IDENTIFIKASI KONTEN NEGATIF PADA TWITTER DENGAN *DEEP LEARNING*

Disusun Oleh:

Nama : Anggara Chandra Dwinata

NIM : 17523030

PROGRAM STUDI INFORMATIKA–PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

IDENTIFIKASI KONTEN NEGATIF PADA TWITTER DENGAN *DEEP LEARNING*

TUGAS AKHIR

Disusun Oleh:

N a m a : Anggara Chandra Dwinata
NIM : 17523030

Yogyakarta, 04 Januari 2023

Pembimbing,

(Chanifah Indah Ratnasari, S.Kom., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

IDENTIFIKASI KONTEN NEGATIF PADA TWITTER DENGAN *DEEP LEARNING*

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Teknik Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 04 Januari 2023

Tim Penguji

Chanifah Indah Ratnasari, S.Kom., M.Kom.

Anggota 1

Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D.

Anggota 2

Septia Rani, S.T., M.Cs.

Mengetahui,

Ketua Program Studi Teknik Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Anggara Chandra Dwinata

NIM : 17523030

Tugas akhir dengan judul:

IDENTIFIKASI KONTEN NEGATIF PADA TWITTER DENGAN *DEEP LEARNING*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 04 Januari 2023

(Anggara Chandra Dwinata)

HALAMAN PERSEMBAHAN

Setiap pagi saya berterima kasih kepada Allah SWT atas bantuannya. Setiap pagi, saya berterima kasih pada diri sendiri karena telah menjadi diri saya. Saya berterima kasih juga karena telah mendukung dan selalu bersama saya, apapun yang terjadi. Waktu adalah hal yang paling berharga dalam hidup kita, orang yang rela mengorbankan waktu untuk orang lain harus dihormati dan dihargai.

Saya persembahkan skripsi ini kepada ayah, ibu, dan kakak saya, dan mereka telah membantu banyak saya dalam menyelesaikan perkuliahan ini. Skripsi ini adalah hadiah kecil untuk keluarga saya. Tidak lupa kepada teman-teman saya yang selalu mendukung saya dalam segala kondisi. Saya berjanji untuk tidak menyia-nyiakan segalanya, dan berharap melakukan yang terbaik untuk setiap kepercayaan yang saya dapatkan. Saya akan menjadikan diri saya yang terbaik.

HALAMAN MOTO

"Nilai akhir dari proses pendidikan, sejatinya terekapitulasi dari keberhasilannya menciptakan perubahan pada dirinya dan lingkungan. Itulah fungsi daripada pendidikan yang sesungguhnya" - Lenang Manggala

"Tujuan pendidikan itu untuk mempertajam kecerdasan, memperkukuh kemauan serta memperhalus perasaan" - Tan Malaka

"Pendidikan mempunyai akar yang pahit, tapi buahnya manis" - Aristoteles

"Menyia-nyiakan waktu lebih buruk dari kematian. Karena kematian memisahkanmu dari dunia, sementara menyia-nyiakan waktu memisahkanmu dari Allah" - Ali bin Abi Thalib

"Allah tidak akan membeban seseorang melainkan sesuai kesanggupannya" - QS Al-Baqarah: 286

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi ini dengan judul **"IDENTIFIKASI KONTEN NEGATIF PADA TWITTER DENGAN *DEEP LEARNING*"**. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Komputer Jurusan Informatika pada Fakultas Teknologi Industri Universitas Islam Indonesia. Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan skripsi ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga penulis yang memberikan dukungan, doa, dan kekuatan.
2. Bapak Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
3. Bapak Prof., Dr., Ir. Hari Purnomo M.T, selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Hendrik, ST., M.Eng., selaku Ketua Jurusan Informatika Universitas Islam Indonesia.
5. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Program Studi Informatika Universitas Islam Indonesia.
6. Bapak Ahmad Fathan Hidayatullah, S.T., M.Cs., yang telah membantu penulis dalam menyelesaikan tugas akhir ini.
7. Chanifah Indah Ratnasari, S.Kom., M.Kom., selaku dosen pembimbing.
8. Bapak Hartono selaku pemilik kos tempat tinggal penulis selama kuliah di Universitas Islam Indonesia.
9. Semua pihak yang secara langsung maupun tidak langsung memberikan bantuan dalam bentuk apapun.

Untuk semua bantuan yang diberikan, sebagai penulis, saya ingin mengucapkan terima kasih. Jika saya mengerjakan tugas akhir ini, saya tidak akan lupa meminta maaf kesalahan atau kekurangan. Penulis berharap artikel ini bermanfaat bagi pembaca dan dapat mengembangkan penelitian ini di masa yang mendatang.

Yogyakarta, 04 Januari 2023

(Anggara Chandra Dwinata)

SARI

Media sosial telah menjadi media komunikasi antara satu pengguna dengan pengguna lainnya, salah satunya Twitter. Namun, tidak semua orang yang menggunakan Twitter dengan benar dan bijak. Banyak orang Indonesia yang menulis kata-kata berbau SARA (suku, agama, ras, dan antargolongan), atau mengeluarkan kalimat yang tidak pantas dan menyinggung seperti menulis nama binatang, alat kelamin, dan lain-lain. Oleh karena itu, perlu dikenali dengan melihat konteks kalimat secara keseluruhan. Pada penelitian ini peneliti, akan melakukan klasifikasi menggunakan metode CNN dan LSTM dengan ekstraksi fitur TF-IDF. TF-IDF digunakan untuk merubah *tweet* berbentuk teks yang akan ditampilkan ke dalam bentuk *vector,* sedangkan metode CNN dan LSTM digunakan untuk mendapatkan model terbaik untuk klasifikasi *tweet* sehingga dapat mengidentifikasi *tweet* kedalam tiga buah kategori yaitu netral (tidak mengandung konten negatif), kata kasar, dan konten pornografi. Berdasarkan hasil penelitian yang dilakukan, diperoleh bahwa metode CNN dan LSTM memiliki performa klasifikasi yang baik karena dari evaluasi *epoch* yang digunakan menghasilkan tingkat akurasi sebesar 100%, hal ini dapat dilihat pada *epoch* ke 97-100 dan hasil *confusion matrix* dengan rata-rata *precision* untuk kelas netral sebesar 0.99, *recall* sebesar 0.99, dan *f1-score* sebesar 0.99.

Kata kunci: *Tweet*, *LSTM*, *CNN, Deep Learning*

GLOSARIUM

CNN (*Convolutional Neural Network*)	jaringan saraf tiruan yang memberi umpan balik di mana jaringan saraf mempertahankan struktur hierarki dengan mempelajari representasi fitur internal dan menggeneralisasi fitur-fitur dalam masalah gambar secara umum
Confusion Matrix	metode untuk menguji algoritma
Dataset	sekumpulan data yang digunakan pada penelitian
Epoch	jumlah tahapan pelatihan untuk seluruh data
Klasifikasi	rangkaian proses pengkategorian terhadap sekumpulan dokumen menjadi beberapa kelas
Labelling	proses untuk melabelkan data
LSTM (*Long Short-Term Memory*)	merupakan bagian dari *recurrent neural network* (RNN) pada *deep learning*
Preprocessing	salah satu langkah untuk menghilangkan masalah yang dapat menganggu hasil pengolahan data
Python	bahasa pemrograman
TF-IDF	metode untuk menghitung bobot suatu kata (*term*) terhadap dokumen
Tweet	tulisan yang diunggah oleh seseorang di Twitter

DAFTAR ISI

HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI ix
GLOSARIUM x
DAFTAR ISI xi
DAFTAR GAMBAR xiii
DAFTAR TABEL 1
BAB I PENDAHULUAN 2
1.1 Latar Belakang 2
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 3
1.5 Manfaat Penelitian 4
1.6 Sistematika Penelitian 4
BAB II LANDASAN TEORI 4
2.1 Penelitian Terkait 5
2.2 Dasar Teori 12
2.2.1 *Deep Learning* 12
2.2.2 Klasifikasi Teks 13
2.2.3 *Text Mining* 14
2.2.4 Pra-Pengolahan (*Preprocessing*) 14
2.2.5 *Term Frequency Inverse Document Frequency* (TF-IDF) 16
2.2.6 *Convolutional Neural Network* (CNN) 17
2.2.7 *Long Short-Term Memory* (LSTM) 18
2.2.8 *Confusion Matrix* 21
BAB III METODOLOGI PENELITIAN 24

3.1 Langkah-Langkah Penelitian 24
3.1.1 Pengumpulan Data 25
3.1.2 *Labelling* 25
3.1.3 *Preprocessing* 26
3.1.4 Pembagian Data 26
3.1.5 Ekstraksi Fitur 26
3.1.6 Klasifikasi 27
3.1.7 Evaluasi 27
3.1.8 Deteksi Konten 28
BAB IV HASIL DAN PEMBAHASAN 29
4.1 Pengumpulan Data 29
4.2 *Input* Dataset 29
4.3 *Preprocessing* 30
4.3.1 *Case Folding* 30
4.3.2 *Tokenizing* 31
4.3.3 *Filtering* 33
4.3.4 *Stemming* 34
4.4 Ekstrasi Fitur 35
4.5 Klasifikasi 35
4.6 Evaluasi 37
4.7 Deteksi Konten 43
BAB V KESIMPULAN DAN SARAN 46
5.1 Kesimpulan 46
5.2 Saran 46
DAFTAR PUSTAKA 47
LAMPIRAN **50**

DAFTAR GAMBAR

Gambar 2.1 Proses klasifikasi teks 13
Gambar 2.2 Contoh Tahap *Tokenizing* 15
Gambar 2.3 Contoh Tahap *Filtering* 15
Gambar 2.4 Contoh Tahap *Stemming* 15
Gambar 2.5 Arsitektur CNN 18
Gambar 2.6 Arsitektur LSTM 19
Gambar 3.1 Langkah Klasifikasi 24
Gambar 3.2 *Dashboard* Twitter Developers 25
Gambar 4.1 Kode Tahapan *Split* Data 29
Gambar 4.2 Kode Tahapan *Case Folding* 30
Gambar 4.3 Kode Tahapan *Tokenizing* 32
Gambar 4.4 Kode Tahapan *Stemming* 34
Gambar 4.5 Kode Tahapan Klasifikasi CNN+LSTM 36
Gambar 4.6 Kode Tahapan *Confusion Matrix* 37
Gambar 4.7 Kode Tahapan *Epoch* Model 41
Gambar 4.8 Kode Tahapan *Accuracy* Model 41
Gambar 4.9 Hasil *Accuracy* Model CNN + LSTM 42
Gambar 4.10 Kode Tahapan *Loss* Model 42
Gambar 4.11 Hasil *Loss* Model 43
Gambar 4.12 Hasil Deteksi Konten Pornografi 44
Gambar 4.13 Hasil Deteksi Konten Netral 44
Gambar 4.14 Hasil Deteksi Konten Umpatan 45

DAFTAR TABEL

Tabel 2.1 Penelitian Terkait 9
Tabel 2.2 *Confusion Matrix* 21
Tabel 4.1 Dataset Ujaran *Tweet* 29
Tabel 4.2 Hasil Tahapan *Case Folding* 31
Tabel 4.3 Hasil Tahapan *Tokenizing* 32
Tabel 4.4 Kode Tahapan *Filtering* 33
Tabel 4.5 Hasil Tahapan *Stemming* 34

1 BAB I
PENDAHULUAN

1.1 Latar Belakang

Twitter merupakan salah satu media pertukaran informasi yang mudah penggunaannya dan populer. Twitter juga termasuk salah satu media sosial yang banyak digunakan di Indonesia karena memiliki persebaran informasi yang sangat cepat (Berliana, dkk, 2018).

Dalam penggunaan sosial media, masih banyak orang kurang bijak dalam bersosialisasi di media Twitter. Banyak orang Indonesia yang menulis kata-kata berbau SARA (suku, agama, ras, dan antargolongan), atau mengeluarkan kalimat yang tidak pantas, dan menyinggung. Oleh karena itu, perlu dikenali dengan melihat kesuluruhan konteks kalimat (Hidayatullah, dkk, 2019) . Akan tetapi, untuk menganalisis atau mengenali konteks kalimat memiliki tantangan tersendiri untuk menyelesaikannya. Mulai dari bagaimana memisahkan secara konteks terkait dengan kata-kata yang ada di dalam *tweet*, mengandung kata-kata yang dimungkinkan menyerempet dengan kata kasar atau konten pornografi, akan tetapi sebenarnya tidak, karena dapat dimungkinkan secara konteks keseluruhan *tweet* adalah sebuah berita. Oleh karena itu, dilakukan penelitian identifikasi konten negatif pada Twitter menggunakan *deep learning*. *Deep learning* merupakan teknik *machine learning* yang menggunakan metode *artificial neural networks* dalam mengerjakan tugasnya. *Deep learning* mempelajari data dengan abstraksi yang tinggi dengan memanfaatkan arsitektur hirarki (Arham, 2018).

Dalam penelitian ini, peneliti menggunakan metode *deep learning Convolutional Neural Network* (CNN) dan *Long Short-Term Memory* (LSTM). CNN merupakan salah satu algoritma dari *deep learning* yang merupakan perkembangan dari *Multi Layer Perceptron* (MLP). CNN memiliki kelebihan dalam menggabungkan ekstraksi fitur yang berbasis *feature learning* (pembelajaran fitur) dan klasifikasi dalam sebuah proses pembelajaran (Juliano & Firdaus, 2019) . *Long Short-Term Memor*y (LSTM) yang merupakan bagian dari *Recurrent Neural Network* (RNN) pada *deep learning* (Amelia, dkk, 2021) . Kedua metode tersebut dipilih karena memiliki tingkat akurasi yang sangat baik apabila digabungkan (Abdulrahman & Baykara, 2020).

Tujuan dari penelitian ini adalah untuk mengidentifikasi tiga kategori dari sebuah *tweet*: netral (tidak mengandung konten negatif), umpatan, dan konten pornografi. Proses

identifikasi dilakukan dengan model dari kumpulan *tweet* berbahasa Indonesia yang berisi tiga kategori di atas. Penelitian ini bertujuan untuk mengetahui hasil akurasi dari penggabungan (*combined*) metode CNN dan LSTM, sehingga diharapkan penerapan kedua metode tersebut memiliki nilai akurasi yang tinggi karena berdasarkan penelitian yang dilakukan oleh (Abdulrahman & Baykara, 2020) penggabungan CNN dan LSTM memiliki tingkat akurasi mencapai 100%.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah, maka rumusan yang diangkat sebagai berikut:

a. Bagaimana melakukan klasifikasi terhadap konten *tweet* menjadi jenis: konten netral (tidak berisi konten negatif), umpatan, dan konten pornografi dengan metode *deep learning*?
b. Bagaimana performa model klasifikasi gabungan CNN dan LSTM dalam melakukan klasifikasi untuk mengidentifikasi konten negatif pada Twitter?

1.3 Batasan Masalah

Ada beberapa batasan masalah yang digunakan untuk membatasi sasaran utama tugas akhir ini adalah sebagai berikut:

a. Data yang diambil adalah data *tweet* yang diperoleh dari Twitter API.
b. Data *tweet* yang digunakan merupakan bahasa Indonesia.
c. Hasil akhir berupa *prototype* yang akan menghitung hasil akurasi terhadap data teks dan memberikan *output* berupa persentase.
d. Data *preprocessing* menggunakan *library Sastrawi*, *pandas*, *numpy* dan NLTK.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

a. Untuk melakukan penghitungan akurasi terhadap tiga jenis konten *tweet*: konten netral (tidak berisi konten negatif), umpatan, dan konten pornografi dengan metode *deep learning*. Dengan begitu, dapat diketahui konten negatif di Twitter dan dapat dihindari atau dilaporkan.
b. Untuk mengetahui performa model klasifikasi menggunakan penggabungan metode CNN dan LSTM dalam melakukan klasifikasi konten negatif pada Twitter,

sehingga hasil performa yang didapatkan dijadikan acuan atau referensi dalam melakukan penelitian yang akan dilakukannya selanjutnya.

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari penelitian ini adalah sebagai berikut:

a. Mengetahui hasil penggunaan gabungan metode CNN dan LSTM dalam pengklasifikasian konten *tweet* bahasa Indonesia dapat diterapkan dengan baik atau tidak.
b. Mengetahui seberapa banyak masyarakat Twitter yang membuat konten *tweet* negatif. Hasil ini juga dapat menjadi salah satu acuan bagi berbagai pihak yang berkepentingan seperti pihak Twitter yang sudah menerapkan *filter* terkait konten negatif.

1.6 Sistematika Penelitian

Sistematika penulisan penelitian ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab pendahuluan akan membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penelitian.

BAB II LANDASAN TEORI

Bab landasan teori akan membahas penelitian sebelumnya dan dasar teori. Bagian ini akan memperkenalkan operasi dalam klasifikasi teks.

BAB III METODOLOGI PENELITIAN

Bab metodologi penelitian berisi deskripsi mengenai langkah-langkah yang dilakukan dalam penelitian.

BAB IV HASIL DAN PEMBAHASAN

Bab hasil dan pembahasan berisikan hasil klasifikasi *tweet* bahasa Indonesia serta pembahasan dari penelitian yang telah dilakukan.

BAB V KESIMPULAN DAN SARAN

Bab kesimpulan dan saran berisikan kesimpulan yang diambil setelah melakukan penelitian dan pemberian saran untuk perbaikan pada penelitian berikutnya.

2 BAB II
LANDASAN TEORI

2.1 Penelitian Terkait

Pada bagian penelitian terkait ini akan membahas penelitian-penelitian yang telah dilakukan sebelumnya dan menjadi sumber rujukan pada penelitian ini. Tabel 2.1 menunjukkan penelitian yang pernah dilakukan. Beberapa hasil penelitian hanya menyimpulkan garis besar dari penelitian yang sudah diteliti.

