TESIS

DETEKSI HATE SPEECH DI TWITTER MENGGUNAKAN METODE MULTI-LAYER LONG SHORT-TERM MEMORY

Disusun oleh:

Nama	**: Firman Sriyono Hadi Sudiro**
NIM	**: 20.51.1292**
Konsentrasi	**: Business Intelligence**

PROGRAM STUDI S2 TEKNIK INFORMATIKA

PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA

YOGYAKARTA

2022

TESIS

DETEKSI HATE SPEECH DI TWITTER MENGGUNAKAN METODE MULTI-LAYER LONG SHORT-TERM MEMORY

DETECTING HATE SPEECH ON TWITTER USING MULTI-LAYER LONG SHORT-TERM MEMORY METHOD

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Magister

Disusun oleh:

Nama	**: Firman Sriyono Hadi Sudiro**
NIM	**: 20.51.1292**
Konsentrasi	**: Business Intelligence**

PROGRAM STUDI S2 TEKNIK INFORMATIKA

PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA

YOGYAKARTA

2022

HALAMAN PENGESAHAN

DETEKSI HATE SPEECH DI TWITTER MENGGUNAKAN METODE MULTI-LAYER LONG SHORT-TERM MEMORY

DETECTING HATE SPEECH ON TWITTER USING MULTI-LAYER LONG SHORT-TERM MEMORY METHOD

Dipersiapkan dan Disusun oleh

Firman Sriyono Hadi Sudiro

20.51.1292

Telah Diujikan dan Dipertahankan dalam Ujian Tesis
Program Studi S2 Teknik Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Rabu, 2 Maret 2022

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 10 Maret 2022
Rektor

Prof. Dr. M. Suyanto, M.M.
NIK. 190302001

HALAMAN PERSETUJUAN

DETEKSI HATE SPEECH DI TWITTER MENGGUNAKAN METODE MULTI-LAYER LONG SHORT-TERM MEMORY

DETECTING HATE SPEECH ON TWITTER USING MULTI-LAYER LONG SHORT-TERM MEMORY METHOD

Dipersiapkan dan Disusun oleh

Firman Sriyono Hadi Sudiro

20.51.1292

Telah Diujikan dan Dipertahankan dalam Ujian Tesis
Program Studi S2 Teknik Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari Rabu, 2 Maret 2022

Pembimbing Utama

Prof. Dr. Kusrini, M.Kom
NIK. 0513127901

Anggota Tim Penguji

Prof. Dr. Ema Utami, S.St., M.Kom
NIK. 0521027501

Alva Hendi Muhammad, S.T., M.Eng., Ph.D.
NIK. 0518078401

Pembimbing Pendamping

Drs. Asro Nasiri, M.Kom
NIK. 0407106704

Prof. Dr. Kusrini, M.Kom
NIK. 0513127901

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 10 Maret 2022
Direktur Program Pascasarjana

Prof. Dr. Kusrini, M.Kom
NIK. 190302106

HALAMAN PERNYATAAN KEASLIAN TESIS

Yang bertandatangan di bawah ini,

Nama mahasiswa : Firman Sriyono Hadi Sudiro
NIM : 20.51.1292
Konsentrasi : Business Intelligence

Menyatakan bahwa Tesis dengan judul berikut:
DETEKSI HATE SPEECH DI TWITTER MENGGUNAKAN METODE MULTI-LAYER LONG SHORT-TERM MEMORY

Dosen Pembimbing Utama : Prof. Dr. Kusrini, M.Kom
Dosen Pembimbing Pendamping : Drs. Asro Nasiri, M.Kom

1. Karya tulis ini adalah benar-benar ASLI dan BELUM PERNAH diajukan untuk mendapatkan gelar akademik, baik di Universitas AMIKOM Yogyakarta maupun di Perguruan Tinggi lainnya
2. Karya tulis ini merupakan gagasan, rumusan dan penelitian SAYA sendiri, tanpa bantuan pihak lain kecuali arahan dari Tim Dosen Pembimbing
3. Dalam karya tulis ini tidak terdapat karya atau pendapat orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan disebutkan dalam Daftar Pustaka pada karya tulis ini
4. Perangkat lunak yang digunakan dalam penelitian ini sepenuhnya menjadi tanggung jawab SAYA, bukan tanggung jawab Universitas AMIKOM Yogyakarta
5. Pernyataan ini SAYA buat dengan sesungguhnya, apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka SAYA bersedia menerima SANKSI AKADEMIK dengan pencabutan gelar yang sudah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi

Yogyakarta, 19 April 2022
Yang Menyatakan,

Firman Sriyono Hadi Sudiro

HALAMAN PERSEMBAHAN

Puji syukur saya ucapkan kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga saya dapat menyelesaikan penyusunan tesis yang berjudul "Deteksi Hate Speech di Twitter menggunakan Metode Multi-Layer Long Short-term memory" dan diajukan sebagai salah satu syarat untuk mendapat gelar Magister Teknik Informatika pada Jurusan MTI, Fakultas Teknik Informatika, Universitas Amikom Yogyakarta. Saya mengucapkan terimakasih kepada pihak yang terlibat langsung maupun tidak langsung atas selesainya tesis ini

1. Ayah Sutiyono, sang tulang punggung keluarga. Walaupun dengan penyakit jantung dan diabetesnya tetap gigih memperjuangkan kehidupan anak - anaknya. Semoga engkau diberi kesehatan wal afiat oleh Allah swt, tunggu sebentar lagi putra bontotmu ini akan membahagiakanmu. Thanks for everything, Ayah!

2. Ibu Sugiyanti. Seorang wanita tangguh dan perkasa menyabet gelar kepahlawanan dalam membentuk karakter anak – anaknya. Dengan tanpa gelar akademik sekalipun tetap menjadi suksesor pasca sarjana bagi anak – anaknya. Izinkan aku membentuk senyum simpul manis di ujung bibirnya ketika sukses nanti.

3. Ketiga kakak – kakakku yang tidak bisa kusebutkan satu persatu. Ketiga saudara tersebut yang selalu support. Teruntuk buat ponakan – ponakan terhebatku, Handi Adillah, Afarzki Sitompul, Hanzani mutmaenah, Hafizah Nuraini, Zakariah Mas'ud dan Muhammad Solahun. Kalian adalah ponakan terhebat yang pernah aku miliki. Terimakasih telah menyemangatiku!

4. Ibu Prof. Kusrini sebagai pembimbing utama dan Pak Asro Nasiri sebagai pembimbing pendamping yang telah memberikan kontribusi berupa bimbingan, motivasi, kritik dan saran atas terselesaikannya tesis ini.

HALAMAN MOTTO

1. Janganlah menyesali kegagalan karena dari kegagalanlah kita bisa mendapatkan kesuksesan.
2. Waktu tidak akan pernah berhenti, jadi pergunakan waktu dengan sebaik-baiknya.
3. Berpikir adalah kegiatan tersulit yang pernah ada. Oleh karena itu hanya sedikit yang melakukannya (Henry Ford).
4. Orang bijak belajar ketika mereka bisa. Orang bodoh belajar ketika mereka harus (Arthur Wellesley)

KATA PENGANTAR

Alhamdulillah puji syukur kehadirat Allah SWT, tuhan semesta alam, yang telah melimpahkan rahmat, karunia, serta petunjuk-Nya, shalawat serta salam penulis panjatkan kepada junjungan alam Nabi Muhammad SAW berserta kelaurga dan sahabatnya, sehingga penulis dapat menyelesaikan tesis dengan judul **"DETEKSI HATE SPEECH DI TWITTER MENGGUNAKAN METODE MULTI-LAYER LONG SHORT-TERM MEMORY"**.

Tesis ini merupakan salah satu syarat untuk mendapatkan gelar Magister Teknik Informatika pada program Pascasarjana Teknik Informatika Universitas Amikom Yogyakarta. Oleh karena itu, dengan segala kerendahan hati pada kesempatan ini penulis mengucapkan banyak terimakasih kepada:

1. Kedua orang tua, adik, dan keluarga besar yang telah memberi banyak dukungan, bimbingan, doa, kasih saying, dan semua yang telah diberikan.
2. Ibu Prof. Dr. Kusrini, M.Kom. selaku **Direktur Program Pascasarjana** Universitas Amikom Yogyakarta, yang telah memberikan kesempatan kepada penuis untuk belajar di program studi Magister Teknik Informatika Universitas Amikom Yogyakarta.
3. Ibu Prof. Dr. Kusrini, M.Kom. selaku pembimbing utama atas segala saran, masukan, dan kesabaran yang diberikan kepada penulis untuk menyelesaikan tesis ini.
4. Bapak Drs. Asro Nasiri, M.Kom selaku pemmbimbing pendamping atas segala saran, masukan, dan kesabaran yang diberikan kepada penulis untuk menyelesaikan tesis ini.
5. Bapak dan Ibu Dosen penguji yang telah memberikan saran dan masukan untuk hasil penelitian ini yang lebih baik.
6. Bapak dan Ibu Dosen Program Studi Magister Teknik Informatika Universitas Amikom Yogyakarta atas ilmu yang telah diberikan.
7. Seluruh staff karyawan dan pengelola Universitas Amikom Yogyakarta yang telah banyak memberikan bantuan kepada penulis.

8. Seluruh teman teman Magister Teknik Informatika Universitas Amikom Yogyakarta angkatan 2020 yang telah banyak memberikan bantuan kepada penulis.
9. Semua pihak yang telah memberikan bantuan dan dukungan kepada penulis dalam penyusunan Tesis ini.

Akhir kata penulis berharap semoga Tesis ini dapat memberikan manfaat bagi seluruh umat manusia, terutama bagi perkembangan ilmu pengetahuan khususnya teknik informatika. Apabila ada saran, kritik yang bersifat membangun sangat diharapkan untuk kebaikan tesis ini. Semoga tesis ini dapat bermanfaat bagi kata semua.

Yogyakarta, 19 April 2022

Penulis,

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

Perkembangan teknologi informasi saat ini melaju sangat cepat, sehingga menjadi kebutuhan hidup masyarakat di seluruh dunia. Salah satu dampak dari perkembangan teknologi informasi adalah munculnya berbagai macam situs jejaring sosial atau media sosial seperti *Facebook*, *Twitter* dan *Instagram*. Perkembangan teknologi tidak hanya berdampak positif saja, namun juga berdampak negatif yaitu tindak pidana penghinaan atau ujaran kebencian (*Hate Speech*).

Penelitian ini bertujuan untuk melakukan pengklasifikasian terhadap kalimat *Hate Speech* berbahasa Indonesia dengan menggunakan metode *Long Short-Term Memory* (LSTM). Pada proses pengujian, Metode LSTM 1 Layer akan dibandingkan dengan Metode LSTM 2 Layer berdasarkan perhitungan dari nilai akurasi, *precision, recall dan f-measure*.

Hasil pengujian menunjukkan bahwa metode *Long Short-Term Memory* 2 Layer memiliki hasil akurasi yang lebih baik yaitu 77.54%, *precision* 77.00%, *recall* 77.00% dan *f-measure* 77.00% dibandingkan dengan metode LSTM 1 Layer dengan nilai *akurasi* 75.63%, *precision* 76.00%, *recall* 76.00% dan *f-measure* 76.00%.

Kata kunci: Sentimen Analisis, *Hate Speech*, *Long Short-Term Memory*, *Multi Layer Long Short-Term Memory*.

ABSTRACT

The development of information technology is currently grow very fast, so it becomes a necessity for the lives of people around the world. One of the impacts in development of information technology is the emergence of various kinds of social networking sites or social media such as Facebook, Twitter and Instagram. Technological developments not only have a positive impact, but also have a negative impact which called criminal acts of humiliation or hate speech.

This study is classify hate speech sentences in Indonesian using the Long Short-Term Memory (LSTM) method. In the testing process, the LSTM 1 Layer Method will be compared with the LSTM 2 Layer Method based on the calculation of the values of accuracy, precision, recall and f-measure.

The test results show that Long Short-Term Memory 2 Layers method has better accuracy 77.54%, precision 77.00%, recall 77.00% and f-measure 77.00% compared with LSTM 1 Layer method with an accuracy value of 75.63%, 76.00% precision, 76.00% recall and 76.00% f-measure.

Keyword: Sentiment Analysis, Hate Speech, Long Short-Term Memory, Multi Layer Long Short-Term Memory.

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Perkembangan teknologi informasi saat ini melaju sangat cepat, sehingga menjadi kebutuhan hidup masyarakat di seluruh dunia. Jumlah penduduk Indonesia yang selalu bertambah setiap tahunnya karena angka kelahiran yang terus bertambah, sehingga pemanfaatan teknologi sangat dibutuhkan untuk menunjang aktivitas sehari-hari. Salah satu dampak dari perkembangan teknologi informasi adalah munculnya berbagai macam situs jejaring sosial atau media sosial, pengguna situs jejaring sosial atau media sosial ini mencakup berbagai kalangan mulai dari anak-anak, pelajar, ibu rumah tangga, pedagang, karyawan dan lain sebagainya. Media sosial banyak digunakan oleh masyarakat Indonesia dan dapat kita temukan melalui mesin pencari seperti Google, namun yang paling populer di kalangan pengguna media sosial adalah Facebook, Twitter, BBM, WhatsApp, Instagram. Permasalahan hukum yang sering dihadapi berkaitan dengan penyampaian informasi dan komunikasi khususnya dalam hal pembuktian dan hal-hal yang berkaitan dengan hukum yang dilaksanakan melalui sistem elektronik (undang-undang republik indonesia nomor 11 tahun 2008, https://jdih.kemenkeu.go.id/fulltext/2008/11TAHUN2008UUPenj.htm, diakses pada 02 Februari 2022 20:38). Akibat dari perkembangan tersebut, teknologi informasi dengan sendirinya juga telah mengubah perilaku manusia

dari peradaban global. Namun perkembangan teknologi tidak hanya berdampak positif, tetapi juga berdampak negatif yaitu berupa tindak pidana penghinaan atau ujaran kebencian dan penyebaran informasi yang bertujuan menimbulkan kebencian atau permusuhan antar individu atau kelompok tertentu berdasarkan suku, agama dan ras.

Ujaran kebencian adalah tindakan komunikasi yang dilakukan oleh individu atau kelompok dalam bentuk provokasi, hasutan, atau penghinaan kepada individu atau kelompok lain dalam berbagai aspek seperti ras, warna kulit, suku, jenis kelamin, kecacatan, orientasi seksual, kebangsaan, agama, dan sebagainya (wikipedia, https://id.wikipedia.org/wiki/Ucapan_kebencian, diakses pada 02 februari 2022 21:00). Dalam pengertian hukum, ujaran kebencian adalah perkataan, tingkah laku, tulisan, atau perbuatan yang dilarang karena dapat memicu tindakan kekerasan dan prasangka baik dari pihak pelaku maupun korban dari perbuatan tersebut.

Ujaran kebencian atau *Hate Speech* sendiri dapat dibedakan dalam beberapa konteks. Contohnya adalah dalam konteks bahasa, bidang tertentu, siapa saja respondenya dan waktu terjadinya. Jika dilihat dalam konteks bahasa, kalimat *Hate Speech* di indonesia tentu saja berbeda dengan kalimat *Hate Speech* di Saudi Arabia. Jika dilihat dalam konteks bidang, kalimat hate speech dalam bidang politik tentu saja berbeda dengan bidang kesehatan (contohnya *Hate Speech* pada Covid-19).

