implementasi dan hasil dari evaluasi model berdasarkan fungsi aktivasi

A screen shot of a computer code

Description automatically generated with low confidence

Gambar 4. 43 Implementasi dan hasil dari evaluasi model berdasarkan optimizer

Setelah dilakukan evaluasi, maka hasil training akan ditinjau melalui grafik yang memetakan akurasi, loss dan f1-score. Grafik untuk sumbu x merupakan nilai epoch dan grafik untuk sumbu y merupakan nilai dari akurasi, loss maupun f1-score. Untuk detail implementasi, dapat dilihat pada gambar 4.44 sampai dengan 4.48.



Gambar 4. 44 Implementasi grafik hasil training model berdasarkan arsitektur model

A screenshot of a computer

Description automatically generated with medium confidence

Gambar 4. 45 Implementasi grafik akurasi hasil training model berdasarkan fungsi aktivasi

A screenshot of a computer

Description automatically generated with medium confidence

Gambar 4. 46 Implementasi grafik loss hasil training model berdasarkan fungsi aktivasi

A screenshot of a computer

Description automatically generated with medium confidence

Gambar 4. 47 Implementasi grafik f1-score hasil training model berdasarkan fungsi aktivasi

Text

Description automatically generated

Gambar 4. 48 Implementasi grafik hasil training model berdasarkan optimizer

## 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 fokus, yaitu model dengan *layer* LSTM dan model dengan *layer* Bi-LSTM.

### Pengujian Model Sederhana

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

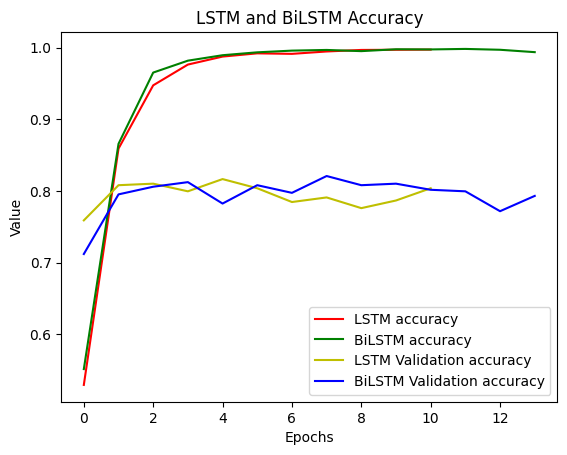
#### Pengujian Terhadap Arsitektur Model

Pada bagian ini, akan membandingkan antara arsitektur model dengan *layer* LSTM dengan model dengan *layer* Bi-LSTM. Model yang akan dibagun sederhana, dengan komposisi input *layer* berupa Embedding *layer*, hidden *layer* berupa satu *layer* LSTM ataupun Bi-LSTM, 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 crossentropy. 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 | 11 | 0.9279 | 78.31% | 78.68% |
| 2 | * Embedding (16) * Bi-LSTM(64/tanh) * Flatten * Dense(3/softmax) | Adam | Categorical | 14 | 0.7966 | 81.57% | 81.97% |

Tabel 4. 2 Hasil evaluasi training perbandingan arsitektur *layer* model.

Setelah dilakukan training terhadap dua model dengan parameter yang sama, dapat dilihat bahwa model dengan arsitektur *layer* Bi-LSTM 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 LSTM berhenti terlebih dahulu di epochs ke lima belas dan menghasilkan akurasi yang lebih rendah dibandingkan dengan model Bi-LSTM. Hal ini disebabkan karena Bi-LSTM 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 new product”. Bi-LSTM 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* Bi-LSTM sangatlah cocok dan terbukti lebih baik untuk mengklasifikasikan ujaran kebencian.



Gambar 4. 49 Akurasi dan validasi akurasi dari training model LSTM dan Bi-LSTM

Grafik di atas menunjukan bahwa kedua model yang dilatih masih mengalami overfitting yang lumayan besar. Selain itu, pada data validasi, terlihat bahwa grafik validasi di kedua model berjalan tidak semulus dengan grafik akurasi di kedua model. Overfitting merupakan kondisi dimana model kurang bisa mengenali data variasi baru dibandingkan dengan data yang telah digunakan saat proses training model. Model Bi-LSTM cenderung lebih unggul baik di akurasi, validasi akurasi maupun saat dilakukan evaluasi model yang hasilnya dapat dilihat pada tabel 4.2.

A picture containing text, diagram, plot, screenshot

Description automatically generated

Gambar 4. 50 Loss dan validasi loss dari training model LSTM dan Bi-LSTM

Dari grafik di atas, terlihat kedua model mengalami overfitting pada loss. Grafik validasi cenderung naik turun dan nilainya semakin membesar seiring dengan epoch yang berjalan. Model Bi-LSTM cenderung lebih rendah baik di loss, validasi loss maupun saat dilakukan evaluasi model yang hasilnya dapat dilihat pada tabel 4.2.