1. Penelitian sebelumnya pernah dilakukan oleh Abdulloh & Hidayatullah (2020) , yang bertujuan untuk dari mengidentifikasi konten yang mengandung makna perundungan secara daring (c*yberbullying*) pada media sosial. Dalam kasus ini, penulis memilih media sosial Twitter sebagai objek penelitian. Setidaknya terdapat 1.971 baris data yang telah dikumpulkan. Data-data tersebut berisi dua jenis cuitan baik cuitan yang memiliki kecenderungan c*yberbullying* dan yang tidak. Untuk mencapai tujuan penelitian, peneliti menggunakan lima langkah penelitian, yaitu pengumpulan data, *preprocessing*, ekstraksi fitur, klasifikasi, dan evaluasi. Empat algoritma *Machine Learning* diimplementasikan dalam penelitian ini, yaitu *K-Nearest Neighbor (*KNN*), Multinomial Naïve Bayes, Logistic Regression*, dan *Support Vector Machine with linear kernel (*SVM*).* Dapat disimpulkan bahwa keempat algoritma tersebut memiliki performa yang relatif sama. Akurasi dari masing masing algoritma dituliskan sebagai berikut *Multinomial Naïve Bayes* 0.9616, *Logistic Regression* 0.9949, *Support Vector Machine* dengan linear kernel 0.9975, dan *K-Nearest Neighbor* (KNN) 0.9188.
2. Penelitian yang dilakukan oleh Amelia, Aldino, & Isnain (2021). Tujuan penelitian ini untuk melakukan analisis sentimen terhadap salah satu grup WhatsApp yaitu grup *chat* yang bernama "Shinubi" untuk dilakukan analisis sentimen menggunakan metode *Long Short Term Memory* terhadap respons pengguna di grup terkait fitur-fitur yang tersedia pada Whatsapp. Metode LSTM digunakan untuk mengetahui tingkat akurasi terhadap data yang digunakan dan penelitian ini juga menggunakan fitur *word2vec*. Hasil yang didapat pada penelitian analisis sentimen ini yaitu, mendapatkan tingkat akurasi sebesar 99%, *precision* 99%, dan *recall* sebesar 1%. Metode LSTM ini memiliki akurasi yang baik pada pemrosesan analisis sentimen *chat* grup WhatsApp ini.
3. Penelitian yang dilakukan oleh Hidayatullah, Hakim, & Sembada, (2019) . Dalam penelitian ini, peneliti mempresentasikan klasifikasi konten dewasa pada data Twitter menggunakan *Long Short-Term Memory* (LSTM) *Neural Network*. Untuk menemukan model klasifikasi terbaik, pada penelitian ini dibangun empat model LSTM dengan skenario yang berbeda untuk setiap model LSTM. Penelitian ini juga membandingkan

kinerja metode LSTM dengan beberapa metode pembelajaran mesin tradisional, termasuk *Multinomial Naïve Bayes*, Regresi Logistik, dan Klasifikasi Vektor Dukungan. Berdasarkan percobaan, model terbaik diperoleh dengan menerapkan dua lapisan LSTM dengan *dropout* dengan akurasi 98,38%. Pencapaian ini sedikit lebih tinggi 0,06% dari akurasi model SVC. Pada penelitian ini juga menemukan bahwa adanya *dropout* mempengaruhi nilai *loss* dan nilai akurasi. Hasil penelitian menunjukkan bahwa nilai *loss* menurun dari 12,88% menjadi 5,08% dan akurasi meningkat dari 97,89% menjadi 98,39%.

4. Penelitian terkait lainnya juga pernah dilakukan Abdulrahman & Baykara (2020) . Penelitian ini membahas klasifikasi berita palsu di media sosial telah mendapatkan banyak perhatian dalam satu dekade terakhir karena kemudahan menambahkan konten palsu melalui situs media sosial. Selain itu, orang lebih suka mendapatkan berita di media sosial daripada di televisi nasional. Tren ini telah menyebabkan meningkatnya minat pada berita palsu. Penelitian ini berfokus pada pengklasifikasian berita palsu di media sosial dengan konten tekstual (klasifikasi teks). Dalam klasifikasi ini, empat metode tradisional diterapkan untuk mengekstrak fitur dari teks (frekuensi istilah – frekuensi dokumen terbalik, vektor hitungan, vektor level karakter, dan vektor *level N-Gram*), menggunakan 10 pengklasifikasi pembelajaran mesin (*machine learning*) dan pembelajaran mendalam (*deep learning*) yang berbeda untuk mengkategorikan kumpulan data berita yang palsu. Hasil yang diperoleh menunjukkan bahwa berita palsu dengan konten tekstual memang dapat diklasifikasikan, terutama menggunakan jaringan saraf *convolutional*. Penelitian ini memperoleh rentang akurasi 81% hingga 100% dengan menggunakan pengklasifikasi yang berbeda.
5. Penelitian sebelumnya juga pernah dilakukan oleh Badjrie, Pratiwi, & Anggana (2021). Algoritma CNN (*Convolutional Neural Network*) merupakan algoritma *deep learning* yang dapat menggunakan gambar sebagai masukan, menetapkan kepentingan untuk berbagai aspek dan objek dalam gambar agar dapat membedakan satu dengan yang lain dan memiliki akurasi yang tinggi, sehingga dalam penelitian analisis sentimen *review* produk IndiHome dan First Media. Penelitian ini bertujuan untuk melakukan penilaian produk terhadap *provider* menggunakan metode analisis sentimen *review customer* dari tiap *tweets* yang pelanggan berikan dengan algoritma *convolutional neural network*. Hasil akurasi yang didapatkan, diperoleh akurasi tertinggi sebesar 98% untuk *provider* IndiHome dan 91% untuk *provider* First Media.

6. Penelitian selanjutnya dilakukan oleh Ihsan, Iskandar, Harahap, & Agustian (2021) Penelitian ini membahas mengenai ujaran kebencian dan bahasa kasar. Masalah utama ujaran kebencian dan bahasa kasar mudah ditemukan di dalam komunikasi tertulis di media sosial seperti Twitter, yang dapat memicu terjadinya persengketaan di antara korban dan pengujarnya. Penelitian ini bertujuan untuk membangun sistem untuk mengklasifikasi *tweet* apakah mengandung ujaran kebencian dan kata-kata kasar. *Dataset* yang digunakan terdiri dari 13.126 *tweet* asli dari Twitter. *Word embedding* digunakan untuk fitur dari teks. Algoritma *decision tree* digunakan untuk klasifikasi. Rekayasa fitur dan pengaturan parameter menunjukkan peningkatan performa deteksi. Fitur leksikon di klasifikasi *decision tree* menghasilkan akurasi tertinggi untuk deteksi ketiga kelas, yaitu kelas ujaran kebencian, kata-kata kasar dan level ujaran kebencian, daripada rekayasa fitur khusus dan fitur tekstual. Rata-rata akurasi dari ketiga kelas meningkat dari 69,77% menjadi 70,48% untuk komposisi data latih-uji 90:10, dan dari 69,35% menjadi 69,54% untuk komposisi 80 banding 20.
7. Penelitian terkait juga dilakukan oleh Dwitama & Hidayat (2021) . Penelitian ini membahas mengenai terjadinya peningkatan yang signifikan dalam aktivitas komunikasi antar pengguna internet di media *online* karena adanya peningkatan pengguna media sosial. Misalnya, pengguna Twitter dapat mengirim pesan melalui *tweet* mereka. Namun, *tweet* juga bisa mengandung makna negatif. Oleh karena itu, perlu mendapat perhatian khusus karena berpotensi mengandung ujaran kebencian. Bahkan pemerintah memandang perlu untuk menerbitkan regulasi untuk menangani kasus ujaran kebencian seperti Undang-Undang Informasi dan Transaksi Elektronik (UU ITE) yang diterbitkan pada tahun 2018 Pasal 28 ayat 2 Ujaran Kebencian. *Machine learning* (ML) merupakan salah satu teknik yang dapat digunakan dalam mengidentifikasi pola. Ada berbagai jenis data yang dapat diterapkan ML, termasuk teks (dikenal sebagai *text analytic*). Penelitian sebelumnya telah menggunakan metode *Support Vector Machine* (SVM) untuk mengidentifikasi ujaran kebencian pada teks Twitter dengan lebih dari satu label (*multilabel*). Tujuan dari penelitian ini adalah untuk mengidentifikasi ujaran kebencian di Twitter dengan label lebih dari satu (*multilabel*) melalui *Convolutional Neural Network* (CNN). Penelitian tersebut memperoleh model CNN terbaik dengan akurasi 98,76% dari dataset *multilabel* tentang ujaran kebencian dalam teks berbahasa Indonesia.

8. Penelitian terkait juga dilakukan oleh Ramdhani (2021) . Penelitian ini membahas mengenai pernyataan Mendikbud Republik Indonesia mengenai keberlanjutan pembelajaran daring di awal tahun 2021 yang memperoleh komentar positif dan negatif dari masyarakat Indonesia. Komentar tersebut sulit dipilah untuk mendapatkan *term* atau kata hasil dari komentar positif atau negatif karena penggunaan ragam bahasa dalam media sosial di antaranya tidak formal, menggunakan simbol, singkatan, bahasa asing, dan bahasa daerah. Tujuan penelitian ini adalah untuk melakukan pengklasifikasian terhadap sentimen positif, negatif, dan netral terhadap data uji (*tweet* terkait pembelajaran daring), serta mengetahui akurasi model klasifikasi dengan menggunakan metode *deep learning*. Tahapan penelitian ini terdiri dari pengambilan data mentah (*crawling*), *pre-processing* data, klasifikasi dengan metode *deep learning*, dan evaluasi *cross validation* dengan menggunakan tools *RapidMiner*. Hasil percobaan pada penelitian menunjukan bahwa metode terbaik pada data *tweet* adalah metode *deep learning* yaitu dengan akurasi sebesar 100%, ketika dibandingkan dengan metode *Naïve Bayes* yang memiliki akurasi sebesar 99%, dan k-NN yang memiliki akurasi sebesar 82%. Diketahui juga bahwa sebanyak 73% komentar positif, 14% komentar negatif, dan 13% komentar netral.
9. Penelitian terkait lainnya dilakukan oleh Zamil (2019) . Pada penelitian ini membahas mengenai kalimat pada *tweet* yang berisikan konten ofensif atau menyinggung dapat mengajak kepada permusuhan. Penyaringan *tweet* secara manual tidak terukur dan membutuhkan waktu yang lama sehingga dibutuhkan aplikasi untuk melakukan penyaringan. Penelitian ini menggunakan metode klasifikasi yaitu *Naïve Bayes Classifier*. Data yang digunakan yaitu *tweet* tentang politik. Data *tweet* diperoleh berdasarkan *hashtag* terkait tentang politik dan memanfaatkan Twitter API (*Application Programming Interface*). Penelitian ini menggunakan dataset berjumlah 1.000 data yang dibagi menjadi 500 data kelas ofensif dan 500 data kelas tidak ofensif. Hasil pengujian menunjukkan nilai akurasi tertinggi yaitu pada *bigram* sebesar 84%, nilai *precision* tertinggi yaitu pada *trigram* sebesar 97.33%, dan nilai *recall* tertinggi yaitu pada *bigram* sebesar 81.84%. Berdasarkan hasil penelitian dapat disimpulkan bahwa *Naïve Bayes Classifier* berhasil mengklasifikasikan *tweet* yang bersifat ofensif atau tidak ofensif.
10. Penelitian terkait lainnya juga dilakukan oleh Fadli & Hidayatullah (2021) . Penelitian ini membahas mengenai *cyberbullying* merupakan masalah yang harus menjadi

perhatian penting oleh masyarakat. *Cyberbullying* termasuk kebiasaan buruk yang berdampak mengerikan, mulai dari gangguan psikologis korban, hingga munculnya kasus bunuh diri. Tujuan dari penelitian ini adalah mengidentifikasi konten yang mengandung makna perundungan secara daring (*cyberbullying*) pada media sosial khususnya Twitter. Dalam kasus ini, sumber data penelitian ini berasal dari media sosial Twitter. Setidaknya, ada 6.835 data yang telah dikumpulkan. Data tersebut terdiri dari dua jenis cuitan dengan masing-masing cuitan memiliki kecenderungan *cyberbullying* dan *noncyberbullying*. Tujuan penelitian akan tercapai dengan melakukan beberapa langkah yang pertama yaitu pengumpulan data, lalu *preprocessing*, kemudian ekstraksi fitur, setelah itu klasifikasi, dan terakhir evaluasi. Dua algoritma *deep learnin*g diimplementasikan dalam penelitian ini, yaitu LSTM dan BiLSTM. Dalam penelitian ini disimpulkan bahwa kedua algoritma tersebut memiliki performa yang relatif sama. Akurasi dari masing-masing algoritma dituliskan sebagai berikut L*ong Short-Term Memory* 81,60% dan *Bidirectional Long Short-Term Memory* 81,78%. Lalu, untuk nilai dari *F1-Score* dari masing-masing algoritma sebagai berikut *Long Short-Term Memory* 77,88% dan *Bidirectional Long Short-Term Memory* 77,89%.

Berdasarkan penelitian terkait di atas, dirangkum pada Tabel 2.1.

Tabel 2.1 Penelitian terkait

Peneliti	**Judul**	**Metode**	**Hasil**
(Abdulloh & Hidayatullah, 2020)	Deteksi *Cyberbullying* pada Cuitan Media Sosial Twitter	KNN, SVM, *Logistic Regression*, dan *Multinomial Naïve Bayes*	Nilai *accuracy, precision, recall, dan F1-Score* dari *Linear SVM* paling tinggi dengan nilai masing-masing 0,997; 1,00; 1,00; 1,00.
(Amelia, Aldino, & Isnain, 2021)	Teks dan analisis sentimen pada *chat* grup whatsapp Menggunakan *long short-term memory* (lstm)	LSTM	Hasil yang didapat pada penelitian analisis sentimen ini yaitu, mendapatkan tingkat akurasi sebesar 99%, *precision* 99%, dan *recall* sebesar 1%

(Hidayatullah, Hakim, & Sembada, 2019)	*Adult Content Classification on Indonesian Tweets using LSTM Neural Network*	LSTM, *Multinomial Naïve Bayes, Logistic Regression, Support Vector Classification*	Model terbaik didapatkan dengan menerapkan 2 lapisan LSTM dengan akurasi 98,38%. Pencapaian tersebut lebih tinggi 0,06% lebih tinggi daripada model SVC. Hasil dari *loss value* berkurang dari 12,88% menjadi 5,08% dan akurasi meningkat dari 97,89% menjadi 98,39%.
(Abdulrahman & Baykara, 2020)	*Fake News Detection Using Machine Learning and Deep Learning Algorithms*	*Simple Neural Network*, RNN + LSTM, CNN + LSTM	Penelitian ini memperoleh hasil Akurasi klasifikasi sangat kuat, terutama saat menggunakan teknologi CNN + LSTM, karena teknik ini memberikan akurasi 100% dalam waktu yang sangat singkat.
(Badjrie, Pratiwi, & Anggana, 2021)	Analisis Sentimen *Review Customer* Terhadap Produk Indihome Dan First Media Menggunakan Algoritma *Convolutional Neural Network*	*Convolutional Neural Network*	Hasil akurasi yang didapatkan, memperoleh akurasi tertinggi untuk *provider* IndiHome sebesar 0,98 *precision*, 0,98 *recall*, 0,98 F1-*score* 0,98.
(Ihsan, Iskandar, Harahap, & Agustian,	Algoritme decision tree untuk mendeteksi ujaran kebencian dan	*Clasification*	Fitur leksikon di klasifikasi *Decision Tree* menghasilkan akurasi tertinggi untuk deteksi

2021)	bahasa kasar multilabel pada Twitter berbahasa Indonesia		ketiga kelas, yaitu kelas ujaran kebencian, kata-kata kasar dan level ujaran kebencian, daripada rekayasa fitur khusus dan fitur tekstual. Rata-rata akurasi dari ketiga kelas meningkat dari 69,77% menjadi 70,48% untuk komposisi data latih-uji 90:10, dan dari 69,35% menjadi 69,54% untuk komposisi 80:20.
(Dwitama & Hidayat., 2021)	Identifikasi Ujaran Kebencian Multilabel Pada Teks Twitter Berbahasa Indonesia Menggunakan *Convolution Neural Network*	*Convolution Neural Network*	Penelitian tersebut memperoleh model CNN terbaik dengan akurasi 98,76% dari dataset multilabel tentang ujaran kebencian dalam teks berbahasa Indonesia
(Ramdhani N., 2021)	ANALISIS SENTIMEN PENGGUNA TWITTER TERHADAP BELAJAR DARING SELAMA PANDEMI COVID-19 DENGAN DEEP	KNN, *Naïve Bayes, Deep Learning*	Hasil percobaan pada penelitian menunjukan bahwa metode terbaik pada data *tweet* adalah metode *Deep Learning* yaitu dengan akurasi sebesar 100%, ketika dibandingkan dengan metode *Naïve Bayes* yang memiliki akurasi sebesar 99%, dan k-NN yang memiliki

	LEARNING		akurasi sebesar 82%. Diketahui juga bahwa sebanyak 73% komentar positif, 14% komentar negatif, dan 13% komentar netral.
(Zamil, 2019)	Klasifikasi Kalimat Ofensif Pada Media Sosial Twitter Menggunakan Metode *Naïve Bayes Classifier*	*Naive Bayes Classifier*	Hasil akurasi tertinggi sebesar 84,00% dengan model *bigram*, nilai *precision* pada model *trigram* sebesar 97,33% dan nilai *recall* tertinggi yaitu pada model *bigram* sebesar 81,48%.
(Fadli & Hidayatullah, 2021)	Identifikasi Cyberbullying pada Media Sosial Twitter Menggunakan Metode LSTM dan BiLSTM.	LSTM, BiLSTM	Model BiLSTM mendapatkan nilai *accruacy* sebesar 94,51; 95,24 dan nilai *F1-Score* sebesar 92,75; 93,84. Pada model LSTM, tidak memiliki perbandingan yang jauh dengan hasil *accruacy* 93,77; 94,87 dan nilai *F1-Score* sebesar 92,02; 93,33.