Salah satu cara untuk mengurangi ujaran kebencian di Twitter adalah dengan menambahkan filter pada bagian tweet. Sedangkan cara mudah untuk

menghentikan ujaran kebencian di Twitter adalah dengan memblokir tweet untuk setiap suku kata hingga kalimat yang dianggap ujaran kebencian.

Kearifan lokal (*local wisdom*) tentu juga mempunyai pengaruh dalam masyarakat terutama masyarakat Banjar. Tradisi masyarakat Banjar sangat kuat didominasi unsur-unsur agama Islam, salah satunya tradisinya yaitu Sasindiran. Tradisi Sasindiran adalah tradisi untuk nasihat-menasihati antar sesama, yang mana tradisi ini sendiri biasanya turun-temurun diajarkan oleh nenek moyang dalam suatu lingkungan masyarakat Banjar. Contohnya yaitu dalam masyarakat Banjar ada suatu nasihat "jangan mengada-ada" (jangan melebih-lebihkan) yang biasanya diucapkan ketika seseorang mendengar berita yang tidak jelas kebenarannya. Sehingga, orang yang mendengar nasihat ini tidak akan mudah percaya. Tradisi masyarakat Banjar ini dipercaya dapat mencegah dan menangkal kasus ujaran kebencian saat ini.

Metode pendeteksian ujaran kebencian yang diusulkan oleh Mutanga et al (2020) menggunakan metode CNN (Convolutional Neural Network), GRU (Gated Recurrent Units) dan BERT (Bidirectional Encoder Representations From Transformers) untuk mendeteksi ujaran kebencian. Metode ini memiliki performansi yang cukup baik namun akurasi yang dihasilkan dirasa masih kurang dan juga fitur semantik antar kata jarang diperhatikan dalam klasifikasi teks. Metode LSTM yang telah digunakan oleh Zimmerman (2019) memiliki hasil yang baik dibandingkan dengan metode Convolutional Neural Network.

Peneliti mengambil kasus pendeteksian ujaran kebencian karena masih banyak sebagian oknum yang tidak bertanggung jawab menggunakan media

sosial *twitter* sebagai media penyebaran ujaran kebencian. Kasus ujaran kebencian di Indonesia menjamur dan merajalela serta meresahkan masyarakat luas. Hal ini membuat dunia media sosial tercemar oleh ulah oknum-oknum tersebut.

Berdasarkan latar belakang yang telah dijelaskan, penelitian ini menggunakan pendekatan deep learning untuk mendeteksi ujaran kebencian di Twitter menggunakan Metode Multi-Layer Long Short-Term Memory serta Word2vec sebagai ekstraksi fitur.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah disebutkan diatas, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

a. Bagaimana cara yang efektif untuk mendeteksi *Hate Speech* ?
b. Metode manakah yang paling akurat untuk mendeteksi *Hate Speech* ?

1.3. Batasan Masalah

Batasan masalah yang digunakan pada penelitian ini adalah sebagai berikut:

a. Data yang digunakan dalam penelitian ini adalah data dari twitter Indonesia.
b. Data yang digunakan berupa *tweet* dengan hastag #Jokowi2Periode, #TurunkanJokowi.
c. Fitur ekstraksi yang digunakan adalah Word2Vec.
d. Parameter metode LSTM yang digunakan adalah sigmoid, tanh dan relu.

1.4. Tujuan Penelitian

Tujuan penelitianya adalah:

a. Menggunakan Metode *Long Short-Term Memory* untuk mendeteksi *hate speech* yang ada di media sosial *Twitter*.
b. Membuktikan bahwa Metode Long Short-Term Memory dapat digunakan untuk mendeteksi *Hate Speech* yang ada di media sosial Twitter dengan tepat dan akurat.
c. Memudahkan klasifikasi *tweet* yang terindikasi *Hate Speech*.
d. Untuk mengevaluasi metode LSTM dalam mendeteksi *Hate Speech*.

1.5. Manfaat Penelitian

Manfaat yang dapat diperoleh pada penelitian ini yaitu:

a. Dapat mengetahui lebih jelas *tweet* yang terindikasi *Hate Speech* atau tidak terindikasi *Hate Speech*.
b. Manfaat bagi penulis adalah menambah pengetahuan tentang Metode mendeteksi *Hate Speech*.
c. Memberi gagasan peneliti-peneliti berikutnya dalam memilih Metode untuk mendeteksi *Hate Speech*.

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Beberapa penelitian tentang *Hate Speech* yang dijadikan kajian pustaka dalam penelitian ini adalah penelitian (Alshalan and Al-Khalifa, 2020) , (Zhang and Luo, 2019), (MacAvaney *et al.*, 2019), (Mutanga, Naicker and Olugbara, 2020), (Biere, 2018), (Zimmerman, Fox and Kruschwitz, 2019).

Alshalan and Al-Khalifa (2020) menggunakan Metode CNN (*Convolutional Neural Network*), GRU (*Gated Recurrent Units*) dan BERT (*Bidirectional Encoder Representations From Transformers*) untuk mendeteksi ujaran kebencian. Dari penelitian yang telah dilakukan, hasilnya menunjukkan bahwa Metode CNN berhasil mengungguli metode GRU, dengan akurasi F1 79 % dan AUROC 89%. Hasil penelitian juga menunjukkan bahwa Metode BERT gagal untuk memperbaiki hasil baseline dan metode yang dievaluasi lainnya.

Penelitian yang dilakukan oleh Zhang and Luo (2019) menggunakan Metode DNN (*Deep Neural Network*) dan CNN (*Convolutional Neural Networks*). Dari penelitian yang dilakukan, didapatkan hasil yang menunjukkan bahwa, sebagian besar hasil yang didapatkan memiliki skor keunikan yang rendah. Skor ini khususnya didapatkan dari hasil WY dan WZ serta pj dataset, di mana sebagian besar data yang ditambahan memiliki skor keunikan yang sangat rendah. Sebagian besar dari data (antara 50 % sampai 60 %) dinyatakan bahwa data tweet tersebut tidak memiliki kata-kata yang terindikasi *Hate Speech*.

Penelitian menggunakan Metode SVM (*Support Vector Machine*) dilakukan oleh MacAvaney *et al* (2019). Hasil penelitian menunjukkan bahwa pengklasifikasi meta memiliki bobot 4 n-gram dan kata unigram sebagai kontributor tertinggi untuk skor keseluruhan. Kata yang termasuk 4 n-gram seperti "jew", "ape", "mud", "egro" adalah salah satu sinyal terkuat dari ujaran kebencian. Fitur Unigram seperti "invasi" dan "kekerasan" berkontribusi tinggi pada klasifikasi ujaran kebencian, dan tampaknya masuk dalam aspek ujaran kebencian. Penelitian tersebut menemukan bahwa keakuratan semua pengklasifikasian setidaknya mempunyai akurasi 2% lebih rendah.

Penelitian yang dilakukan oleh Mutanga, Naicker and Olugbara (2020) menjelaskan bahwa Metode BERT (*Bidirectional Encoder Representations From Transformers)* yang termasuk dalam NLP (Natural Language Processing) yang diusulkan dibandingkan dengan Metode XLNet, RoBERTa dan LSTM. Penelitian tersebut membagi dataset dengan rasio masing-masing 80:20 untuk pelatihan dan pengujian model. Hasil yang didapatkan menunjukkan bahwa DistilBERT (distilbert-base-uncased) mencatat skor F-measure 75 % sedangkan LSTM dengan attention mencatat skor F-measure terendah 66 %. Meskipun DistilBERT memiliki lebih sedikit lapisan dan parameter, ia unggul dalam semua algoritma transformator lain yang dieksplorasi dalam penelitian ini.

Penelitian yang dilakukan oleh Biere (2018) menggunakan Metode Natural Language Processing (NLP) and Machine Learning (ML). Hasil penelitianya menghasilkan model yang memprediksi setiap kategori dengan akurasi 91%. Model terakhir memberikan presisi keseluruhan 91%, recall 90% dan skor F1 90%.

Penelitian tersebut dapat disimpulkan bahwa algoritma yang digunakan mengidentifikasi hampir 80% dari data tweet diklasifikasikan ke dalam *Hate Speech*.

Penelitian yang dilakukan Zimmerman, Fox and Kruschwitz (2019) menggunakan Metode CNN (*Convolutional Neural Networks*) untuk mendeteksi ujaran kebencian. Hasil penelitianya mendapatkan akurasi dari sub-model sebesar 75.65% dan akurasi dari metode ensemble 78.62% serta menyimpulkan bahwa algoritma metode ensemble mempunyai kinerja lebih baik pada pengujian dataset dibandingkan dengan submodel rata-rata.

2.2. Keaslian Penelitian

Berdasarkan referensi dan kajian pustaka yang dimiliki penulis, penelitian yang membahas tentang mendeteksi *Hate Speech* menggunakan metode *Long Short-Term Memory* sudah pernah dilakukan. Pada penelitian ini penulis meggunakan Metode *Long Short-Term Memory* dengan digabungkan dengan fitur *Word2Vec* untuk mempermudah mendapatkan hasil persentase deteksi *Hate Speech*. Walaupun demikian, terdapat penelitian terdahulu tentang topik dan metode sejenis yang dipaparkan dalam kajian pustaka. Literatur *review* penelitian ini dapat ditunjukan pada tabel 2.1 sebagai berikut.

Tabel 2.1. Matriks literatur review dan posisi penelitian Mendeteksi Hate Speech di Twitter menggunakan Metode Long Short-Term Memory dan Naïve Bayes

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
1	A deep learning approach for automatic Hate Speech detection in the saudi twittersphere	Alshalan and Al-Khalifa, Elsevier (2020)	Bertujuan untuk menyelidiki beberapa model jaringan saraf berbasis	Memperluas eksperimen di luar area domain dan mengevaluasi	Belum ada keterangan target *Hate Speech* dan tingkat	Menambahkan fitur *Word2Vec* untuk mendeteksi *Hate Speech* dan

Tabel 2.1. Matriks literatur review dan posisi penelitian Mendeteksi Hate Speech di Twitter menggunakan Metode Long Short-Term Memory dan Naïve Bayes (lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			jaringan saraf konvolusional (CNN) dan jaringan saraf berulang (RNN) untuk mendeteksi ujaran kebencian dalam tweet berbahasa Arab.	model yang diusulkan pada kumpulan data *Hate Speech*. Hal tersebut membantu memahami cara mem-bedakan antara *Hate Speech* dan bahasa yang kasar atau menyinggung.	keagresifanya misal lemah, sedang, kuat.	menambahkan Parameter *Sigmoid*, *tanh* dan *relu* pada LSTM untuk meningkatkan hasil akurasinya.
2	Hate Speech Detection A Solved Problem: The Challenging Case of Long Tail on Twitter	Zhang and Luo, Elsevier (2019)	Mengembangkan struktur Deep Neural Network yang berfungsi	Membuat evaluasi terhadap kelas non Hate	Kurangnya pemahaman tentang efek normalisasi	Menambahkan fitur *Word2Vec* untuk mendeteksi

Tabel 2.1. Matriks literatur review dan posisi penelitian Mendeteksi Hate Speech di Twitter menggunakan Metode Long Short-Term Memory dan Naïve Bayes (lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			sebagai fitur ekstraktor yang sangat efektif untuk menganalisa semantik kata kata *Hate Speech*.	Speech yang dominan dalam kumpulan data, menambahkan kemampuan metode untuk mengidentifikasi *Hate Speech*.	tweet pada keakuratan pengklasifikasian, sehingga bisa menjadi masalah kompleks seperti yang ditunjukkan oleh analisis awal yang tidak terdapat korelasi antara ukuran OOV dan hasil kinerja.	menambahkan Parameter *Sigmoid*, *tanh* dan *relu* pada LSTM untuk meningkatkan hasil akurasinya.
3	Hate Speech detection Challenges and solutions	MacAvaney *et al., Elsevier* (2019)	Mengusulkan metode SVM multi-layer yang mutakhir, lebih	Mengusulkan pendekatan baru yang lebih baik dari sistem yang	Mengkategorikan posting yang salah klasifikasi berdasarkan fitur	Menambahkan fitur *Word2Vec* untuk mendeteksi

Tabel 2.1. Matriks literatur review dan posisi penelitian Mendeteksi Hate Speech di Twitter menggunakan Metode Long Short-Term Memory dan Naïve Bayes (lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			sederhana dan memberikan hasil yang lebih akurat.	telah ada, dengan manfaat tambahan berupa deteksi *Hate Speech* yang lebih baik.	mutual linguistik dan fitur semantik.	*Hate Speech* dan menambahkan Parameter *Sigmoid, tanh* dan *relu* pada LSTM untuk meningkatkan hasil akurasinya.
4	Hate Speech Detection in Twitter using Transformer Methods	Mutanga *et al*, Elsevier (2020)	Menganalisa metode berbasis transformator dan mengujinya pada korpus ujaran kebencian multi level yang	Metode DistilBERT, versi lain dari BERT, lebih unggul dari semua metode baseline	Data yang digunakan dalam penelitian ini terbatas pada teks Twitter tekstual saja, sedangkan ujaran kebencian	Menambahkan fitur *Word2Vec* untuk mendeteksi *Hate Speech* dan menambahkan

Tabel 2.1. Matriks literatur review dan posisi penelitian Mendeteksi Hate Speech di Twitter menggunakan Metode Long Short-Term Memory dan Naïve Bayes (lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			berisi 24783 tweet yang berlabel.	berbasis transformator dan LSTM.	di Twitter dapat diungkapkan melalui format data yang berbeda seperti gambar dan video. Misalnya, pengguna yang memposting video yang menghasut ujaran kebencian di Twitter tetap tidak terdeteksi.	Parameter *Sigmoid*, *tanh* dan *relu* pada LSTM untuk meningkatkan hasil akurasinya.
5	Hate Speech Detection Using Natural Language Processing Techniques	Biere, Elsevier (2018)	Meneliti bagaimana metode NLP	Dari hasil tersebut dapat disimpulkan	Masih terdapat banyak tweet yang salah	Menambahkan fitur *Word2Vec* untuk

Tabel 2.1. Matriks literatur review dan posisi penelitian Mendeteksi Hate Speech di Twitter menggunakan Metode Long Short-Term Memory dan Naïve Bayes (lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			(*Natural Language Processing*) digunakan dalam mendeteksi kata-kata *Hate Speech*.	CNN sederhana yang digunakan oleh Kim (2014) memperoleh performansi yang baik (akurasi 91%).	klasifikasi yaitu dalam mengklasifikasikan ujaran kebencian.	mendeteksi *Hate Speech* dan menambahkan Parameter *Sigmoid*, *tanh* dan *relu* pada LSTM untuk meningkatkan hasil akurasinya.
6	Improving Hate Speech Detection with Deep Learning Ensembles	Zimmerman, Fox and Kruschwitz, Elsevier (2019)	Mengembangan metode baru untuk mengidentifikasi dan mengatasi diskriminasi	Perbandingan lebih banyak metrik evaluasi daripada ukuran makro F-1 agregat	Inisialisasi bobot sering tidak dilaporkan oleh penulis lain ditambah dengan masalah	Menambahkan fitur *Word2Vec* untuk mendeteksi *Hate Speech* dan

Tabel 2.1. Matriks literatur review dan posisi penelitian Mendeteksi Hate Speech di Twitter menggunakan Metode Long Short-Term Memory dan Naïve Bayes (lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			dengan lebih baik sambil melindungi kebebasan berekspresi.	berbobot skor.	perpustakaan pembelajaran yang kurang dapat direpresentasi karena pengaturan sumber.	menambahkan Parameter *Sigmoid*, *tanh* dan *relu* pada LSTM untuk meningkatkan hasil akurasinya.