A picture containing text, screenshot, diagram, line

Description automatically generated

Gambar 4. 51 F1-score dan validasi f1-score dari training model LSTM dan Bi-LSTM

Dari grafik di atas, terlihat bahwa f1-score dari kedua model masih mengalami overfitting, akan tetapi tidak sejauh akurasi dan loss. Grafik validasi cenderung naik turun dan tidak konstan tidak seperti grafik f1-score. Dari nilai dan grafik f1-score yang ada, mengindikasikan bahwa model yang dimiliki masih memiliki nilai recall dan precision yang belum seimbang/harmonis. Model Bi-LSTM cenderung lebih unggul baik di loss, validasi loss maupun saat dilakukan evaluasi model yang hasilnya dapat dilihat pada tabel 4.2.

Model yang digunakan untuk tahapan evaluasi merupakan model dengan performa terbaik yang didapatkan dari hasil callback EarlyStopping, sehingga ditemukan model yang dilatih tidak sampai dengan 20 epoch karena model tersebut tidak menunjukan sebuah peningkatan performa untuk nilai validasi akurasi.

Dari percobaan arsitektur model ini antara model dengan *layer* LSTM dengan Bi-LSTM dapat dilihat bahwa Bi-LSTM unggul di akurasi, loss dan juga f1-score dibandingkan dengan model LSTM, dan hal ini dikarenakan Bi-LSTM 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 ataso berdasarkan model yang pada umumnya digunakan sesuai dengan kondisi yang ada. Model diuji menggunakan parameter yang sama, yaitu arsitektur Bi-LSTM, untuk fungsi loss menggunakan categorical crossentropy 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 | 9 | 0.5836 | 82.73% | 83.19% |
| 2 | * Relu (hidden) * Softmax (output) | Adam | Categorical | 7 | 0.4453 | 84.26% | 84.20% |
| 3 | * Sigmoid (hidden) * Sigmooid (output) | Adam | Categorical | 16 | 0.9002 | 79.46% | 71.46% |
| 4 | * Sigmoid (hidden) * Softmax (output) | Adam | Categorical | 14 | 0.8918 | 75.24% | 75.98% |
| 5 | * Tanh (hidden) * Sigmoid (output) | Adam | Categorical | 7 | 0.5462 | 81.77% | 77.45% |
| 6 | * Tanh (hidden) * Softmax (output) | Adam | Categorical | 7 | 0.5089 | 81.57% | 81.82% |

Tabel 4. 3 Hasil evaluasi training perbandingan fungsi aktifasi menggunakan *layer* Bi-LSTM

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 dengan 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 4.52 sampai dengan gambar 4.57.

A picture containing text, screenshot, line, diagram

Description automatically generated

Gambar 4. 52 Grafik perbandingan akurasi model berdasarkan fungsi aktivasi

A picture containing text, screenshot, diagram, line

Description automatically generated

Gambar 4. 53 Grafik perbandingan validasi akurasi model berdasarkan fungsi aktivasi

A picture containing text, screenshot, plot, line

Description automatically generated

Gambar 4. 54 Grafik perbandingan loss model berdasarkan fungsi aktivasi

A picture containing text, diagram, plot, line

Description automatically generated

Gambar 4. 55 Grafik perbandingan validasi loss model berdasarkan fungsi aktivasi

A picture containing text, screenshot, diagram, line

Description automatically generated

Gambar 4. 56 Grafik perbandingan f1-score model berdasarkan fungsi aktifasi

A picture containing text, diagram, line, plot

Description automatically generated

Gambar 4. 57 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. Model masih mengalami overfitting pada f1-score, akan tetapi tidak terlalu noisy dan jauh dibandingkan dengan akurasi model. Hal ini berarti model memiliki nilai precision dan recall yang seimbang. Sedangkan di bagian validasi akurasi, validasi loss serta validasi f1-score, terlihat beberapa perbedaan yang cukup berbeda satu sama lain. Model dengan hidden *layer* sigmoid 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.

Model yang digunakan untuk tahapan evaluasi merupakan model dengan performa terbaik yang didapatkan dari hasil callback EarlyStopping, sehingga ditemukan model yang dilatih tidak sampai dengan 20 epoch karena model tersebut tidak menunjukan sebuah peningkatan performa untuk nilai validasi akurasi.

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 optimizer Adam dengan RMSPROP. Pemilihan optimizer berdasarkan model yang pada umumnya digunakan sesuai dengan kondisi yang ada. Model diuji menggunakan parameter arsitektur Bi-LSTM, untuk fungsi loss menggunakan categorical crossentropy dan untuk fungsi aktifasi pada hidden *layer* menggunakan relu serta untuk output *layer* menggunakan softmax yang dilatih selama 20 epochs menggunakan *callback* EarlyStoping.