2.2 Dasar Teori

2.2.1 *Deep Learning*

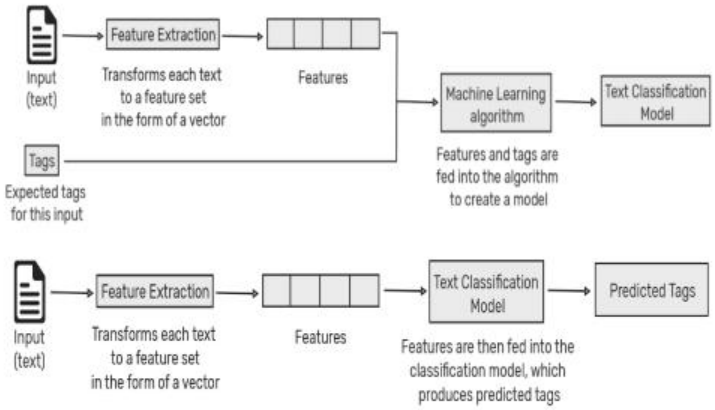
Deep learning merupakan metode *learning* yang memanfaatkan *artificial neural network* yang berlapis-lapis (*multilayer*). *Artifical neural network* kini dibuat mirip otak manusia, di mana *neuron-neuron* terkoneksi satu sama lain sehingga membentuk sebuah jaringan *neuron* yang sangat rumit (Nugroho, Fenriana, & Arijanto, 2020) . *Deep learning*

juga merupakan sebuah pendekatan dalam melakukan penyelesaian masalah pada sistem-sistem pembelajaran komputer dengan memakai konsep hierarki (Badjrie, dkk, 2021).

Deep learning merupakan salah satu bidang dari *machine learning* yang memanfaatkan jaringan syaraf tiruan untuk implementasi permasalahan dengan dataset yang besar dan menggunakan metode *artificial neural network*. Teknik *deep learning* memberikan arsitektur yang sangat kuat untuk *supervised learning* dengan menambahkan lebih banyak lapisan, maka model pembelajaran tersebut dapat mewakili data citra berlabel dengan lebih baik. Pada *machine learning* terdapat teknik untuk menggunakan ekstraksi fitur dari data pelatihan dan algoritma pembelajaran khusus untuk mengklasifikasi citra maupun untuk mengenali suara. Namun, metode ini masih memiliki beberapa kekurangan baik dalam hal kecepatan dan akurasi (Triakno, 2018).

2.2.2 Klasifikasi Teks

Klasifikasi merupakan rangkaian proses pengkategorian terhadap sekumpulan dokumen menjadi beberapa kelas (Abdulloh & Hidayatullah, 2020). Tujuan dari klasifikasi teks adalah menetapkan dokumen teks ke dalam satu atau beberapa kategori yang telah ditentukan.



Gambar 2.1 Proses Klasifikasi Teks (Sumber: devopedia.org)

Gambar 2.1 merupakan proses dari klasifikasi teks. Langkah yang dilakukan dalam melakukan klasifikasi teks adalah dengan mengubah teks menjadi numerik dalam bentuk vektor menggunakan *feature extraction*. Kemudian, *feature set* dan *tags* digunakan dalam algoritma pembelajaran mesin untuk membuat model klasifikasi. Proses ini disebut proses pelatihan (*training*). Lalu, model klasifikasi yang dihasilkan digunakan untuk memprediksi teks. Proses ini disebut proses pengujian (*testing*) (Hidayatullah, dkk, 2019).

2.2.3 *Text Mining*

Text mining merupakan salah satu teknik yang dapat digunakan untuk melakukan klasifikasi dokumen, *clustering, information extraction*, analisis sentimen, dan *information retrival. Text mining* merupakan variasi dari *data mining* yang berusaha menemukan pola yang menarik dari sekumpulan data tekstual yang berjumlah besar (Sudiantoro & Zuliarso, 2018) . *Text mining* merupakan penambangan data yang berupa teks di mana sumber data biasanya didapatkan dari dokumen dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisis keterhubungan antara dokumen (Simatupang & Utomo, 2019).

2.2.4 Pra-Pengolahan (*Preprocessing*)

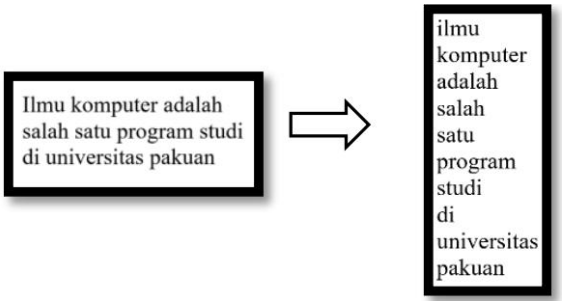
Pra-Pengolahan merupakan proses awal klasifikasi dokumen yang bertujuan menyiapkan data agar menjadi terstuktur. Pra-Pengolahan merupakan salah satu langkah yang penting dalam analisis sentimen. Tahapan dalam pra-pengolahan yaitu (Badjrie, dkk, 2021):

1. *Case Folding*

   Tidak semua teks dalam dokumen konsisten dalam penggunaan huruf kapital. Maka dari itu, *case folding* digunakan dalam mengonversi keseluruhan teks dalam dokumen menjadi suatu bentuk standar (menjadi *lowercase*). Sebagai contoh, *user* ingin mendapatkan informasi “KOMPUTER” dengan mengetik “KomPuter” atau “kompUTER”, tetapi diberikan hasil *retrieval* yang sama yakni “komputer”. Dalam hal ini, *case folding* digunakan untuk mengubah semua huruf dalam dokumen menjadi huruf kecil (Tjut Awaliyah Zuraiyah, 2017).

2. *Tokenizing*

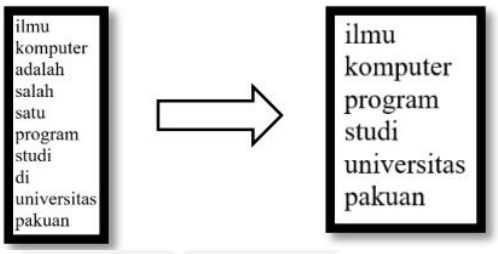
   Tahapan *tokenizing* merupakan tahapan pemotongan *string input* berdasarkan tiap kata yang menyusunnya (Tjut Awaliyah Zuraiyah, 2017). Contoh dari *tokenizing* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Contoh Tahap *tokenizing*

3. *Filtering*

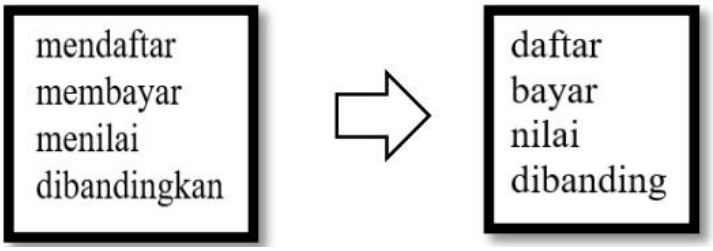
Tahap *filtering* merupakan tahap mengambil kata-kata penting dari hasil *token* dengan menggunakan *stoplist* (membuang kata kurang penting) atau *wordlist* (menyimpan kata penting) (Tjut Awaliyah Zuraiyah, 2017) . Contohnya dapat dilihat pada Gambar 2.3.



Gambar 2.3 Contoh Tahap *filtering*

4. *Stemming*

Teknik *stemming* diperlukan untuk memperkecil jumlah indeks yang berbeda dari suatu dokumen dan juga untuk melakukan pengelompokan kata-kata lain yang memiliki kata dasar dan arti yang serupa namun memiliki bentuk atau *form* yang berbeda (Tjut Awaliyah Zuraiyah, 2017) . Gambar 2.4 menunjukkan contoh tahap *stemming*.



Gambar 2.4 Contoh Tahap *Stemming*

2.2.5 *Term Frequency Inverse Document Frequency* (TF-IDF)

Metode TF-IDF merupakan metode untuk menghitung bobot suatu kata (*term*) terhadap dokumen. Metode ini juga terkenal efisien, mudah, dan memiliki hasil yang akurat. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu frekuensi kemunculan sebuah kata di dalam sebuah dokumen tertentu dan *inverse* frekuensi dokumen yang mengandung kata tersebut. Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata tersebut di dalam dokumen (Naf'an, dkk, 2019).

Deep learning tetap membutuhkan *feature extractor*, hal ini dikarenakan pada kasus klasifikasi diperlukan pengolahan sentimen berdasarkan kemunculan frekuensi. Hal ini didasari dengan kemampuan TF-IDF itu sendiri yang di mana memberikan pembobotan kata yang lebih besar pada kata yang memiliki frekuensi kata yang lebih sedikit.

Pendekatan TF-IDF menyajikan teks dengan ruang tabel yang di setiap ciri dalam teks sesuai dengan satu kata. TF (*Term Frequency*) akan menghitung frekuensi kemunculan sebuah kata dan dibandingkan jumlah seluruh kata yang ada di dalam dokumen. Persamaan yang digunakan untuk menghitung TF ditunjukkan pada Persamaan (1).

$$tf\ (i) = \frac{freq\ (t_1)}{\sum freq\ (t)} \tag{1}$$

Keterangan:

tf(i) : Nilai *term frequency* sebuah kata dalam sebuah dokumen.

Freq (t_i) : Frekuensi kemunculan sebuah kata dalam sebuah dokumen.

$\sum freq\ (t)$: Jumlah keseluruhan kata dalam dokumen.

Sementara IDF (*Inverse Document Frequency*) menghitung algoritma dari jumlah seluruh dokumen dan dibandingkan dengan jumlah dokumen di mana dalam dokumen tersebut kata (*t*) yang dimaksud muncul. Rumus untuk menghitung IDF ditunjukkan pada Persamaan (2).

$$tdf\ (i) = \log \frac{|D|}{|\{d:t_i \in d\}|} \tag{2}$$

Keterangan:

idf(i) : Nilai *inverse document frequency* sebuah kata di seluruh isi dokumen.

|D| : Jumlah seluruh dokumen.

|(d: ti∈ d}| : Jumlah dokumen yang mengandung kata (t).

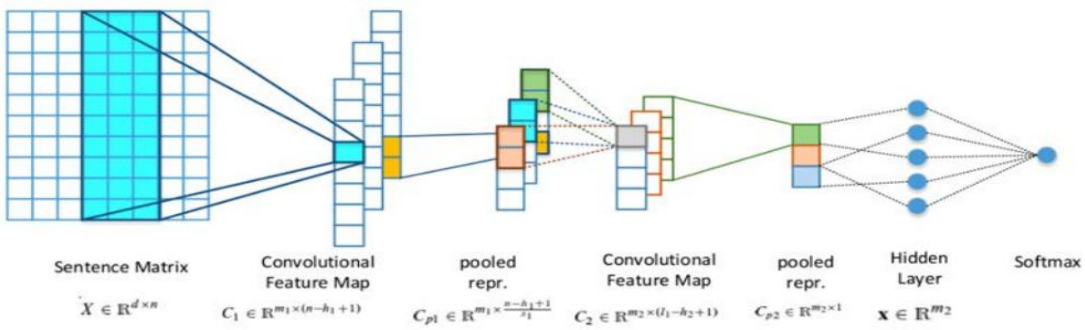
Dengan kedua persamaan tersebut maka dapat ditentukan nilai bobot (*w*) sebuah kata dalam sekumpulan dokumen, dengan menghitung perkalian dari kedua persamaan sebelumnya. Persamaan (3) untuk menentukan nilai bobot (*w*) sebuah kata.

$$Weight\ (tf - idf)_i = tf(i)\ \times idf(i) \tag{3}$$

Informasi terkait *time series* akan hilang setelah proses TF-IDF, namun dapat dilakukan *representation learning*. *Representation learning* merupakan metode untuk melakukan kompresi *feature vector* menggunakan *neural network*. Proses melakukan kompresi disebut *encoding*, hasil *feature vector* dalam bentuk terkompres disebut *coding*. Solusi dari informasi atau *time series* yang hilang dilakukan *decoding*. *Decoding* merupakan proses mengembalikan hasil kompresi ke bentuk awal.

2.2.6 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan jaringan saraf tiruan yang memberi umpan balik di mana jaringan saraf mempertahankan struktur hierarki dengan mempelajari representasi fitur internal dan menggeneralisasi fitur-fitur dalam masalah gambar secara umum seperti pengenalan objek dan masalah *computer vision*. CNN dikenal juga sebagai ConvNets, yang merupakan bagian dari *artificial neural network*. Penggunaannya tidak terbatas pada gambar, dapat juga digunakan untuk mencapai hasil dalam masalah pemrosesan bahasa alami dan pengenalan ucapan (Hermanto, dkk, 2021) . Pada penelitian ini, CNN melakukan ekstraksi fitur menggunakan lapisan konvolusi dan lapisan *pooling*. Arsitektur dari CNN tergambar pada Gambar 2.5.

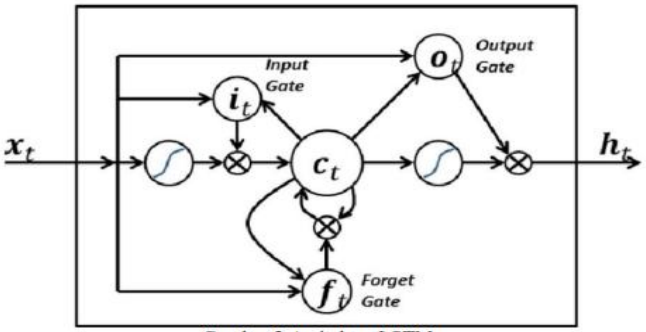


Gambar 2.5 Arsitektur CNN

Gambar 2.5 menunjukkan model *layer* yang dibuat pada penelitian ini. Terdapat beberapa *layer* di antaranya *sentence matrix, convolutional feature map, pooled repr, convolutional feature map, pooled repr, hidden layer* dan *softmax.* CNN yang akan diimplementasikan tidak menggunakan banyak lapisan, hal ini bertujuan untuk menghemat waktu *training*. Arsitektur CNN yang sederhana sudah dirasa mencukupi untuk menyelesaikan klasifikasi data, mengingat kategori untuk klasifikasi hanya sedikit. Adapun sedikitnya jumlah data yang digunakan untuk pembelajaran dan diprediksi akan dihasilkan akurasi yang kurang maksimal.

2.2.7 *Long Short-Term Memory* (LSTM)

*Long Short-Term Memor*y (LSTM) yang merupakan bagian dari *Recurrent Neural Network* (RNN) pada *deep learning* (Amelia, dkk, 2021) . Kemudahan mengatasi ketergantungan jangka panjang (*long term dependencies*) pada masukannya, yaitu salah satu fungsi LSTM. *Memory block* pada LSTM dapat menentukan nilai mana yang akan dipilih sebagai keluaran yang relevan terhadap masukan yang diberikan. Hal ini merupakan kelebihan yang dimiliki oleh LSTM dibandingkan RNN (Hastomo & Satyo, 2019). Penggunaan LSTM digunakan untuk menyimpan *output* yang dihasilkan dari lapisan CNN ke dalam lapisan LSTM. Penggabungan metode CNN dan LSTM dilakukan dikarenakan LSTM sendiri masih kurang tepat dalam melakukan klasifikasi dan tidak dapat melakukan pengambilan bobot yang lebih bermakna, sehingga memerlukan lapisan konvolusi dan lapisan *pooling* serta *fully connected layer*. Arsitektur LSTM ditunjukkan pada Gambar 2.6.



Gambar 3 Arsitektur LSTM

Gambar 2.6 Arsitektur *LSTM*

Pada Gambar 2.6 Arsitektur LSTM terdapat 2 hasil nilai *output*, yang di antaranya *output* pertama merupakan keluaran sebenarnya yang dilanjutkan kembali ke *cell* berikutnya dan menjadi *output* dari *cell* itu. Pada sistem LSTM terdapat tiga buah pintu atau *gates*, di antaranya *input*, *forget*, dan *output gat*e. Pada *input gate* bertujuan untuk memasukkan data baru, sedangkan menghapus informasi yang tidak penting terdapat pada *forget gate* dan memengaruhi keluaran pada saat waktu bersamaan merupakan tugas dari *output gate*.