2.3. Landasan Teori

2.3.1 Deep Learning

Deep Learning adalah pengembangan dari Jaringan Saraf Tiruan (*Artificial Neural Network*) yang memiliki lebih banyak lapisan atau layer. Dengan Lapisan yang lebih banyak, Deep Learning diharapkan untuk dapat mengenali proses yang lebih kompleks (Pumsirirat and Yan, 2018). Metode ini efektif dalam mengidentifikasi pola dari data. *Deep Learning* dapat meningkatkan semua bagian dari kecerdasan buatan, mulai dari pemrosesan bahasa alami hingga *machine vision*. Deep Learning dapat berperan sebagai otak yang lebih baik yang dapat meningkatkan cara belajar komputer. Ia dapat meningkatkan kemampuan asisten virtual seperti Google Now untuk menangani hal-hal yang belum dikenali dengan baik oleh kedua asisten virtual tersebut. Pemrosesan video dan pembuatan klip juga sangat mungkin dilakukan oleh *Deep Learning*. Algoritma Deep Learning memiliki beberapa jenis diantaranya adalah Supervised Learning dan Unsupervised Learning.

2.3.2 Supervised Learning

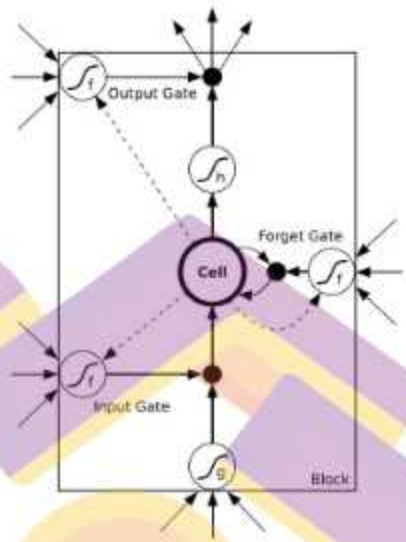
Supervised learning biasanya digunakan untuk menyelesaikan masalah klasifikasi dan regresi. Algoritma supervised learning sangat bergantung pada kesesuaian antara input dan output pada dataset yang diberikan, sehingga kita (user/data scientist) berperan besar dalam memvalidasi input dan output tersebut (Ray, 2019). Input disini maksudnya adalah variabel-variabel atau fitur-fitur yang menjadi ukuran penentu hasilnya (output).

2.3.3 Unsupervised Learning

Pada algoritma unsupervised learning, data tidak memiliki label secara eksplisit dan model mampu belajar dari data dengan menemukan pola yang implisit. Sangat berbeda dengan supervised learning, unsupervised learning merupakan jenis learning yang hanya mempunyai variabel input tapi tidak mempunyai variabel output yang berhubungan. Tujuan dari Machine Learning ini adalah untuk memodelkan struktur data dan menyimpulkan fungsi yang mendeskripsikan data tersebut (Buslim and Iswara, 2019).

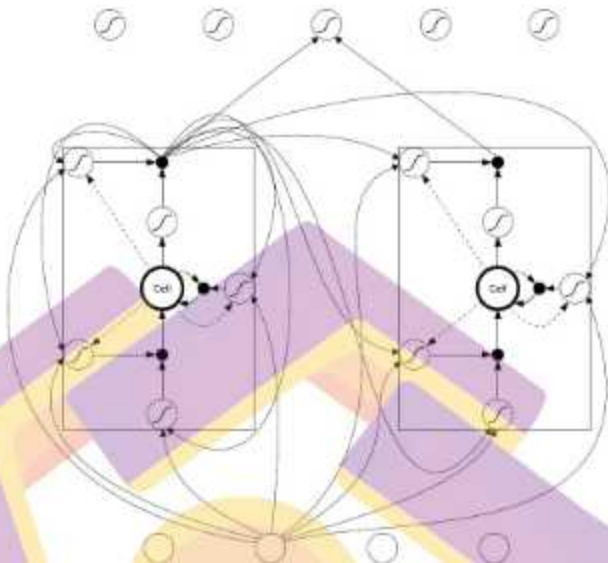
2.3.4 Metode Long Short-Term Memory

Long Short-Term Memory (LSTM) merupakan varian dari Recurrent Neural Network (RNN). LSTM dapat mengingat informasi jangka panjang. LSTM menggantikan simpul RNN di hidden layer dengan sel LSTM yang dirancang untuk menyimpan informasi terdahulu (Buslim and Iswara, 2019). LSTM menggunakan tiga gerbang yaitu input gate, forget gate, output gate untuk mengendalikan penggunaan dan update informasi teks terdahulu. Sel memori dan tiga gerbang dirancang untuk memungkinkan LSTM membaca, menyimpan, dan memperbaharui informasi terdahulu. Diagram struktur metode LSTM dapat dilihat pada gambar 2.1 berikut ini:



Gambar 2.1 Diagram struktur cell LSTM

Berbeda dengan arsitektur RNN konvensional, LSTM dikembangkan sebagai solusi dari masalah *vanishing gradient* yang biasa ditemui pada RNN konvensional. RNN konvensional tidak bisa mengingat hal-hal yang telah terjadi di jaringan dalam jangka waktu yang terlalu lama. Gerbang-gerbang gates pada LSTM memungkinkan memory cell untuk menyimpan dan mengakses informasi dalam jangka waktu yang lebih lama, sehingga mengurangi masalah *vanishing gradient*. Gambar 2.2 memberikan ilustasi *memory block* LSTM dengan *single cell*. Jaringan LSTM sama dengan RNN, kecuali unit inti pada *hidden layer* digantikan oleh *memory block*, seperti yang diilustrasikan sebagai berikut:



Gambar 2.2 Diagram jaringan LSTM

Langkah pertama pada LSTM adalah menentukan apakah informasi dari input Xt-1 dan Xt pantas lewat atau tidak dari *cell state*. Keputusan ini dibuat oleh lapisan sigmoid yang disebut "*forget gate*". Output 1 artinya "biarkan lewat" dan 0 artinya "lupakan informasi". Menghitung nilai forget gate dengan persamaan 1.

$$ft = \sigma\,(Wf\,.[ht{-}1, xt]{+}bi) \qquad (1)$$

Kemudian langkah selanjutnya adalah menentukan informasi baru apa yang akan disimpan di cell state. Pertama lapisan sigmoid yang disebut input gate yang menentukan bagian mana yang akan di update. Selanjutnya, lapisan tanh yang membuat vektor nilai kandidat baru, $\tilde{c}t$,

yang dapat ditambahkan ke cell state. Pada langkah selanjutnya, digabungkan keduanya untuk membuat update ke state. Untuk menghitung nilai input gate dengan persamaan 2 dan nilai kandidat baru dengan persamaan 3.

$$it = \sigma\,(Wi\,.[ht-1, xt] + bi) \tag{2}$$

$$\tilde{c}t = tanh\,(Wc\,.[ht-1, xt] + bc) \tag{3}$$

Selanjutnya memperbarui cell state lama, $Ct-1$, ke cell state baru Ct. Dengan mengalikan cell state lama dengan forget gate ft kemudian ditambah $it * \tilde{c}t$. Untuk lebih jelas pada persamaan 4.

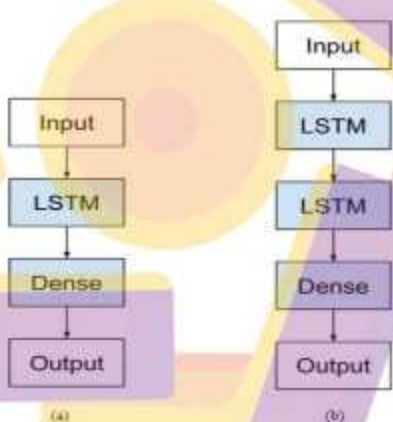
$$Ct = ft * Ct-1 + it * \tilde{C}t \tag{4}$$

Terakhir adalah output gate. Pertama menjalankan lapisan sigmoid yang menentukan sel mana yang akan menjadi output, kemudian menempatkan cell state melalui tanh dan memperbanyak output dari sigmoid gate, sehingga hanya bagian yang kita tentukan yang menjadi output. Perhitungan output gate dengan persamaan 5 dan 6.

$$Ot = \sigma\,(Wo\,.[ht-1, xt] + bo) \tag{5}$$

$$ht = Ot * tanh(Ct) \tag{6}$$

Model LSTM terdiri dari lapisan LSTM tersembunyi tunggal diikuti oleh lapisan output seperti pada Gambar 2.3 bagian (a). Deep LSTM adalah ekstensi untuk model LSTM yang memiliki beberapa layer LSTM tersembunyi di mana setiap lapisan berisi beberapa sel memori/neuron seperti yang ditunjukkan pada Gambar 2.3 bagian (b). Deep LSTM menyatakan kedalaman jaringan lebih penting daripada jumlah sel memori dalam lapisan yang diberikan untuk pembelajaran model. Deep LSTM memungkinkan representasi yang lebih kompleks untuk menangkap informasi pada skala yang berbeda yang digunakan untuk pembelajaran pada model.



Gambar 2.3 Arsitektur LSTM 1 layer (a) dan 2 layer (b)

2.3.5 Hate Speech

Hate Speech (Ujaran Penghinaan atau kebencian) adalah tindakan komunikasi yang dilakukan oleh suatu individu atau kelompok dalam bentuk provokasi, hasutan, ataupun hinaan kepada individu atau kelompok

yang lain dalam hal berbagai aspek seperti ras, warna kulit, etnis, gender, cacat, orientasi seksual,kewarganegaraan, agama, dan lain-lain. Dalam arti hukum, *Hate Speech* adalah perkataan, perilaku, tulisan, ataupun pertunjukan yang dilarang karena dapat memicu terjadinya tindakan kekerasan dan sikap prasangka entah dari pihak pelaku ataupun korban dari tindakan tersebut. Website yang menggunakan atau menerapkan *Hate Speech* ini disebut *Hatesite* (Burnap and Williams, 2018). Kebanyakan dari situs ini menggunakan Forum Internet dan Berita untuk mempertegas suatu sudut pandang tertentu. Para kritikus berpendapat bahwa istilah *Hate Speech* merupakan contoh modern dari novel Newspeak, ketika *Hate Speech* dipakai untuk memberikan kritik secara diam-diam kepada kebijakan sosial yang diimplementasikan dengan buruk dan terburu-buru seakan-akan kebijakan tersebut terlihat benar secara politik. Sampai saat ini, belum ada pengertian atau definisi secara hukum mengenai apa yang disebut *Hate Speech* dan pencemaran nama baik dalam bahasa Indonesia. Dalam bahasa Inggris, pencemaran nama baik diartikan sebagai sebagai *defamation*, *libel*, dan *slander* yang jika diterjemahkan ke dalam bahasa Indonesia adalah fitnah (*defamation*), fitnah lisan (*slander*), fitnah tertulis (*libel*). Dalam bahasa Indonesia, belum ada istilah yang sah untuk membedakan ketiga kata tersebut (Zhang and Luo, 2019).

2.3.6 Hate Speech Dalam Internet

Etika dalam dunia online perlu ditegaskan, mengingat dunia online merupakan hal yang sudah dianggap penting bagi masyarakat dunia. Namun,

semakin banyak pihak yang menyalahgunakan dunia maya untuk menyebarluaskan hal-hal yang tidak lazim mengenai sesuatu, seperti suku bangsa, agama, dan ras. Penyebaran berita yang sifatnya fitnah di dunia Internet, misalnya, menjadi hal yang patut diperhatikan. Internet Service Provider (ISP) biasanya menjadi pihak yang dianggap bertanggung jawab atas segala isi yang mengandung fitnah. Sesungguhnya, isi yang mengandung fitnah berada di luar tanggung jawab ISP; terlebih ada pihak ke tiga yang memasukkannya tanpa sepengetahuan ISP. Sama halnya seperti manajemen dalam toko buku, dunia Internet membedakan peran antara distributor dan publisher. Dalam hal ini, ISP sekadar bertindak sebagai publisher yang mengontrak distributor untuk mengelola jaringan mereka. Hal di ataslah yang sering disebut dengan Libel yakni sebuah pernyataan ataupun ekspresi seseorang yang mengakibatkan rusaknya reputasi orang lain dalam komunitas tertentu karena ekspresinya itu (Zhang and Luo, 2019). Ataupun bisa dalam bentuk pembunuhan karakter dan dalam dunia professional sekalipun. Dalam bukunya yang berjudul 'The New Communication Technology', Mirabito menyatakan ada 12 ribu pengguna Internet yang menjadi korban kejahatan di Internet yang berkenaan dengan: suku bangsa, ras, agama, etnik, orientasi seksual, hingga gender. Nyatanya, kemajuan Internet berjalan seiring dengan peningkatan teror di dunia maya. Contoh kasus pada seorang anak muda berusia 19 tahun yang menggunakan komputer di sekolahnya untuk mengirim surat elektronik berisi ancaman pembunuhan pada 62 siswa lain yang keturunan Asia-Amerika. Contoh kasus di atas adalah

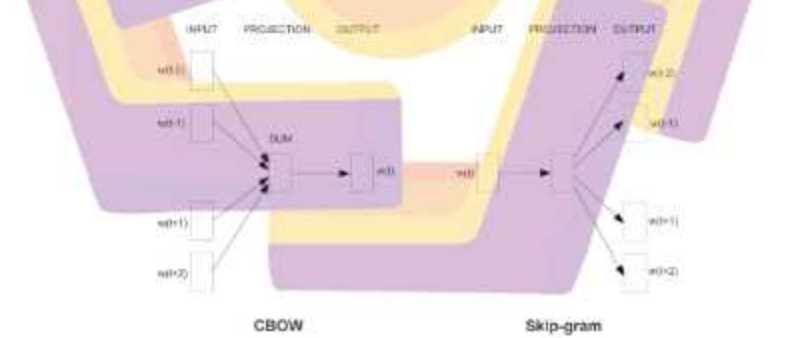
salah satu contoh kasus mengenai istilah hate yang sering dihadapi oleh Amerika dan merupakan sebuah dilema dari kebebasan berekspresi dari first amandment mereka. Kejahatan Hate merupakan masalah serius yang dihadapi oleh Amerika, pada tahun 2001 sendiri terdapat 12.000 individu yang menjadi korban dari kejahatan Hate ini biasanya dikarenakan ras, etnis, negara asal, agama atau kepercayaan mereka, orientasi sex, atau bahkan karena gender mereka (Alfina and Ekanata, 2018). Di Amerika, pernah muncul sebuah aksi yang bernama *The Hate Crime Prevention Act of 2003* yang masih diperdebatkan dalam kongres yang ke-108. Jika aksi ini disahkan kedalam hukum, maka perlindungan dari *Hate Speech* akan semakin terjamin dari lembaga federal. Aksi tersebut didasarkan pada premis legal yaitu:

a. Individu yang menjadi target *Hate Crime* akan mencoba untuk pergi keluar batas negara agar tidak menjadi korban penghinaan.
b. Pelaku kejahatan *Hate Crime* akan mencoba untuk pergi melewati batas negara untuk melakukan penghinaan terhadap korban.
c. Pelaku mungkin menggunakan artikel, termasuk komputer yang mampu menyebarkan informasi ke berbagai negara, untuk melakukan *Hate Crime*.