Setelah dilakukan tahapan training, didapatkan hasil yang dapat dilihat pada tabel 4.4.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Model | Optimizer | Fungsi loss | Early stopping | Loss | Accuracy | F1-score |
| 1 | * Embedding (16) * Bi-LSTM(64/relu) * Flatten * Dense(3/softmax) | Adam | Categorical | 13 | 0.6350 | 81.57% | 82.04% |
| 2 | * Embedding (16) * Bi-LSTM(64/relu) * Flatten * Dense(3/softmax) | RMSPROP | Categorical | 14 | 0.7578 | 84.45% | 84.71% |

Tabel 4. 4 Hasil evaluasi training model terhadap optimizer

Dari tabel di atas, terlihat bahwa model dengan optimizer RMSPROP lebih unggul dalam akurasi dan f1-score, akan tetapi lebih tinggi pada loss dibandingkan dengan model Adam. Untuk melihat lebih detail perbandingan hasil training dari semua model yang ada, dapat dilihat pada gambar 4.58 sampai dengan gambar 4.60.

A picture containing text, diagram, line, plot

Description automatically generated

Gambar 4. 58 Grafik akurasi dan validasi akurasi dari model berdasarkan optimizer

Grafik di atas menunjukan bahwa kedua model yang dilatih masih mengalami overfitting, walaupun tidak sebesar pada percobaan sebelumnya. Grafik akurasi training dari optimizer Adam terlihat lebih unggul dibandingkan dengan RMSPROP. Akan tetapi, Ketika dilakukan validasi pada saat training, didapatkan optimizer RMSPROP cenderung lebih dominan unggul dibandingkan optimizer Adam. Model dengan optimizer RMSPROP cenderung lebih unggul baik di akurasi, validasi akurasi maupun saat dilakukan evaluasi model yang hasilnya dapat dilihat pada tabel 4.4.

A picture containing text, diagram, plot, line

Description automatically generated

Gambar 4. 59 Grafik loss dan validasi loss dari model berdasarkan optimizer

Pada grafik tersebut, terlihat bahwa nilai loss yang dialami kedua model pada validasi, tidak konstan, dan pada nilai training model Adam lebih unggul terhadap RMSPROP, tetapi ketika validasi, model RMSPROP lebih dominan unggul lebih rendah dibandingkan model Adam. Kedua model masih mengalami overfitting pada nilai lossnya. Dari grafik di atas dan berdasarkan hasil evaluasi model pada tabel 4.4, terlihat bahwa model dengan optimizer Adam mendapatkan nilai loss yang cenderung lebih kecil.

A picture containing text, diagram, plot, line

Description automatically generated

Gambar 4. 60 Grafik f1-score dan validasi f1-score dari model berdasarkan optimizer

Grafik di atas menunjukan bahwa nilai dari f1-score di kedua model masih mengalami overfitting, walaupun tidak sebesar pada akurasi. Grafik pada validasi sudah Mulai konstan walaupun masih terdapat naik turun di beberapa epoch. Dari nilai dan grafik f1-score yang ada, mengindikasikan bahwa model yang dimiliki masih memiliki nilai recall dan precision yang belum seimbang/harmonis. Model dengan optimizer RMSPROP lebih unggul pada f1-score pada validasi dan evaluasi dibandingkan dengan model dengan optimizer Adam.

Model yang digunakan untuk tahapan evaluasi merupakan model dengan performa terbaik yang didapatkan dari hasil callback EarlyStopping, sehingga ditemukan model yang dilatih tidak sampai dengan 20 epoch karena model tersebut tidak menunjukan sebuah peningkatan performa untuk nilai validasi akurasi.

Dan untuk skenario sederhana pengujian optimizer, ditemukan bahwa optimizer RMSPROP menghasilkan model dengan akurasi,dan f1-score yang lebih baik dibandingkan dengan model yang menggunakan optimizer Adam, akan tetapi lebih tinggi pada loss. Hal ini dikarenakan ketika berhadapan dengan model yang tersusun dari arsitektur yang tidak rumit dan dataset yang cenderung sedikit, RMSPROP ditemukan lebih baik dalam beberapa kasus (Zhang, 2018). Karena model yang dibangun saat ini adalah model sederhana dan menggunakan dataset yang cukup sedikit, maka hasil penelitian untuk skenario ini didapatkan optimizer RMSPROP lebih unggul pada akurasi, loss serta f1-score di skenario ini. Akan tetapi, perlu diujikan kembali penggunaan optimizer pada skenario kompleks untuk mengetahui apakah hasil yang dihasilkan sama ataupun tidak mengingat RMSPROP bekerja baik pada model yang tidak kompleks.

### Pengujian Model Kompleks

Dalam tahap pengujian ini, akan dilakukan pengujian menggunakan arsitektur yang lebih kompleks lagi, dan akan dilakukan secara bertahap untuk membandingkan *layer*, optimizer serta training model dengan menambah epochs dan tanpa menggunakan callback EarlyStopping untuk arsitektur model untuk menemukan parameter model yang paling optimal untuk kasus ini. Untuk fungsi aktivasi yang digunakan pada *hidden layer* adalah relu dan untuk output *layer* adalah softmax. Model akan dilatih sebanyak 20 epoch, dengan validation\_split sebesar 0.1 serta menggunakan EarlyStopping *callback*.