1. *Forget Gate*

   Forget gate dapat dirumuskan sebagai berikut (Aldi et al., 2018).

$$Ft = \sigma(Wf.[ht-1, xt] + bf) \quad (4)$$

   Keterangan:

   f_t : *Forget gate*

   σ : Fungsi *sigmoid*

   Wf : Nilai *weight* untuk *forget gate*

   ht-1 : Nilai *output* sebelum orde ke-t

   x_t : Nilai *input* pada orde ke-t

   b_f : Nilai bias pada *forget gate*

2. *Input Gate*

   Formula dari *input gate* sebagai berikut (Arfan & Lussiana ETP, 2019).

$$It = \sigma(wi.[ht-1, xt] + bi) \quad (5)$$

   Keterangan:

   it : *Input gate*

σ : Fungsi *sigmoid*

Wt : Nilai *weight* untuk *input gate*

*h*t-1 : Nilai *output* sebelum orde ke-t

*x*t : Nilai *input* pada orde ke-t

bi : Nilai bias pada *input gate*

3. *Cell Aksen Gate*

Cell aksen gate merupakan *memory cell* yang diaktivasi menggunakan *tanh* dengan rumus sebagai berikut (Arfan & Lussiana ETP, 2019).

$$C't = tanh(wc.[ht-1, xt] + bc) \quad (6)$$

Keterangan:

$C't$: Nilai baru yanng ditambahkan ke *cell state*

Tanh : Fungsi *tanh*

Wc : Nilai *weight* untuk *cell state*

ht-1 : Nilai *output* sebelum Orde ke-*t*

xt : Nilai *input* pada orde ke-t

bc : Nilai bias pada *cell state*

4. *Output State*

Formula yang digunakan pada *output state* adalah sebagai berikut (Lubis & Kharisudin, 2021)

$$Ot = \sigma\,(wo.[ht-1, xt] + bo \quad (7)$$

Keterangan:

Ot : *Output gate*

σ : Fungsi *sigmoid*

wo : Nilai *weight* untuk *output gate*

ht-1 : Nilai *output* sebelum orde ke-t

xt : Nilai *input* pada orde ke-t

bo : Nilai bias pada *output gate*

5. *Cell State*

Dengan rumus sebagai berikut (Lubis & Kharisudin, 2021):

$$Ct = ft * Ct-1 + it * C't \quad (8)$$

Keterangan:

Ct : *Cell state*

Ft : *Forget gate*

Ct-1 : *Cell state* sebelum orde ke-t

It : *Input gate*

C't : Nilai baru yang ditambahkan ke *cell state*

6. *H-State*

Formula dari *h-state* sebagai berikut (Arfan & Lussiana ETP, 2019).

$$ht = ot * tanh\,(Ct) \quad (9)$$

Keterangan:

ht : Nilai *output*

Ot : *Output gate*

Tanh : Fungsi *tanh*

Ct : *New cell state*

2.2.8 *Confusion Matrix*

Metode untuk menguji algoritma adalah metode *confusion matrix*. *Confusion matrix* merupakan suatu instrumen yang digunakan untuk mengevaluasi performa dari model klasifikasi yang telah dihasilkan. Akan tetapi, tidak semua algoritma dapat diuji dengan menggunakan *confusion matrix*, hanya *supervised learning* yang digunakan untuk klasifikasi (Andraini & Mahdiyah, 2022). Pada *confusion matrix* terdapat *true positives* adalah jumlah *record* positif yang diklasifikasikan sebagai positif, *false positives* adalah jumlah *record* negatif yang diklasifikasikan sebagai positif, *false negatives* adalah jumlah *record* positif yang diklasifikasikan sebagai negatif. Data uji yang dimasukkan ke dalam *confusion matrix* dihitung dan menghasilkan nilai *sensitivity* (*recall*), *specificity*, *precision*, dan *accuracy*. Adapun detail dari *confusion matrix* dapat dilihat pada Tabel 2.2.

Tabel 2.2 *Confusion Matrix*

		True Value	
		True	*False*
Pre	*True*	TP	FP

		Correct result	*Unexpected Result*
	False	FN *Missing Result*	TN *Corect Absence of result*

1. *True Positive* (TP) artinya seberapa banyak data yang aktual kelasnya positif, dan model juga memprediksi positif.
2. *True Negative* (TN) artinya seberapa banyak data yang aktual kelasnya negatif, dan model memprediksi negatif.
3. *False Positive* (FP) artinya seberapa banyak data yang aktual kelasnya negatif, namun model memprediksi positif.
4. *False Negative* (FN) artinya seberapa banyak data yang aktual kelasnya positif, namun model memprediksi negatif.

Melalui hasil 4 data diatas dapat diperoleh data-data lain yang berguna untuk mengukur perfoma sebuah model, sebagai berikut:

1. *Accuracy* atau biasa disebut akurasi mengukur jumlah total prediksi yang benar dibandingkan dengan total data (Ramdhani, Andreswari, & Hasibuan, 2018) . Rumus untuk menghitung *accuracy* ditunjukkan pada Persamaan (10).

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (10)$$

2. *Precision* adalah rasio *item* yang relevan dipilih untuk semua *item* dipilih (Ramdhani, Andreswari, & Hasibuan, 2018) . Rumus untuk menghitung *precision* ditunjukkan pada Persamaan (11).

$$Precision = \frac{TP}{TP+FP} \quad (11)$$

3. *Recall* didefinisikan sebagai rasio *item* yang relevan dipilih dengan jumlah total *item* yang relevan (Ramdhani, Andreswari, & Hasibuan, 2018). Rumus untuk menghitung *recall* ditunjukkan pada Persamaan (12).

$$Recall = \frac{TP}{TP+FN} \quad (12)$$

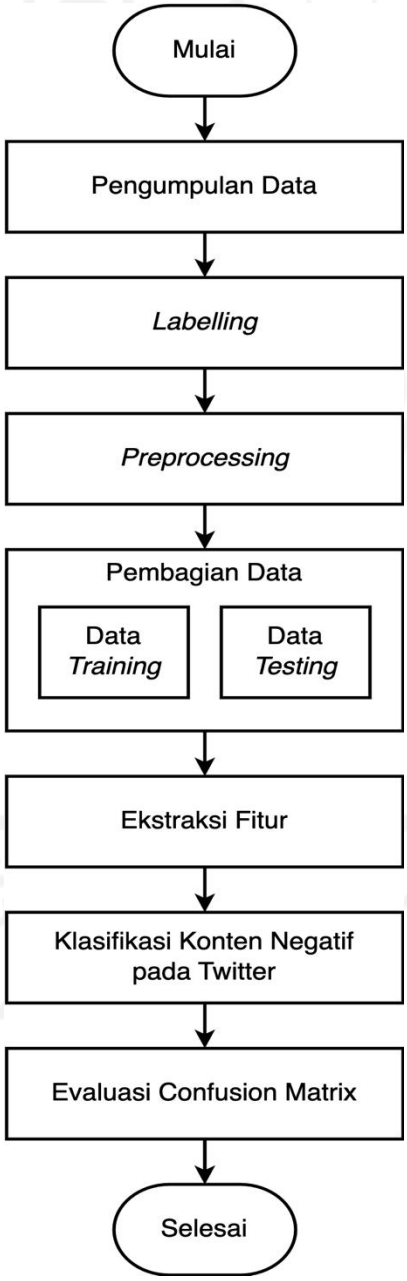
4. *F1-Measure* adalah kombinasi rata-rata *harmonic precision* dan *recall* yang berbanding lurus dengan nilai keduanya (Ramdhani, Andreswari, & Hasibuan, 2018). Rumus untuk menghitung *F1-Measure* ditunjukkan pada Persamaan (13).

$$F1 - Measure = \frac{2 * precision * recall}{precision + recall} \quad (13)$$

3 BAB III
METODOLOGI PENELITIAN

3.1 Langkah-Langkah Penelitian

Langkah-langkah penelitian merupakan rancangan alur yang digunakan untuk gambaran umum terkait dengan alur penelitian yang dilakukan, dari awal hingga akhir penelitian. Langkah-langkah pengerjaan penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Langkah Klasifikasi

Berdasarkan langkah-langkah pada Gambar 3.1, berikut penjelasan setiap langkah penelitian yang dilakukan.

3.1.1 Pengumpulan Data

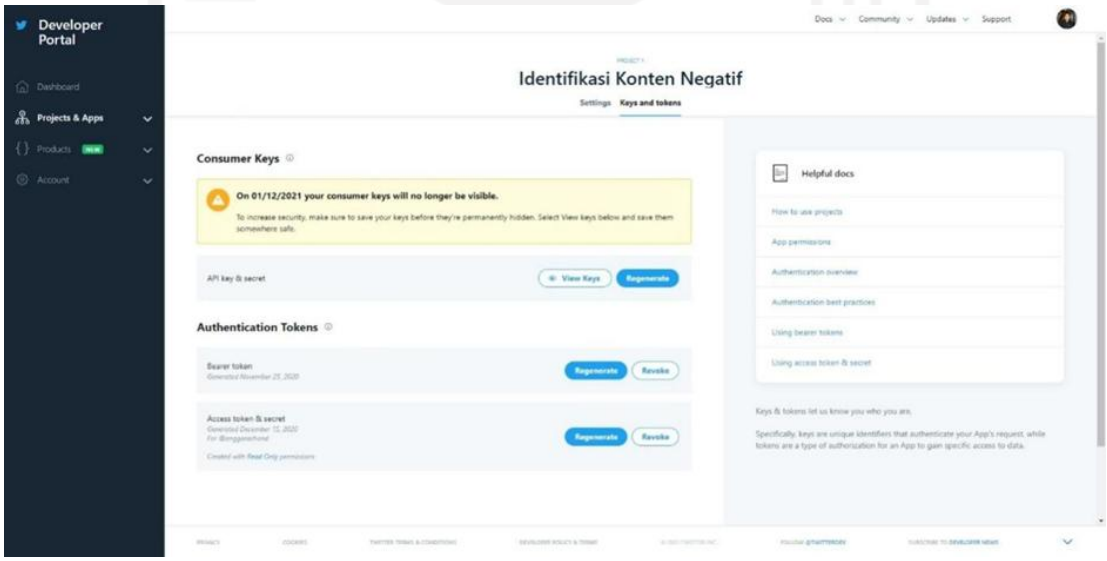
Pada penelitian ini tahap pengumpulan data dilakukan dengan cara mencari studi literatur dan pengambilan data Twitter.

1. Studi literatur

   Pada tahap ini peneliti mencari informasi terkait penelitian identifikasi konten negatif berdasarkan jurnal, buku, dan penelitian-penelitian sebelumnya.

2. Pengambilan Data Twitter

   Pada tahap ini dikumpulkan data Twitter dapat dilakukan dengan teknik *web scrapping* memanfaatkan Twitter API dan *library tweepy* yang disediakan dalam bahasa pemrograman *python*. Dataset yang digunakan dalam penelitian ini adalah data yang dikumpulkan oleh (Abdulloh & Hidayatullah, 2020) , jumlah data yang terkumpul sebanyak 46.970 kumpulan data mentah dari Twitter. Pengambilan data dilakukan pada bulan Desember tahun 2019. Namun, dalam penelitian ini data yang digunakan hanya 3.000 data. Gambar 3.2 merupakan tampilan *dashboard* Twitter Developers, di mana dapat dilakukan proses pengambilan data.



Gambar 3.2 *Dashboard* Twitter Developers

3.1.2 *Labelling*

Pada tahap *labelling,* peneliti menentukan apakah *tweet* tersebut tergolong netral, umpatan, ataupun pornografi. Dalam proses *labelling* ini seharusnya dilakukan oleh seorang ahli di bidangnya dan membutuhkan lebih dari satu orang untuk menghindari perbedaan

pendapat yang sama dalam melakukan sentimen *tweet* dari hasil data *crawling.* Pada penelitian ini, peneliti melabelkan data sebanyak 3.000 data terdiri dari 1.000 data netral, 1.000 data pornografi, dan 1.000 data umpatan. Proses *labelling* dilakukan oleh penulis seorang yang bukan dari pakar bahasa. Konsekuensi proses pelabelan data ini adalah terdapat beberapa *tweet* yang tidak sesuai dengan sentimen analisis sebelumnya. Proses pelabelan data Twitter ini dilakukan dengan cara manual, dan proses ini memakan waktu yang cukup lama. Hal tersebut menjadi kelemahan pada saat jumlah data yang akan diproses pelabelan sangat banyak.

3.1.3 *Preprocessing*

Preprocessing merupakan membersihkan data *tweet* yang terdiri dari kata atau simbol yang tidak penting, sehingga data *tweet* lebih mudah untuk diolah. Adapun langkah yang dilakukan yaitu:

a. Menghilangkan angka.
b. Menghilangkan URL (*Uniform Resource Locator*).
c. Menghilangkan *stopword.*
d. Menghilangkan karakter non-ASCII (*American Standard Code for Information Interchange*).
e. Menghilangkan simbol dan tanda baca.
f. Menghilangkan kata yang terdiri satu huruf.
g. Penyeragaman huruf.
h. Menghapus spasi yang berlebih.
i. Menghilangkan karakter khusus yang ada di Twitter, di antaranya adalah *hashtag* (tanda pagar), *username*, dan RT (*retweet*).

3.1.4 Pembagian Data

Pada tahap ini data akan dibagi menjadi data *training* dan data *testing*. Data *training* adalah data yang digunakan selama proses *training* dengan algoritma CNN dan LSTM, sedangkan data *testing* adalah data yang akan diuji.

3.1.5 Ekstraksi Fitur

Setelah data *tweet* melewati tahap proses *preprocessing* dan bentuknya menjadi lebih terstruktur, tahap selanjutnya adalah ekstraksi fitur. Pada proses ini, *tweet* yang berbentuk

teks akan ditampilkan ke dalam bentuk *vector*. *Tweet* yang berupa *array* akan dijadikan dalam bentuk matriks, di mana baris matriks mewakili baris teks pada dokumen dan kolom matriks mewakili kata-kata pada dokumen. Setelah mengonversi teks menjadi vektor kata, langkah selanjutnya adalah memberikan pembobotan terhadap setiap kata dalam setiap kalimat dengan menggunakan metode *tf-idf*. Tahapan pertama pada metode *tf-idf* yaitu menghitung nilai dari *Term Frequency* (TF) yaitu banyaknya kata baku (*term*) yang muncul dalam *file* dataset. Tahap selanjutnya, setelah mendapat nilai *tf-idf*, untuk mendapatkan bobot akhir dari *tf-idf* diformulasikan dengan rumus $IDF\,(word) = \log\frac{td}{df}$ di mana w ($word_i$) adalah nilai bobot dari setiap kata, TF ($word_i$) adalah hasil dari perhitungan dari perhitungan IDF.

3.1.6 Klasifikasi

Pada tahapan ini dilakukan proses klasifikasi data yang telah melalui tahap ekstraksi fitur. Terdapat dua tahapan dalam proses CNN dan LSTM. Adapun tahapan tersebut adalah sebagai berikut:

1. Pada tahap CNN diterapkan dua tahapan lagi yaitu proses konvolusi dan proses *pooling*. Proses konvolusi memanfaatkan apa yang disebut sebagai *filter*. Seperti layaknya gambar, *filter* memiliki ukuran tinggi, lebar, dan tebal tertentu. *Filter* ini diinisialisasi dengan nilai tertentu, dan nilai dari *filter* inilah yang menjadi parameter yang akan di-*update* dalam proses *learning*. Sedangkan proses *pooling* merupakan proses untuk mereduksi *input* secara spasial dengan operasi *down-sampling*. Umumnya, metode *pooling* yang digunakan adalah *max pooling* atau mengambil nilai terbesar dari bagian tersebut.
2. Hasil dari CNN digunakan sebagai masukan pada LSTM. Pada tahap ini pula data dibagi menjadi dua yaitu data latih (*training*) dan data uji (*test*), dengan menggunakan 80% data *training* sebanyak 2.400 data dan 20% data *test* sebanyak 600 data. Pada tahap tersebut data diklasifikasikan menjadi tiga kelas yaitu konten netral (tidak berisi konten negatif), umpatan, dan konten pornografi.

3.1.7 Evaluasi

Hasil dari implementasi, selanjutnya yaitu dilakukan proses pengujian dengan *confusion matrix*. Evaluasi merupakan tahapan yang bertujuan untuk mengukur kinerja dari model yang telah dibuat. *Confusion matrix* memuat informasi klasifikasi aktual dan prediksi

dari sistem. *Confusion matrix* dapat ditunjukkan pada Tabel 2.2. Berdasarkan nilai dari *confusion matrix* dapat diperoleh nilai *accuracy, precision, recall, dan f1-score* dengan menggunakan persamaan (4), (5), (6), dan (7). Nilai *accuracy, precision, recall, dan f1-score* dapat dihitung sebagai nilai ukur evaluasi terhadap model. *Accuracy* adalah tingkat kedekatan antara nilai prediksi dengan nilai aktual. *Precision* merupakan tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. *Recall* merupakan tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. *F1-score* merupakan perbandingan rata-rata *precision* dan *recall* yang dibobotkan. *F1-score* berguna ketika memiliki jumlah distribusi data yang tidak seimbang. Data yang digunakan dalam pengujian adalah data *tweet* yang merupakan hasil klasifikasi, pengujian dilakukan otomatis oleh sistem.

3.1.8 Deteksi Konten

Setelah melalui semua tahap, maka tahap terakhir adalah deteksi konten. Pada tahap ini peneliti akan mengetahui apakah *tweet* tersebut termasuk ke dalam kategori netral, umpatan, atau pornografi.

4 BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Dataset tweet yang digunakan pada penelitian ini merupakan data yang diperoleh menggunakan teknik *scrapping* dari Twitter. Total data yang didapatkan sejumlah 3.000 data *tweet*.

4.2 *Input* Dataset

Dataset berisi data *training* dan data *testing*. Data *training* adalah data yang digunakan selama proses *training* dengan algoritma CNN dan LSTM, sedangkan data *testing* adalah data yang akan diuji. Untuk data *training* 80%, sedangkan untuk data *testing* 20%, dengan sebanyak 2.400 data *training* dan 600 data *test*. Hal ini dapat dilihat pada kodingan berikut pada Gambar 4.1.

```
# Split data train and data test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=0.2) ## Split data 80:20 -> Random state bisa di atur
jika ingin random
```

Gambar 4.1 Kode Tahapan *Split* Data

Pada Gambar 4.1 kode tahapan *split* data, baris pertama kode merupakan kode untuk mengimport *library* data *train test* yang digunakan, dan di baris kedua merupakan *source code* untuk melakukan peng-*input*-an data *train test.*

Tabel 4.1 merupakan contoh dataset yang digunakan, di mana data yang digunakan adalah beberapa *tweet*, yang mengandung konten pornografi ataupun tidak.