2.3.7 Word2Vec

Word2vec, yang dipublikasikan oleh *Google* pada tahun 2013, adalah implementasi jaringan syaraf tiruan yang mempelajari representasi terdistribusi kata. Vektor kata terdistribusi powerful dan dapat digunakan

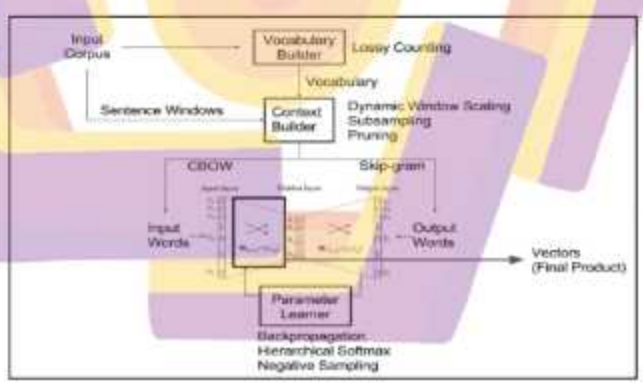
dalam prediksi kata dan terjemahan. *Word2vec* adalah sekelompok model yang digunakan untuk menghasilkan *word embedding*. Representasi vektor dari pembelajaran kata dengan menggunakan model *word2vec* telah terbukti membawa makna semantik dan berguna dalam berbagai tugas NLP. *Word2vec* dikembangkan dengan menggunakan jaringan syaraf dua lapisan. Diperlukan data teks atau korpus teks sebagai masukan dan menghasilkan sekumpulan vektor dari teks yang diberikan. Model *word2vec* menghasilkan satu set vektor atau ruang vektor dari teks. Di ruang vektor, setiap kata unik dalam korpus diberikan vektor yang sesuai. Jadi, ruang vektor merupakan representasi vektor dari semua kata yang ada dalam korpus teks. Terdapat dua model arsitektur yang dapat digunakan pada *word2vec*, yaitu *CBOW* dan *Skip-Gram* . Kedua model tersebut dapat dilihat pada gambar 2.4 sebagai berikut:



Gambar 2.4 Arsitektur CBOW (kiri) dan Skip-gram (kanan)

Word2vec memiliki dua buah arsitektur yang berbeda yaitu *Continuous Bag-of-word* (CBOW) dan *Skip-gram*. Dalam metode CBOW

tujuannya adalah untuk memprediksi kata yang diberikan oleh kata-kata disekitarnya. Pada Gambar 2.4 diilustrasikan arsitektur CBOW dalam memprediksi kata saat ini didasarkan pada konteks. Sebaliknya pada *Skip-gram*, yaitu untuk memprediksi jendela kata yang diberikan oleh kata tunggal. Kedua metode menggunakan jaringan syaraf tiruan sebagai algoritmanya. Awalnya setiap kata dalam kosakata adalah vektor acak N dimensi. Selama pelatihan, algoritma belajar vektor optimal untuk setiap kata menggunakan metode CBOW atau *skip-gram*. Komponen utama arsitektur model *word2vec* adalah jaringan syaraf. *Word2vec* memiliki dua lapisan pada jaringan syaraf untuk menghasilkan bentuk vektor kata. Seperti yang ditunjukkan pada Gambar 2.5, ada tiga blok utama model word2vec yaitu *vocabulary builder*, *context builder*, dan jaringan syaraf dua lapisan.



Gambar 2.5 Word2vec block

Vocabulary builder adalah blok pertama dari model *word2vec*. Dibutuhkan data teks yang sebagian besar dalam bentuk kalimat. Vocabulary

builder digunakan untuk membangun kosakata dari korpus teks yang diberikan. Ini akan mengambil semua kata-kata unik dari korpus. Setiap kata dalam kosakata memiliki hubunga dengan objek kosakata, yang berisi indeks dan jumlah kata. Context builder menggunakan output dari vocabulary builder. Context window seperti sliding window yang dapat ditentukan ukuran window sesuai aplikasi NLP saat menggunakna word2vec. Umumnya aplikasi NLP menggunakan ukuran window lima sampai sepuluh kata. Jika digunakan sepuluh kata, maka akan dipertimbangkan sepuluh kata disisi kiri dari kata tengah dan sepuluh kata disisi kanan kata tengah. Dengan demikian, dapat menangkap informasi tentang kata-kata disekitar kata pusat. Sebagai contoh, ukuran context window=1, dengan kalimat "aku suka belajar komputer.", dan 'belajar' adalah kata pusat. Jadi, harus dipertimbangkan kata-kata disekitarnya sesuai ukuran window seperti kata 'suka' dan 'komputer'. Kemudian iterasi berikutnya 'komputer', kata disekitarnya yaitu 'dalam' dan pada akhir kalimat tanda titik.

Setelah didapatkan pasangan kata, ini akan menjadi input pada jaringan syaraf. Lapisan *input* memiliki banyak neuron karena ada kata-kata dalam kosakata untuk pelatihan. Ukuran lapisan tersembunyi dalam jumlah neuronnya adalah dimensi dari vektor kata yang dihasilkan. Lapisan *output* memiliki jumlah neuron yang sama dengan lapisan input. *Input* pada lapisan *input* pertama adalah kata dengan pengkodean *one-hot*. Asumsikan bahwa ukuran kosakata untuk mempelajari vektor kata adalah V, yang berarti ada nomor V dari kata-kata yang berbeda dalam korpus. Dalam hal ini, posisi kata yang mewakili dirinya dikodekan sebagai 1 dan semua posisi lainnya dikodekan sebagai 0. Misalkan

dimensi dari kata-kata ini adalah N. Jadi, input ke koneksi lapisan tersembunyi dapat diwakili oleh matriks *W1* (matriks *input*) dari ukuran V * N, dengan setiap baris dari matriks *W1* yang mewakili kata dari kosakata. Demikian pula, koneksi dari lapisan tersembunyi ke lapisan output berarti output dari lapisan tersembunyi dapat dijelaskan oleh matriks output lapisan tersembunyi *W0* (matriks *hidden layer*). Matriks *W0* berukuran N * V. Dalam hal ini, setiap kolom dari matriks *W0* mewakili kata dari kosakata yang diberikan.

Untuk mempermudah memahami *Word2vec* maka akan diilustrasikan cara kerja W*ord2vec*, misal kita mempunyai sebuah kalimat yaitu: "Anoa melihat seekor kucing, anoa mengejar kucing, kucing memanjat sebuah pohon", korpus kosakata dari kalimat tersebut memiliki delapan kata, setelah disusun menurut abjad, setiap kata dapat direferensikan oleh indeks seperti yang dapat dilihat dalam Tabel 2.2.

Tabel 2.2 Contoh korpus word2vec

Kosakata	Index
Berenang	1
Hiu	2
Memangsa	3
Memakan	4
Mengincar	5
Manusia	6
Paus	7
Seekor	8

Untuk contoh diatas, jaringan syaraf akan memiliki delapan *neuron input* dan delapan *neuron output*, Kita asumsikan bahwa kita memutuskan

untuk menggunakan tiga *neuron* pada lapisan tersembunyi. Ini berarti matrik W_1 dan W_0 didapat dari matriks 8x3 dan 3x8. Sebelum pelatihan dimulai, matriks ini diinisialisasikan dengan nilai-nilai acak yang kecil seperti biasa dalam pelatihan jaringan syaraf tiruan. Kita asumsikan W_1 dan W_0 diinisialisasikan dengan nilai-nilai berikut:

$$W1=\begin{bmatrix}0.2 & 0.3 & 0.4\\ 0.1 & 0.5 & 0.8\\ 0.3 & 0.2 & 0.6\\ 0.5 & 0.3 & 0.4\\ 0.8 & 0.6 & 0.2\\ 0.2 & 0.3 & 0.4\\ 0.1 & 0.6 & 0.5\\ 0.3 & 0.4 & 0.7\end{bmatrix}$$

$$W0=\begin{bmatrix}0.4 & 0.6 & 0.1 & 0.5 & 0.8 & 0.4 & 0.5 & 0.2\\ 0.9 & 0.2 & 0.1 & 0.5 & 0.7 & 0.3 & 0.2 & 0.4\\ 0.4 & 0.2 & 0.5 & 0.1 & 0.3 & 0.6 & 0.1 & 0.8\end{bmatrix}$$

Misalkan kita ingin jaringan untuk mempelajari hubungan antara kata "hiu" dan kata "memakan". Artinya jaringan harus menunjukkan probabilitas tinggi untuk kata "memakan" ketika kata "hiu" dimasukkan ke jaringan. Dalam terminologi *word embedding*, kata "hiu" disebut sebagai konteks dan kata "memakan" disebut sebagai sasaran kata (target kata). Dalam hal ini, vektor input dari kata "hiu" akan dikodekan menjadi [01000000], hanya indeks kedua dari vektor yang bernilai 1, hal ini dikarenakan kata *input* "hiu" berada pada posisi nomor dua dalam daftar yang diurutkan didalam *corpus* kata. Sedangkan kata yang menjadi target yaitu kata "memakan", akan dikodekan menjadi [00010000], terlihat hanya indeks keempat dari vektor

yang bernilai 1 dikarenakan kata input "memakan" berada pada posisi nomor empat dalam daftar yang diurutkan didalam *corpus* kata.

Dengan vektor *input* menggunakan kata "kucing", *output* pada *neuron* lapisan tersembunyi dapat dihitung dengan persamaan 3.1

$$H = X * W_1 = [0.1 \quad 0.5 \quad 0.8] \tag{3.1}$$

Dengan *H* adalah *hidden layer*, *X* adalah *input neuron* sebelumnya dan W_1 adalah bobot, *output* dari *neuron* tersembunyi yaitu vektor *H* meniru bobot dari baris kedua matriks dari W_1 hal ini dikarenakan input yang diberikan ke jaringan dikodekan menggunakan representasi "*1-out of-V*" yang berarti bahwa hanya satu baris masukan yang nilainya bernilai satu dan sisanya dari jalur input diatur ke nol. Jadi fungsi dari input ke koneksi lapisan tersembunyi pada dasarnya adalah dengan menyalin vektor kata input ke lapisan tersembunyi, kemudian untuk output dari lapisan tersembunyi didapat dari persamaan 3.2

$$H = X * W_0 = [0.81 \quad 0.32 \quad 0.46 \quad 0.38 \quad 0.67 \quad 0.67 \quad 0.23 \quad 0.86] \tag{3.2}$$

Selanjutnya untuk menghasilkan probabilitas untuk kata-kata, *Word2vec* menggunakan fungsi *softmax* pada lapisan *output*, dengan demikian probabilitas untuk delapan kata-kata dalam korpus dapat dihitung menggunakan persamaan 3.3.

$$yk = P_r(\text{kata}_k \mid \text{kata}_{\text{contex}}) = \frac{\exp(k)}{\sum n \exp(n)} \tag{3.3}$$

Dengan demikian probabilitas untuk delapan kata-kata dalam korpus adalah 0.158285 0.096969 0.111541 **0.102965** 0.137606 0.137606 0.088623

0.166400. Nilai probabilitas untuk kata "memakan" dicetak tebal, hal ini mengingat vektor sasaran berada pada indeks keempat didalam corpus kata [00010000], vektor kesalahan untuk lapisan output dengan mudah dihitung dengan mengurangkan vektor probabilitas dengan vektor sasaran. Setelah kesalahan diketahui, bobot dalam matriks W_1 dan W_0 dapat diperbarui menggunakan *backpropagation*. Dengan demikian, pelatihan dapat dilanjutkan dengan menghadirkan berbagai pasangan kata konteks-target dari korpus.

BAB III

METODE PENELITIAN

3.1. Jenis, Sifat, dan Pendekatan Penelitian

Adapun jenis, sifat dan pendekatan penelitian ini adalah:

a) Jenis Penelitian

Jenis penelitian yang digunakan dalam penelitian ini adalah Eksperimental. penelitian eksperimen adalah suatu metode penelitian sistematis yang berusaha untuk mencari pengaruh dari suatu perlakuan tertentu yang diberikan pada variabel terhadap variabel yang lain yang tanpa diberikan perlakuan dengan kondisi yang dikendalikan. Penelitian eksperimen merupakan salah satu jenis penelitian kuantitatif yang sangat kuat mengukur hubungan sebab akibat. Penelitian eksperimen dimaksudkan untuk membuktikan suatu hipotesis.

b) Sifat Penelitian

Penelitian eksperimen memiliki 4 faktor utama yaitu hipotesis, variabel independen, variabel dependen, dan subyek. Hipotesis dalam penelitian ini didapatkan melalui kesimpulan pertama yang dilakukan seorang peneliti sebelum melaksanakan penelitian. Variabel yang digunakan dalam penelitian ini terbagi menjadi dua jenis yakni variabel eksperimental dan variabel non-eskperimental. Variabel eksperimental ditetapkan secara langsung sesuai dengan tujuan penelitian, sedangkan variabel non-eksperimental dilakukan secara tidak sengaja.

c) Pendekatan Penelitian

Jenis Penelitian pada proposal ini menggunakan penelitian kuantitatif. Penelitian kuantitatif yaitu metode penelitian yang berlandaskan pada filsafat positivisme digunakan untuk meneliti pada populasi atau sampel tertentu, pengumpulan data menggunakan instrumen penelitian, analisis data bersifat kuantitatif atau statistik, dengan tujuan untuk menguji hipotesis yang telah ditetapkan (Sugiono, 2010).

3.2. Metode Pengumpulan Data

Data dikumpulkan dari cuitan status pengguna di Twitter. Data didapatkan dengan cara *crawling* dengan menggunakan aplikasi RStudio. Data yang didapatkan dari aplikasi Rstudio sebanyak 2440 buah data. Data yang dikumpulkan hanya berupa cuitan status pengguna berupa ujaran kebencian, artinya tulisan status tersebut terindikasi atau terdeteksi sebagai *Hate Speech* (ujaran kebencian). Kalimat *tweet* akan terdeteksi *Hate Speech* jika berhubungan dengan beberapa persamaan kata seperti agama, ras, fisik, gender atau orientasi jenis kelamin, dan kata-kata umpatan lainnya. Pendeteksian juga mampu mengklasifikasikan target, kategori, dan level ujaran kebencian itu sendiri. Ujaran kebencian diklasifikasikan pada tiga level yaitu:

a. *Weak Hate Speech* yaitu level dari kata-kata *Hate Speech* yang ditujukan pada individu tanpa unsur provokasi.
b. *Moderate Hate Speech* adalah level umpatan yang ditujukan kepada kelompok tanpa provokasi.

c. *Strong Hate Speech* adalah level umpatan yang memprovokasi dan berpotensi membuka konflik.