#### Pengujian Terhadap Arsitektur Model

Pada bagian ini, akan membandingkan Kembali arsitektur model antara LSTM dengan Bi-LSTM. Model yang dibangun adalah model kompleks yang terdiri dari komposisi Embedding *layer*, tiga *layer* LSTM maupun Bi-LSTM, Dense *layer* sebagai hidden *layer* dan output *layer*. Jumlah unit tiap *layer* ditambah lebih banyak. Optimizer yang digunakan ialah RMSPROP dan untuk fungsi loss yang digunakan ialah categorical crossentropy. Setelah dilakukan percobaan, didapatkan hasil yang dapat dilihat pada tabel 4.5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Model | Optimizer | Fungsi loss | Early stopping | Loss | Accuracy | F1-score |
| 1 | * Embedding (128) * LSTM(128/relu/*Dropout* 0.2), * LSTM(128/relu/*Dropout* 0.2), * LSTM(128/relu/*Dropout* 0.2), * Flatten * Dense(512/relu) * *Dropout* (0.5) * Dense(3/softmax) | RMSPROP | Categorical | 14 | 0.7413 | 86.95% | 86.70% |
| 2 | * Embedding (128) * Bi-LSTM(128/relu/*Dropout* 0.2), * Bi-LSTM(128/relu/*Dropout* 0.2), * Bi-LSTM(128/relu/*Dropout* 0.2), * Flatten * Dense(512/relu) * *Dropout*(0.5) * Dense(3/softmax) | RMSPROP | Categorical | 9 | 0.3686 | 89.44% | 89.16% |

Tabel 4. 5 Hasil training model terhadap model kompleks

Dari tabel di atas, terihat bahwa model kompleks yang tersusun dari *layer* Bi-LSTM unggul baik dalam loss, akurasi maupun f1-score. Hal ini membuktikan bahwa dalam kondisi arsitektur yang berbeda pun, Bi-LSTM lebih dapat memberikan hasil yang optimal terhadap dataset dan kasus yang ada karena Bi-LSTM dilatih secara ke depan dan ke belakang sehingga dapat mengenali konteks suatu data serta dapat mengatasi ketergantungan data jangka panjang.



Gambar 4. 61 Grafik akurasi dan validasi akurasi hasil training model terhadap model kompleks

Dari grafik di atas, terlihat bahwa akurasi yang dihasilkan model pada skenario kompleks ini meningkat drastis, walaupun masih terdapat overfitting yang tidak terlalu besar dibandingkan dengan percobaan pada skenario sederhana. Hal itu dikarenakan penggunaan *dropout* *layer* pada skenario ini yang bertugas memungkinkan overfitting berkurang. Model Bi-LSTM cenderung lebih unggul pada akurasi, validasi akurasi serta evaluasi dibandingkan dengan model LSTM.

A picture containing text, diagram, plot, line

Description automatically generated

Gambar 4. 62 Grafik loss dan validasi loss hasill training model terhadap model kompleks

Pada grafik di atas, terlihat bahwa loss untuk kedua model sudah relatif menurun dengan signifikan, dan model Bi-LSTM lebih rendah nilai lossnya dibandingkan dengan LSTM. Akan tetapi, pada saat memasuki validasi, terlihat bahwa kedua model mengalami overfitting, terutama validasi loss model Bi-LSTM yang nilainya naik secara signifikan. Walaupun demikiaan, ketika memasuki fase evaluasi, loss dari model Bi-LSTM cenderung lebih rendah dan lebih baik dibandingkan dengan model LSTM.

A picture containing text, diagram, line, screenshot

Description automatically generated

Gambar 4. 63 Grafik f1-score dan validasi f1-score hasil training terhadap model kompleks

Dari grafik di atas, terlihat bahwa nilai f1-score meningkat seiring dengan meningkatnya nilai akurasi. Selain itu, f1-score mengalami overfitting yang tidak sebesar akurasi dan nilainya yang tinggi menandakan bahwa nilai recall dan precision memiliki proporsi yang hampir sama ataupun dapat disebut harmonis.

Model yang digunakan untuk tahapan evaluasi merupakan model dengan performa terbaik yang didapatkan dari hasil callback EarlyStopping, sehingga ditemukan model yang dilatih tidak sampai dengan 20 epoch karena model tersebut tidak menunjukan sebuah peningkatan performa untuk nilai validasi akurasi.

Dari percobaan di atas, terlihat bahwa pada skenario kompleks dengan parameter yang sama, model dengan *layer* Bi-LSTM lebih unggul dari segi akurasi, loss dan f1-score dibandingkan dengan model LSTM. Hal ini dikarenakan model Bi-LSTM dapat mengatasi permasalahan konteks dan ketergantungan data jangka panjang, dibandingkan LSTM yang hanya bisa mengatasi ketergantungan jangka panjang (Zhou, Long, & Ou).