Tabel 4.1 Dataset Ujaran *Tweet*

Doc	Text	Kelas
1	KPPS Diduga Sebar Amplop https://t.co/pPWMdTZ0rv	Netral
2	Tiada Kursi buat Sandi di DKI https://t.co/yOepnxtipa	Netral

3	Sertifikasi Halal Jangan Bebani UMKM https://t.co/aB37HMqBkU	Netral
4	Caleg Stres Bisa Berobat Pakai BPJS https://t.co/GWIu1JkP5V	Netral
5	Punya temen kek pantek Ngasih spoiler suka ga tanggung"	Umpatan
6	@ill_lyana @dents_mazlan @chedetofficial jgn panggil dorg BABI, memang muka dah tebal mcm binatang tu. panggil dorg anak2 keturunan JALANG atau SUNDAL! baru DEEP!	Umpatan
7	@yudharuto Kan semok ga muluk muluk bantet	Pornografi
8	@yudharuto Aku semok lah	Pornografi
9	@pussyhut sempurna...sexy semok...10	Pornografi
10	@bariah_semok @BundaJumbo Mantap Bun	Pornografi

4.3 *Preprocessing*

Preprocessing merupakan data teks yang perlu dibersihkan dan dikodekan ke nilai numerik sebelum diberikan ke model pembelajaran mesin. Pada penelitian dilakukan beberapa tahapan *preprocessing* sebagai berikut:

4.3.1 *Case Folding*

Case folding adalah salah satu bentuk *text preprocessing* yang paling sederhana dan efektif meskipun sering diabaikan. Tujuan dari *case folding* untuk mengubah semua huruf dalam dokumen menjadi huruf kecil. Untuk penjelasan kode program tersebut, di bagian kode program *text lower* digunakan untuk mesin pencarian dokumen yang mengandung "Indonesia". Sedangkan dibagian kode program *text = re.sub* digunakan untuk menghapus angka. Berikut kode program *case folding* pada Gambar 4.2:

```
    def caseFolding(text):
    text = str(text).lower()
    text = ' '.join(re.sub("[@#][A-Za-z0-9_]+|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)+|<.*?>|^https?:\/\/.*[\r\n]*"," ", text).split(' '))
      text = re.sub(r"\d+", "", text)
       return text
```

Gambar 4.2 Kode Tahapan *Case Folding*

Di bagian kode program *lower case* digunakan untuk mengonversi semua huruf besar dalam bentuk *string* menjadi huruf kecil. *Source code* baris ketiga *.join,* berfungsi untuk mengambil semua *item* dalam *iterable* dan menggabungkannya menjadi satu *string*. *Iterable* yaitu objek apapun yang dapat diubah di mana semua nilai yang dikembalikan adalah *string*. Sedangkan di bagian kode program *text = re.sub*, digunakan untuk mengganti satu atau banyak kecocokan dengan *string*.

Adapun contoh hasil dari tahapan *case folding* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil Tahapan *Case Folding*

Doc	Text	Kelas
1	kpps diduga sebar amplop	Netral
2	tiada kursi buat sandi di dki	Netral
3	sertifikasi halal jangan bebani umkm	Netral
4	caleg stres bisa berobat pakai bpjs	Netral
5	punya temen kek pantek ngasih spoiler suka ga tanggung	Umpatan
6	illlyana dentsmazlan chedetofficial jgn panggil dorg babi memang muka dah tebal mcm binatang tu panggil dorg anak keturunan jalang atau sundal baru deep	Umpatan
7	yudharuto kan semok ga muluk muluk bantet	Pornografi
8	yudharuto aku semok lah	Pornografi
9	pussyhut sempurna sexy semok	Pornografi
10	bariah semok bundajumbo mantap bun	Pornografi

4.3.2 *Tokenizing*

Tokenizing adalah proses pemisahan teks menjadi potongan-potongan, seperti kata, angka, simbol, tanda baca dan entitas penting lainnya dapat dianggap sebagai *token.* Berikut kode program *tokenizing* pada Gambar 4.3

```
def tokenizing(text):
    text = nltk.tokenize.word_tokenize(text)
    return text
```

Gambar 4.3 Kode Tahapan *Tokenizing*

Source code 4.3 menjelaskan proses pemisahan teks dengan metode *tokenizing*, di baris pertama '*def tokenizing*' merupakan fungsi pemanggilan proses *tokenizing,* di baris kedua merupakan fungsi proses *tokenizing* dan baris ketiga merupakan fungsi untuk kembali ke teks dan menampilkan proses *tokenizing*. Adapun contoh hasil dari tahapan *tokenizing* dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Tahapan *Tokenizing*

Doc	**Text**	**Kelas**
1	["kpps", "duga", "sebar", "amplop"]	Netral
2	["tiada", "kursi", "buat", "sandi", "dki"]	Netral
3	["sertifikasi", "halal", "jangan", "bebani", "umkm"]	Netral
4	["caleg", "stress", "bisa", "berobat", "pakai", "bpjs"]	Netral
5	["punya", "temen", "kek", "pantek", "ngasih", "spoiler", "suka", "ga", "tanggung"]	Umpatan
6	["jgn", "panggil", "dorg", "babi", "memang", "muka", "dah", "tebal", "mcm", "binatang", "tu", "panggil", "dorg", "anak", "keturunan", "jalang", "atau", "sundal", "baru", "deep"]	Umpatan
7	["semok", "ga", "bantet"]	Pornografi
8	["aku", "semok"]	Pornografi
9	["sempurna", "sexy", "semok"]	Pornografi
10	["bariah", "semok", "mantap"]	Pornografi

4.3.3 *Filtering*

Filtering adalah tahap mengambil kata-kata penting dari hasil *token* dengan menggunakan algoritma *stoplist* (membuang kata kurang penting) atau *wordlist* (menyimpan kata penting). Berikut kode program *filtering token* pada Gambar 4.4.

```
def filtering(token):
    filter_ = [t for t in token if t not in new_stopwords]
    return " ".join(filter_)
```

Gambar 4.4 Kode Tahapan *Filtering*

Source code filtering digunakan untuk menyaring kata yang penting dari *token,* di baris kedua, merupakan fungsi *filter* kata yang disaring dari *token* dengan *stopwords* dan baris ketiga menjelaskan fungsi *return* atau kembali ke fungsi *filtering*. Adapun contoh hasil dari tahapan *filtering* dapat dilihat pada Tabel 4.4.

Tabel 4.4 Hasil Tahapan *Filtering*

Doc	Text	Kelas
1	kpps duga sebar amplop	Netral
2	tiada kursi buat sandi dki	Netral
3	sertifikasi halal jangan bebani umkm	Netral
4	caleg stress bisa berobat pakai bpjs	Netral
5	punya temen pantek ngasih spoiler suka tanggung	Umpatan
6	jgn panggil dorg babi memang muka tebal mcm binatang panggil dorg anak keturunan jalang sundal baru deep	Umpatan
7	semok ga bantet	Pornografi
8	aku semok	Pornografi
9	sempurna sexy semok	Pornografi
10	bariah semok mantap	Pornografi

4.3.4 *Stemming*

Stemming adalah proses menghilangkan infleksi kata ke bentuk dasarnya, namun bentuk dasar tersebut tidak berarti sama dengan akar kata (*root word*). Berikut kode program *stemming* pada Gambar 4.5.

```
def stemming(filtered_token):
    stemmed = stemmer.stem(filtered_token)
    return stemmed
```

Gambar 4.5 Kode Tahapan *Stemming*

Gambar 4.5 merupakan *source code* yang berfungsi untuk memanggil atau membaca hasil tahapan dari *stemming* yang sudah di *train* untuk kemudian ditampilkan.

Adapun contoh hasil dari tahapan *Stemming* dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil Tahapan *Stemming*

Doc	**Text**	**Kelas**
1	kpps duga sebar amplop	Netral
2	tiada kursi buat sandi dki	Netral
3	sertifikasi halal jangan bebani umkm	Netral
4	caleg stress bisa berobat pakai bpjs	Netral
5	punya temen pantek ngasih spoiler suka tanggung	Umpatan
6	jgn panggil dorg babi memang muka tebal mcm binatang panggil dorg anak keturunan jalang sundal baru deep	Umpatan
7	semok ga bantet	Pornografi
8	aku semok	Pornografi
9	sempurna sexy semok	Pornografi
10	bariah semok mantap	Pornografi

4.4 Ekstraksi Fitur

Pada proses ini, *tweet* yang berbentuk teks akan ditampilkan ke dalam bentuk *vector*. *Tweet* yang berupa *array* akan dijadikan dalam bentuk matriks, di mana baris matriks mewakili baris teks pada dokumen dan kolom matriks mewakili kata-kata pada dokumen. Adapun *source code* yang digunakan dapat dilihat pada Gambar 4.6.

```
# proses tfidf
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(norm='l2') # Membuat TF ID-F Vectorizer
response   = vectorizer.fit_transform(X_train) # mendapatkan kata dari data
train
print(response)
```

Gambar 4.6 *Source Code* TF-IDF

Penjelasan *source code* pada Gambar 4.6, pada baris pertama merupakan fungsi untuk meng-*import* TF-IDF *Vectorizer* yang digunakan untuk melakukan perhitungan TF-IDF. Baris merupakan fungsi untuk membuat TF-IDF *Vectorizer,* kemudian pada baris ketiga merupakan fungsi untuk mendapatkan kata dari data *train*.

4.5 Klasifikasi

Klasifikasi dilakukan untuk mendapatkan model terbaik untuk klasifikasi *tweet* yang mengandung umpatan, pornografi, atau *tweet* yang bermuatan netral. Untuk penjelasan kode program proses CNN dan LSTM menggunakan model *sequential*. Dalam *keras* terdapat dua cara dalam membangun model yaitu model *sequential* dan *function API*. *Function API* digunakan untuk menentukan model kompleks. Model yang digunakan adalah *sequential* yang artinya dalam membangun model, tiap *layer* akan disusun *layer* per *layer*. *Layer* yang digunakan seperti *embedding (Embedding), conv1d (Conv1D), max_pooling1d (MaxPooling1D), lstm (*LSTM*), dropout (Dropout), dan dense (Dense)*. Pada LSTM 100 merupakan angka yang menunjukkan banyak *neuron* yang digunakan. Pada *dense layer* 3 merupakan banyaknya kategori *tweet* yang digunakan yaitu, netral, pornografi, dan umpatan. Gambar 4.7 merupakan kode program klasifikasi CNN dan LSTM.

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D,
Dropout, Dense, LSTM
from tensorflow.keras.losses import CategoricalCrossentropy
max_features    = 20000
embedding_dim   = 64
sequence_length = max_len
model = Sequential()
model.add(Embedding(max_features +1, embedding_dim,
input_length=sequence_length,\
embeddings_regularizer = regularizers.l2(0.0005)))

model.add(Conv1D(128,3, activation='relu',\
 kernel_regularizer = regularizers.l2(0.0005),\
 bias_regularizer = regularizers.l2(0.0005)))

model.add(MaxPooling1D())
model.add(LSTM(100)) #LSTM with 100 neutron

model.add(Dropout(0.5))
model.add(Dense(3, activation='sigmoid',
                kernel_regularizer=regularizers.l2(0.001),
                bias_regularizer=regularizers.l2(0.001)))
model.summary()
model.compile(loss=CategoricalCrossentropy(from_logits=True),
optimizer='Adam', metrics=["accuracy"])
```

Gambar 4.7 Kode Tahapan Klasifikasi CNN+LSTM

Penjelasan *source code* Gambar 4.7, di baris pertama merupakan fungsi untuk meng-*import sequential* dari *tensorflow*.keras, kemudian di baris kedua merupakan fungsi untuk meng-*import* data *embedding*, data *Conv1d* dan *pooling*. Baris keempat merupakan fungsi untuk meng-*import* kategori *crossentropy,* di baris ke empat hingga ke sebelas merupakan proses dari peng-*input*-an data, di antaranya nilai *max_features,* nilai *max_features* merupakan nilai bilangan bulat atau ukuran kosakata di mana pada penelitian ini peneliti menggunakan nilai maksimum sebesar 20000, hal tersebut dilakukan karena banyaknya jumlah kata yang digunakan pada penelitian ini. *Hyperparameter* di atas didapat dari *tensorflow.keras* itu sendiri. Kemudian terdapat *embedding_dim*, nilai *embedding_dim* merupakan nilai bilangan bulat dari *dense embedding,* di mana pada penelitian ini peneliti menggunakan dimensi *output* sebesar 64. Kemudian fungsi "model.add(Dense(3,

activation='sigmoid'" merupakan fungsi untuk memproses pengambilan informasi yang mendalam dilakukan dilapisan konvolusi dan *pooling* dan informasi yang diambil ini dilakukan aktivasi untuk penentu kelas yang dilakukan serta dilakukan *regularizers* yang bertujuan untuk menyederhanakan model dengan melakukan pengurangan nilai bobot pada *hidden layer* yang membuat nilai bobot kecil mendekati 0, namun tidak membuat bobot sama dengan 0, dan baris terakhir menggunakan *optimizer* "*Adam*" dan matriks akurasi untuk mendapatkan nilai "*accuracy*".

4.6 Evaluasi

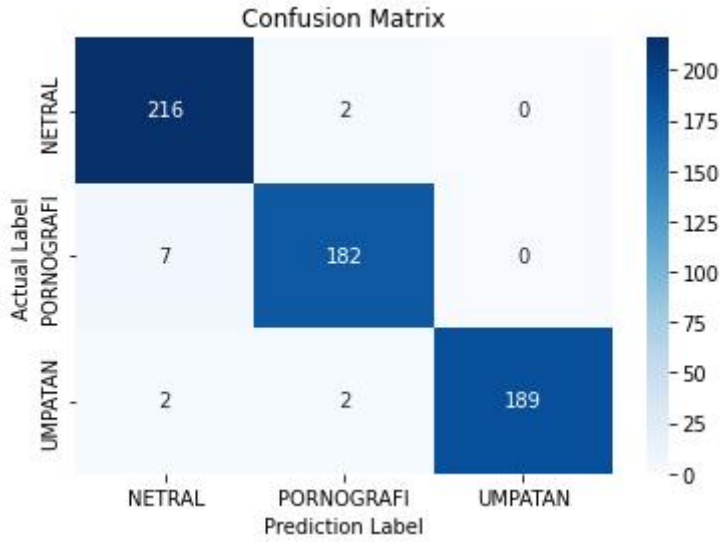
Evaluasi merupakan tahapan yang mempunyai tujuan untuk mengukur kerja dari model yang telah dibuat. Metode evaluasi yang digunakan menggunakan *confusion matrix* dan *epoch* model. Metode ini membagi data ke dalam tiga bagian yakni data latih, data validasi, dan data uji.

Confusion matrix merupakan nilai *True positive* didefinisikan sebagai *positive tupple* yang diklasifikasikan dengan benar oleh model. *True negative* adalah *negative tupple* yang diklasifikasikan dengan benar oleh model. Sementara itu, *false positive* adalah *negative tupple* yang diklasifikasikan sebagai kelas positif oleh model. *False negative* adalah *positive tupple* yang diklasifikasikan sebagai kelas negatif oleh model klasifikasi. Gambar 4.8 merupakan kode program *confusion matrix*

```
print(classification_report(test_data['label'].tolist(),test_data['pred_
label'].tolist()))
```

Gambar 4.8 Kode Tahapan *Confusion Matrix*

Berdasarkan Gambar 4.8 merupakan *source code* yang mana fungsi '*print*' pada *source code* berfungsi untuk mencetak hasil klasifikasi data *test* dengan *confusion matrix.* Adapun tabel *confusion matrix* dapat dilihat pada Gambar 4.9.



Gambar 4.9 *Confusion Matrix*

Diketahui:

1) ***Class Netral***

   a) ***False Negative***

   $FN = (cell\ 2 + cell\ 3)$

   $FN = (2 + 0)$

   $FN = 2$

   b) ***False Positive***

   $FP = (cell\ 4 + cell\ 7)$

   $FP = (7 + 2)$

   $FP = 9$

   c) ***True Negative***

   $TN = (cell\ 5 + cell\ 6 + cell\ 8 + cell\ 9)$

   $TN = (182 ++ 0 + 2 + 189)$

   $TN = 373$

   d) ***True Positive***

   $TP = (cell\ 1)$

   $TP = 216$

2) **Class Pornografi**

   a) ***True Positive***

   $TP = cell\ 5$

   $TP = 182$

b) *False Negative*

$FN = (cell\ 4 + cell\ 6)$

$FN = (7 + 0)$

$FN = 7$

c) *False Positive*

$FP = (cell\ 2 + cell\ 8)$

$FP = (2 + 2)$

$FP = 4$

d) *True Negative*

$TN = (cell\ 1 + cell\ 3 + cell\ 7 + cell\ 9)$

$TN = (216 + 0 + 2 + 189)$

$TN = 407$

3) *Class* Umpatan

a) *True Positive*

$TP = cell\ 9$

$TP = 189$

b) *False Negative*

$FN = (cell\ 4 + cell\ 6)$

$FN = (7 + 0)$

$FN = 7$

c) *False Positive*

$FP = (cell\ 2 + cell\ 8)$

$FP = (2 + 2)$

$FP = 4$

d) *True Negative*

$TN = (cell\ 1 + cell\ 3 + cell\ 7 + cell\ 9)$

$TN = (216 + 0 + 2 + 189)$

$TN = 407$

Jadi:

- TP (Nilai TP dijumlahkan dari “Class Netral, Class Pornografi, Class Umpatan”)
  TP = 216+182+189 = 587
- TN (Nilai TN dijumlahkan dari “Class Netral, Class Pornografi, Class Umpatan”)
  TN = 373+407+407 = 1187
- FP (Nilai FP dijumlahkan dari “Class Netral, Class Pornografi, Class Umpatan”)
  FP = 9+4+4 = 17

- FN (Nilai FN dijumlahkan dari “Class Netral, Class Pornografi, Class Umpatan”)

  FN = 2+7+7 = 16

Mencari Nilai Akurasi:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} * 100\%$$

$$Accuracy = \frac{587 + 1187}{587 + 1187 + 17 + 16} * 100\%$$

$$Accuracy = \frac{1774}{1807} * 100\%$$

$$Accuracy = \ 0{,}98 * 100\%$$

$$Accuracy = \ 98\%$$

Mencari Nilai Presisi:

$$Precision = \frac{TP}{TP + FP} * 100\%$$

$$Precision = \frac{587}{587 + 17} * 100\%$$

$$Precision = \frac{587}{604} * 100\%$$

$$Precision = \ 0{,}97 * 100\%$$

$$Precision = \ 97\%$$

Mencari Nilai *Recall*:

$$Recall = \frac{TP}{TP + FN} * 100\%$$

$$Recall = \frac{587}{587 + 16} * 100\%$$

$$Recall = \frac{587}{603} * 100\%$$

$$Recall = \ 0{,}97 * 100\%$$

$$Recall = \ 97\%$$

$$F1 - Measure = \ \frac{2 * 97 * 97}{97 + 97}$$

$$F1 - Measure = \ \frac{18818}{194}$$

$$F1 - Measure = \ \frac{18818}{194}$$

$$F1 - Measure = \ 97\%$$

Dalam melakukan *fit* model, peneliti menggunakan *epochs*-nya = 100 *epochs*. *Epoch* berarti berapa kali jaringan akan melihat seluruh kumpulan data, dan untuk *batch_size* = 128, yang artinya *batch_size* adalah jumlah contoh pelatihan dalam satu *forward* atau *backward pass*. Semakin tinggi nilai *batch_size*, maka akan semakin banyak memori yang dibutuhkan. Berikut kode program *epochs*:

```
epochs     = 100
batch_size = 128
# Fit the model using the train and test datasets.
history = model.fit(train_ds.batch(batch_size),
                    epochs= epochs ,
                    validation_data=valid_ds.batch(batch_size),
                    verbose=1)
```

Gambar 4.10 Kode Tahapan *Epoch* Model

Source code Gambar 4.10 merupakan proses dari *epoch* model, di baris pertama dan baris kedua merupakan nilai dari *epochs* dan *batch_size*. Di baris ketiga merupakan fungsi untuk menghitung *epoch*, '*validation_data*' merupakan fungsi untuk memvalidasi data *batch_size* hingga mendapatkan nilai = 1.

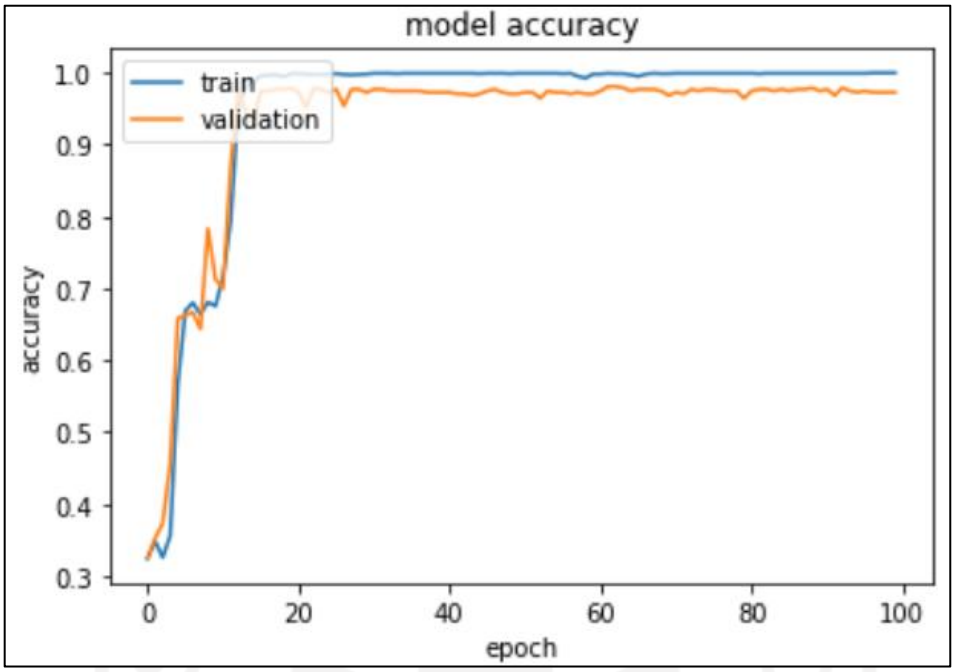
```
# list all data in history
print(history.history.keys())

# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

Gambar 4.11 Kode Tahapan *Accuracy* Model

Berdasarkan Gambar 4.11 dapat dijelaskan bahwa kode di atas merupakan fungsi yang digunakan untuk menampilkan data ke dalam bentuk grafik. Baris pertama pada *source code* berfungsi untuk mencetak data riwayat akurasi, kemudian baris kedua merupakan fungsi

untuk menampilkan model akurasi yang sudah di *train*, seperti hasil *epoch*, dan akurasi. Adapun hasil tampilan grafik dari *accuracy* model dapat dilihat pada Gambar 4.12.



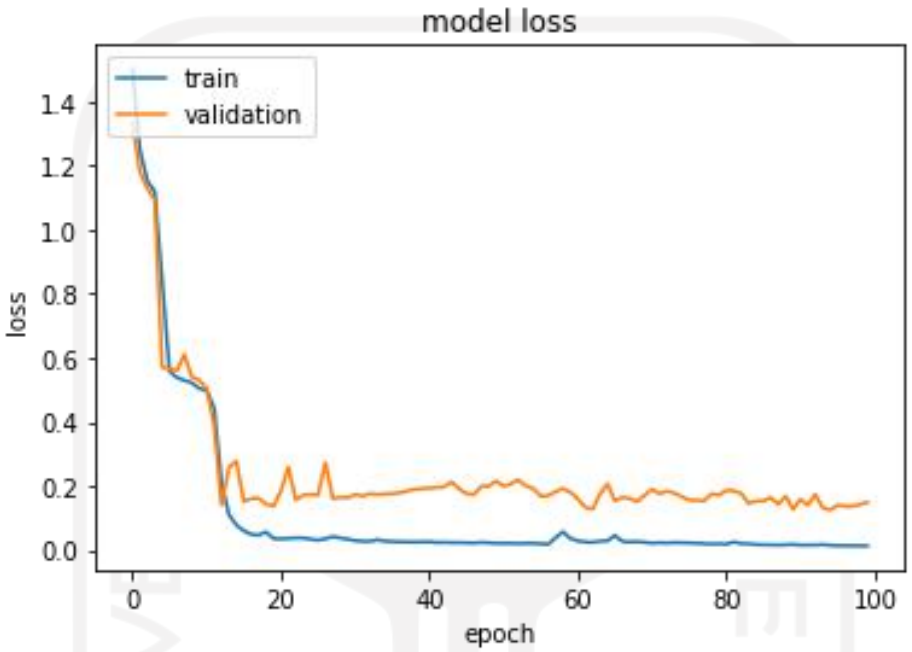
Gambar 4.12 Hasil *Accuracy* Model CNN + LSTM

Pada Gambar 4.12 menghasilkan nilai akurasi dan *loss* pada data *training* dan validasi. Proses *training* menggunakan jumlah *epoch* sebanyak 100 dan *batch size* sebanyak 128. Dengan parameter yang sudah ditentukan akan dilihat berapa akurasi dari data *training* dan melihat kenaikan nilai *loss*. Model akan menyimpan *epoch* yang optimal pada nilai *loss* yang tinggi selama proses *epoch* berlangsung. Untuk nilai akurasi yang diperoleh sebesar 0.98, dan nilai *loss* mengalami kenaikan *epoch* ke-100.

```
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

Gambar 4.13 Kode Tahapan *Loss* Model

Berdasarkan Gambar 4.13 dapat dijelakan bahwa kode di atas merupakan fungsi yang digunakan untuk menampilkan data model *loss* kedalam bentuk grafik, *loss* model merupakan nilai dari penghitungan *loss function* dari *training dataset*, lalu ditampilkan dalam bentuk grafik sehingga diketahui berapa penurunan nilai *loss* atau dapat diartikan fungsi yang digunakan untuk melihat performa dari model CNN. Adapun hasil tampilan grafik dari model *loss* dapat dilihat pada Gambar 4.14.



Gambar 4.14 Hasil *Loss* Model

Gambar 4.14 adalah grafik pada saat *training* dari model yang dibuat dengan algoritma CNN dan LSTM. Dilihat dari gambar tersebut, kedua algoritma CNN dan LSTM mengalami penurunan nilai *loss* pada *validation set* dan mengalami berbanding lurus dengan *training loss* pada *epoch* ke-100. Nilai akurasi yang diperoleh sebesar 1.45, dan nilai *loss* nya mengalami penurunan *epoch* ke-100.

4.7 Deteksi Konten

CNN dan LSTM merupakan gabungan model dengan menghasilkan akurasi terbaik, maka langkah selanjutnya adalah menguji kalimat apakah termasuk ke dalam kategori netral, pornografi, atau umpatan. Berikut hasil deteksi konten untuk kategori kata yang pornografi dapat dilihat pada Gambar 4.15.

```
HTML(value='Form Klasifikasi kata')

Text(value='', placeholder='Type Here!')

Button(description='Klasifikasi', style=ButtonStyle())

['berita kampus tantang deddy corbuzier lucinta luna bugil']
kalimat Awal: berita kampus tantang deddy corbuzier lucinta luna bugil
kalimat stem= ['berita' 'bugil' 'corbuzier' 'deddy' 'kampus' 'lucinta' 'luna' 'tantang']
hasil tf-idf=   (0, 1)  0.35355339059327373
  (0, 6)        0.35355339059327373
  (0, 5)        0.35355339059327373
  (0, 2)        0.35355339059327373
  (0, 3)        0.35355339059327373
  (0, 7)        0.35355339059327373
  (0, 4)        0.35355339059327373
  (0, 0)        0.35355339059327373
hasil sequnce= [[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 2 3 4 5 6 7 8 9]]
[[0.12710601 0.99858606 0.04981852]]
pornografi
```

Gambar 4.15 Hasil Deteksi Konten Pornografi

Berdasarkan Gambar 4.15 data yang di *input* merupakan data berupa kalimat yang kemudian terdeteksi sebagai konten pornografi dikarenakan memiliki nilai probabilitas 0.99858606. Sedangkan dengan hasil deteksi konten dengan kategori netral dapat dilihat pada Gambar 4.15.

```
[0]
['sentuh level psikologis nampaknya ihsg coba uji kuat']
kalimat Awal: Setelah menyentuh level psikologis di 6,500 nampaknya IHSG akan mencoba menguji kekuatan tersebut. h
ttps://t.co/yKG8DQI9LW
hasil sequnce : [[1047  631    1 2942  975  190 1735  120    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0]]
[[0.99787533 0.07211401 0.04806   ]]
NETRAL
```

Gambar 4.16 Hasil Deteksi Konten Netral

Berdasarkan Gambar 4.16 data yang di *input* merupakan data berupa kalimat yang kemudian terdeteksi sebagai konten netral dikarenakan memiliki nilai probabilitas 0.99787533.

Untuk hasil deteksi konten dengan kategori umpatan memperoleh hasil seperti yang dapat dilihat pada Gambar 4.17.

```
['gue udah sempet koar temen bhw gue pilih psi imbang kuat parlemen program amp manuver psi seksi utk lirik pd tit
ik si wanita sundal kicau anti poligami tolak perda syariah sungguh sakit']
kalimat Awal: @nutdn @greynatalino @shaniason @atsariats @psi_id Gue udah sempet koar2 sama temen2 bhw gue bakalan
pilih PSI, supaya ada perimbangan kekuatan di parlemen. Program &amp;amp; manuver PSI cukup seksi utk dilirik waktu it
u. Sampai pd titik ketika si wanita sundal itu berkicau soal anti poligami dan menolak perda syariah. Sungguh saki
t!
hasil sequnce : [[  31   34    1 1590  418    1   31   36  534 4472  120  962  811   91
     1  534  388  588 2209    1 5867   41  601    8    1 1078  871  793
     1 1872 1162  159    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0]]
[[0.18627779 0.17987242 0.9114275 ]]
UMPATAN
```

Gambar 4.17 Hasil Deteksi Konten Umpatan

Berdasarkan Gambar 4.17 data yang di *input* terdeteksi konten umpatan dikarenakan memiliki nilai probabilitas 0.9114275.

Berdasarkan hasil deteksi konten di atas, semua kalimat yang diuji sudah tepat. Akan tetapi, penelitian ini masih ada kekurangan yaitu beberapa kalimat (*tweet*) yang dimaknai tidak semestinya. Hal tersebut terjadi karena pada dataset ini lebih banyak pada konten netral dibandingkan dengan pornografi dan umpatan.

5 BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil yang telah didapatkan, penelitian ini telah berhasil melakukan identifikasi *tweet* pada Twitter dengan klasifikasi antara tiga kelas: netral, pornografi, dan umpatan yang tersedia pada *dataset* menggunakan algoritma CNN dan LSTM *(Long Short-Term Memory).*

1. Penelitian ini berhasil melakukan klasifikasi terhadap tiga jenis konten *tweet* dengan gabungan model CNN dan LSTM untuk mendeteksi *tweet* yang mengandung konten netral, pornografi, dan umpatan. Dalam melakukan klasifikasi *tweet* yang mengandung ketiga konten diatas, langkah yang perlu dilakukan yaitu pengumpulan data, *preprocessing,* klasifikasi dan evaluasi menggunakan *epoch* model dan *confusion matrix*.
2. Penelitian ini menghasilkan performa klasifikasi yang baik karena dari evaluasi *epoch* yang digunakan menghasilkan tingkat akurasi sebesar 100%, hal ini dapat dilihat pada *epoch* ke 97-100 dan hasil *confusion matrix* dengan rata-rata *precision* untuk kelas netral sebesar 0.99, *recall* sebesar 0.99, dan *f1-score* sebesar 0.99. Sedangkan untuk kelas pornografi, rata-rata *precision* sebesar 0.99, *recall* sebesar 0.98, dan untuk *f1-score* sebesar 0.99. Untuk kelas umpatan, rata-rata *precision* sebesar 1.00, *recall* sebesar 1.00, dan *f1-score* sebesar 1.00.

5.2 Saran

Penelitian ini masih menggunakan *tweet* berbahasa Indonesia. Peneliti berharap penelitian ini dapat dikembangkan oleh peneliti-peneliti di masa mendatang dengan beberapa saran seperti:

1. Penambahan selain proses klasifikasi dan mencoba menggunakan model lain pada klasifikasi, agar dapat membandingkan model mana yang lebih baik.
2. Menggunakan pakar bahasa pada saat proses *labelling*, agar menghindari perbedaan pendapat yang sama dalam melakukan sentimen *tweet* dari hasil data *crawling.*

.

DAFTAR PUSTAKA

Abdulloh, N., & Hidayatullah, A. (2020). Deteksi Cyberbullying pada Cuitan Media Sosial Twitter. *AUTOMATA 1.1*.

Abdulrahman, A., & Baykara, M. (2020). Fake News Detection Using Machine Learning and Deep Learning Algorithms. *International Conference on Advanced Science and Engineering (ICOASE). IEEE*.

Amelia, D., Aldino, A., & Isnain, A. (2021). Teks Dan Analisis Sentimen Pada Chat Grup Whatsapp Menggunakan Long Short Term Memory (LSTM). *Jurnal Teknologi dan Sistem Informasi 2.4*, 56-61.

Andraini, F., & Mahdiyah, E. (2022). Analisis Sentimen Twitter Terhadap Peperangan Rusia Dan Ukraina Menggunakan Algoritma Support Vector Machine. *Jurnal Aplikasi Komputer (JAK)* , 2(1), pp. 48-58.

Arham, A. (2018). Klasifikasi Ulasan Buku Menggunakan Algoritma Convolutional Neural Network-Long Short Term Memory. *Diss. Institut Teknologi Sepuluh Nopember*.

Badjrie, S., Pratiwi, O., & Anggana, H. (2021). Analisis Sentimen Review Customer Terhadap Produk Indihome Dan First Media Menggunakan Algoritma Convolutional Neural Network. *eProceedings of Engineering 8.5*, Vol.8, No.5.

Berliana, G., Shaufiah, S., & Sa'adah, S. (2018). Klasifikasi Posting Tweet Mengenai Kebijakan Pemerintah Menggunakan Naïve Bayesian Classification. *eProceedings of Engineering 5.1*.

Dwitama, A., & Hidayat. , S. (2021). Identifikasi Ujaran Kebencian Multilabel Pada Teks Twitter Berbahasa Indonesia Menggunakan Convolution Neural Network. *Jurnal Sistem Komputer dan Informatika (JSON) 3.2*, 117-127.

Fadli, H., & Hidayatullah, A. (2021). Identifikasi Cyberbullying pada Media Sosial Twitter Menggunakan Metode LSTM dan BiLSTM. *AUTOMATA 2.1*.

Hastomo, W., & Satyo, A. (2019). Long Short Term Memory Machine Learning Untuk Memprediksi Akurasi Nilai Tukar IDR Terhadap USD. *Prosiding SeNTIK 3.1, 7*(4), 115-124.

Hermanto, D., Setyanto, A., & Luthfi, E. (2021). Algoritma LSTM-CNN untuk Binary Klasifikasi dengan Word2vec pada Media Online. *Creative Information Technology Journal 8.1*, 64-77.

Hidayatullah, A., Hakim, A., & Sembada, A. (2019). Adult Content Classification on Indonesian Tweets using LSTM Neural Network. *International Conference on Advanced Computer Science and information Systems (ICACSIS). IEEE*.

Ihsan, F., Iskandar, I., Harahap, N. S., & Agustian, S. (2021). Algoritme decision tree untuk mendeteksi ujaran kebencian dan bahasa kasar. *Jurnal Teknologi dan Sistem Komputer*, vol. 9, No. 4, pp. 199-204.

Juliano, A., & Firdaus, F. (2019). Klasifikasi Atrial Fibrillation Pada Ekg Dengan Menggunakan Convolutional Neural Network (CNN). *Diss. Sriwijaya University*.

Naf'an, M., Burhanuddin, A., & Riyani, A. (2019). Penerapan Cosine Similarity dan Pembobotan TF-IDF untuk Mendeteksi Kemiripan Dokumen. *Jurnal Linguistik Komputasional 2.1*, 23-27.

Nugroho dkk, D. G. (2016). Analisis Sentimen Pada Jasa Ojek Online Menggunakan Metode Naive Bayes. *Prosiding SNST Fakultas Teknik 1.1*.

Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (CNN) Pada Ekspresi Manusia. *JURNAL ALGOR*, 2(1), pp. 21-21.

Ramdhani, N. (2021). Analisis Sentimen Pengguna Twitter Terhadap Belajar Daring Selama Pandemi Covid-19 Dengan Deep Learning. *Jurnal Siliwangi Seri Sains dan Teknologi 7.2*.

Ramdhani, S. L., Andreswari, R., & Hasibuan, M. A. (2018). Sentiment Analysis of Product Reviews using Naive Bayes Algorithm: A Case Study. *Sentiment Analysis of Product Reviews using Naive Bayes Algorithm: A Case Study*.

Simatupang, M., & Utomo, D. (2019). Analisa Testimonial Dengan Menggunakan Algoritma Text Mining Dan Term Frequency-Inverse Document Frequence (Tf-Idf) Pada Toko Allmeeart. *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer) 3.1*.

Sudiantoro, A., & Zuliarso, E. (2018). Analisis sentimen twitter menggunakan text mining dengan algoritma Naïve Bayes Classifier.

Tjut Awaliyah Zuraiyah, D. K. (2017). IMPLEMENTASI CHATBOT PADA PENDAFTARAN MAHASISWA BARU MENGGUNAKAN RECURRENT NEURAL NETWORK.

Triakno, N. (2018). Implementasi deep learning untuk image classification menggunakan algoritma Convolutional Neural Network (CNN) pada citra wayang golek. *Universitas Islam Indonesia, Yogyakarta*.

Utama, P. (2018). Identifikasi Hoax pada Media Sosial dengan Pendekatan Machine Learning. *Widya Duta: Jurnal Ilmiah Ilmu Agama dan Ilmu Sosial Budaya 13.1*, 69-76.

Zamil, M. (2019). Klasifikasi Kalimat Ofensif pada Media Sosial Twitter Menggunakan Metode Naive Bayes Classifier. *Diss. universitas Islam Negeri Sultan Syarif Kasim Riau*.

LAMPIRAN

Dalam melakukan penelitian ini, peneliti membuat tahapan atau *flowchart* terkait penerapan CNN dan LSTM pada Identifikasi Konten Negatif Pada Twitter yang dapat dilihat pada Gambar 3.1. Dari Gambar 3.1 dapat dijelaskan sebagai berikut:

1. **Pengumpulan Data**

   Dataset *tweet* yang digunakan pada penelitian ini merupakan data yang diperoleh menggunakan teknik *scrapping* dari Twitter. Total data yang didapatkan sejumlah 3.000 data *tweet*. Hasil dari pengumpulan data ini akan digunakan sebagai *input* dalam penelitian ini. Tabel 1 merupakan data sampel dari hasil pengumpulan data.

Tabel 1 Dataset Sampel

Doc	Text
1	KPPS Diduga Sebar Amplop https://t.co/pPWMdTZ0rv
2	Tiada Kursi buat Sandi di DKI https://t.co/yOepnxtipa
3	Sertifikasi Halal Jangan Bebani UMKM https://t.co/aB37HMqBkU
4	@yudharuto Kan semok ga muluk muluk bantet
5	@yudharuto Aku semok lah
6	@pussyhut sempurna...sexy semok...10

2. ***Labelling***

   Setelah mendapatkan data, selanjutnya adalah dilakukan pelabelan. Pada tahap ini dilakukan pelabelan netral, ponografi dan umpatan, namun dalam contoh perhitungan ini menggunakan kelas netral dan pornografi.

Tabel 2 *Labelling* Data

Doc	Text	Kelas
1	KPPS Diduga Sebar Amplop https://t.co/pPWMdTZ0rv	Netral
2	Tiada Kursi buat Sandi di DKI https://t.co/yOepnxtipa	Netral
3	Sertifikasi Halal Jangan Bebani UMKM https://t.co/aB37HMqBkU	Netral
4	@yudharuto Kan semok ga muluk muluk bantet	Pornografi
5	@yudharuto Aku semok lah	Pornografi
6	@pussyhut sempurna...sexy semok...10	Pornografi

3. ***Preprocessing***

Preprocessing merupakan data teks yang perlu dibersihkan dan dikodekan ke nilai numerik sebelum diberikan ke model pembelajaran mesin. Pada penelitian dilakukan tahapan *preprocessing* seperti tahap *case folding, tokenizing, filtering* dan *stemming*.

a. *Case Folding*

Case folding bertujuan untuk mengubah semua huruf dalam dokumen menjadi huruf kecil.

Tabel 3. *Case Folding*

Doc	**Text**	***Case-Folding***
1	KPPS Diduga Sebar Amplop https://t.co/pPWMdTZ0rv	kpps diduga sebar amplop
2	Tiada Kursi buat Sandi di DKI https://t.co/yOepnxtipa	tiada kursi buat sandi di dki
3	Sertifikasi Halal Jangan Bebani UMKM https://t.co/aB37HMqBkU	sertifikasi halal jangan bebani umkm
4	@yudharuto Kan semok ga muluk muluk bantet	yudharuto kan semok ga muluk muluk bantet
5	@yudharuto Aku semok lah	yudharuto aku semok lah
6	@pussyhut sempurna...sexy semok...10	pussyhut sempurna sexy semok

b. *Tokenizing*

Tokenizing bertujuan untuk pemisahan teks menjadi potongan-potongan, seperti kata, angka, simbol, tanda baca dan entitas penting lainnya dapat dianggap sebagai *token*.

Tabel 4 *Tokenizing*

Case-Folding	***Tokenizing***
kpps diduga sebar amplop	[“kpps”, “duga”, “sebar”, “amplop”]
tiada kursi buat sandi di dki	[“tiada”, “kursi”, “buat”, “sandi”, “dki”]

sertifikasi halal jangan bebani umkm	[“sertifikasi”, “halal”, “jangan”, “bebani”, “umkm”]
yudharuto kan semok ga muluk muluk bantet	[“semok”, “ga”, “bantet”]
yudharuto aku semok lah	[“aku”, “semok”]
pussyhut sempurna sexy semok	[“sempurna”, “sexy”, “semok”]

c. *Filtering*

Filtering bertujuan mengambil kata-kata penting dari hasil *token* dengan menggunakan algoritma *stoplist* (membuang kata kurang penting) atau *wordlist* (menyimpan kata penting).

Tabel 5 *Filtering*

Tokenizing	***Filtering***
[“kpps”, “duga”, “sebar”, “amplop”]	kpps duga sebar amplop
[“tiada”, “kursi”, “buat”, “sandi”, “dki”]	tiada kursi buat sandi dki
[“sertifikasi”, “halal”, “jangan”, “bebani”, “umkm”]	sertifikasi halal jangan bebani umkm
[“semok”, “ga”, “bantet”]	semok ga bantet
[“aku”, “semok”]	aku semok
[“sempurna”, “sexy”, “semok”]	sempurna sexy semok

d. *Stemming*

Stemming adalah proses menghilangkan infleksi kata ke bentuk dasarnya, namun bentuk dasar tersebut tidak berarti sama dengan akar kata (*root word*).

Tabel 6 *Stemming*

Filtering	***Stemming***
kpps duga sebar amplop	kpps duga sebar amplop
tiada kursi buat sandi dki	tiada kursi buat sandi dki
sertifikasi halal jangan bebani umkm	sertifikasi halal jangan bebani umkm
semok ga bantet	semok ga bantet
aku semok	aku semok

sempurna sexy semok	sempurna sexy semok

4. ***Split* Data**

Pembagian data menjadi data *training* dan data *testing* dengan perbandingan data *training* dan *test* adalah 80% (data *train*) dan 20% (data *testing*)

Tabel 7 *Split* Data

Doc	Stemming	Kelas
1	kpps duga sebar amplop	Netral
2	tiada kursi buat sandi dki	Netral
3	sertifikasi halal jangan bebani umkm	Netral
4	semok ga bantet	Pornografi
5	aku semok	Pornografi
6	sempurna sexy semok	Pornografi

Tabel 8 Keterangan *Split* Data

	Data *Train*
	Data *Test*

5. **Ekstrasi Fitur Menggunakan TF-IDF**

Pada perhitungan TF-IDF data yang akan digunakan adalah data *training*, sedangkan perhitungan TF-IDF untuk data *testing* akan menggunakan df_t dan idf_t yang dihasilkan pada data *training*. df_t merupakan jumlah dokumen yang mengandung *term* *t* dan idf_t merupakan kebalikan dari df_t.

a. Menghitung TF

Perhitungan TF dilakukan dengan menghitung kemunculan setiap *term* (kata) pada tiap dokumen. Contoh kata **aku** di D1 = 0, yang artinya **aku** di D1 memiliki frekuensi kemunculan 0.

Tabel 9 Hasil Perhitungan *Term* untuk Setiap Dokumen

***Term Frequency* (TF)**				
	D1	**D2**	**D3**	**D4**
aku	0	0	0	1

amplop	1	0	0	0
bantet	0	0	1	0
buat	0	1	0	0
dki	0	1	0	0
duga	1	0	0	0
ga	0	0	1	0
kpps	1	0	0	0
kursi	0	1	0	0
sandi	0	1	0	0
sebar	1	0	0	0
semok	0	0	1	1
tiada	0	1	0	0

b. Menghitung IDF

Perhitungan IDF memiliki beberapa cara, namun pada *sklearn* (TF-IDF *Vectorizer*) sendiri terdapat perbedaan formula yang digunakan jika menggunakan L2 *norm* (merupakan *default*). Formula pada IDF pada *sklearn* sendiri sebagai berikut (*sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 1.1.3 documentation*, n.d.).

$$IDF = LN\left(\frac{1+n}{1+DF(t,d)}\right)+1$$

Keterangan:

n = Total dokumen

DF(t,d) = Total kemunculan kata di seluruh dokumen

Tabel 10 Hasil Perhitungan IDF Setiap *Term*

	DF	IDF
aku	1	1.916291
amplop	1	1.916291
bantet	1	1.916291
buat	1	1.916291
dki	1	1.916291

duga	1	1.916291
ga	1	1.916291
kpps	1	1.916291
kursi	1	1.916291
sandi	1	1.916291
sebar	1	1.916291
semok	2	1.510826
tiada	1	1.916291

Nilai DF(t,d) didapatkan dengan menghitung (*count*) total kemunculan kata di seluruh dokumen. Contoh pada kata **aku** memiliki DF(t,d) = 1, artinya kata **aku** muncul di 1 dokumen (perhatikan tabel TF, kemunculan kata **aku** terdapat di D4) dan untuk kata lainnnya untuk mendapatkan DF(t,d) seperti halnya kata **aku.** Untuk menghitung nilai IDF tinggal memasukkan dengan rumus yang dijelaskan di atas.

$IDF_{aku} = LN\left(\frac{1+4}{1+1}\right) + 1 = 1.916291$

$IDF_{amplop} = LN\left(\frac{1+4}{1+1}\right) + 1 = 1.916291$

$IDF_{bantet} = LN\left(\frac{1+4}{1+1}\right) + 1 = 1.916291$

c. Menghitung TF-IDF

Perhitungan TF-IDF didapatkan dari mengkalikan hasil TF dengan IDF yang didapatkan. Formula dari TF-IDF dapat dilihat pada persamaan rumus (3).

Tabel 11 Hasil Perhitungan TF-IDF Setiap *Term*

Term Frequency Inverse Document (TF-IDF)				
D1	**D2**	**D3**	**D4**	
0	0	0	1.916290732	**aku**
1.916291	0	0	0	**amplop**
0	0	1.916291	0	**bantet**
0	1.916291	0	0	**buat**
0	1.916291	0	0	**dki**
1.916291	0	0	0	**duga**

0	0	1.916291	0	**ga**
1.916291	0	0	0	**kpps**
0	1.916291	0	0	**kursi**
0	1.916291	0	0	**sandi**
1.916291	0	0	0	**sebar**
0	0	1.510826	1.510825624	**semok**
0	1.916291	0	0	**tiada**

Contoh perhitungan adalah sebagai berikut:

$TF - IDF_{aku,D1} = 0 * 1.916 = 0$

$TF - IDF_{aku,D2} = 0 * 1.916 = 0$

$TF - IDF_{aku,D3} = 1 * 1.916 = 0$

d. Perhitungan L2 *Norm*

Perhitungan L2 *norm* merupakan *default* yang ditetapkan oleh *sklearn* pada saat menggunakan TF-IDF *Vectorizer*. Formula dari L2 *norm* sendiri adalah sebagai berikut (*L^2-Norm -- from Wolfram MathWorld*, n.d.)

$$||X|| = \sqrt{X_1^2 + X_2^2 + .. + X_n^2}$$

Untuk melakukan perhitungan L2, perlu dilakukan *transpose matrix* dari hasil TF-IDF sebelumnya, sehingga hasil lengkap dari L2 *norm* sendiri untuk setiap dokumen adalah sebagai berikut:

Tabel 12 L2 *Norm*

D1	3.832581
D2	4.284956
D3	3.10273
D4	2.440239

Contoh perhitungan:

$||D1|| = \sqrt{0^2 + 1.916290732^2 + 0^2 + ..}$

$||D1|| = 3.832581464$

Begitu juga dengan menghitung L2 *Norm* pada dokumen lainnya. Selanjutnya menghitung hasil TF-IDF L2 *Norm*.

e. Perhitungan TF-IDF L2 *Norm*.

Perhitungan TF-IDF L2 *Norm* didapatkan dengan cara membagi setiap hasil TF-IDF *term* dengan nilai L2 tiap dokumen. Sehingga didapatkan hasil seperti berikut:

Tabel 13 TF-IDF L2 *Norm*

	D1	D2	D3	D4
aku	0	0	0	0.785288
amplop	0.5	0	0	0
bantet	0	0	0.617614	0
buat	0	0.447214	0	0
dki	0	0.447214	0	0
duga	0.5	0	0	0
ga	0	0	0.617614	0
kpps	0.5	0	0	0
kursi	0	0.447214	0	0
sandi	0	0.447214	0	0
sebar	0.5	0	0	0
semok	0	0	0.486934	0.61913
tiada	0	0.447214	0	0

Contoh perhitungan:

$$TF - IDF\ L2\ Norm_{aku,D1} = \frac{0}{1.916} = 0$$

Pada perhitungan data *testing* untuk TF-IDF mendapatkan hasil sebagai berikut:

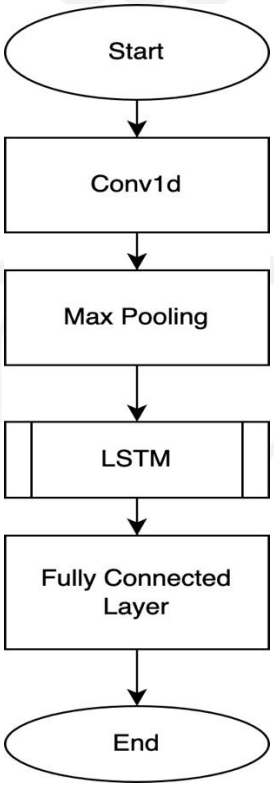
Tabel 14 TF-IDF Data *Testing*

	D5	**D6**
aku	0	0
amplop	0	0
bantet	0	0
buat	0	0
dki	0	0
duga	0	0

ga	0	0
kpps	0	0
kursi	0	0
sandi	0	0
sebar	0	0
semok	1	0
tiada	0	0

6. Klasifikasi Konten Negatif pada Twitter

   Proses klasifikasi akan dilakukan dengan menggunakan hasil TF-IDF sebelumnya. Pada TF-IDF memiliki ukuran *vector* yang sama tiap datanya, sehingga tidak diperlukan *tokenizer*, *text to sequences* dan *padded sequences*, di mana ketiga proses tersebut berguna mengubah kata menjadi kumpulan *vector* yang *output*nya memiliki panjang *vector* yang sama untuk tiap ukuran data yang dimiliki. Pada TF-IDF *output* keluarannya merupakan hasil *weight* dari proses TF-IDF itu sendiri, sehingga proses *input* ke dalam *deep neural network* (CNN dan LSTM) tidak diperlukan *embedding layer*.



Gambar 1 Proses Klasifikasi CNN + LSTM

Dari Gambar 1, terdapat tahapan-tahapan proses klasifikasi CNN + LSTM yang dapat dijelaskan sebagai berikut:

a. Conv1D

CNN memiliki performa yang lebih baik melakukan ekstraksi fitur daripada LSTM (Widhiyasana et al., 2021), sehingga pada tahapan gabungan antara CNN + LSTM, fungsi CNN digunakan melakukan ekstraksi fitur menggunakan lapisan konvolusi dan lapisan *pooling*. Pada CNN terdapat banyak *convolutional* salah satunya 1D. *Convolutional* 1D atau yang disingkat Conv1D merupakan *layer* yang lebih cocok digunakan untuk permasalahan *Natural Language Processing*, hal ini dikarenakan Conv1D menerima *input*-an 1 dimensional dan jika masalah NLP menggunakan Conv2D maka tidak cocok hal ini dikarenakan Conv2D menerima *input*-an 2 dimensi (Shorten et al., 2021) (Yan et al., 2021). Conv1D memiliki 2 *input*-an penting yaitu ukuran *kernel* (dimensi *kernel* yang dikenal dengan *filter kernel* di mana pada tahapan ini akan menggunakan 1x2) dan aktivasi (fungsi digunakan untuk melakukan hasil aktivasi pada perkalian antar *weight* TF-IDF dengan *kernel* Conv1D (yang didapatkan secara *random* dengan rentang antara -1 hingga tidak terbatas) dan fungsi aktivasi yang digunakan adalah Relu Non-Linarity yang memiliki rumus seperti berikut (Firmansyah & Hayadi, 2022).

$$ReLu = MAX(x, 0)$$

$$X = Wi * Xj$$

Keterangan

X = Hasil perkalian *kernel* dengan *weight* TF-IDF

MAX = Mencari nilai terbesar antara x dengan 0

Wi = Bobot *kernel*

Xj = Nilai *input*-an *weight* TF-IDF

$ReLu$ = Fungsi aktivasi

Fungsi *kernel* hasil dari Conv1D sebagai berikut:

Tabel 15 Nilai *Filter Kernel*

-0.5390143	-0.69772494

Nilai *filter kernel* sendiri ditentukan berdasarkan ukuran matriksnya. Pada hasil diatas ukuran matriks *kernel*-nya adalah 1x2 (1 baris dan 2 kolom). Untuk nilai dari matriks kernelnya itu tidak ada ketentuan nilai yang digunakan. Nilai *kernel* ini akan digunakan sebagai hasil dari Conv1D menggunakan formula Relu di atas, sehingga mendapatkan hasil yang dapat dilihat pada Tabel 16.

Tabel 16 Hasil Conv1D

	aku	**amplop**	**bantet**	**buat**	**dki**	**duga**	**ga**	**kpps**	**kursi**	**sandi**	**sebar**	**semok**	**tiada**
D1	0	0	0	0	0	0	0	0	0	0	0	0	0
D2	0	0	0	0	0	0	0	0	0	0	0	0	0
D3	0	0	0	0	0	0	0	0	0	0	0	0	0
D4	0	0	0	0	0	0	0	0	0	0	0	0	0

Manualisasi mendapatkan nilai Relu:

$$X\ (aku) = (0 * - 0.539) + \ (0 * - 0.698) = 0$$

Nilai 0 didapatkan dari hasil *weight* dari TF-IDF sebelumnya, nilai -0.539 dan -0.698 didapatkan dari nilai *matrix kernel* di atas. Hasil dari X ini dimasukkan ke dalam fungsi aktivasi Relu dan mendapatkan hasil sebagai berikut:

$$Relu = MAX(0,0) = 0$$

Hasil 0 didapat dari perbandingan nilai terbesar antara hasil X dengan nilai 0. Hal ini dikarenakan *output* Relu memiliki *range* nilai dari 0 hingga x, di mana jika nilai $x \leq 0 = 0$ dan jika nilai $x > 0$ maka nilai $x = x$ (Sitepu & Sigiro, 2021).

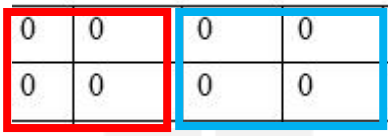
b. *Max Pooling*

Input-an dari tahapan ini berasal dari Tabel 16, hal ini dikarenakan *inception* memiliki beberapa cabang yang nantinya akan disatukan dalam bentuk *depth* (tumpukan) (Haq, 2020).

Tabel 17 *Max Polling*

D1	0	0	0	0	0	0
D2	0	0	0	0	0	0
D3	0	0	0	0	0	0
D4	0	0	0	0	0	0

Pada *max pooling layer inception*, *stride* (pergeseran) adalah 2, artinya pegeseran yang dilakukan akan melewati 2 kolom.

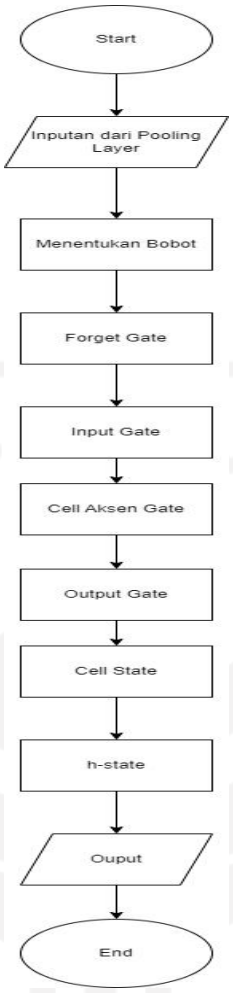


Gambar 2 Contoh *Stide* 2 pada *Max Pooling*

Tahapan *max pooling* ini merupakan akhir dari proses CNN dengan LSTM. *Output* dari CNN di lapisan *max pooling* ini adalah ekstraksi fitur yang telah mengurangi pengurangan dimensi yang sebelumnya pada tahapan Conv1D 13 fitur dan setelah melewati tahapan *max pooling* memiliki 6 fitur. Hal ini dapat berubah tergantung nilai *stride* yang di atur.

c. LSTM

Terdapat *flow* dalam implementasi LSTM pada penelitian ini yaitu:



Gambar 3 *Flowchart* dari LSTM

1) *Input* Dataset Hasil *Max Pooling*

   Dataset yang digunakan adalah dataset dari hasil *max pooling* sebelumnya yang terdapat pada Tabel 17.

2) Menentukan Bobot

   Penentuan bobot bertujuan untuk dapat menghitung *gate-gate* yang ada di LSTM. Rentang bobot bermula dari -1 hinga 1 dan bersifat *random* (*Understanding of LSTM Networks - GeeksforGeeks*, 2021)

Tabel 18 Bobot pada *Forget Gate*

W_f	**Uf**	**b_f**
0.280286	0.010539	1

Keterangan:

- W_f = Bobot pada *forget gate*

- b_f = Bias *forget gate*

Tabel 18 merupakan bobot, *kernel* dan bias pada *forget gate*, nilai tersebut didapatkan secara *random* dengan rentang -1 hinga 1. *Forget gate* sendiri akan menentukan apakah nilai akan diteruskan kepada *input gate* dengan syarat nilai yang didapatkan adalah 1, dan jika 0 maka tidak akan diteruskan kepada *input gate*. *Kernel* nantinya akan dikalikan pada *h-state* (*output* pada nilai sebelumnya) yang di mana setiap *gate* akan memiliki biasnya masing-masing.

Tabel 19 Bobot pada *Input Gate*

W_I	**Ui**	**b_i**
0.858629	-0.63881	0

Keterangan:

- W_i = Bobot pada *forget gate* atau *kernel* pada *gate*
- b_i = Bias *input*

Bobot pada *input gate* juga memiliki sifat dengan *forget gate*, di mana bobot didapatkan secara *random* dengan rentang -1 hingga 1. *Kernel* nantinya akan dikalikan pada *h-state* (*output* pada nilai sebelumnya) yang di mana setiap *gate* akan memiliki biasnya masing-masing, *kernel* sendiri dapat dikatakan bobot pada *forget*.

Tabel 20 Bobot pada *Cell Aksen Gate*

W_c	**Uc**	**b_c**
0.343074	-0.05939	0

Keterangan

- W_f = Bobot pada *Cell Aksen Gate*
- b_f = Bias *Cell Aksen Gate*

Bobot pada *cell aksen gate* juga memiliki sifat dengan *forget gate*, di mana bobot didapatkan secara *random* dengan rentang -1 hingga 1. *Kernel* nantinya

akan dikalikan pada *h-state* (*output* pada nilai sebelumnya) yang di mana setiap *gate* akan memiliki biasnya masing-masing.

Tabel 21 Bobot *Output Gate*

W_o	Uo	b_o
0.077388	-0.76699	0

Keterangan

- W_f = Bobot pada *output gate*
- b_f = Bias *output*

Bobot pada *output gate* juga memiliki sifat dengan *forget gate*, di mana bobot didapatkan secara *random* dengan rentang -1 hingga 1. *Kernel* nantinya akan dikalikan pada *h-state* (*output* pada nilai sebelumnya) yang di mana setiap *gate* akan memiliki biasnya masing-masing.

Tabel 22 Bobot *Cell gate* dan *H-state*

ht-1	ct-1
0	0

Keterangan

- ht-1 = Bobot *h-state* sebelumnya
- ct-1 = Bobot *cell gate* sebelumnya

Untuk nilai *cell gate* dan *h-state* untuk *input*-an pertama = 0 dan untuk *input*-an kedua akan menerima *cell gate* dan *h-state* dari *input*-an pertama, dan seterusnya.

3) Menghitung *Forget Gate*

Pada *forget gate* informasi pada setiap data masukan diolah dan dipilih data mana saja yang akan disimpan atau dibuang pada *memory cells*. Fungsi aktivasi yang digunakan pada *forget gate* ini adalah fungsi aktivasi *sigmoid*. Di mana hasil keluarannya antara 0 dan 1. Jika keluarannya adalah 1 maka semua data akan disimpan dan sebaliknya jika keluarannya 0 maka semua data akan dibuang. Dalam menghitung *forget gate* menggunakan persamaan rumus (4)

dengan perbandingan data *train* dan *test* adalah 80% : 20% dan didapatkan hasil *forget gate* pada data *train* sebagai berikut:

Tabel 23 Hasil *Forget Gate*

No	**Forget**								
	Wf.xt						**Uf.ht-1**	**bf**	**Aktivasi**
	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6			
Data 1	0	0	0	0	0	0	0	1	0.731059
Data 2	0	0	0	0	0	0	0	1	0.731059
Data 3	0	0	0	0	0	0	0	1	0.731059
Data 4	0	0	0	0	0	0	0	1	0.731059

Dari Tabel 23, dapat dijabarkan perhitungannya sebagai berikut:

$Wf.[ht-1, xt]\ bf = (0 * 0.2802862) + (0.01053863 * 0) + 1 = 1$

Nilai Wf berasal dari nilai bobot *random* yang telah didapatkan sebelumnya di mana nilai xt yang digunakan t1 (0.28 dan 0.0105) dan ht-1 nya adalah 0. Setelah mendapatkan hasil di atas, selanjutnya akan dilakukan aktivasi *sigmoid*

$Ft1 = \sigma(1)$

$= \frac{1}{1+e^{(1)}} = 0..7311$

Perhitungan di atas juga berlakukan nilai *series*-t yang lain.

4) *Input Gate*

Pada *input gate* terdapat dua *gates* yang dilaksanakan, pertama diputuskan nilai mana yang diperbarui menggunakan fungsi aktivasi *sigmoid.* Pada tahap ini dapat menggunakan persamaan rumus (5).

Tabel 24 Hasil *Input Gate*

No	**Input**

	Wi.xt						Ui.ht-1	bi	Aktivasi
	Col 1	Col 2	Col 3	Col 4	Col 5	Col6			
Data 1	0	0	0	0	0	0	0	0	0.5
Data 2	0	0	0	0	0	0	0	0	0.5
Data 3	0	0	0	0	0	0	0	0	0.5
Data 4	0	0	0	0	0	0	0	0	0.5

Dari Tabel 24, dapat dijabarkan perhitungannya sebagai berikut:

$Wi.[ht-1, xt]\ bi = (0.85862935 * \ 0) + (-0.639 * 0) + 0 = 0$

Setelah mendapatkan hasil di atas, selanjutnya akan dilakukan aktivasi *sigmoid*

$It1 = \ \sigma(0)$

$= \frac{1}{1+e^{(0)}} = 0.5$

Perhitungan di atas juga berlakukan nilai *series*-t yang lain.

5) *Cell Aksen Gate*

Cell aksen gate merupakan *memory cell* yang diaktivasi menggunakan *tanh*. Dalam menghitung *cell aksen gate* dapat menggunakan persamaan rumus (6).

Tabel 25 Hasil *Cell Aksen Gate*

No	C'								
	Wc.xt						Uc.ht-1	bc'	Aktivasi
	Col 1	Col 2	Col 3	Col 4	Col 5	Col6			
Data 1	0	0	0	0	0	0	0	0	0
Data 2	0	0	0	0	0	0	0	0	0
Data	0	0	0	0	0	0	0	0	0

3									
Data 4	0	0	0	0	0	0	0	0	0

Dari Tabel 25 dapat dijelaskan perhitungannya sebagai berikut:

$Wc.[ht-1, xt]\ bc = (0.34307384 * 0) + (-0.05939113 * 0) + 0 = 0$

Setelah mendapatkan hasil di atas, selanjutnya akan dilakukan aktivasi *tanh*

$C't1 = tanh(0)$

$= tanh(0) = 0$

Perhitungan di atas juga berlakukan nilai *series*-t yang lain.

6) *Output Gate*

Pada *output gate* terdapat dua *gates* yang dilaksanakan, pertama diputuskan nilai pada bagian *memory cell* mana yang dikeluarkan dengan menggunakan fungsi aktivasi *sigmoid*. Pada tahap ini dapat menggunakan persamaan rumus (7).

Tabel 26 Hasil *Output Gate*

No	**Output**								
	Wo.xt						**Uo.ht -1**	**bo**	**Aktivasi**
	Col 1	Col 2	Col 3	Col 4	Col 5	Col6			
Data 1	0	0	0	0	0	0	0	0	0.5
Data 2	0	0	0	0	0	0	0	0	0.5
Data 3	0	0	0	0	0	0	0	0	0.5
Data 4	0	0	0	0	0	0	0	0	0.5

Berdasarkan Tabel 26 dapat dijelaskan perhitungan pada *output gate* sebagai berikut:

$Wo.[ht-1, xt]\ bo = (0.07738841 *\ \ 0\ ) + (-0.76699406 * 0) +\ \ 0 = 0$

Setelah mendapatkan hasil di atas, selanjutnya akan dilakukan aktivasi *sigmoid*

$Ot1 =\ \sigma(0)$

$= \frac{1}{1+ e^{(0)}} = 0.5$

Perhitungan di atas juga berlakukan nilai *series*-t yang lain.

7) *Cell State*

Pada *cell state* mengganti nilai pada *memory cell* sebelumnya dengan nilai *memory cell* yang baru. Di mana nilai ini didapatkan dari menggabungkan nilai yang terdapat pada *forget gate* dan *input gate*.

Berdasarkan persamaan rumus (8), dibutuhkan nilai *forget gate*, Ct-1, *input gate* dan *cell aksen gate*. Untuk Ct-1 didapatkan pada penentuan bobot yaitu 0 dan alasannya 0 karena Ct-1 sebelum ct tidak ada. Oleh karena itu diset dengan 0. Tabel 27 merupakan hasil dari perhitungan *Cell State*.

Tabel 27 Hasil *Cell State*

No	C
Data 1	0
Data 2	0
Data 3	0
Data 4	0

$Ct1 =\ 0.7311 *\ 0 +\ 0.5\ \ * 0 =\ 0$

Sehingga didapatkan Ct1 = 0. Selanjutnya menghitung *h-state*.

8) *H-State*

H-state merupakan perkalian antara *output gate* dengan aktivasi *tanh cell state*. Pada tahap ini menggunakan persamaan rumus (9) dan menghasilkan *h-state* yang dapat dilihat pada Tabel 28.

$$ht1 = 0.5 * tanh(0) = 0$$

Tabel 28 Hasil *H-State*

No	h
Data 1	0
Data 2	0
Data 3	0
Data 4	0

d. *Fully Connected Layer*

Pada *fully connected layer* menggunakan metode *multilayer perceptron* yaitu *backpropagation* untuk pembaruan bobot (W) dan bias (b) (Haq, 2020). Formula dari *fully connected layer* adalah sebagai berikut:

$$y_i = b_i + \sum_{i=0,..n} w_i + x_i$$

Keterangan:

y = Jaringan label ke-i

b = Bias

w = Bobot dari setiap jaringan

x = *Input*-an dari hasil *flatten*

Hasil dari *fully connected layer* adalah sebagai berikut (contoh perhitungan data nomor 1). Untuk nilai bobot dapat ditentukan secara *random* dengan kisaran 0 sampai 1 serta nilai bias yang digunakan dapat dilihat pada Tabel 29 dan Tabel 30.

Tabel 29 Bobot Yang Digunakan

	V
V1	0.49112546

Tabel 30 Bias Yang Digunakan

Bias1

b1	-0.00053225

Sehingga hasil dari *fully connected layer* adalah sebagai berikut:

Tabel 31 Hasil *Fully Connected Layer*

No	h	Bobot * h + b	Aktivasi	Prediksi
Data 1	0	-0.00053225	0.49987	0
Data 2	0	-0.00053225	0.49987	0
Data 3	0	-0.00053225	0.49987	0
Data 4	0	-0.00053225	0.49987	0

Dari Tabel 31, dapat dijelaskan perhitungannya sebagai berikut:

$W_{,data1} = \sum_{i=0,..n} (0.49112546 * 0) + - 0.00053225 = - 0.00053225$

$W_{,data2} = \sum_{i=0,..n} (0.49112546 * 0) + - 0.00053225 = - 0.00053225$

$W_{,data3} = \sum_{i=0,..n} (0.49112546 * 0) + - 0.00053225 = - 0.00053225$

$W_{,data4} = \sum_{i=0,..n} (0.49112546 * 0) + - 0.00053225 = - 0.00053225$

Pada perhitungan aktivasi menggunakan aktivasi *sigmoid*, di mana formula aktivasi *sigmoid* adalah sebagai berikut:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Keterangan:

X = Nilai W

Sehingga didapatkan hasil pada Tabel 31 sebagai berkut:

$f(D1) = \frac{1}{1 + e^{-(-0.00053225)}} = 0.49986693$

$f(D2) = \frac{1}{1+e^{-(-0.00053225)}} = 0.49986693$

$f(D3) = \frac{1}{1 + e^{-(-0.00053225)}} = 0.49986693$

$f(D4) = \frac{1}{1 + e^{-(-0.00053225)}} = 0.49986693$

Untuk menentukan kelas dari hasil aktivasi, jika >= 0.5 akan termasuk ke dalam kelas 1 dan < 0.5 termasuk ke dalam kelas 0. Sehingga, 0.49986693 masuk ke dalam kelas 0 atau kelas netral dikarenakan 0.49986693 kurang dari 0.5.