Ujaran kebencian juga dapat dibagi berdasarkan tingkat sosial yaitu :

a. *Ascribed Hate Speech Status*

Merupakan ujaran kebencian berdasarkan kedudukan seseorang dalam masyarakat yang diperoleh dengan sendirinya, biasanya karena faktor keturunan. Contohnya seorang anak bangsawan akan mendapatkan kehormatan dari masyarakat karena mendapat status yang diwariskan dari orang tuanya. Contoh kalimat Hate Speech *Ascribed Hate Speech Status*: "Pemimpin serakah yang hobi bicara sara siapa lagi kalo bukan raja kita! Sultan HB IX!"

b. *Achieved Hate Speech Status*

Merupakan ujaran kebencian berdasarkan status yang diperoleh seseorang melalui usaha-usaha yang disengaja. Contohnya setiap orang bisa menjadi hakim asalkan memenuhi persyaratan tertentu, seperti lulusan fakultas hukum, memiliki pengalaman kerja dalam bidang hukum, dan lulus ujian sebagai hakim.Contoh kalimat Hate Speech *Achieved Hate Speech Status*: "Kepicikanmu tak menjadi soal, menganggap semua orang sedangkal akalmu, Itu soal serius seorang hakim".

c. *Assigned Hate Speech Status*

Merupakan ujaran kebencian berdasarkan status yang diperoleh dari pemberian pihak lain. Maksudnya adalah suatu kelompok atau golongan tertentu memberikan status yang lebih tinggi kepada seseorang yang

berjasa. Status ini diberikan karena orang tersebut telah memperjuangkan sesuatu untuk memenuhi kebutuhan ataupun kepentingan masyarakat. Contoh assigned status adalah gelar pahlawan revolusi , siswa teladan dan peraih Kalpataru. Contoh kalimat Hate Speech *Assigned Hate Speech Status*: "PDRI hingga Soeharto tidak masuk Keppres sama aja kalo Jasmerah itu gak kan guna! kami tidak akan mengakui pemerintahan dan kepahlawanan suharto!"

(wikipedia, https://id.wikipedia.org/wiki/Status_sosial, diakses pada 08/03/2022 15:30).

Setelah data terkumpul, selanjutnya dilakukan filtering data sehingga data yang dihasilkan sesuai untuk penelitian ujaran kebencian di Twitter. Hasil pengumpulan data dapat dilihat pada tabel 3.1 sebagai berikut:

Tabel 3.1 Tabel hasil pengumpulan tweet bahasa indonesia

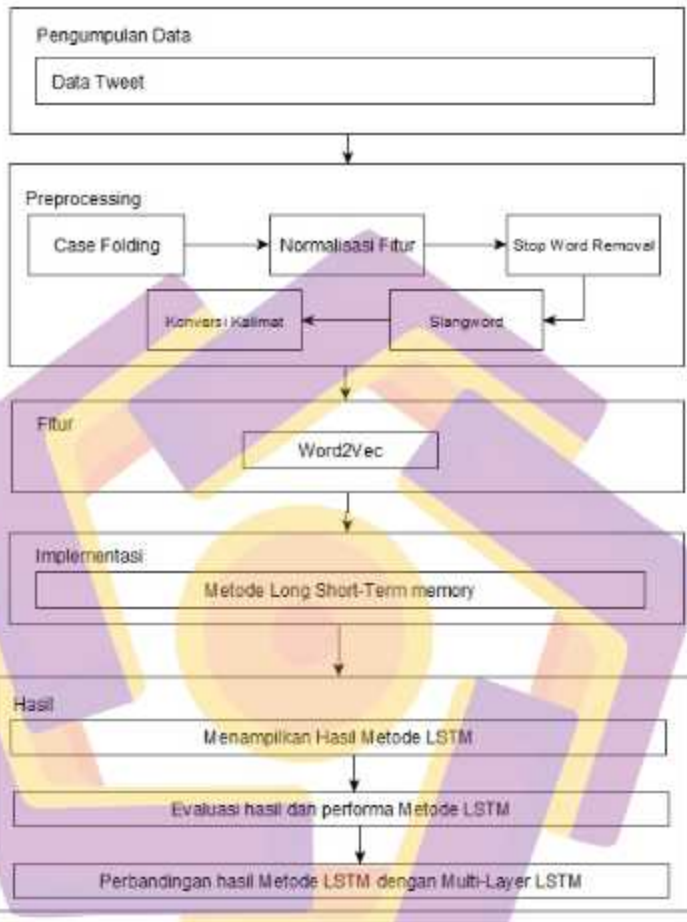
Data Tweet	Indikasi Hate Speech
Sejak kapan politik dan agama menjadi sesuatu yang tidak bisa dipisahkan ya... Sejak jaman nenek anda ya?	Positif
Lawan jangan takut . Anis harus dilantik . Memang nya mereka yang punya Indonesia berbuat semau hatinya ?	Positif
Efek dari ketakutan politisi yang takut akan kekalahan di pilkada mendatang...yg sudah kalah karena kita peduli akan agama.. agama kuat negara pun ikut kuat...	Negatif
Memilihlah berdasar kecerdasan seseorang yang anda pilih dengan begitu anda akan terlihat cerdas juga.	Negatif

3.3. Metode Analisis Data

Metode analisis data pada penelitian ini adalah dengan cara mengumpulan data di *Twitter* sebanyak 2440 data, kemudian dilakukan *preprocessing*. Tahapan yang dilakukan pada *preprocessing* yaitu *Case Folding*, Normalisasi Fitur, *Stop Word Removal*, *Slangword* dan Konversi kalimat. Kemudian dilanjutkan fitur pemrosesan data dengan *bag-of-word vector*. Setelah selesai dengan fitur pemrosesan, data diolah dengan Metode LSTM.

3.4. Alur Penelitian

Alur penelitian yang akan dilakukan penulis adalah sebagai berikut:



Gambar 3.1 Rancangan dan penelitian

Berdasarkan desain penelitian pada gambar 3.1, maka dapat dijabarkan sebagai berikut :

1. Pengumpulan Data

   Data dikumpulkan dari cuitan status pengguna di Twitter. Data didapatkan dengan cara *crawling* dengan menggunakan aplikasi RStudio. Data yang

didapatkan dari aplikasi Rstudio sebanyak 2440 buah data. Data yang dikumpulkan hanya berupa cuitan status pengguna berupa ujaran kebencian, artinya tulisan status tersebut terindikasi atau terdeteksi sebagai *Hate Speech* (ujaran kebencian). Setelah data terkumpul, selanjutnya dilakukan filtering data sehingga data yang dihasilkan sesuai untuk penelitian ujaran kebencian di Twitter.

2. Preprocessing

Tahapan *preprocessing* adalah mengelola data *tweet* agar lebih mudah dalam melakukan proses selanjutnya. Dalam preprocessing terdapat dua tahapan yang harus dilakukan yaitu:

a. Case Folding

Tidak semua dokumen teks konsisten dalam penggunaan huruf. Oleh karena itu, tahap ini bertujuan untuk merubah karakter huruf di dalam komentar menjadi karakter huruf kecil semua. Hasil tahap case folding dapat ditunjukkan pada tabel 3.2 sebagai berikut:

Tabel 3.2 Tabel tahap casefolding

Input	Output
Sejak kapan politik dan agama menjadi sesuatu yang tidak bisa dipisahkan ya... Sejak jaman nenek anda ya?	sejak kapan politik dan agama menjadi sesuatu yang tidak bisa dipisahkan ya... sejak jaman nenek anda ya?

b. Normalisasi Fitur

Tahap ini bertujuan untuk menghapus karakter khusus dalam komentar seperti tanda baca (titik(.), koma(,), tanda tanya(?), tanda seru(!)

dan sebagainya), angka numerik (0-9), dan karakter lainnya ($, %, *, dan sebagainya). Hal ini dilakukan agar proses klasifikasi menjadi lebih akurat. Hasil tahap normalisasi fitur dapat ditunjukkan pada tabel 3.3 sebagai berikut:

Tabel 3.3 Tabel tahap normalisasi fitur

Input	Output
sejak kapan politik dan agama menjadi sesuatu yang tidak bisa dipisahkan ya... sejak jaman nenek anda ya?	sejak kapan politik dan agama menjadi sesuatu yang tidak bisa dipisahkan ya sejak jaman nenek anda ya

c. Stop Word Removal

Penghapusan Stopword merupakan proses penghilangan kata stopword. Stopword adalah kata- kata yang sering kali muncul dalam dokumen namun arti dari kata kata tersebut tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu. Misalnya "di", "oleh", "pada", "sebuah", "karena" dan lain sebagainya. Hasil tahap *stop word removal* dapat dilihat pada tabel 3.4 sebagai berikut.

Tabel 3.4 Tabel hasil stopword

Kalimat	Hasil Stopword Removal
Sejak	Sejak
Kapan	Kapan
Politik	Politik
Dan	Agama
Agama	Menjadi

Tabel 3.4 Tabel hasil stopword (lanjutan)

Kalimat	Hasil Stopword Removal
Menjadi	Dipisahkan
Sesuatu	Jaman
Yang	Nenek
Tidak	Anda
Bisa	
Dipisahkan	
Ya	
Sejak	
Jaman	
Nenek	
Anda	
Ya	

d. Slangword

Slangword merupakan proses mengubah kata yang tidak baku menjadi kata yang baku. Tahap ini dilakukan dengan menggunakan bantuan kamus *slangword* dan padanan dalam kata baku. Tahapan ini akan memeriksa kata yang terdapat dalam kamus *slangword*. Contoh dari *Slangword* dapat dilihat pada Tabel 3.5

Tabel 3.5 Tabel hasil slangword

Kalimat	Hasil Slangword
Aq	Aku
Alus	Halus
Bkn	Bukan
Cm	Cuma

Tabel 3.5 Tabel hasil slangword (lanjutan)

Kalimat	Hasil Slangword
Dah	Sudah
Gak	Tidak
Kalo	Kalau

e. Konversi Kalimat

Konversi kalimat dilakukan dalam beberapa tahap yaitu pembuatan kamus kata, mengubah kalimat menjadi angka, dan *padding*. Proses ini dilakukan agar siap digunakan sebagai input pada metode LSTM. Proses diawali dengan membuat kamus kata yang digunakan untuk memberikan id kata yang terdapat dalam kalimat pada data *Hate Speech* yang telah melalui proses *preprocessing*. Dalam penelitian ini kamus kata yang digunakan yaitu 5000. Contoh data *Hate Speech* yang sudah melalui proses *preprocessing* dapat dilihat pada Tabel 3.6

Tabel 3.6 Kalimat *Hate Speech* yang telah melalui *preprocessing*

No	Kalimat
1	lawan jangan takut anis dilantik mereka indonesia berbuat semau hatinya
2	efek takut politisi takut kalah pilkada mendatang kalah kita peduli agama negara kuat
3	memilihlah berdasar kecerdasan seseorang anda pilih terlihat cerdas.

Dari data *Hate Speech* yang telah melalui proses *preprocessing* seperti pada Tabel 3.6, kemudian akan dibuat kamus kata. Ada tiga tahap yang harus dilakukan, pertama memisahkan kalimat menjadi satuan kata.

Kedua yaitu menghapus duplikasi kata. Ketiga adalah memberi id pada masing-masing kata secara berurutan berdasarkan kata yang sering muncul. Kata pertama diberi id=1 dan seterusnya hingga kata terakhir. Contoh kamus kata dapat dilihat pada Gambar 3.2

lawan = 1	berbuat = 8	kalah = 15	berdasar = 22
jangan = 2	semau = 9	kita = 16	kecerdasan = 23
takut = 3	hatinya = 10	peduli = 17	seseorang = 24
anis = 4	efek = 11	agama = 18	anda = 25
dilantik = 5	politisi = 12	negara = 19	pilih = 26
mereka = 6	pilkada = 13	kuat = 20	terlihat = 27
indonesia = 7	mendatang = 14	memilihlah = 21	cerdas = 28

Gambar 3.2 Kamus kata

Tahap selanjutnya mengubah kalimat menjadi angka yaitu mengubah kata yang menyusun kalimat *Hate Speech* menjadi angka berdasarkan kamus kata yang telah dibuat pada tahap sebelumnya. Data kalimat *Hate Speech* pada Tabel 3.6 akan diubah menjadi angka berdasarkan kamus kata seperti pada Gambar 3.2 Contoh hasil konversi kalimat dapat dilihat pada Gambar 3.3

1, 2, 3, 4, 5, 6, 7, 8, 9, 10
11, 3, 12, 3, 15, 13, 14, 15, 16, 17, 18, 19, 20
21, 22, 23, 24, 25, 26, 27, 28

Gambar 3.3 Hasil konversi kalimat

Tahap terakhir adalah memberi *padding* pada kalimat agar panjang kalimatnya sama. *Padding* dilakukan dengan menentukan panjang maksimal kata pada suatu kalimat atau mencari jumlah kalimat terpanjang pada data *Hate Speech*. Pada Gambar 5 dapat dilihat kalimat terpanjang pada kalimat kedua dengan 13 kata, sedangkan kalimat pertama memiliki 10 kata dan kalimat ketiga memiliki 8

kata. Oleh karena itu, kalimat pertama dan kedua harus disesuaikan dengan kalimat terpanjang. Kalimat pertama dan kedua harus dilakukan *padding* dengan menambahkan angka 0 agar panjangnya sama dengan kalimat terpanjang. Penambahan angka 0 dapat dilakukan diawal maupun diakhir kalimat. Hasil padding diawal kalimat dapat dilihat pada Gambar 3.4

Kalimat 1	0	0	0	1	2	3	4	5	6	7	8	9	10
Kalimat 2	11	3	12	3	15	13	14	15	16	17	18	19	20
Kalimat 3	0	0	0	0	0	21	22	23	24	25	26	27	28

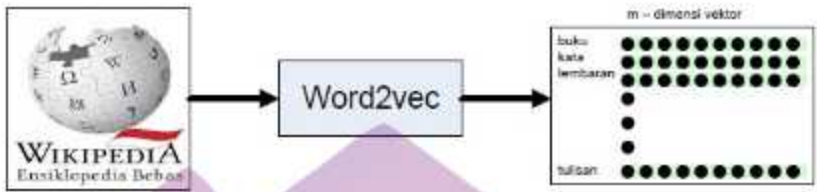
Gambar 3.4 Proses *padding* kalimat *Hate Speech*

3. Word2vec

Pada tahap ini diawali dengan mengambil data korpus Wikipedia Indonesia (https://dumps.wikimedia.org/). Kemudian data tersebut dikonversi kedalam format teks menggunakan bahasa pemrograman *python*. Data teks Wikipedia Indonesia tersebut akan dijadikan input untuk membuat model *word2vec* dan output yang dihasilkan berupa kata dan satu set vektor.

Pada proses ini memanfaatkan *library* gensim 3.4.0 (https://pypi.python.org/pypi/gensim). Library tersebut berfungsi untuk membaca file teks Wikipedia Indonesia yang kemudian di *training* dengan jaringan syaraf tiruan untuk mendapatkan nilai vektor yang kemudian disimpan dalam format file *txt*. Arsitektur *word2vec* yang digunakan pada penelitian ini adalah arsitektur CBOW (*Continuous Bag-of-word*) dan *skipgram* dengan ukuran parameter yang sama yaitu vektor 300 dimensi, windows ukuran 10, dan jumlah minimal kata

sebanyak 5 kata. Proses pembuatan model *word2vec* dapat dilihat pada Gambar 4.6.



Gambar 3.5 Proses pembuatan model word2vec

4. Konversi Vektor

Konversi vektor dilakukan dengan menggunakan kamus kata yang telah dibuat dan akan direpresentasikan pada model *word2vec* yang telah dibuat pada tahap sebelumnya. Jika kata dapat ditemukan dalam model *word2vec* (memiliki representasi vektor untuk sebuah kata), maka akan digunakan vektor *word2vec* tersebut untuk mewakili kata tersebut. Namun jika kata tidak ditemukan dalam model, maka akan diganti dengan nilai vektor acak dari rata-rata semua nilai vektor pada model. Selanjutnya didapatkan nilai vektor kamus kata berdasarkan model *word2vec*. Jika diasumsikan *tweet* memiliki 5 kata, dan vektor *word2vec* 300 dimensi, maka input menjadi 5 x 300 = 1500 dimensi.

5. Implementasi Sistem

a. Metode LSTM

Pada penelitian ini arsitektur LSTM yang diujikan terbagi menjadi dua bagian yaitu :

1. LSTM dengan 1 LSTM layer, arsitekturnya dapat dilihat pada gambar 3.6
2. LSTM dengan 2 LSTM layer, arsitekturnya dapat dilihat pada gambar 3.7

Gambar 3.6 Arsitektur LSTM dengan 1 layer LSTM

Gambar 3.7 Arsitektur LSTM dengan 2 layer LSTM

Layer pertama adalah *input* dari LSTM yang berupa *tweet* pengguna *twitter* berbahasa Indonesia dengan ukuran tertentu. Pada penelitian ini *input* yang digunakan berukuran 50 yang artinya 50 adalah panjang maksimum kalimat dari *tweet* yang sudah melalui *preprocessing*. Selanjutnya data tweet tersebut akan diproses menjadi input vektor pada *embedding layer*.

Tujuan *embedding layer* adalah mempelajari pemetaan setiap kata dalam *vocabulary* ke dimensi vektor yang lebih rendah. Pada tahapan ini fitur *Word2vec* digunakan dalam memproses teks. *Input* berupa korpus kata dari hasil *preprocessing*. Pada penelitian ini, kalimat terpanjang adalah 50 kata dan menggunakan model *word2vec* 150 dimensi. Oleh karena itu, input akan menjadi 50

x 150 = 7500 dimensi. *Output* dari lapisan ini berupa satu set vektor akan menjadi input pada lapisan selanjutnya.

Input dari LSTM *layer* adalah output dari *embedding layer* yaitu matriks 50 kali 150 dimensi. Setiap kata diwakili oleh 150 dimensi. Dan 50 kata tersebut akan masuk ke jaringan LSTM secara berurutan dengan memasukkan setiap kata 150 dimensi. Kemudian diberi *batch size* sehingga input timesteps tergantung pada jumlah batch size yang diberikan.

Output yang dihasilkan oleh LSTM *layer* adalah sejumlah *neuron* LSTM yang ditentukan. Output diperoleh dari LSTM *layer* pada timesteps terakhir yang dihasilkan dari kalkulasi seluruh kata yang menjadi input LSTM. Kemudian dalam LSTM 2 *layer*, setiap output dari LSTM *layer* pertama akan menjadi input pada LSTM *layer* kedua.

Layer output merupakan lapisan terakhir dimana pada layer ini terdapat 3 neuron didalamnya yang mewakili dari 3 kalimat dari kalimat ujaran kebencian yang telah diklasifikasikan sebelumnya. Layer ini terhubung secara penuh dengan layer sebelumnya. Layer ini merupakan softmax layer sehingga jika nilai aktivasi pada masing-masing neuron ditotal bernilai sama dengan 1.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1 Implementasi Perangkat Lunak dan Keras

Sistem yang dibangun dalam penelitian ini diimplementasikan berdasarkan rancangan sistem pada bab sebelumnya. Sistem yang dibangun diimplementasikan menggunakan Bahasa pemrograman python 3.8. Bahasa pemrograman menggunakan python karena python memiliki library yang luas dan bisa dipakai untuk segala keperluan. Perangkat keras dan perangkat lunak yang digunakan dalam implementasi sistem adalah :

1. Perangkat keras

a. Processor Intel® Core i5-8250U CPU @1.6GHz

b. Hard Disk Drive 1000 GB

c. RAM 4 GB

d. GPU NVIDIA GeForce MX130

2. Perangkat lunak

a. Sistem operasi Windows 10 Pro 64-bit

b. Python 3.6

c. Library Keras

d. Library Tensorflow

e. Library NLTK

f. Library Numpy

g. Library Pandas

h. Library Sastrawi

i. Library gensim

j. Library scikit-learn

k. Library twitteR

l. Library ROAuth

m. Library RStudio

4.2 Implementasi Pengumpulan Data

Pada tahap ini merupakan implementasi pengumpulan data. Implementasi pengumpulan data dilakukan dengan teknik *crawling*. Teknik *crawling* dilakukan untuk mengambil data di media sosial twitter. Proses ini dilakukan dengan menggunakan aplikasi RStudio. RStudio dapat mengambil data dari media sosial twitter dengan cara memasukkan kata kunci yang diinginkan pada aplikasinya. Langkah Pertama yang dilakukan adalah memanggil *library twitteR* dan *library ROAuth*, kemudian memasukkan *activation keys* dan *secret keys* akun twitter yang digunakan lalu dilajutkan pengambilan data dengan cara memasukkan kata kunci. Kata kunci yang digunakan pada penelitian ini adalah #TurunkanJokowi dan #Jokowi2Periode. Kemudian hasil *crawling* datanya disimpan dalam format file csv. Implementasi *crawling* pengambilan data diperlihatkan pada gambar 4.1

```
> library(twitteR)
> library(ROAuth)
>
setup_twitter_oauth('j8UesNlEqAwgV54HsEMfaILAe','2a6CBdGQm
ZVd60Uo8l8GwFy5g8uUlVzufEbhPIdvmNwskoVX0L','10195463455756
12416-
```

```
YnAj8GWYMfGSkJG9rHMXa3utDlxtTR','qYNkxhDPAFgoNcVO7gA75ziGL
L2bBFDanbuO2pFjvvVrt')
[1] "using direct authentication"
> 1
[1] 1
> tweets<-searchTwitter("#Jokowi2Periode",n=2000)
> Jokowi.df<-twListToDF(tweets)
> view(Jokowi.df)
> write.csv(Jokowi.df,"E:/Amikom/Thesis/Dataset/Jokowi.csv")
```

Gambar 4.1 *Crawling* mengambil data di Rstudio

Baris 1 menjelaskan penggunaan *Library twitteR*. Baris 2 menjelaskan penggunaan Library ROAuth. Baris 3 menjelaskan langkah untuk memasukkan kode *api key*, kode *api key secret*, *token key*, *token key secret* pada akun *twitter* yang digunakan. Baris 4 menjelaskan langkah memasukkan kata kunci dan jumlah data yang ingin dihasilkan pada twitter. Langkah 5 menjelaskan perintah untuk melihat isi variabel *tweet* dalam bentuk data frame. Baris 6 menjelaskan langkah untuk menampilkan hasil data yang diambil di *twitter*. Baris 7 menjelaskan langkah menyimpan data yang diambil dalam format .csv

4.3 Implementasi Preprocessing

Tujuan dari *Preprocessing* adalah mendapatkan data *Hate Speech* agar proses perhitungan akurasi menjadi lebih akurat. Tahap tahap dalam *presprocessing* adalah *casefolding, normalisasi fitur, stopword removal* dan konversi *slangwords*. Proses *preprocessing* ditunjukkan pada gambar 4.2

```
1 from Sastrawi.StopWordRemover.StopWordRemoverFactory import
2 StopWordRemoverFactory
3 import re
4 import pandas as pd
5 from numpy import nan
6 nan == nan
7 #Open Datasets
```

```
file = pd.read_csv('E:/amikom/Laporan Thesis/Dataset/Data_Hate
Speech_1.csv',encoding='cp1252')
#Preprocessing Slangword
def konversi_slangword(kamus):
    kamus_slangword = eval(open("E:/amikom/Laporan
Thesis/Dataset/slangwordsku.txt").read())
    pattern = re.compile(r'\b( ' + '|'.join
(kamus_slangword.keys())+r')\b')
    content = []
    for kata in kamus:
        filteredSlang = pattern.sub(lambda x:
kamus_slangword[x.group()],kata)
        content.append(filteredSlang.lower())
    kamus = content
    return kamus
#Pengkategorian Hate Speech : Hate Speech, netral
sentiments = file.kategori.unique()
dict={}
for i,kategori in enumerate(sentiments):
    dict[kategori]=i
labels = file.kategori.apply(lambda x:dict[x])
print (labels)
def casefolding (text):
    text = text.lower()
    return text
def normalisasi (textt):
    textt = re.sub('[^\x00-\x7f]',r'', textt)
    return textt
def stopword (teks):
    factory  = StopWordRemoverFactory()
    stoplist = factory.get_stop_words()
    stopword = factory.create_stop_word_remover()
    teks1= stopword.remove(teks)
    return teks1
text = [file]
for text1 in text:
    label = text1['kategori']
    text1 = text1['text']
    text1 = text1.apply(casefolding)
    text1 = text1.apply(normalisasi)
    text1 = text1.apply(stopword)
    text1 = text1.apply("".join)
```

```
50      textl = textl.apply(konversi_slangword)
51      textl = textl.apply("".join)
52      print(textl)
53  tweet_data = {'text':textl, 'kategori': label}
54  df = pd.DataFrame (tweet_data, columns = ['text', 'kategori'])
55  print(df.info())
56  df.to_csv('data_Hate_Speech_okey.csv', sep=',', encoding='utf-
57  8')
```

Gambar 4.2 Koding Preprocessing

4.3.1 Casefolding

Proses *casefolding* merupakan proses merubah huruf besar menjadi huruf kecil. Pada gambar 4.2 baris 30 sampai baris 32 menunjukkan implementasi dari *Casefolding*. Secara manual tahap casefolding dapat dilihat pada tabel 4.1 sebagai berikut.

Tabel 4.1 Tabel tahap casefolding

Input	Output
Sejak kapan politik dan agama menjadi sesuatu yang tidak bisa dipisahkan ya... Sejak jaman nenek anda ya?	sejak kapan politik dan agama menjadi sesuatu yang tidak bisa dipisahkan ya... sejak jaman nenek anda ya?

4.3.2 Normalisasi Fitur

Proses normalisasi fitur merupakan proses menghilangkan tanda baca (titik (.), tanda seru (!), koma (,), tanda tanya (?), dan sebagainya). Pada gambar 4.2 baris 33 sampai baris 35 menunjukkan implementasi dari normalisasi fitur. Secara manual tahap Normalisasi Fitur dapat dilihat pada tabel 4.2 sebagai berikut.

Tabel 4.2 Tabel tahap normalisasi fitur

Input	Output
sejak kapan politik dan agama menjadi sesuatu yang tidak bisa dipisahkan ya... sejak jaman nenek anda ya?	sejak kapan politik dan agama menjadi sesuatu yang tidak bisa dipisahkan ya sejak jaman nenek anda ya

4.3.3 Stopword Removal

Proses *stopword removal* merupakan proses menghilangkan kata-kata yang tidak memiliki arti yang berkesinambungan dalam sebuah kalimat (dan, yang, ke, dari, dan sebagainya). Pada gambar 4.2 baris 36 sampai baris 41 menunjukkan implementasi dari normalisasi fitur. Secara manual tahap Normalisasi Fitur dapat dilihat pada tabel 4.2 sebagai berikut. Secara manual tahap Stopword Removal dapat dilihat pada tabel 4.3 sebagai berikut.

Tabel 4.3 Tabel tahap stopword removal

Kalimat	Hasil Stopword Removal
Sejak	Sejak
Kapan	Kapan
Politik	Politik
Dan	Agama
Agama	Menjadi
Menjadi	Dipisahkan
Sesuatu	Jaman
Yang	Nenek
Tidak	Anda
Bisa	
Dipisahkan	

Tabel 4.3 Tabel tahap stopword removal (lanjutan)

Kalimat	Hasil Stopword Removal
Ya	
Sejak	
Jaman	
Nenek	
Anda	
Ya	

4.3.4 Konversi Slangword

Proses konversi *slangwords* merupakan proses merubah kata-kata yang tidak baku menjadi kata yang baku (contoh: kata aq menjadi kata aku, kata gak menjadi kata tidak, dan sebagainya). Pada gambar 4.2 baris 10 sampai baris 22 menunjukkan implementasi dari normalisasi fitur. Secara manual tahap Konversi Slangword dapat dilihat pada tabel 4.4 sebagai berikut.

Tabel 4.4 Tabel tahap konversi slangword

Kalimat	Hasil Slangword
Aq	Aku
Alus	Halus
Bkn	Bukan
Cm	Cuma
Dah	Sudah
Gak	Tidak
Kalo	Kalau

4.3.5 Konversi Kalimat

Implementasi konversi kalimat kedalam bentuk numerik dimulai dengan membuat kamus kata untuk memberikan id pada setiap kata yang terdapat didalam data *Hate Speech*. Pembuatan kamus memanfaatkan library keras. Langkah pertama menginisialisasi tokenizer dengan parameter num_words=NUM_WORDS yaitu jumlah kamus kata dibatasi berdasarkan kata yang memiliki frekuensi kemunculan teratas sebanyak NUM_WORDS. Kemudian fit_on_texts secara otomatis melakukan pencarian id kata dari data *Hate Speech* yang telah melalui proses *preprocessing*. Kamus kata akan disimpan dalam variable word_index yang berisi id kata untuk setiap kata dalam data *Hate Speech* yang diurutkan berdasarkan jumlah kemunculan kata. Gambar 5.4 baris 1 sampai dengan baris 3 menjelaskan proses pembuatan kamus kata.

```
1 tokenizer = Tokenizer(num_words=NUM_WORDS)
2 tokenizer.fit_on_texts(texts)
3 word_index = tokenizer.word_index
4 sequences_train = tokenizer.texts_to_sequences(train_data.text)
5 sequences_test = tokenizer.texts_to_sequences(test_data.text)
6 X_train = pad_sequences(sequences_train)
7 X_test = pad_sequences(sequences_test,maxlen=X_train.shape[1])
8 y_train = to_categorical(np.asarray(labels[train_data.index]))
9 y_test = to_categorical(np.asarray(labels[test_data.index]))
```

Gambar 4.3 Koding konversi kalimat

Tahap selanjutnya adalah mengubah kalimat *Hate Speech* menjadi susunan angka berdasarkan kamus kata yang sudah dibuat. Gambar 4.3 baris 4 dan 5

dijelaskan bahwa texts_to_sequences digunakan untuk mengkonversi teks kalimat menjadi susunan bilangan bulat berdasarkan kamus kata dalam word_index. Setelah berhasil dikonversi akan dilakukan *padding* yaitu untuk menyamakan panjang kalimat input. Pada Gambar 4.3 baris 7 menunjukkan pad_sequences digunakan untuk menyamakan panjang kalimat sesuai dengan jumlah yang terdapat pada parameter maxlen. Demikian juga label dikonversi ke format one-hot vector yang dapat dipahami model LSTM. Gambar 4.3 baris 8 dan 9 dijelaskan to_categorical akan mengubah tiga label menjadi one-hot vector.

4.3.6 Konversi Vektor

Proses konversi kamus kata kedalam bentuk vektor yang didapat dari model *word2vec* dapat dilihat pada gambar 4.4. Pada baris 1 sampai baris 13 menjelaskan proses dari konversi vektor. Baris 21 menjelaskan proses penyimpanan data pada output 1. Baris 22 menjelaskan hasil dari output 2.

```
1  word_vectors = KeyedVectors.load_word2vec_format ('E:\...\Kode Laporan
   Thesis\wiki.id.text.model.model200skipgram.txt', encoding="utf8")
2  EMBEDDING_DIM=200
3  vocabulary_size=min(len(word_index)+1,NUM_WORDS)
4  embedding_matrix = np.zeros((vocabulary_size, EMBEDDING_DIM))
5  for word, i in word_index.items():
6    if i>=NUM_WORDS:
7      continue
8    try:
```

```
9       embedding_vector = word_vectors[word]
10      embedding_matrix[i] = embedding_vector
11    except KeyError:
12  embedding_matrix[i]=np.random.normal(0,np.sqrt(0.25),EMBEDDING_DIM)
13  del(word_vectors)


14  from keras.layers import Embedding
15  embedding_layer = Embedding(vocabulary_size,
16                              EMBEDDING_DIM, weights=[embedding_matrix],
17                              trainable=False)


18  from tensorflow.keras.optimizers import Adam
19  from keras.preprocessing.sequence import pad_sequences
20  from keras.models import Sequential
21  from keras.layers import LSTM
22  embed_dim = 128
23  lstm_out = 128
24  sequence_length = X_train.shape[1]
25  inputs = Input(shape=(sequence_length,))
26  embedding = embedding_layer(inputs)
27  reshape = Reshape((sequence_length,EMBEDDING_DIM,1))(embedding)
    adam = Adam(lr=1e-1)
```

Gambar 4.4 Koding Konversi vektor

4.4 Implementasi Long Short-Term Memory

Kode program pada sistem ini dibuat menggunakan *Library Keras* dan *Library tensorflow*, sebuah library berbasis bahasa pemrograman *python* yang

memiliki kemampuan pada proses komputasi secara efisien. Proses komputasi ini efisien karena berjalan secara lancar untuk proses komputasi GPU. Arsitektur LSTM dengan 1 layer dapat dilihat pada Gambar 4.5 dan arsitektur LSTM dengan 2 LSTM *layer* tampak Gambar 4.6.

```
1  model = Sequential()
2  model.add(Embedding(input_dim = len(word_index)+1, output_dim =
   EMBEDDING_DIM, input_length = sequence_length,
   weights=[embedding_matrix], trainable=False))
3  model.add(LSTM(lstm_out))
4  model.add(Dense(2,activation='softmax'))
5  model.compile(loss = 'binary_crossentropy', optimizer='adam', metrics =
   ['accuracy'])
6  history = model.fit(X_train, y_train, batch_size=64, epochs=5,
   validation_data=(X_test, y_test), shuffle = True)
7  print(model.summary())
8  scores = model.evaluate(X_test, y_test, verbose=0)
9  print("Accuracy: %.2f%%" % (scores[1]*100))
```

Gambar 4.5 Kode program LSTM 1 Layer

```
1  model = Sequential()
2  model.add(Embedding(input_dim = len(word_index)+1, output_dim =
   EMBEDDING_DIM, input_length = sequence_length,
   weights=[embedding_matrix], trainable=False))
3  model.add(LSTM(((lstm_out)), return_sequences=True, activation='sigmoid'))
4  model.add(LSTM((2), activation='sigmoid'))
5  model.add(Dense(2, activation='softmax'))
```

```
6  model.compile(loss = 'binary_crossentropy', optimizer='adam', metrics =
   ['accuracy'])
   history = model.fit(X_train, y_train, batch_size=64, epochs=15,
7  validation_data=(X_test, y_test), shuffle = True)
   print(model.summary())
8  scores = model.evaluate(X_test, y_test, verbose=0)
9  print("Accuracy: %.2f%%" % (scores[1]*100))
10
```

Gambar 4.6 Kode program LSTM 2 Layer

4.4.1 Implementasi Input Layer

Input dari model LSTM diperoleh dari hasil *crawling* pada kalimat *Hate Speech*. Setiap input harus memiliki ukuran yang sama. Dalam penelitian ini panjang maksimal kata adalah sebanyak 50 kata.

4.4.2 Implementasi Embedding Layer

Keras menyediakan kode pustaka untuk melakukan embedding dengan *b*menggunakan *Keras.layers.Embedding* yang dapat dilihat pada Gambar 4.7 baris 1 sampai dengan baris 4. Input dari lapisan Embedding memiliki 4 dimensi. `embedding_dim` mendefinisikan input_shape dari lapisan berikutnya. Ukuran pada parameter ini menyesuaikan dengan ukuran vektor yang dihasilkan oleh *word2vec*. `Weights` merupakan bobot dari setiap kata yang dihasilkan dari vektor *word2vec*. `input_length` yaitu jumlah kata pada suatu kalimat yang dimasukkan ke jaringan. `trainable=False` merupakan bobot setiap kata yang tidak akan diupdate.

```
1  embedding_layer = Embedding (input_dim = vocabulary_size, output_dim =
2                               embedding_dim, weights = [embedding_matrix], input_length
3                               = max_len, trainable=False)
```

Gambar 4.7 Koding embedding layer

4.4.3 Implementasi LSTM Layer

LSTM layer dibangun menggunakan pustaka *python Keras* dengan memanfaatkan library *Keras.layers.LSTM* seperti pada Gambar 4.5 dan Gambar 4.6. Baris 1 pada gambar 4.5 menjelaskan model yang digunakan pada Long Short-Term Memory. Baris 2 adalah kode program untuk proses menentukan lapisan input dengan parameter panjang input sebanyak `sequence_lenght`. Baris 3 sampai baris 9 menjelaskan proses LSTM 1 layer. Pada Gambar 4.6 menjelaskan LSTM 2 layer, yang mana perlu mengubah konfigurasi LSTM layer sebelumnya untuk menghasilkan array 3D sebagai input untuk lapisan LSTM berikutnya dengan mengatur `return_sequences` pada layer menjadi `True`.

4.4.4 Implementasi Output Layer

Implementasi Output layer juga memanfaatkan library *Keras.layers.Dense* seperti pada Gambar 4.5 baris 4 dan Gambar 4.6 baris 5. Pada layer ini terdapat parameter yaitu fungsi aktivasi. Fungsi aktivasi yang digunakan adalah *Softmax* untuk mengelompokkan sentimen kedalam dua kelas yaitu *Hate Speech* dan netral.

4.5 Hasil dan Pembahasan

Tahap ini adalah tahap pengujian sistem yang digunakan untuk mengetahui performa sistem yang dibangun. Pengujian *Hate Speech* digunakan

untuk mengukur akurasi dari *Hate Speech*. Berdasarkan hasil pengujian yang dilakukan maka dapat diketahui akurasi yang terbaik. Metode LSTM 1 Layer dibandingkan dengan Metode LSTM 2 Layer dengan memanfaatkan bahasa pemrograman *python*.

4.5.1 Data

Pada penelitian ini data *Hate Speech* yang digunakan adalah 2440 data yang diambil dari media sosial *twitter*. Data tersebut dibagi menjadi dua sentimen, yaitu sentimen *Hate Speech* dan sentimen netral. Kemudian data tersebut dibagi menjadi dua, yaitu data *training* sejumlah 1440 data dan data *testing* sejumlah 1000 data. Distribusi data *Hate Speech* untuk setiap sentimen ditunjukkan pada tabel 4.5 dan tabel *confusion matrix* datanya ditunjukkan pada tabel 4.6

Tabel 4.5 Distribusi data sentimen

Sentimen	Jumlah Data
Hate Speech	228
Netral	2212

Tabel 4.6 Confusion Matrix Data Hate Speech

n = 2440	Positive	Negative
Positive	192	512
Negative	36	1700

Berdasarkan tabel Confusion Matrix diatas didapatkan hasil True Positive (TP) sebanyak 192 data, False Positive (FP) 512 data, False Negative (FN) sebanyak 36 data, dan False Negatve (TN) sebanyak 1700 data.

4.5.2 Pengujian LSTM

Eksperimen dilakukan untuk mencari model terbaik dalam mengenali sentimen pada *Hate Speech*. Pengujian dilakukan untuk mencari parameter yang menghasilkan akurasi, *precision*, *recall*, dan *f-measure* terbaik dari arsitektur LSTM yang dibuat. Pengujian dilakukan menggunakan dua arsitektur LSTM yaitu LSTM 1 *layer* dan LSTM 2 *layer*. Jika pada pengujian parameter pertama telah didapatkan nilai terbaik, maka akan digunakan dalam pengujian parameter selanjutnya. Parameter dan nilai yang diujikan dapat dilihat pada Tabel 4.7. Selain itu juga akan diuji dua arsitektur *word2vec* yaitu arsitektur CBOW (*Continuous bag-of-word*) dan arsitektur *skip-gram*.

Tabel 4.7 Parameter yang diuji

Epoch	Fungsi Aktivasi
75	Tanh
100	Sigmoid
125	Relu

4.5.3 Pengujian LSTM 1 Layer

4.5.3.1 Pengujian Word2Vec

Pengujian ini diawali dengan pengujian *word2vec* yaitu arsitektur *skipgram dan Continuous bag-of-word* (CBOW). Pengujian ini dilakukan menggunakan metode LSTM 1 layer dengan parameter lain yang dipilih

secara acak. Arsitektur terbaik dari model *word2vec* akan digunakan pada pengujian selanjutnya.

Tabel 4.8 Pengujian word2vec pada LSTM 1 Layer

Arsitektur Word2vec	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
Skipgram	20.13	75.63	76.00	76.00	76.00
CBOW	22.25	69.85	70.00	70.00	69.00

Pada Tabel 4.8 dijelaskan bahwa arsitektur *skipgram* memiliki akurasi lebih baik daripada arsitektur CBOW. Hal ini dikarenakan arsitektur *skipgram* dapat menghasilkan *word embedding* yang lebih baik sehingga dapat meningkatkan akurasi.

4.5.3.2 Pengujian Jumlah Epoch

Pengujian berikutnya adalah *epoch* yaitu dengan menentukan jumlah *epoch* yang akan diuji. Dalam penelitian ini jumlah *epoch* yang diujikan yaitu 75, 100 dan 125. Pada Tabel 4.9 dijelaskan bahwa akurasi pada *epoch* 75 merupakan akurasi terbaik dengan 78.31%, tetapi saat *epoch* diubah menjadi 100 hasil akurasi mengalami penurunan menjadi 76.33%. Sedangkan pada *epoch* 125 juga mengalami penurunan dengan akurasi 75.06%.

Tabel 4.9 Pengujian jumlah epoch

Jumlah Epoch	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
75	11.45	75.63	76.00	76.00	76.00
100	12.32	76.33	74.00	75.00	75.00
125	15.04	75.06	74.00	73.00	73.00

4.5.3.3 Pengujian Fungsi Aktivasi

Pengujian berikutnya adalah pengujian fungsi aktivasi. Pada penelitian ini fungsi aktivasi yang akan diujikan *tanh*, *sigmoid*, dan *relu*. Untuk parameter lain diambil dari nilai parameter yang menghasikan akurasi terbaik pada pengujian sebelumnya yaitu epoch 75. Hasil pengujian fungsi aktivasi dapat dilihat pada Tabel 4.10.

Tabel 4.10 Pengujian fungsi aktivasi

Fungsi Aktivasi	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
Sigmoid	11.45	75.63	76.00	76.00	76.00
Tanh	08.55	72.65	73.00	73.00	72.00
Relu	09.45	70.33	69.00	69.00	70.00

Dari pengujian yang dilakukan menunjukkan bahwa fungsi aktivasi *sigmoid* menghasilkan nilai akurasi terbaik yaitu 75.63%. Dibandingkan dengan fungsi aktivasi *tanh* dengan akurasi 72.65% dan fungsi aktivasi *relu* yang nilai akurasinya 70.33%.

4.5.3.4 Hasil Pengujian LSTM 1 Layer

Setelah dilakukan pengujian terhadap beberapa parameter yang telah ditentukan yaitu arsitektur *word2vec*, jumlah *epoch*, dan fungsi aktivasi, maka hasil keseluruhan pengujian dapat dilihat pada Tabel 4.11

Tabel 4.11 Hasil pengujian LSTM 1 Layer

Parameter	Nilai	Akurasi
Arsitektur *Word2vec*	*Skipgram*	75.63%
Epoch	75	
Fungsi Aktivasi	Sigmoid	

4.5.4 Pengujian LSTM 2 Layer

4.5.4.1 Pengujian Word2Vec

Pengujian ini diawali dengan pengujian *word2vec* yaitu arsitektur *skipgram dan Continuous bag-of-word* (CBOW). Pengujian ini dilakukan menggunakan metode LSTM 2 layer dengan parameter lain yang dipilih secara acak. Arsitektur terbaik dari model *word2vec* akan digunakan pada pengujian selanjutnya.

Tabel 4.12 Pengujian Word2Vec pada LSTM 2 Layer

Arsitektur Word2vec	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
Skipgram	26.22	77.54	77.00	77.00	77.00
CBOW	27.34	72.19	73.00	73.00	73.00

Pada Tabel 4.12 dijelaskan bahwa arsitektur *skipgram* memiliki akurasi lebih baik daripada arsitektur CBOW. Hal ini dikarenakan arsitektur *skipgram* dapat menghasilkan *word embedding* yang lebih baik sehingga dapat meningkatkan akurasi. Jadi, arsitektur *skipgram* akan digunakan pada pengujian selanjutnya.

4.5.4.2 Pengujian Jumlah Epoch

Jumlah *epoch* yang diujikan pada LSTM 2 *layer* ini sama dengan pengujian *epoch* pada LSTM 1 *layer* yaitu jumlah *epoch* 75, 100, dan 125. Sedangkan nilai parameter lain yang digunakan yaitu jumlah *neuron* 75 pada setiap *layer* dan arsitektur *skipgram*.

Tabel 4.13 Pengujian jumlah epoch

Jumlah Epoch	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
75	14.56	77.54	77.00	77.00	77.00
100	15.02	73.53	75.00	73.00	73.00
125	19.34	75.46	74.00	74.00	73.00

Pada Tabel 4.13 dijelaskan bahwa akurasi pada *epoch* 75 merupakan akurasi terbaik dengan 77.54%, tetapi saat *epoch* diubah menjadi 100 hasil akurasi mengalami penurunan menjadi 75.53%. Sedangkan pada *epoch* 125 juga mengalami penurunan dengan akurasi 75.46%.

4.5.4.3 Pengujian Fungsi Aktivasi

Pengujian fungsi aktivasi ini sama dengan pegujian fungsi aktivasi pada LSTM 1 *layer*. Fungsi aktivasi yang diujikan adalah fungsi aktivasi *tanh*, fungsi aktivasi *sigmoid* dan fungsi aktivasi *relu*. Nilai parameter yang digunakan yaitu jumlah epoch 75. Hasil pengujian dapat ditunjukkan pada Tabel 4.14.

Tabel 4.14 Pengujian fungsi aktivasi

Fungsi Aktivasi	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
Sigmoid	14.56	77.54	77.00	77.00	77.00
Tanh	13.40	74.09	74.00	75.00	74.00
Relu	14.21	68.30	69.00	69.00	69.00

Dari pengujian yang dilakukan menunjukkan bahwa fungsi aktivasi *sigmoid* menghasilkan nilai akurasi terbaik yaitu 77.54%. Dibandingkan dengan fungsi aktivasi *tanh* dengan akurasi 74.09% dan fungsi aktivasi *relu* yang nilai akurasinya 68.30%.

4.5.4.4 Hasil Pengujian LSTM 2 Layer

Setelah dilakukan pengujian terhadap beberapa parameter yang telah ditentukan yaitu arsitektur *word2vec*, jumlah *epoch*, dan fungsi aktivasi, maka hasil keseluruhan pengujian dapat dilihat pada Tabel 4.15

Tabel 4.15 Hasil pengujian LSTM 2 Layer

Parameter	Nilai	Akurasi
Arsitektur *Word2vec*	*Skipgram*	77.54%
Epoch	75	
Fungsi Aktivasi	Sigmoid	

4.5.5 Pengujian LSTM 1 Layer Multi Class Labeling

4.5.5.1 Pengujian Word2Vec Multi Class Labeling

Pengujian ini diawali dengan pengujian *word2vec* yaitu arsitektur *skipgram dan Continuous bag-of-word* (CBOW). Pengujian ini dilakukan menggunakan metode LSTM 1 layer dengan multi class labeling dan parameter lain yang dipilih secara acak. Arsitektur terbaik dari model *word2vec* akan digunakan pada pengujian selanjutnya.

Tabel 4.16 Pengujian word2vec pada LSTM 1 layer multi class labeling

Arsitektur Word2vec	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
Skipgram	22.23	80.84	80.00	80.00	80.00
CBOW	25.45	78.25	78.00	78.00	77.00

Pada Tabel 4.16 dijelaskan bahwa arsitektur *skipgram* memiliki akurasi lebih baik daripada arsitektur CBOW. Hal ini dikarenakan arsitektur *skipgram* dapat menghasilkan *word embedding* yang lebih baik sehingga dapat meningkatkan akurasi.

4.5.5.2 Pengujian Jumlah Epoch Multi Class Labeling

Pengujian berikutnya adalah *epoch* yaitu dengan menentukan jumlah *epoch* yang akan diuji. Dalam penelitian ini jumlah *epoch* yang diujikan yaitu 75, 100 dan 125. Pada Tabel 4.17 dijelaskan bahwa akurasi pada *epoch* 75 merupakan akurasi terbaik dengan 80.84%, tetapi saat *epoch* diubah menjadi 100 hasil akurasi mengalami penurunan menjadi 79.03%. Sedangkan pada *epoch* 125 juga mengalami penurunan dengan akurasi 78.75%.

Tabel 4.17 Pengujian jumlah epoch multi class labeling

Jumlah Epoch	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
75	11.44	80.84	80.00	80.00	80.00
100	12.02	79.03	79.00	79.00	79.00
125	16.04	78.75	78.00	77.00	77.00

4.5.5.3 Pengujian Fungsi Aktivasi Multi Class Labeling

Pengujian berikutnya adalah pengujian fungsi aktivasi. Pada penelitian ini fungsi aktivasi yang akan diujikan *tanh*, *sigmoid*, dan *relu*. Untuk parameter lain diambil dari nilai parameter yang menghasikan akurasi terbaik pada pengujian sebelumnya yaitu epoch 75. Hasil pengujian fungsi aktivasi dapat dilihat pada Tabel 4.18.

Tabel 4.18 Pengujian fungsi aktivasi multi class labeling

Fungsi Aktivasi	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
Sigmoid	11.55	80.84	80.00	80.00	80.00
Tanh	09.01	76.65	76.00	76.00	76.00
Relu	08.55	72.39	72.00	72.00	72.00

Dari pengujian yang dilakukan menunjukkan bahwa fungsi aktivasi *sigmoid* menghasilkan nilai akurasi terbaik yaitu 80.84%. Dibandingkan dengan fungsi aktivasi *tanh* dengan akurasi 76.65% dan fungsi aktivasi *relu* yang nilai akurasinya 72.39%.

4.5.5.4 Hasil Pengujian LSTM 1 Layer Multi Class Labeling

Setelah dilakukan pengujian terhadap beberapa parameter yang telah ditentukan yaitu arsitektur *word2vec*, jumlah *epoch*, dan fungsi aktivasi, maka hasil keseluruhan pengujian dapat dilihat pada Tabel 4.19

Tabel 4.19 Hasil pengujian LSTM 1 layer multi class labeling

Parameter	Nilai	Akurasi
Arsitektur *Word2vec*	*Skipgram*	80.84%
Epoch	75	
Fungsi Aktivasi	Sigmoid	

4.5.6 Pengujian LSTM 2 Layer Multi Class Labeling

4.5.6.1 Pengujian Word2Vec Multi Class Labeling

Pengujian ini diawali dengan pengujian *word2vec* yaitu arsitektur *skipgram dan Continuous bag-of-word* (CBOW). Pengujian ini dilakukan menggunakan metode LSTM 2 layer dengan parameter lain yang dipilih secara acak. Arsitektur terbaik dari model *word2vec* akan digunakan pada pengujian selanjutnya.

Tabel 4.20 Pengujian Word2Vec pada LSTM 2 Layer Multi Class Labeling

Arsitektur Word2vec	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
Skipgram	29.13	84.41	84.00	84.00	84.00
CBOW	28.44	81.07	81.00	81.00	80.00

Pada Tabel 4.20 dijelaskan bahwa arsitektur *skipgram* memiliki akurasi lebih baik daripada arsitektur CBOW. Hal ini dikarenakan arsitektur *skipgram* dapat menghasilkan *word embedding* yang lebih baik sehingga dapat meningkatkan akurasi. Jadi, arsitektur *skipgram* akan digunakan pada pengujian selanjutnya.

4.5.6.2 Pengujian Jumlah Epoch Multi Class Labeling

Jumlah *epoch* yang diujikan pada LSTM 2 *layer* ini sama dengan pengujian *epoch* pada LSTM 1 *layer* yaitu jumlah *epoch* 75, 100, dan 125.

Sedangkan nilai parameter lain yang digunakan yaitu jumlah *neuron* 75 pada setiap *layer* dan arsitektur *skipgram*.

Tabel 4.21 Pengujian jumlah epoch multi class labeling

Jumlah Epoch	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)
75	15.09	84.41	84.00	84.00	84.00
100	16.32	81.77	81.00	81.00	80.00
125	19.15	79.62	79.00	79.00	78.00

Pada Tabel 4.21 dijelaskan bahwa akurasi pada *epoch* 75 merupakan akurasi terbaik dengan 84.41%, tetapi saat *epoch* diubah menjadi 100 hasil akurasi mengalami penurunan menjadi 81.77%. Sedangkan pada *epoch* 125 juga mengalami penurunan dengan akurasi 79.62%.

4.5.6.3 Pengujian Fungsi Aktivasi Multi Class Labeling

Pengujian fungsi aktivasi ini sama dengan pegujian fungsi aktivasi pada LSTM 1 *layer Multi Class Labeling*. Fungsi aktivasi yang diujikan adalah fungsi aktivasi *tanh*, fungsi aktivasi *sigmoid* dan fungsi aktivasi *relu*. Nilai parameter yang digunakan yaitu jumlah epoch 75. Hasil pengujian dapat ditunjukkan pada Tabel 4.22.

Tabel 4.22 Pengujian fungsi aktivasi multi class labeling

Fungsi Aktivasi	Waktu (menit)	Akurasi (%)	Precision (%)	Recall (%)	f-measure (%)

Sigmoid	15.50	84.41	84.00	84.00	84.00
Tanh	14.36	81.09	80.00	80.00	80.00
Relu	13.29	80.35	80.00	79.00	79.00

Dari pengujian yang dilakukan menunjukkan bahwa fungsi aktivasi *sigmoid* menghasilkan nilai akurasi terbaik yaitu 84.41%. Dibandingkan dengan fungsi aktivasi *tanh* dengan akurasi 81.09% dan fungsi aktivasi *relu* yang nilai akurasinya 80.35%.

4.5.6.4 Hasil Pengujian LSTM 2 Layer Multi Class Labeling

Setelah dilakukan pengujian terhadap beberapa parameter yang telah ditentukan yaitu arsitektur *word2vec*, jumlah *epoch*, dan fungsi aktivasi, maka hasil keseluruhan pengujian dapat dilihat pada Tabel 4.23

Tabel 4.23 Hasil pengujian LSTM 2 Layer Multi Class Labeling

Parameter	Nilai	Akurasi
Arsitektur *Word2vec*	*Skipgram*	84.41%
Epoch	75	
Fungsi Aktivasi	Sigmoid	

4.5.7 Perbandingan Hasil Sentimen

Perbandingan hasil akurasi dari pengujian klasifikasi menggunakan Metode LSTM 1 Layer dan Metode LSTM 2 Layer serta Metode LSTM 1 *Layer Multi Class Labeling* dan Metode LSTM 2 Layer *Multi Class Labeling* diperlihatkan pada Tabel 4.24.

Tabel 4.24 Perbandingan hasil akurasi klasifikasi

Metode	**Waktu (menit)**	**Akurasi (%)**	**Precision (%)**	**Recall (%)**	**f-measure (%)**
LSTM 1 Layer	11.45	75.63	76.00	76.00	76.00
LSTM 2 Layer	14.56	77.54	77.00	77.00	77.00
LSTM 1 Layer *Multi Class Labeling*	11.55	80.84	80.00	80.00	80.00
LSTM 2 Layer *Multi Class Labeling*	15.50	84.41	84.00	84.00	84.00

BAB V

PENUTUP

5.1. Kesimpulan

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

1. Cara yang paling efektif untuk mendeteksi *Hate Speech* di media sosial adalah dengan cara membandingkan beberapa metode yang dapat digunakan dalam *Sentiment Analysis* yaitu Metode Naïve Bayes, Metode LSTM dan Metode SVM. Setelah dilakukan beberapa eksperimen menggunakan metode tersebut, maka dapat disimpulkan bahwa Metode yang paling efektif dalam mendeteksi Hate Speech adalah menggunakan Metode *Long Short-Term Memory 1 Layer* dan dibandingkan dengan Metode *Long Short-Term Memory 2 Layer*.
2. Metode *Long Short-Term Memory* 2 Layer memiliki hasil akurasi 77.54% lebih baik dibandingkan dengan Metode *Long Short-Term Memory* 1 Layer yang memiliki hasil akurasi 75.63%. Artinya menambahkan *Layer* pada arsitektur Metode LSTM dapat meningkatkan hasil akurasinya.
3. Metode *Long Short-Term Memory* 2 Layer dengan *multi class labeling* memiliki hasil akurasi 84.41% lebih baik dibandingkan dengan Metode *Long Short-Term Memory* 1 Layer dengan *multi class labeling* yang memiliki hasil akurasi 80.84%.

4. Hasil akurasi Metode *Long Short-Term Memory* 1 Layer menggunakan *muti class labeling* dan Metode *Long Short-Term Memory* 2 Layer menggunakan *Muti Class Labeling* dianggap ambigu dan kurang efektif dikarenakan terdapat beberapa *class* dalam label data yang hampir sepadan sehingga menyebabkan pelabelan data menjadi kurang efektif.
5. Preprocessing yang sudah dilakukan pada penelitian ini berlaku general dan dapat digunakan untuk berbagai bahasa, berbagai kasus dan berbagai bidang berdasarkan penelitian-penelitian sebelumnya.
6. Penggunaan Metode LSTM Multi Layer dengan fitur Word2Vec dapat
7. Tolak ukur efektifitas dapat diukur berdasarkan perbandingan efektifitas penelitian sebelumnya dan perbandingan metode sebelumnya yakni Metode Naive Bayes dengan hasil akurasi 55%, sedangkan perbandingan efektifitas menggunakan Metode LSTM 2 Layer mendapatkan hasil akurasi 77.54%.

5.2. Saran

Dalam penelitian ini masih terdapat beberapa kekurangan yang dapat disempurnakan. Pada penelitian berikutnya dapat menggunakan beberapa saran berikut ini:

1. Penelitian ini masih menggunakan data dalam jumlah terbatas, penelitian berikutnya dapat menggunakan data yang lebih besar pada kasus yang berbeda.

2. Penelitian berikutnya dapat menggunakan perhitungan parameter jumlah *neuron* dan L2 *regularization* untuk memproses data.
3. Penelitian berikutnya dapat mengkombinasikan metode *Long Short-Term Memory* dengan metode *Convulutional Neural Network* untuk klasifikasi sentimen.

DAFTAR PUSTAKA

PUSTAKA MAJALAH, JURNAL ILMIAH ATAU PROSIDING

Alshalan, R. and Al-Khalifa, H. (2020) 'A deep learning approach for automatic Hate Speech detection in the saudi twittersphere', *Applied Sciences (Switzerland)*, 10(23), pp. 1–16. doi: 10.3390/app10238614.

Biere, S. (2018) 'Hate Speech Detection Using Natural Language Processing Techniques', *Vrije Universiteit Amsterdam*, p. 30.

Buslim, N. and Iswara, R. P. (2019) 'Pengembangan Algoritma Unsupervised Learning Technique Pada Big Data Analysis di Media Sosial sebagai media promosi Online Bagi Masyarakat', *Jurnal Teknik Informatika*, 12(1), pp. 79–96. doi: 10.15408/jti.v12i1.11342.

MacAvaney, S. *et al.* (2019) 'Hate Speech detection: Challenges and solutions', *PLoS ONE*, 14(8), pp. 1–16. doi: 10.1371/journal.pone.0221152.

Mutanga, R. T., Naicker, N. and Olugbara, O. O. (2020) 'Hate Speech detection in twitter using transformer methods', *International Journal of Advanced Computer Science and Applications*, 11(9), pp. 614–620. doi: 10.14569/IJACSA.2020.0110972.

Pumsirirat, A. and Yan, L. (2018) 'Credit card fraud detection using deep learning based on auto-encoder and restricted Boltzmann machine', *International Journal of Advanced Computer Science and Applications*, 9(1), pp. 18–25. doi: 10.14569/IJACSA.2018.090103.

Ray, S. (2019) 'A Comparative Analysis and Testing of Supervised Machine Learning Algorithms', (August 2018). doi: 10.13140/RG.2.2.16803.60967.

Zhang, Z. and Luo, L. (2019) 'Hate Speech detection: A solved problem? The challenging case of long tail on Twitter', *Semantic Web*, 10(5), pp. 925–945. doi: 10.3233/SW-180338.

Zimmerman, S., Fox, C. and Kruschwitz, U. (2019) 'Improving Hate Speech detection with deep learning ensembles', *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, pp. 2546–2553.

Manaswi, N. K. (2018).'Deep Learning with Applications Using Python'. Apress.

PUSTAKA ELEKTRONIK

https://colah.github.io/posts/2015-08-Understanding-LSTMs, diakses rabu 17/02/2021 20:30.

https://jdih.kemenkeu.go.id/fulltext/2008/11TAHUN2008UUPenj.htm, diakses rabu 02/02/2022 20:38

https://id.wikipedia.org/wiki/Ucapan_kebencian, diakses rabu 02/02/2022 21:00

https://id.wikipedia.org/wiki/Status_sosial diakses pada selasa 08/03/2022 15:30