#### Pengujian Terhadap Optimizer Model

Pada skenario ini, akan diujikan model kompleks menggunakan optimizer Adam dengan model kompleks menggunakan optimizer RMSPROP. Pengujian ini ditujukan untuk memeriksa apakah pada saat model yang disusun lebih kompleks, kinerja dari optimizer akan berbeda dibandingkan dengan skenario pengujian yang ada pada tahap sebelumnya. Model akan disusun menggunakan arsitektur Bi-LSTM hasil skenario sebelumnya, fungsi loss yang akan digunakan ialah categorical crossentropy, dan fungsi aktivasi untuk hidden *layer* ialah relu, sedangkan untuk output *layer* menggunakan softmax. Setelah dilakukan training, didapatkanlah hasil yang dapat dilihat pada tabel 4.6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Model | Optimizer | Fungsi loss | Early stopping | Loss | Accuracy | F1-score |
| 1 | * Embedding (128) * Bi-LSTM(128/relu/*Dropout* 0.2), * Bi-LSTM(128/relu/*Dropout* 0.2), * Bi-LSTM(128/relu/*Dropout* 0.2), * Flatten * Dense(512/relu) * *Dropout*(0.5) * Dense(3/softmax) | Adam | Categorical | 11 | 0.6145 | 89.64% | 89.90% |
| 2 | * Embedding (128) * Bi-LSTM(128/relu/*Dropout* 0.2), * Bi-LSTM(128/relu/*Dropout* 0.2), * Bi-LSTM(128/relu/*Dropout* 0.2), * Flatten * Dense(512/relu) * *Dropout*(0.5) * Dense(3/softmax) | RMSPROP | Categorical | 7 | 0.3635 | 88.48% | 88.88% |

Tabel 4. 6 Hasil training model kompleks terhadap optimizer

Dari hasil pada tabel di atas, terlihat bahwa terjadi perubahan dimana model dengan optimizer Adam lebih baik dibandikan dengan model dengan optimizer RMSPROP pada akurasi dan f1-score hasil training, untuk loss model RMSPROP lebih unggul. Pada tahapan ini, didapatkan peningkatan akurasi dan f1-score yang sangat signifikan pada kedua model dimana kedua model dapat menghasilkan akurasi dan f1-score hampir 90%.

A picture containing text, diagram, line, screenshot

Description automatically generated

Gambar 4. 64 Hasil akurasi training model kompleks berdasarkan optimizer

Dari grafik di atas, terlihat bahwa akurasi yang dihasilkan model pada skenario kompleks ini meningkat drastis, walaupun masih terdapat overfitting yang tidak terlalu besar dibandingkan dengan percobaan pada skenario sederhana. Hal itu dikarenakan penggunaan *dropout* *layer* pada skenario ini yang bertugas memungkinkan overfitting berkurang. Jika ditinjau lagi dari percobaan skenario sederhana pada optimizer, RMSPROP lebih unggul dibandingkan dengan Adam, tetapi pada skenario ini yang terjadi adalah sebaliknya dimana model dengan optimizer Adam lebih unggul pada akurasi, validasi akurasi serta pada evaluasi yang telah dijalankan.

A picture containing text, diagram, plot, line

Description automatically generated

Gambar 4. 65 Hasil loss training model kompleks terhadap optimizer

Pada grafik di atas, terlihat bahwa nilai loss cenderung grafiknya tidak konsisten dibandingkan dengan percobaan sebelumnya. Nilai loss yang ada meningkat dan lebih overfitting dibandingkan dengan skenario percobaan pada arsitektur model pada skenario kompleks. Model dengan optimizer Adam nilai lossnya lebih tinggi dibandingkan dengan model dengan optimizer RMSPROP.

A picture containing text, screenshot, diagram, line

Description automatically generated

Gambar 4. 66 Hasil f1-score training model kompleks terhadap optimizer

Dari grafik di atas, terlihat bahwa kedua model mengalami overfitting, tetapi nilainya semakin mengecil dibandingkan dengan f1-score pada percobaan sebelumnya. Dari nilai dan grafik f1-score yang ada, mengindikasikan bahwa model yang dimiliki memiliki nilai recall dan precision yang cenderung sudah membaik dan sudah mulai seimbang/harmonis. Model dengan optimizer Adam cenderung unggul dan grafiknya cenderung mulai konstan dibandingkan dengan model dengan optimizer RMSPROP.

Model yang digunakan untuk tahapan evaluasi merupakan model dengan performa terbaik yang didapatkan dari hasil callback EarlyStopping, sehingga ditemukan model yang dilatih tidak sampai dengan 20 epoch karena model tersebut tidak menunjukan sebuah peningkatan performa untuk nilai validasi akurasi.

Dari percobaan ini, didapatkan bahwa model dengan optimizer Adam lebih unggul secara akurasi, loss dan f1-score dibandingkan dengan model yang menggunakan optimizer RMSPROP. Hal ini dikarenakan ketika menghadapi model dengan arsitektur yang lebih kompleks, RMSPROP kurang baik dalam meningkatkan performa, berkebalikan dengan yang telah dilakukan pada percobaan skenario sederhana berdasarkan optimizer.

# 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 Bi-LSTM dapat melakukan analisis sentimen ujaran kebencian di Twitter.
2. Setelah diuji menggunakan dua skenario, dapat diketahui bahwa model dengan *layer* Bi-LSTM lebih unggul terhadap model dengan *layer* LSTM saja di skenario sederhana dengan akurasi 82.34% dan skenario kompleks dengan akurasi 89.44 %.
3. Dari penelitian yang telah dilakukan, maka ditemukanlah model *deep learning* yang paling optimal yang dapat menghasilkan akurasi sebesar 89.64%, f1-score sebesar 89.90% dan untuk nilai lossnya sebesar 0.6145. Parameter dari model tersebut adalah menggunakan model arsitektur Bi-LSTM, fungsi aktivasi pada *hidden layer* menggunakan relu, fungsi aktivasi *output layer* menggunakan softmax, serta menggunakan *optimizer* adam.
4. Dari setiap scenario yang telah dijalankan, maka ditemukan hasil untuk setiap parameter yang telah diujikan sebagai berikut:
   1. Arsitektur layer Bi-LSTM optimal untuk digunakan pada kasus analisa sentiment ujaran kebencian Twitter baik pada scenario sederhana maupun kompleks karena dapat memahami konteks data lebih luas.
   2. Fungsi aktivasi yang optimal untuk digunakan pada *hidden* *layer* adalah relu dan untuk output *layer* adalah softmax. Relu dapat mengatasi gradien yang hilan (*vanishing gradient descent*) serta softmax cocok untuk digunakan pada kasus klasifikasi *multiclass.*
   3. Optimizer Adam dan RMSPROP tidaklah jauh berbeda dalam memberikan efek terhadap model. Optimizer RMSPROP lebih unggul digunakan ketika arsitektur model sederhana dan tidak menggunakan dataset yang terhitung banyak, sedangkan untuk Adam lebih unggul ketika arsitektur model lebih kompleks dan data yang digunakan cenderung lebih banyak.
5. 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 *retweet*.
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

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

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (n.d.). ImageNet Classification with Deep Convolutional. Retrieved from http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

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

Zhang, S. Z. (2018). An empirical study of Adam and its variants for deep learning.

Zhou, K., Long, F., & Ou, W. (n.d.). Sentiment Analysis of Text Based on Bidirectional LSTM With Multi-Head Attention. Retrieved from https://ieeexplore.ieee.org/abstract/document/8845615

# Daftar Istilah / Glosarium

|  |  |  |
| --- | --- | --- |
| No | Istilah | Penjelasan |
| 1 | deep learning | metode pembelajaran mesin yang menggunakan jaringan neural dengan banyak lapisan untuk memproses data dan melakukan prediksi. |
| 2 | lstm | jenis arsitektur jaringan neural yang umumnya digunakan untuk memproses data yang memiliki urutan atau seri seperti teks dan suara. |
| 3 | bidirectional lstm | Bidirectional LSTM: varian LSTM yang memproses data dalam kedua arah, mundur dan maju, sehingga mampu menangkap konteks dari kedua arah. |
| 4 | optimizer | sebuah algoritme yang digunakan untuk mengoptimalkan parameter-model dalam pembelajaran mesin. |
| 5 | adam | optimizer pembelajaran mesin yang umum digunakan, dengan singkatan Adaptive Moment Estimation. |
| 6 | rmsprop | optimizer pembelajaran mesin yang menggunakan rata-rata bergerak dari gradien kuadrat untuk memperbarui parameter-model. |
| 7 | activation function/fungsi aktivasi | fungsi matematis yang diterapkan pada keluaran suatu lapisan dalam jaringan neural untuk memungkinkan non-linearitas. |
| 8 | relu | fungsi aktivasi yang paling umum digunakan dalam jaringan neural, menghasilkan keluaran 0 jika masukan negatif dan masukan itu sendiri jika positif. |
| 9 | tan h | fungsi aktivasi yang menghasilkan keluaran dalam rentang -1 hingga 1, umumnya digunakan dalam arsitektur jaringan neural klasik seperti NLP. |
| 10 | softmax | fungsi aktivasi yang menghasilkan keluaran dalam rentang 0 hingga 1, umumnya digunakan dalam lapisan akhir jaringan neural klasik untuk klasifikasi. |
| 11 | loss | ukuran yang digunakan untuk mengevaluasi seberapa baik model memprediksi label yang benar dari data pelatihan. |
| 12 | categorical crossentropy | salah satu jenis loss function yang digunakan untuk masalah klasifikasi multikelas. |
| 13 | mean squared error | Mean Squared Error (MSE) adalah metrik yang umum digunakan untuk mengevaluasi kinerja model regresi. |
| 14 | tweets | pesan singkat yang ditulis di Twitter. |
| 15 | influencer | seseorang yang memiliki pengaruh besar dalam media sosial. |
| 16 | hate speech | bahasa atau tindakan yang merendahkan atau mendiskreditkan seseorang atau kelompok berdasarkan ras, agama, gender, atau orientasi seksual. |
| 17 | natural language processing | cabang ilmu komputer yang berfokus pada pengolahan bahasa manusia oleh mesin. |
| 18 | RNN (recurrent neural network) | jenis arsitektur jaringan neural yang digunakan untuk memproses data berurutan. |
| 19 | support vector machine | metode pembelajaran mesin yang digunakan untuk klasifikasi dan regresi. |
| 20 | random forest | metode pembelajaran mesin yang menggunakan beberapa pohon keputusan untuk meningkatkan akurasi dan mencegah overfitting. |
| 21 | naive bayes | metode pembelajaran mesin yang berdasarkan pada teorema Bayes untuk klasifikasi. |
| 22 | decision tree | metode pembelajaran mesin yang menggunakan pohon untuk membuat keputusan berdasarkan fitur data. |
| 23 | convolutional neural network | jenis arsitektur jaringan neural yang digunakan untuk memproses data spasial seperti gambar. |
| 24 | k nearest neighboor | k nearest neighbor adalah metode dalam machine learning yang digunakan untuk mengklasifikasikan suatu data dengan cara membandingkan data tersebut dengan kumpulan data yang sudah ada di dalam dataset. Klasifikasi dilakukan dengan mencari jarak antara data baru dengan data yang sudah ada, dan memilih k data terdekat sebagai tetangga terdekat. |
| 25 | fold | Fold dalam cross-validation adalah bagian dari dataset yang dipilih untuk dijadikan sebagai data uji. Dataset akan dibagi menjadi beberapa fold, di mana setiap fold akan diambil bergantian sebagai data uji dan data latih. |
| 26 | cross validation | Cross-validation adalah metode dalam machine learning untuk mengevaluasi performa model pada dataset yang telah ada. Dataset akan dibagi menjadi beberapa fold, di mana setiap fold akan diambil bergantian sebagai data uji dan data latih. Setelah proses dilakukan pada setiap fold, performa dari model akan dihitung dan diambil rata-rata dari hasil evaluasi pada setiap fold. |
| 27 | GRU (*gate* recurrent unit) | tipe dari Recurrent Neural Network (RNN) yang memiliki *gate* untuk mengontrol aliran informasi. GRU digunakan untuk memproses data sekuensial pada NLP (Natural Language Processing) seperti teks dan audio. |
| 28 | feature extraction | Feature extraction adalah proses untuk mengubah data mentah menjadi fitur-fitur yang dapat digunakan dalam proses klasifikasi atau regresi pada machine learning. |
| 29 | word embedding | metode untuk merepresentasikan kata-kata dalam bentuk vektor pada machine learning. Vektor ini merepresentasikan makna kata dan dapat digunakan pada proses NLP seperti klasifikasi teks dan sentiment analysis. |
| 30 | machine learning | salah satu cabang dari Artificial Intelligence (AI) yang bertujuan untuk mengembangkan algoritma yang dapat belajar dari data dan dapat membuat keputusan atau prediksi secara otomatis. |
| 31 | input layer | Input layer adalah layer pertama dari suatu neural network yang menerima input data dari dataset. |
| 32 | hidden layer | Hidden layer adalah layer yang berada di antara input layer dan output layer pada neural network. Hidden layer dapat melakukan proses transformasi pada data. |
| 33 | output layer | layer terakhir dari suatu neural network yang menghasilkan output berupa hasil prediksi atau klasifikasi pada dataset. |
| 34 | dense layer | layer neural network dimana setiap neuron dari layer sebelumnya terhubung dengan setiap neuron di layer saat ini. |
| 35 | embedding layer | layer pada neural network yang digunakan untuk membuat vektor representasi untuk data input. Embedding layer digunakan pada data dengan tipe data kategori seperti teks. |
| 36 | flatten layer | layer pada neural network yang digunakan untuk mengubah data input menjadi bentuk satu dimensi sebelum dioperasikan pada layer selanjutnya. |
| 37 | backpropagation | proses yang digunakan pada neural network untuk menghitung gradien dari setiap parameter dalam model dengan menggunakan chain rule. |
| 38 | multilayer | neural network dengan lebih dari satu hidden layer |
| 39 | output | hasil akhir dari proses klasifikasi atau regresi pada dataset |
| 40 | input | data mentah yang akan diolah |
| 41 | dropout | eknik regularisasi yang digunakan dalam machine learning untuk mencegah overfitting. Dropout secara acak mengabaikan beberapa unit (neuron) dalam jaringan saraf selama proses pelatihan, sehingga mencegah ketergantungan yang terlalu besar pada unit tertentu dan meningkatkan generalisasi model. |
| 42 | training | Proses mengajari model machine learning dari data pelatihan. |
| 43 | lowercasing | Proses mengubah semua huruf dalam teks menjadi huruf kecil. |
| 44 | stemming | Proses penghapusan imbuhan dalam kata untuk menghasilkan bentuk kata dasar atau akar. |
| 45 | stopword removal | Proses penghapusan kata-kata umum yang tidak memiliki arti khusus dalam bahasa tertentu (seperti 'the', 'a', 'an', 'dan' dalam bahasa Inggris) dari teks yang diolah. |
| 46 | non alphanumeric | Karakter yang tidak termasuk dalam huruf atau angka. |
| 47 | lemmatization | Proses pengubah kata-kata dalam bentuk kata kerja atau kata sifat ke bentuk dasar atau kata benda. |
| 48 | web scrapping | Proses ekstraksi data dari halaman web secara otomatis menggunakan bot atau skrip. |
| 49 | retweet | Tindakan mengirim ulang tweet orang lain di Twitter. |
| 50 | repost | Tindakan mengirim ulang konten orang lain di platform sosial media. |
| 51 | alphanumeric | Karakter yang termasuk dalam huruf atau angka. |
| 52 | stopword | Kata-kata umum yang tidak memiliki arti khusus dalam bahasa tertentu dan dihapus dari teks saat memproses. |
| 53 | open source | Software yang lisensinya memungkinkan orang untuk melihat, mengedit, dan mendistribusikan kode sumbernya secara bebas. |
| 54 | tokenizing | Proses memecah teks menjadi bagian-bagian yang lebih kecil, disebut token, seperti kata atau frase. |
| 55 | one hot encoded | Representasi numerik dari data kategorikal, di mana setiap nilai kategori diwakili oleh vektor biner tunggal. |
| 56 | forget *gate* | Komponen dalam model GRU (*Gate* Recurrent Unit) yang membantu model melupakan informasi yang tidak lagi relevan dari waktu sebelumnya. |
| 57 | input *gate* | Komponen dalam model GRU (*Gate* Recurrent Unit) yang membantu model menentukan informasi mana yang akan disimpan dalam memori jangka pendek. |
| 58 | output *gate* | Komponen dalam model GRU (*Gate* Recurrent Unit) yang membantu model mengambil informasi dari memori jangka pendek dan menghasilkan output. |
| 59 | *cell state* | Komponen dalam model LSTM (Long Short-Term Memory) yang bertanggung jawab untuk memelihara informasi jangka panjang dalam jaringan. |
| 60 | modelling | Proses pembuatan model yang mewakili ketergantungan dalam data. |
| 61 | epochs | Satuan waktu dalam pelatihan model machine learning, yang mengacu pada satu iterasi melalui seluruh dataset pelatihan. |
| 62 | validation split | Pembagian data pelatihan menjadi dua bagian, di mana satu bagian digunakan untuk pelatihan model dan bagian lainnya digunakan untuk validasi. |
| 63 | callback | Fungsi yang dipanggil oleh model machine learning selama proses pelatihan, seperti saat mencapai kondisi berhenti dini (early stopping). |
| 64 | early stopping | teknik dalam pembelajaran mesin di mana pelatihan model dihentikan sebelum mencapai konvergensi penuh untuk mencegah overfitting. Penghentian dini ini dilakukan berdasarkan perubahan kriteria evaluasi yang diamati pada set validasi. |
| 65 | patience | parameter dalam teknik early stopping yang menentukan berapa banyak iterasi pelatihan yang diizinkan tanpa perbaikan dalam kriteria evaluasi sebelum pelatihan dihentikan. Jika patience diatur terlalu rendah, pelatihan dapat dihentikan terlalu cepat sebelum model mencapai konvergensi penuh. Jika diatur terlalu tinggi, pelatihan dapat berlangsung terlalu lama, menyebabkan pemborosan waktu komputasi. |
| 66 | testing | tahap akhir dari pembelajaran mesin di mana model yang telah dilatih dievaluasi pada set data yang belum pernah dilihat sebelumnya untuk mengukur kinerjanya. Hasil pengujian digunakan untuk memperkirakan kinerja model pada data yang belum pernah dilihat sebelumnya. |
| 67 | labelling | proses memberikan label pada data, yaitu memberikan kategori atau nilai target pada setiap contoh data dalam set pelatihan. |
| 68 | forward | istilah dalam jaringan saraf yang mengacu pada arus data dari input ke output melalui berbagai lapisan jaringan. |
| 69 | backward | istilah dalam jaringan saraf yang mengacu pada algoritma pelatihan yang menggunakan backpropagation untuk menghitung gradien fungsi kesalahan terhadap bobot setiap neuron dalam jaringan. |
| 70 | stochastic gradient descent | algoritma optimisasi yang digunakan dalam pelatihan model yang menghitung gradien fungsi kesalahan terhadap bobot dalam jaringan dan memperbarui bobot dengan cara menuruni gradien. |
| 71 | noise | sinyal acak yang terdapat dalam data yang dapat memengaruhi kemampuan model untuk mempelajari pola dan tren yang relevan dalam data. Teknik preprocessing seperti filtrasi atau normalisasi dapat digunakan untuk mengurangi noise dalam data. |

# 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 |
|  | Pengumpulan data |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |