KLASIFIKASI *MULTI-LABEL* TERJEMAHAN AL-QUR'AN BAHASA INDONESIA MENGGUNAKAN MODEL *LONG SHORT-TERM MEMORY*

THESIS

Oleh :
ISMAIL AKBAR
NIM. 200605220004

PROGRAM STUDI MAGISTER INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

KLASIFIKASI *MULTI-LABEL* TERJEMAHAN AL-QUR'AN BAHASA INDONESIA MENGGUNAKAN MODEL *LONG SHORT-TERM MEMORY*

THESIS

Diajukan kepada :
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Magister Komputer (M.Kom)

Oleh :
ISMAIL AKBAR
NIM. 200605220004

PROGRAM STUDI MAGISTER INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

HALAMAN PENGAJUAN

KLASIFIKASI *MULTI-LABEL* TERJEMAHAN AL-QUR'AN BAHASA INDONESIA MENGGUNAKAN MODEL *LONG SHORT-TERM MEMORY*

THESIS

Diajukan kepada :
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Magister Komputer (M.Kom)

Oleh :
ISMAIL AKBAR
NIM. 200605220004

PROGRAM STUDI MAGISTER INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

HALAMAN PERSETUJUAN

KLASIFIKASI *MULTI-LABEL* TERJEMAHAN AL-QUR'AN BAHASA INDONESIA MENGGUNAKAN MODEL *LONG SHORT-TERM MEMORY*

THESIS

Oleh :
ISMAIL AKBAR
NIM. 200605220004

Telah diperiksa dan disetujui untuk diuji :
Tanggal : 15 Desember 2023

Pembimbing I,	Pembimbing II,
Dr. Muhammad Faisal, M.T	Dr. Totok Chamidy, M. Kom
NIP. 19740510 200501 1 007	NIP. 19691222 200604 1 001

Mengetahui,
Ketua Program Studi Magister Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PENGESAHAN

KLASIFIKASI *MULTI-LABEL* TERJEMAHAN AL-QUR'AN BAHASA INDONESIA MENGGUNAKAN MODEL *LONG SHORT-TERM MEMORY*

THESIS

Oleh :
ISMAIL AKBAR
NIM. 200605220004

Telah Dipertahankan di Depan Dewan Penguji Thesis
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Magister Komputer (M.Kom)
Tanggal: ……

Susunan Dewan Penguji			**Tanda Tangan**
Penguji Utama	:	Prof. Dr. Suhartono, M.Kom NIP 19680519 200312 1 001	( )
Ketua Penguji	:	Dr. Fresy Nugroho, M.T NIP. 19710722 201101 1 001	( )
Sekertaris Penguji	:	Dr. Totok Chamidy, M. Kom NIP. 19691222 200604 1 001	( )
Anggota Penguji	:	Dr. Muhammad Faisal, M.T NIP 19740510 200501 1 007	( )

Mengetahui,
Ketua Program Studi Magister Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini :

Nama	: Ismail Akbar
NIM	: 200605220004
Program Studi	: Magister Informatika
Fakultas	: Sains dan Teknologi

Menyatakan dengan sebenarnya bahwa Thesis yang saya tulis ini benar-banar merupakan hasil karya saya sendiri, bukan merupakan pengambilalihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan Thesis ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 15 Desember 2023
Yang membuat pernyataan,

Ismail Akbar
NIM. 200605220004

MOTTO

Manusia mempunyai kemampuan untuk merencanakan berbagai aspek kehidupan. Mulai dari menetapkan tujuan, merencanakan, hingga berusaha mencapai apa yang diinginkan. Namun, dalam hidup, manusia sering kali menghadapi berbagai tantangan dan ketidakpastian. Pada akhirnya, berbagai tantangan dan ketidakpastian menyadarkan manusia bahwa segala rencana mungkin terjadi, namun hanya Allah yang mampu menentukan takdir.

وَاللّٰهُ يَعْلَمُ وَاَنْتُمْ لَا تَعْلَمُوْنَ

"Allah mengetahui, sedangkan kamu tidak mengetahui."

HALAMAN PERSEMBAHAN

Alhamdulillahi robbil a'lamin …

Sujud syukur kusembahkan kepada-Mu ya Allah Yang Maha Besar lagi Maha Agung. Berkat takdirmu, aku bisa menjadi orang yang beriman, berilmu, percaya diri dan sabar. Semoga kesuksesan ini menjadi awal masa depan saya, menuju terwujudnya impian saya. Sholawat dan salam selalu terpanjatkan kepada Nabi Muhammad SAW, beserta keluarga, sahabat dan umatnya.

Penulis ingin mempersembahkan karya ini kepada kedua orang tuanya dengan segala keikhlasan, ini adalah hadiah kecil yang penulis kirimkan kepada kedua orang tua, tidak lupa istri dan anak saya yang selalu mendukung dan menyemangati penulis untuk menyelesaikan karya ini. Terima kasih ibu, bapak dan istri yang terus mendoakan saya. Tak lupa saya ucapkan terima kasih kepada pembimbing saya yaitu Bapak Dr. M. Faisal, M.T dan Bapak Dr. Totok Chamidy, M.Kom, terima kasih atas segala kesabaran, perhatian, waktu dan ilmu yang telah diberikan kepada penulis.

Ucapan terima kasih ini juga saya sampaikan kepada seluruh sahabat dan saudara saya yang selalu mendukung dan mendampingi saya serta semua pihak yang turut berkontribusi, yang saya rasa tidak dapat saya sebutkan satu persatu, saya ucapkan terima kasih. Karya ini tidak akan berarti apa-apa tanpa do'a dan motivasi anda.

KATA PENGANTAR

Puji syukur kepada Allah SWT yang telah melimpahkan karunia, rahmat, dan hidayah-Nya sehingga penulis dapat menyelesaikan laporan tesis yang berjudul "Klasifikasi Multi-Label Terjemahan Al-Qur'an Bahasa Indonesia Menggunakan Model *Long Short-Term Memory*". Laporan thesis ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Magister Komputer di Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Penulis menyadari bahwa selama menyelesaikan thesis ini, banyak sekali bantuan, dukungan, dan saran dari semua pihak, baik langsung maupun tidak langsung, selama proses penyusunan tesis hingga selesai. Oleh karena itu, penulis ingin mengucapkan terima kasih dan hormat yang sebesar-besarnya kepada :

1. Bapak Dr. Muhammad Faisal, M.T. selaku Dosen Pembimbing I dan Bapak Dr. Totok Chamidy, M.Kom. selaku Dosen Pembimbing II atas motivasi yang diberikan, dan juga telah banyak memberikan pengarahan dan pengalaman yang berharga.
2. Seluruh dosen Program Studi Magister Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang yang tiada lelah memberikan ilmunya kepada penulis.
3. Kedua orang tua (Bapak Ahmad Zahid dan Ibu Darti), adik (Anifatul Khasanah), Istri (Isbalaikana Larasati, S,Kom) dan Jagoan penulis (Arzan Ravindra Malik Akbar) serta keluarga besar yang telah memberikan dukungan, perhatian, nasihat, motivasi dan doa yang tiada henti kepada penulis untuk menyelesaikan thesis.
4. Seluruh rekan Mahasiswa angkatan ke-3 Program Studi Magister Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang yang telah memberikan banyak pengalaman berupa saran dan kritik yang membangun pada Thesis ini.
5. Semua pihak yang tidak dapat penulis sebut satu persatu, yang telah membantu dalam menyelesaikan thesis ini.

Harapan dan doa penulis semoga semua amal kebaikan dan jasa-jasa dari semua pihak yang telah membantu hingga terselesaikannya thesis ini diterima Allah SWT, serta mendapatkan balasan yang lebih baik dan berlipat ganda. Penulis menyadari bahwa thesis ini belum sempurna, baik dari segi materi maupun penyajiannya. Oleh karena itu, saran dan kritik yang membangun sangat diharapkan untuk menyempurnakan thesis ini. Penulis berharap bahwa thesis ini dapat memberikan hal yang bermanfaat dan menambah wawasan bagi semua pihak.

Malang, 15 Desember 2023

Ismail Akbar

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Akbar, Ismail. 2023. **Klasifikasi Multi-Label Terjemahan Al-Qur'an Bahasa Indonesia Menggunakan Model *Long Short-Term Memory***. Thesis. Program Studi Magister Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim. Pembimbing : (I) Dr. Muhammad Faisal, M.T. (II) Dr. Totok Chamidy, M. Kom.

Kata kunci : Al-Qur'an, Klasifikasi Ayat, LSTM, *Word Embedding*, Terjemahan.

Salah satu ibadah yang paling dianjurkan dalam Islam adalah mempelajari Al-Qur'an. Ketika mempelajari Al-Quran, perlu dipahami bagaimana mengklasifikasikan ayat-ayatnya. Pengelompokkan dapat digunakan untuk mempelajari lebih lanjut suatu ayat dan mempermudah pencarian ayat-ayat yang terkait. Hal yang menarik dari penggolongan ayat-ayat dalam Al-Quran adalah setiap ayat dapat masuk dalam satu atau lebih kategori yang berbeda. Penelitian ini mengeksplorasi model *deep learning* untuk mengklasifikasikan isi terjemahan Al-Qur'an berbahasa Indonesia menjadi kategori-kategori inti seperti Tauhid, Ibadah, Akhlaq, dan Sejarah. Dengan memanfaatkan algoritma *Long Short Term Memory* (LSTM) dan Bi-LSTM serta *word embedding Word2Vec* dan *FastText*, penelitian ini membandingkan teknik-teknik tersebut untuk mengoptimalkan klasifikasi teks Al-Qur'an. Menggunakan dataset dari terjemahan Al-Qur'an juz 4 - 8, penelitian ini mengevaluasi pengaruh *hyperparameter* terhadap kinerja algoritma. Hasil menunjukkan bahwa kombinasi Bi-LSTM + *FastText* + *Hyperparameter* memberikan hasil terbaik dengan akurasi 71,63%, *precision* 64,06%, *recall* 63,60%, dan *hamming loss* 36,17%. Dengan penyetelan parameter terbaik, penelitian ini menawarkan solusi yang berpotensi untuk mengklasifikasikan ayat-ayat Al-Qur'an dalam bahasa Indonesia dengan tingkat akurasi yang baik.

ABSTRACT

Akbar, Ismail. 2023. **Multi-Label Classification of Indonesian Qur'an Translations Using the Long Short-Term Memory Model**. Thesis. Master of Informatics Study Program, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University. Supervisor: (I) Dr. Muhammad Faisal, M.T. (II) Dr. Totok Chamidy, M. Kom.

One of the most recommended acts of worship in Islam is studying the Qur'an. When studying the Quran, it is necessary to understand how to classify its verses. Classifying can be used to further study a verse and make it easier to find related verses. The interesting thing about classifying verses in the Quran is that each verse can fall into one or more different categories. This research explores a deep learning model to classify the content of Indonesian Qur'an translations into core categories such as Tawheed, Worship, Akhlaq, and History. By utilizing Long Short Term Memory (LSTM) and Bi-LSTM algorithms as well as Word2Vec and FastText word embedding, this study compares these techniques to optimize the classification of Qur'anic text. Using a dataset of Qur'anic translations of juz 4 - 8, this study evaluates the effect of hyperparameters on algorithm performance. The results show that the combination of Bi-LSTM + FastText + Hyperparameter gives the best results with 71.63% accuracy, 64.06% precision, 63.60% recall, and 36.17% hamming loss. With the best parameter tuning, this research offers a potential solution for classifying Qur'anic verses in Indonesian with a good level of accuracy.

Keywords: Al-Qur'an, Verse Classification, LSTM, Word Embedding, Translation.

ملخص

أكبر، إسماعيل. 2023. تصنيف متعدد العلامات لترجمات القرآن الإندونيسية باستخدام نموذج الذاكرة الطويلة وقصيرة المدى. أُطرُوحَة. برنامج الماجستير في المعلوماتية كلية العلوم والتكنولوجيا جامعة مولانا مالك إبراهيم الإسلامية الحكومية. المشرف: (I) د. محمد فيصل، طن متري (II) د. توتوك شميدي، م.كوم.

الكلمات المفتاحية: القرآن، تصنيف الآيات، LSTM، تضمين الكلمات، الترجمة.

من أهم العبادات في الإسلام دراسة القرآن. عند دراسة القرآن لا بد من فهم كيفية تصنيف آياته. يمكن استخدام التجميع لمعرفة المزيد عن الآية وتسهيل العثور على الآيات ذات الصلة. والشيء المثير للاهتمام في تصنيف الآيات في القرآن هو أن كل آية يمكن أن تندرج تحت فئة واحدة أو أكثر. يستكشف هذا البحث نموذج التعلم العميق لتصنيف محتويات ترجمات القرآن الإندونيسي إلى فئات أساسية مثل التوحيد والعبادة والأخلاق والتاريخ. من خلال استخدام خوارزميات الذاكرة طويلة المدى (LSTM) وBi-LSTM بالإضافة إلى تضمين الكلمات Word2Vec وFastText، يقارن هذا البحث هذه التقنيات لتحسين تصنيف نص القرآن الكريم. باستخدام مجموعة بيانات من ترجمات الأجزاء 4 - 8 من القرآن الكريم، يقوم هذا البحث بتقييم تأثير المعلمات الفائقة على أداء الخوارزمية. أظهرت النتائج أن الجمع بين Bi-LSTM + FastText + Hyperparameter يوفر أفضل النتائج بدقة تبلغ 71.63%، ودقة 64.06%، واستدعاء 63.60%، وخسارة تشويش بنسبة 36.17%. ومن خلال تحديد أفضل المعايير، يقدم هذا البحث حلاً لديه القدرة على تصنيف آيات القرآن باللغة الإندونيسية بمستوى جيد من الدقة.

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dengan jumlah pemeluk Islam yang signifikan di Indonesia, Al-Qur’an sendiri sebagai kitab suci di antara umat Islam, karena berfungsi sebagai panduan Ilahi atau sebagai pedoman hidup yang menyampaikan ajaran dan perintah Allah Subhanahu Wa Ta'ala kepada umat manusia. Melalui perantara Malaikat Jibril Al-Qur’an ini diwahyukan kepada Nabi Muhammad Shalallaahu Alaihi Wassalaam dalam bentuk teks bahasa Arab.

Sangat penting bagi semua umat Islam dan umat manusia pada umumnya untuk memberikan penekanan yang signifikan pada pembelajaran dan pemahaman isi Al-Qur’an. Salah satu ibadah yang paling dianjurkan dalam Islam adalah belajar dan memahami Al-Qur’an. Lebih jauh lagi, mendalami ajaran Al-Qur'an dapat memungkinkan individu untuk memahami prinsip-prinsip Islam secara lebih komprehensif, memungkinkan mereka untuk berpegang teguh pada ajaran tersebut secara akurat dan tegas. Hal ini Allah Subhanahu Wa Ta'ala berfirman pada surah Al – Isra ayat 9 yang berbunyi :

إِنَّ هَٰذَا ٱلْقُرْءَانَ يَهْدِى لِلَّتِى هِىَ أَقْوَمُ وَيُبَشِّرُ ٱلْمُؤْمِنِينَ ٱلَّذِينَ يَعْمَلُونَ ٱلصَّٰلِحَٰتِ أَنَّ لَهُمْ أَجْرًا كَبِيرًا

Artinya : *Sesungguhnya Al-Qur'an ini memberikan petunjuk kepada (jalan) yang lebih lurus dan memberi khabar gembira kepada orang-orang Mu'min yang mengerjakan amal saleh bahwa bagi mereka ada pahala yang besar* (QS. Al-Isra : 9).

Menurut penafsiran ahli tafsir Syeikh Abu Bakar Jabir Al-Jazairi, seorang ulama Madinah yang cukup terkenal. Makna kata (لِلَّتِي هِيَ أَقْوَمُ) *lillatii hiya aqwam*, menuju jalan yang paling lurus dan benar. Dan (أَنَّ لَهُمْ أَجْرًا كَبِيرًا) *anna lahum ajran kabiiraa* yang artinya surga Darussalam. Maka, makna dari QS. Al-Isra ayat 9 menurut beliau adalah Allah ta'ala mengabarkan bahwa Al-Qur'anul Karim yang diturunkan kepada rasul-Nya Muhammad shallallahu 'alaihi wa sallam, yang telah Dia perjalankan dari Masjidil Haram menuju Masjidil Aqsha, untuk memberi petunjuk dari apa yang terjadi di dalamnya berupa: bukti (hujjah), syariat, dan nasehat menuju jalan yang lebih lurus dari jalan-jalan yang ada. Jalan itu adalah Islam; jalan kebahagiaan dan kesempurnaan di dunia dan akhirat. "Dan memberikan kabar gembira untuk orang-orang yang beriman lagi beramal saleh..." Al-Qur'an memberikan kabar gembira kepada orang-orang yang beriman kepada Allah, rasul-Nya, pertemuan dengan-Nya, janji, dan ancaman-Nya kemudian beramal saleh, baik wajib maupun sunnah, juga meninggalkan dosa-dosa besar dan maksiat, bahwa untuk mereka pahala yang besar yaitu surga.

Oleh karena itu, saat membaca Al-Qur'an, penting untuk berupaya memahami semua ayatnya. Melalui perenungan, pemahaman, dan sering membaca Al-Qur'an, seseorang dapat mengalami kepuasan luar biasa yang

diberikannya. Menurut sebuah hadits yang diriwayatkan oleh Imam al Bukhari, Nabi Shalallaahu Alaihi Wassalaam menyatakan:

خَيْرُكُمْ مَنْ تَعَلَّمَ الْقُرْآنَ وَعَلَّمَهُ

Artinya : *Sebaik-baik kalian adalah orang yang mempelajari al Qur`an dan mengajarkanya* (Diriwayatkan Imam al Bukhari, no. 5027; Fat-hul Bari, 8/692)

Seiring berjalannya waktu, ada keinginan yang tumbuh bagi setiap individu umat islam untuk memperoleh pengetahuan tentang Al-Qur'an dalam bahasa yang dapat dipahami oleh mereka. Salah satunya bahasa Indonesia yang merupakan bahasa yang banyak digunakan untuk menerjemahkan Al-Qur'an. Namun dalam, menerjemahkan Al-Qur'an ke dalam bahasa Indonesia menimbulkan kesulitan karena adanya beberapa konotasi untuk banyak ayat, yang menyebabkan interpretasi yang ambigu. Alhasil, sistem otomatis akan bermanfaat untuk memudahkan penerjemahan Al-Qur'an ke dalam Bahasa Indonesia.

Ketika mempelajari Al-Qur'an, perlu dipahami klasifikasi ayat-ayat yang termasuk di dalamnya. Sangat penting bagi seseorang untuk memahami klasifikasi Al-Qur'an. Dimana klasifikasi dapat digunakan untuk mempelajari lebih lanjut tentang sebuah ayat dan membantu menemukan ayat-ayat terkait dengan lebih mudah. Menurut (Hasyim, 2020) dalam bukunya yang berjudul Akidah Akhlaq yang diterbitkan oleh Kementerian Agama Republik Indonesia, ajaran yang terkandung dalam Al-Qur'an sebagian besar meliputi, Tauhid, Ibadah, Muamalah, Akhlak, Tarikh atau Sejarah dan syariat. Dengan mengkategorikan ayat – ayat Al-Qur'an dapat memudahkan pencarian ayat sesuai yang diinginkan.

Dengan berkembangnya ilmu pengetahuan dan teknologi, sebagian besar masyarakat menggunakan teknologi sebagai sarana pembelajaran dan pencarian informasi. Seperti halnya dalam mencari makna dari terjemahan Al-Qur’an. Dalam penyajian Al-Qur'an telah diklasifikasikan berdasarkan pembahasan ayat-ayat ke dalam Al-Qur'an dalam bentuk fisik atau digital. Yang menarik dari klasifikasi ayat-ayat Al-Qur'an adalah bahwa setiap ayat dapat termasuk dalam satu atau lebih kelas yang berbeda. Fakta ini menunjukkan bahwa klasifikasi ayat-ayat Al-Qur'an berbeda dengan klasifikasi keseluruhan, di mana hanya semua sumber atau dokumen yang diklasifikasikan. Kasus klasifikasi semacam itu dapat disebut sebagai klasifikasi *multi-label*. Ada beberapa teknik klasifikasi menggunakan model *machine learning* dan model *deep learning*.

Metode yang lebih efektif ditunjukkan melalui metode yang melibatkan pembelajaran. Metode ini menggunakan beragam teknik pembelajaran mesin untuk klasifikasi teks. Metode pembelajaran yang diawasi (*supervised learning*) telah digunakan secara luas dan menunjukkan tingkat deteksi yang relatif unggul dibandingkan dengan pendekatan pembelajaran yang tidak diawasi (*unsupervised learning*) (Hanafi et al., 2020). Pada penelitian yang dilakukan oleh Afrian Hanafi, dkk, melakukan klasifikasi *multi-label* pada Hadis Bukhari terjemahan Bahasa Indonesia yang menggunakan model *K-Nearest Neighbor* (KNN). Dalam penelitian tersebut model yang diujikan memiliki kelemahan dalam memproses data dimensi vektor yang tinggi sehingga menyebabkan waktu komputasi menjadi lebih tinggi dan efisiensi dari klasifikasi teks sangat rendah. Dengan model yang sama dalam penelitian Timami Hertza Putrisanni, Dkk, melakukan pengujian *K*-

Nearest Neighbor (KNN) sebagai model untuk klasifikasi terjemahan Al-Qur'an Bahasa Inggris. Dengan meningkatkan seleksi fitur dengan *Information Gain* dapat mengurangi *noise* yang disebabkan oleh fitur-fitur yang tidak relevan untuk mendapatkan hasil akurasi yang lebih tinggi. Sehingga hasil nilai akurasi didapatkan sebesar 64.10%, nilai presisi 65% dan nilai recall 62.68%.

Hal yang sama dilakukan oleh Xiaoyu Luo dalam menerapkan teknik klasifikasi dokumen berbahasa inggris melakukan perbandingan model *machine learning* (*Support Vector Machine*, *Naïve Bayes*, *Logistic Regression*) dengan tiga kategori, yakni *women*, *sport*, dan literatur. Dari hasil eksperimen tersebut diperoleh model *Support Vector Machine* lebih unggul dari model lainnya, dengan tingkat akurasi lebih dari 90% (Luo, 2021).

Lebih baru lagi, model *deep learning* telah diadopsi untuk klasifikasi teks. Dengan peningkatan model *deep learning* ini dapat mendukung dengan tingkat akurasi dan efisiensi yang tinggi pada proses klasifikasi teks, Salah satu strategi untuk meningkatkan kinerja dari *deep learning* yakni mengoptimalkan teknik *Word Embedding*. Dalam penelitian Youssef Taher, Dkk, menggunakan *FastText* dipilih untuk *Word Embedding* dalam penelitian tersebut, hasilnya proses klasifikasi teks meningkat secara signifikan dengan tingkat akurasi 98% (Taher et al., 2022). Pada kasus lain, model *deel learning* diujikan untuk mengklasfikasi teks dengan menerapkan model *hybrid Convolutional Recurrent Neural Network* (CRNN) yang dilakukan oleh Muhammad Yuslan Abu Bakar. Dalam penelitian tersebut mengusulkan model *hybrid* tersebut dengan memilih *word2Vec* sebagai

proses *Word Embedding*, dengan menghasilkan nilai akurasi sebesar 80.79% (Yuslan & Bakar, 2021).

Penelitian lain yang mengimplementasikan algoritma *Long Short Term Memory* (LSTM) pada model *deep learning* dilakukan oleh (Kholifatullah & Prihanto, 2023). Pada penelitian tersebut digunakan Bahasa Indonesia sebagai Bahasa dasar teks klasifikasi. Algoritma *Long Short Term Memory* (LSTM) diterapkan untuk klasifikasi *hate speech*, yang dimana data diambil dari *website* Kaggle. Hasil optimal dicapai oleh LSTM dengan 256 neuron LSTM. Setelah 10 *epoch*, presisi yang dicapai pada data latih mencapai 86,23%, sedangkan data validasi memperoleh tingkat akurasi yang lebih tinggi yaitu 87,10%. Sedangkan pada varian *Long Short Term Memory* (LSTM) lainnya, dilakukan oleh (Af'idah et al., 2022) dengan menggunakan *Bidirectional Long Short Term Memory* (Bi-LSTM) untuk sentiment analisis ulasan destinasi wisata bali dari data tripadvisor.com. Akurasi terbaik dihasilkan oleh kombisasi *Word2Vec* terdiri dari CBOW, *Hierarchical Softmax*, dimensi 200, Bi-LSTM dengan *dropout* sebesar 0,5 dan *learning rate* sebesar 0,0001. Kombinasi tersebut menghasilkan akurasi tertinggi dari keseluruhan 108 kombinasi yaitu sebesar 96,86%, *precission* sebesar 96,53%, *Recall* sebesar 96,31%, *F1 Measure* sebesar 96,41%.

Berdasarkan latar belakang yang telah dipaparkan, penelitian ini bertujuan untuk mengklasifikasikan terjemahan Al-Qur'an Berbahasa Indonesia dalam 4 kelompok, yaitu Tauhid, Ibadah, Akhlaq dan Tarikh atau Sejarah. Pada penelitian ini juga membandingkan teknik *Word Embedding* antara *word2Vec* dan *FastText*. Sehingga pada penelitian ini mampu mendapatkan proses teknik *Word Embedding*

terbaik, guna meningkatkan tingkat akurasi dari varian *Long Short Term Memory* (LSTM). Pada pelatihan model dengan varian LSTM ini, diperlukan parameter - parameter yang terbaik, sehingga pada penelitian ini menggunakan *hyperparameter-tuning* untuk penetapan sebuah parameter. Hasil model ini diharapkan mampu melakukan klasifikasi teks terjemahan Al-Qur'an Berbahasa Indonesia dengan optimal dan juga dapat dimanfaatkan sebagai pengembangan model yang dapat digunakan untuk menyelesaikan tugas pemahaman Bahasa alami lainnya.

1.2. Pernyataan Masalah

Berdasarkan paparan latar belakang penelitian yang telah dijelaskan diatas, maka permasalahan penelitian ini adalah :

1. Bagaimana performa dari varian algortima *Long Short Term Memory* dalam klasifikasi teks terjemahan Al-Qur'an ?
2. Bagaimana perbandingan kinerja teknik *Word Embedding* antara *Word2Vec* dan *FastText* untuk mengoptimalkan varian algoritma *Long Short Term Memory* dalam klasifikasi teks terjemahan Al-Qur'an?
3. Bagaimana pengaruh *hyperparameter* terhadap kinerja varian algoritma *Long Short Term Memory* (LSTM) dalam tugas klasifikasi teks terjemahan Al-Qur'an?

1.3. Tujuan Penelitian

Mengacu pada pernyataan masalah yang telah di sampaikan, maka tujuan dari penelitian ini adalah :

1. Mendapatkan hasil yang optimal dari varian algortima *Long Short Term Memory* dalam klasifikasi teks terjemahan Al-Qur'an.
2. Mengetahui perbandingan teknik *Word Embedding* antara *Word2Vec* dan *FastText* untuk mengoptimalkan varian algoritma *Long Short Term Memory* dalam klasifikasi teks terjemahan Al-Qur'an.
3. Mengetahui pengaruh *hyperparameter-tuning* terhadap kinerja varian algoritma *Long Short Term Memory* (LSTM) dalam tugas klasifikasi teks terjemahan Al-Qur'an.

1.4. Batasan Masalah

Agar penelitian ini terarah dan permasalahan yang dihadapi pada penelitian ini tidak terlalu luas serta sesuai dengan tujuan penulisan, maka ditetapkan batasan terhadap masalah yang sedang diteliti. Adapun batasan masalah dalam penelitian ini :

1. Data yang digunakan dari terjemahan Al-Qur'an dari juz 4 - 8 berbahasa Indonesia diantaranya, surah An-Nisa', surah Al-Maidah dan surah Al-An'am yang diambil pada data terjemahan Al-Qur'an Kementerian Agama Republik Indonesia pada tahun 2022.
2. Sistem klasifikasi *multi-label* terjemahan Al-Qur'an dikategorikan menjadi 4 label, yakni Tauhid, Ibadah, Akhlaq dan Sejarah.
3. Varian algoritma yang digunakan *Long Short Term Memory* (LSTM) dan *Bidirectional* LSTM (Bi-LSTM).
4. *Word Embedding* yang digunakan *word2Vec* dan *FastText* .

5. Parameter yang disetel (*tuning*) menggunakan *hyperparameter* yaitu *learning rate, batch size, dan* jumlah *epoch*.

1.5. Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat bagi beberapa pihak ataupun ahli terkait, diantaranya :

1. Tenaga pengajar Al-Qur’an (guru ngaji, guru Pendidikan Agama Islam), untuk memudahkan kelompok topik pembahasan pada setiap ayat Al-Qur’an.
2. Peneliti bidang pemrosesan bahasa alami, sebagai rujukan dalam penelitian selanjutnya.

BAB II

STUDI PUSTAKA

2.1. Penelitian Terkait

Penelitian terkait mengenai pembahasan klasfikasi teks *multi-label* terhadap bentuk teks dokumen, teks berita, ataupun teks cuitan telah banyak dilakukan pada penelitian sebelumnya.

Pada penelitian pertama yang dilakukan oleh (Antariksa et al., 2019), klasifikasi teks digunakan untuk klasifikasi ujaran kebencian pada cuitan di media sosial Twitter. Hal yang membuat klasfikasi ujaran kebencian sangat sulit, karena tidak ada standar yang baku. Ujaran kebencian sendiri didefinisikan untuk meremehkan orang, kelompok, atau golongan tertentu. Maka dari itu untuk menghentikan ujaran kebencian dalat dengan melakukan penyaringan cuitan di media sosial dengan pembejaran mesin. Karena melalu pembelajaran mesin, sistem akan lebih mudah menganalisis cuitan tersebut mengandung unsur kebencian atau bukan ujaran kebencian. Dalam penelitian ini membandingkan *feature extraction* (TF-IDF, *N-Gram*, dan *Word2Vec*) untuk memperoleh model yang lebih baik dengan mengujikan beberapa model seperti *Naïve Bayes*, *Support Vector Machine* dan *Logistic Regression*. Yang dimana hasil didapatkan bahwa *Logistik Regression + N-Gram* lebih unggul dari pada model lainnya dengan akurasi sebesar 98%.

Dalam penelitian lain yang dilakukan (Lim et al., 2019), mengujikan *feature extraction* (*Word2Vec* dan *FastText*) saat pembentukan vektor kata dan kalimat

untuk menguji performa model dari *deep learning*. Tujuan penelitian yang dilakukan dapat menemukan pendekatan dan parameter klasifikasi yang efektif. Dalam artian mencari kombinasi metode dan pengaturan *FastText* dan *deep learning* terhadap klasifikasi suatu judul post kesehatan di Facebook. Model deep learning yang digunakan yaitu *Convolutional Neural Network* (CNN), *Long Short-Term Memory* dan *bidirectional- Long Short-Term Memory* (Bi-LSTM). Hasil yang di dapatkan dari penelitian tersebut menyatakan performa *FastText* dapa membantu meningkatkan akurasi dari CNN dan Bi-LSTM dalam memilih algoritma klasifikasi berbahasa Indonesia.

Hal yang sama dilakukan oleh (Hana et al., 2020) dalam melakukan klasfikasi teks Bahasa Indonesia, pada kasus ini menggunakan dataset dari media sosial Twitter dalam mengklasifikasikan ujaran kebencian. Metode yang digunakan tidak hanya berbasis *machine learning* yakni *Support Vector Machine* (SVM), akan tetapi dibandingkan dengan model *deep learning*, CNN dan DistilBERT. Parameter yang digunakan dalam klasifikasi teks *multi-label* ini ditentukan dari *hyperparameter* agar mendapatkan parameter terbaik. Hasil yang didapatkan berdasarkan jumlah dataset 5277 *tweets*, model SVM dengan teknik *multi-label Classifier Chains* tanpa proses *stemming*, *stopward removal* dan *translation* lebih unggul dengan akurasi terbaik 74,88%.

Selain penelitian menggunakan Bahasa Indonesia, ada juga yang menggunakan Bahasa Inggris, hal ini dilakukan oleh (Shah et al., 2020). Pada penelitian ini mengklasifikasikan teks berita bersumber dari BBC dengan mengimplementasikan metode klasifikasi *Logistic Regression, Random Forest*

dan *K-Nearest Neighbour*. Dari ketiga metode tersebut nantinya akan dipilih metode terbaik untuk klasifikasi teks berbahasa Inggris. TF-IDF digunakan pada saat proses vectorizer atau mengubah kalimat atau teks menjadi array atau vektor. Yang dihasilkan dalam penelitian ini, model *machine learning* dengan algoritma *Logistic Regression* lebih unggul dibandingkan *Random Forest* dan *K-Nearest Neighbour*. Akurasi tertinggi yang didapatkan *Logistic Regression* dengan fitur TF-IDF sebesar 97%, *Random Forest* 93% dan *K-Nearest Neighbour* 92%.

Pada penelitian lain yang mengimplementasikan klasifikasi *multi-label* terjemahan quran dengan *machine learning*, dilakukan oleh (Abdullahi et al., 2021). Data terjemahan yang digunakan Bahasa Inggris. Adeleke Abdullahi, dkk ini mengujikan *multi-label classification* pada 4 algoritma klasifikasi terbaik dari model *machine learning*, yaitu *Support Vector Machine* (SVM), *Naïve Bayes* (NB), *K-Nearest Neighbors* (K-NN) dan J48. Kemudian hasil klasifikasi tersebut divalidasi dengan enam *conventional performance metrics*, diantaranya : *hamming loss*, *accuracy*, *one error*, *micro-F1, macro-F1,* dan *avg. precision*. Berdasarkan hasil validasi, disimpulkan bahwa algoritma *Support Vector Machine* (SVM) sangat efisien dengan *dataset* yang relatif besar dengan mencapai akuras 70%.

Selain diimplementasikan menggunakan model *machine learning*, klasifikasi dapat juga diterapkan dengan model *deep learning* dalam bentuk dokumen. Klasifikasi dokumen otomatis dapat didefinisikan sebagai penugasan berbasis konten dari satu atau lebih kategori yang telah ditentukan. Pada penelitian (Almuzaini & Azmi, 2020) menyajikan tujuh algoritma berbasis *deep*

learning untuk mengklasifikasikan dokumen Bahasa Arab. Algoritma yang digunakan *Convolutional Neural Network* (CNN), CNN-LSTM (*Long Short-Term Memory*), CNN-GRU (*Gated Recurrent Units*), BiLSTM (*Bidirectional* LSTM), BiGRU, Att-LSTM (*Attention-based* LSTM), and Att-GRU. Peneliti ini menerapkan teknik penyisipan kata atau *word embedding word2vec*. Selain itu juga, peneliti juga mempelajari bagaimana klasifikasi dipengaruhi oleh teknik *stemming*, agar menghasilkan hasil yang signifikan. Yang dimana hasil penggunaan *stemming* dan *word embedding* secara efektif dapat meningkatkan kinerja klasifikasi teks berbasis *deep learning*.

Hal yang sama dilakukan pada penelitian (Taradhita & Putra, 2021), melakukan pengujian klasifikasi teks berbasis *deep learning* dengan algoritma *Convolutional Neural Network* (CNN). Pada penelitian ini menggunakan Bahasa Indonesia dengan kasus ujaran kebencian pada media sosial twitter. TF-IDF dipilih untuk pembobotan pada *feature extraction*. Dari hasil penelitian yang didapatkan, akurasi pelatihan dan akurasi validasi memperoleh nilai optimal sebesar 90,85% dan 88,34 pada 45 *epoch*. Sedangkan akurasi pada tahapan pengujian didapatkan 82,5%. Dari penelitian tersebut disimpulkan bahwa, nilai keakuratan klasifikasi dipengaruhi beberapa faktor, yaitu frekuensi kata-kata yang menghina, kesamaan kata pada *dataset testing* dan *training* dan konotasi kata pada *dataset*.

Klasifikasi teks dengan algoritma dengan *Convolutional Neural Network* (CNN) juga dilakukan oleh (Alsaleh & Larabi-Marie-Sainte, 2021), akan tetapi bahasa yang digunakan Bahasa Arab. Data yang diambil dari sebuah situs berita

digital Al-Riyadh dan Saudi Press diklasifikasikan dalam 6 kelas, yaitu Ekonomi, Politik, Sosial, Olahraga, Berita Umum dan Hiburan. Total data yang diperoleh sebanyak 45.936. Dalam kasus ini, peneliti mencoba mengoptimalkan parameter-parameter dari Algoritma *Convolutional Neural Network* (CNN) menggunakan *Genetic algorithm* (GA). Algoritma Genetika dipilih karena salah satu algoritma optimasi yang paling handal. Operator *Genetic algorithm* (GA) digunakan pada bagian V-B1 dan diatur ke *Tournament Selection*, kemudian probalitas *crossover two-points* 0.65 dan mutasi dengan *rate* 0.05. Hasil penelitian menunjukkan bahwa akurasi klasifikasi mencapai nilai akurasi GA-CNN 88% dan *Convolutional Neural Network* (CNN) 84%, artinya dengan menggunakan *Genetic algorithm* (GA) akurasi meningkat sebesar 4-5% dibandingkan CNN tanpa menggunakan algoritma genetika.

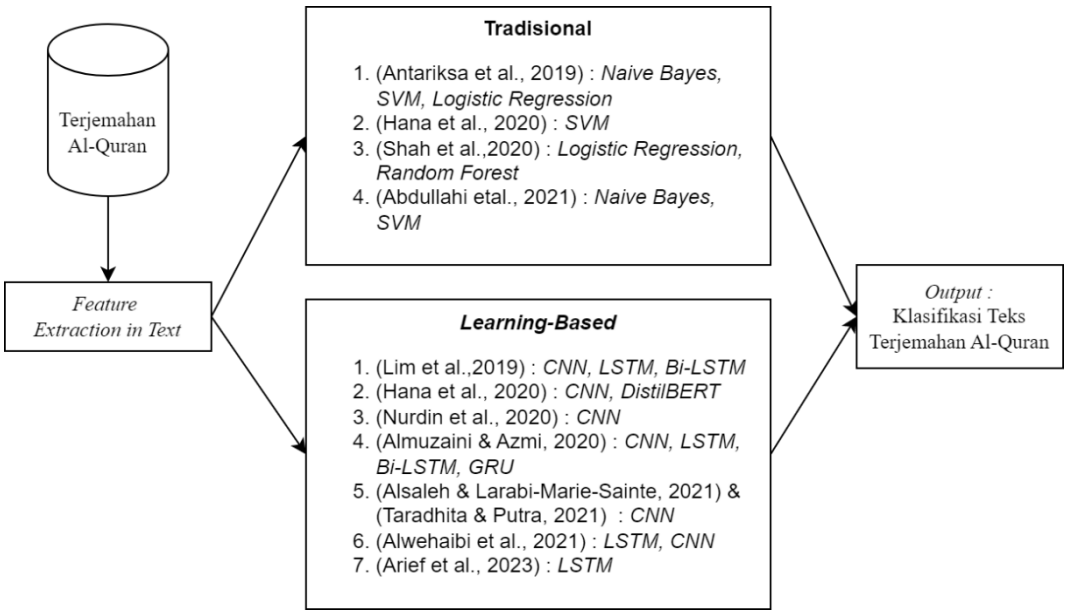
Tidak hanya berfokus pada pengujian model klasifikasi, dalam penelitian lain melakukan pengujian dengan perbandingan kinerja dari *word embedding*, penelitian ini dilakukan oleh (Nurdin et al., 2020). Penelitian tersebut berfokus pada membandingkan kinerja *word embedding* seperti *Word2Vec*, *GloVe* dan *FastText*. Hasil dari *word embedding* tersebut di klasifikasikan dengan algoritma *Convolutional Neural Network* (CNN). Menurut Arliyanti Nurdin, dkk, algoritma *Convolutional Neural Network* (CNN) dipilih karena algoritma tersebut terbukti handal dalam sejumlah permasalahan klasifikasi dalam bidang *Natural Language Processing*, *Deep Learning*. Data yang diambil oleh peneliti berupa *newsgroup* yang terdiri dari sekitar 18.846 artikel yang kemudian dibagi menjadi data latih sebesar 11.314 dan 7.532 data uji. Hasil yang didapatkan dari peneliti

menunjukkan kinerja terbaik *word embedding FastText* lebih unggul dari *word2vec* dan *GloVe*. Dikarenakan *word2vec* dan *GloVe* tidak mampu mempresentasikan vektor dari kata yang tidak ada dalam korpus. Namun, perbedaan kinerja tidak begitu signifikan. Artinya ketiga *word embedding* memiliki kinerja yang kompetitif.

Pengujian teknik *word embedding* juga dilakukan oleh (Alwehaibi et al., 2021). Data yang diambil dari media sosial twitter mempunyai teks berbahasa Arab. Peneliti mengusulkan dalam membandingkan teknik *word embedding* untuk mengoptimalkan teknik klasifikasi. Algoritma yang digunakan pada saat pengujian terdapat 3 model, yaitu *Convolutional Neural Network* (CNN), *Long Short-Term Memory* (LSTM) dan CNN-LSTM. Pada saat pengujian model, peneliti ini menggunakan metode *hyperparameter tuning* untuk mengoptimalkan kinerja dari tugas *deep learning*. Hasil dari percobaan menunjukkan bahwa teknik *word embedding FastText* menghasilkan performa yang lebih baik dibandingkan dengan teknik *embedding* lainnya untuk tugas analisis sentimen pada teks pendek bahasa Arab, yang dimana telah dilakukan penyetelan *hyperparameter*. Model CNN-LSTM menjadi model terbaik dibandingkan *Convolutional Neural Network* (CNN) dan *Long Short-Term Memory* (LSTM) dengan mendapatkan akurasi 96.7%.

2.2. Kerangka Teori

Kerangka teori dalam penelitian ini didasarkan pada penelitian sebelumnya atau studi yang relevan yang dikutip sebagai titik acuan. Berikut kerangka teori dalam penelitian ini dapat dilihat pada Gambar 2.1.



Gambar 2.1 Kerangka Teori

Berdasarkan kerangka teori diatas, dataset terjemahan Al-Qur'an akan menjalani proses *feature extraction* sebagai langkah awal untuk mengungkap perspektif baru dan elemen data penting. Langkah selanjutnya berdasarkan penelitian sebelumnya, pengujian melibatkan berbagai teknik atau metode dalam menangani klasifikasi teks. Adapun penelitian yang menjadi acuan dalam penelitian ini dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu.

No	Peneliti	Kasus	Metode	Perbedaan	Persamaan	Rencana Penelitian
1	(Antariksa et al., 2019)	Klasifikasi ujaran kebencian Bahasa Indonesia dengan *Feature Extraction* (TF-IDF, *N-Gram*, dan *Word2Vec*)	*Naïve Bayes*, SVM, dan *Logistic Regression*	*Feature Extraction* menggunakan *TF-IDF, N-Gram* dan *word2Vec*.	Menguji *Feature Extraction* yang terbaik untuk meningkatkan performa model.	Membandingkan *Feature Extraction word embedding word2Vec* dengan *FastText*.
2	(Lim et al., 2019)	Performa model dalam klasifikasi *stance* suatu judul post kesehatan di Facebook terhadap dengan *FastText Embedding*	CNN, LSTM, Bi-LSTM	Menguji *FastText embedding* untuk meningkatkan performa model dalam klasifikasi *multi-class*, agar mendapat model terbaik.	Bahasa yang digunakan Bahasa Indonesia .	Meningkatkan kinerja LSTM dengan *word embedding* (*Word2Vec* dan *FastText*) untuk klasifikasi *multi-label*.
3	(Hana et al., 2020)	Klasifikasi *multi-label* pada *tweet* ujaran kebencian	SVM, CNN, DistilBERT	Membandingkan 3 model untuk klasifikasi *multi-label*.	Klasifikasi teks Bahasa Indonesia.	Menggunakan Model berbasis *deep learning* untuk klasfikasi teks dengan 4 label.
4	(Nurdin et al., 2020)	Membandingkan *Word Embedding* untuk klasifikasi artikel berita yang di ujikan dengan model CNN	CNN	Klasifikasi artikel dengan model CNN untuk Bahasa Inggris.	Membandingkan *word embedding* untuk klasifikasi teks.	Mengukur kinerja *Word Embedding* dengan model LSTM yang berbasis RNN.

No	Peneliti	Kasus	Metode	Perbedaan	Persamaan	Rencana Penelitian
5	(Shah et al., 2020)	Membandingkan kinerja *Logistic Regression, Random Forest* dan KNN untuk klasifikasi teks berbahasa Inggris	*Logistic Regression*, *Random Forest*, KNN	Mencari model *machine learning* terbaik untuk klasifikasi teks Inggris dengan *Feature Extraction* menggunakan *TF-IDF*.	Mengimplementasikan model untuk klasifikasi *multi-label*.	Pengujian model *deep learning* LSTM untuk klasifikasi teks Bahasa Indonesia.
6	(Almuzaini & Azmi, 2020)	Mempelajari bagaimana klasifikasi dipengaruhi oleh strategi *stemming* dan penyisipan kata	CNN, LSTM, CNN-LSTM, CNN-GRU, BiLSTM, BiGRU, Att-LSTM	Klasifikasi dokumen Bahasa Arab.	Menguji *word2vec* dalam meningkatkan model *deep learning*.	Pengujian teknik *Word Embedding* dengan model *Skip-gram* untuk kinerja model LSTM berbahasa Indonesia.
7	(Abdullahi et al., 2021)	Klasifikasi *multi-label* Terjemahan Al-Qur'an bahasa inggris dengan 3 Kategori, Iman, Ibadah dan Akhlaq	*Naïve Bayes*, SVM, k-NN, J48	Klasifikasi terjemahan Al-Qur'an menggunakan Bahasa Inggris	Klasifikasi *multi-label* Terjemahan Al-Qur'an	Menggunakan terjemahan Bahasa Indonesia dengan model *deep learning*.
8	(Alsaleh & Larabi-Marie-Sainte, 2021)	Klasifikasi teks Bahasa Arab menggunakan model CNN dan Algoritma	CNN	Teks yang digunakan Bahasa Arab dengan model CNN dengan	Menggunakan model *deep learning* untuk klasifikasi teks.	Menggunakan *Hyperparameter-tunning* untuk menentukan

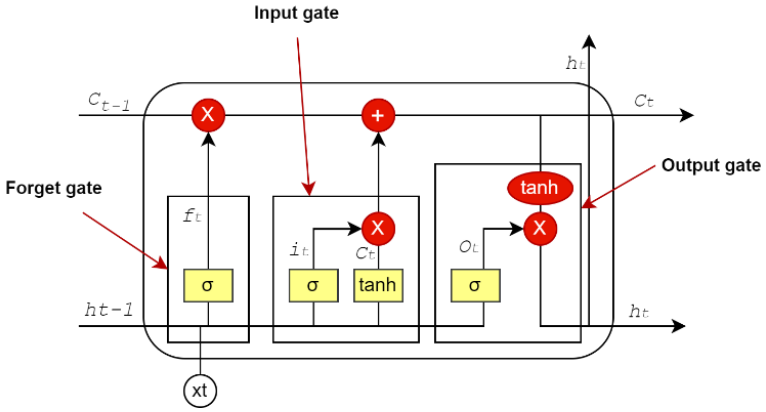
No	Peneliti	Kasus	Metode	Perbedaan	Persamaan	Rencana Penelitian
		Genetika untuk mengoptimalkan parameter CNN		parameter dioptimalkan dengan GA.		parameter LSTM terbaik.
9	(Taradhita & Putra, 2021)	Mengklasifikasikan ujaran kebencian dalam *tweet* berbahasa Indonesia dengan metode CNN	CNN	Model yang digunakan deep learning CNN.	Klasifikasi teks menggunakan bahasa Indonesia.	Mengoptimalkan kinerja *deep Learning* LSTM untuk klasifikasi teks.
10	(Alwehaibi et al., 2021)	Melakukan percobaan dengan menggunakan beberapa teknik *embedding* yang berbeda dan menguji performa model *deep learning* menggunakan data uji yang diambil dari Twitter	LSTM, CNN, CNN-LSTM	Menggabungkan model CNN-LSTM untuk menguji teknik *word embedding*	Penggunaan *Hyperparameter* sebagai penentu parameter.	Menguji LSTM untuk membandingkan kinerja *word embedding*.

Dari hasil rujukan penelitian sebelumnya yang telah diuraikan di atas, maka digunakan Algoritma *Long Short-Term Memory* (LSTM) dan *Bidirectional Long Short Term Memory* (Bi-LSTM) sebagai pengujian teknik *word embedding* antara *word2Vec* dan *FastText*. Sebagai alat bantu analisis penelitian ini digunakanlah bahasa pemrograman *Python*.

2.3. *Long Short-Term Memory*

Long Short-Term Memory (LSTM) merupakan salah satu jenis arsitektur dari RNN (*Recurrent Neural Network*) yang digunakan dalam model *deep learning*. Sejak Schmidhuber dan Hochreiter mengenalkan konsep LSTM pada tahun 1997 dalam bidang *forecasting* dan *recognition* telah mendapat manfaat besar dari penerapannya. Salah satu kelemahan RNN adalah pembelajaran jangka panjang dengan *gradient descent* bisa menghasilkan masalah menghilang atau meledaknya gradien (*vanishing/explode gradient*). Pada saat pembelajaran, ketika nilai gradien dipropagasikan kembali, nilai itu terus dikalikan dengan bobot yang lebih kecil dari satu (<1.0). Oleh karena itu, nilai *gradien* yang dihasilkan saat iterasi lama kelamaan cenderung mengecil atau mengalami kehilangan nilai.

LSTM sendiri dibangun bertujuan untuk memecahkan masalah *vanishing gradient* pada RNN (*Recurrent Neural Network*) yang dimana saat memproses data sekuensial yang panjang, kemiringan fungsi kerugian turun secara eksponensial (Kim & Kang, 2020). Arsitektur umum LSTM dapat dilihat pada Gambar 2.2.



Gambar 2.2 Arsitektur LSTM.

(Pasaribu et al., 2020)

Berdasarkan Gambar 2.2, Arsitektur LSTM terdiri dari sel-sel memori yang disebut sel. Dalam setiap langkah waktu (*time step*), dua state diteruskan ke sel berikutnya, yaitu *cell state* dan *hidden state*. *Cell state* adalah bagian utama yang mengalirkan data sepanjang jaringan, memungkinkan data untuk mengalir maju dengan sedikit perubahan. Meskipun demikian, beberapa transformasi *linier* dapat terjadi pada *cell state*. Data dapat ditambahkan atau dihapus dari *cell state* melalui penggunaan gerbang sigmoid. Gerbang tersebut memiliki struktur yang mirip dengan lapisan atau serangkaian operasi matriks, yang mengandung bobot individu yang berbeda untuk melakukan transformasi pada data (Le et al., 2019).

Gambar 2.2 menggambarkan standarisasi dari arsitektur metode LSTM yang dibagi menjadi empat komponen yaitu, *Input Gate (*i) untuk mengontrol arus input yang masuk ke *neuron*, *Forget Gate* (f) untuk memasukkan *neuron* ke kondisi reset saat ini. keadaan, *Output Gate* (o) untuk mengontrol efek aktivasi *neuron* pada *neuron* lain dan sel memori (c).

Keberadaan *Forget Gate* membantu menentukan derajat lupanya aliran informasi sebelum sel saat ini. Persaamaan *Forget Gate* ditunjukkan pada persamaan (2.1).

$$f_t = \sigma\,(W_f.[h_{t-1}\,.x_t] + b_f) \quad (2.1)$$

Komponen pada persamaan (2.1) adalah f_t nilai dari *forget gate*, σ adalah fungsi sigmoid, yang mengubah nilai input menjadi rentang dari 0 hingga 1. Fungsi sigmoid digunakan di sini untuk memastikan bahwa nilai f_t akan berada di antara 0 dan 1, sesuai dengan rasio bagian-bagian elemen seluler negara harus dilupakan atau dilestarikan. W_f adalah matriks bobot yang digunakan untuk mengalikan status masukan dan *hidden state*. h_{t-1} adalah *hidden state* dari sel LSTM pada *timestep* sebelumnya. $x_t$adalah input pada timestep saat ini. b_f adalah bias.

Fungsi *Input Gate* adalah untuk menentukan besarnya arus informasi yang ditambahkan pada arus informasi. Persamaan *Input Gate* disajikan pada persamaan (2.2) dan (2.3):

$$i_t = \sigma\,(W_i\,.\,[h_{t-1}\,.x_t] + b_i) \quad (2.2)$$

$$C_t = \tanh\,(W_c\,.\,[h_{t-1}\,.x_t] + b_c) \quad (2.3)$$

Setelah informasi melewati *input gate* dan *forget gate*, LSTM memperbarui status sel untuk menghitung keluaran sel LSTM saat ini dan meneruskannya ke sel LSTM berikutnya. Persamaan disajikan pada persamaan (2.4):

$$C_t = f_t * C_{t-1} + i_t * C_t \quad (2.4)$$

Output Gate menggabungkan masukan saat ini dan keadaan sel untuk menentukan keluaran sel LSTM saat ini. Persamaan *output gate* disajikan pada persamaan (2.5) dan (2.6):

$$O_t = \sigma\,(W_o\,.\,[h_{t-1}\,.x_t] + b_0) \tag{2.5}$$

$$h_t = O_t * \tanh\,(C_t) \tag{2.6}$$

Dimana x_t mewakili variabel *input* pada langkah waktu saat ini, $h_t$adalah output dari sel sebelumnya, C_{t-1} adalah keadaan sel sebelumnya yang memberikan informasi masa lalu. Parameter ini digunakan dengan sekumpulan matriks bobot dan vektor bias dalam fungsi *sigmoid* logistik σ dan tanh pada *input gate*, *forget gate* dan *output gate* (Vu et al., 2021).

Fungsi *sigmoid* diterapkan dalam pembelajaran mesin untuk *logistic regression* dan penerapan jaringan saraf. Nilai keluaran dari fungsi *sigmoid* berkisar antara 0 hingga 1. Persamaan yang digunakan dalam fungsi *sigmoid* ditunjukkan pada persamaan (2.7).

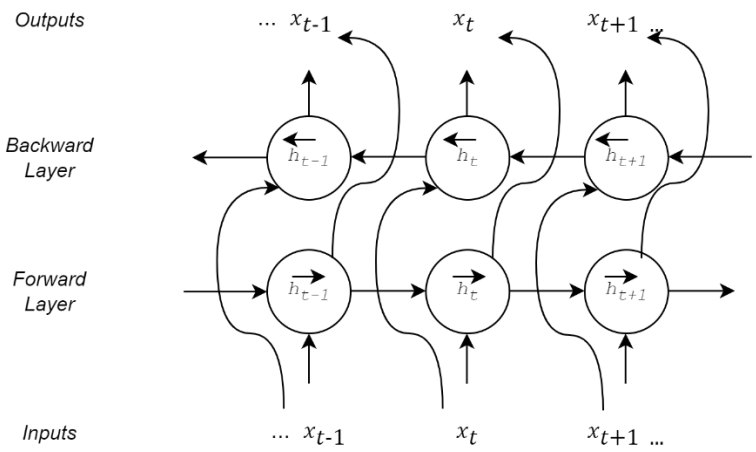
$$f(x) = \frac{1}{1 \div e^{-x}} \tag{2.7}$$

Dengan e adalah bilangan Euler yang merupakan konstanta matematika yang mendekati 2,71828 (Aghaeiboorkheili & Kawagle, 2022).

2.4. *Bidirectional Long Short-Term Memory*

Bidirectional Long-Short Term Memory (Bi-LSTM) merupakan arsitektur jaringan saraf yang terdiri dari dua lapisan LSTM, yaitu LSTM maju (*forward* LSTM) untuk memodelkan konteks sebelumnya dan LSTM mundur (*backward*

LSTM) untuk memodelkan konteks selanjutnya (Ertugrul & Karagoz, 2018). *Bidirectional Long-Short Term Memory* (Bi-LSTM) menghubungkan dua lapisan tersembunyi yang datang dari arah berlawanan dengan keluaran yang sama. Dengan menggunakan bentuk pembelajaran generatif mendalam ini, lapisan *neuron* dapat secara bersamaan memperoleh informasi tentang keadaan masa lalu dan masa depan (Isnain et al., 2020). Gambar 2.3 menunjukkan arsitektur dari *Bidirectional Long-Short Term Memory* (Bi-LSTM).



Gambar 2.3 Arsitektur Bi-LSTM.

(Anjana et al., 2019)

Input diberikan dari $t - 1$ hingga waktu $t - n$ untuk lapisan sebelumnya dan untuk lapisan berikutnya, input diberikan dalam arah berlawanan dari $t - n$ hingga t – 1. *Forward* LSTM operasi dari awal hingga akhir sesuai dengan urutan data. Pada setiap langkah waktu, informasi dari langkah sebelumnya diintegrasikan ke dalam pemrosesan langkah saat ini. Setiap sel LSTM (*Long Short-Term Memory*) memiliki tiga gerbang, yaitu *input gate*, *forget gate*, dan *output gate*, yang

memungkinkan pengorganisasian dan penyimpanan informasi jangka panjang. Persamaan *Forward* LSTM disajikan pada persamaan (2.8).

$$\vec{h}_t = \sigma\left(W_{x\vec{h}} . x_t + W_{\vec{h}\vec{h}} . \vec{h}_{t-1} + b_{\vec{h}}\right) \quad (2.8)$$

Dimana dari persamaan 2.8 :

- $\vec{h}_t$ adalah output yang dihasilkan oleh sel LSTM untuk langkah waktu tertentu.
- σ adalah fungsi aktivasi yang diterapkan pada hasil penjumlahan yang ada di sebelah kanan.
- $W_{x\vec{h}}$ hasil perkalian matriks antara bobot $W_{x\vec{h}}$ dan vektor input x_t.
- $W_{\vec{h}\vec{h}}$ adalah hasil perkalian matriks antara bobot $W_{\vec{h}\vec{h}}$ dan vektor *output* dari langkah waktu sebelumnya $\vec{h}_{t-1}$.
- $b_{\vec{h}}$ adalah bias.

Sedangkan *backward* LSTM beroperasi dari awal hingga akhir mengikuti urutan data yang sama. Seperti *forward* LSTM, *backward* LSTM juga memiliki struktur serupa dengan tiga gerbang, sehingga memungkinkan untuk memahami konteks langkah waktu mundur. Persamaan (2.9) merupakan persamaan dari *backward* LSTM.

$$\overleftarrow{h}_t = \sigma\left(W_{x\overleftarrow{h}} . x_t + W_{\overleftarrow{h}\overleftarrow{h}} . \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right) \quad (2.9)$$

Dimana dari persamaan 2.9 :

- $\overleftarrow{h}_t$ adalah *output* (*hidden state*) pada langkah waktu *t* dari LSTM yang bergerak mundur.
- σ adalah fungsi aktivasi yang diterapkan pada hasil penjumlahan linier dari input dan state sebelumnya.
- $W_{x\overleftarrow{h}}$ hasil matriks bobot yang menghubungkan input x_t ke $\overleftarrow{h}_t$.
- $W_{\overleftarrow{h}\overleftarrow{h}}$ adalah matriks bobot yang menghubungkan $\overleftarrow{h}_t$ pada langkah waktu sebelumnya *t + 1* ke $\overleftarrow{h}_t$ pada langkah waktu *t*.
- $b_{\overleftarrow{h}}$ adalah bias.

Output dari Bi-LSTM, baik *forward* dan *backward*, digabungkan pada setiap langkah waktu. Artinya lapisan Bi-LSTM menghasilkan *vector* keluaran yt, dimana setiap elemen dihitung dengan menggunakan persamaan (2.10).

$$y_t = W_{\vec{h}y}\vec{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y \quad (2.10)$$

Dimana dari persamaan (2.10) :

- y_t : Output pada langkah waktu t.
- $W_{\vec{h}y}$: Bobot yang menghubungkan representasi tersembunyi dari langkah waktu sebelumnya ke ouput y_t.
- $\vec{h}_t$: Representasi tersembunyi pada langkah waktu sebelumnya.
- $W_{\overleftarrow{h}y}$: Bobot yang menghubungkan representasi tersembunyi dari langkah waktu berikutnya ke ouput y_t.
- $\overleftarrow{h}_t$: Representasi tersembunyi pada langkah waktu berikutnya.
- b_y : Bias untuk output.

2.5. *Word Embeddings*

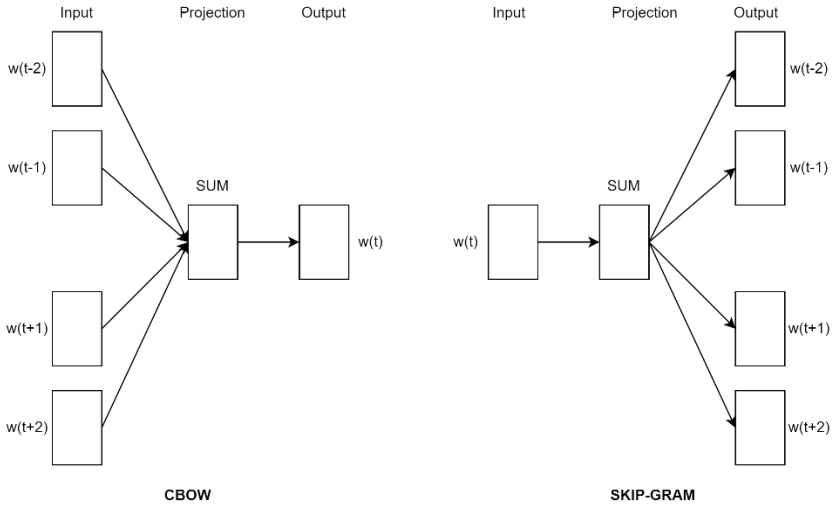
Word Embedding adalah metode merepresentasikan kata menurut nilai vektor tertentu. Awalnya, representasi kata (*word representation*) adalah istilah yang lebih dikenal daripada *Word Embedding*. Selain istilah-istilah tersebut, penyematan kata juga dikenal sebagai representasi terdistribusi (*distributed representation*) karena memiliki kerapatan, menggunakan vektor berdimensi rendah (*low dimentional vector*), dan bernilai nyata (Prabowo et al., 2019). *Word embedding* dapat membantu menentukan kesamaan kata berdasarkan konteks tertentu dilihat dari nilai vektor kata yang tidak berlabel. Nilai kemiripan yang dihasilkan oleh *Word Embedding* berkisar antara -1 hingga 1, dengan 1 merupakan nilai kemiripan tertinggi (Indrapurasih et al., 2018).

Hasil dari *Word Embeddings* dapat digunakan untuk menggambarkan kedekatan suatu kata atau dokumen, namun harus dipahami bahwa kedekatan tersebut dekat secara kontekstual tergantung pada data latih yang digunakan pada saat pelatihan. kata. Penyematan kata bekerja dengan terus melatih sekumpulan vektor dengan panjang tetap. Secara visual pada *Word Embedding* dapat direpresentasikan bahwa setiap kata diwakili oleh sebuah titik pada suatu area tertentu, titik-titik tersebut selanjutnya akan dipelajari dengan perhitungan *Word Embedding* dan suatu titik akan berada lebih jauh atau lebih dekat dengan titik lainnya (Prabowo et al., 2019).

2.5.1. *Word2vec*

Word2vec adalah penyematan kata yang dikembangkan oleh Google. *Word2vec* sendiri termasuk dalam kategori jaringan syaraf tiruan yang

menggunakan lapisan tersembunyi dan beberapa lapisan *non-linier* dalam algoritmanya. Ada dua jenis *word2vec*, yaitu model *Continuous Bag of Words* (CBOW) dan model *Skip-Gram*.



Gambar 2.4 Arsitektur CBOW dan *Skip-Gram*.

Secara umum, perbedaan antara CBOW dan *Skip-Gram* adalah arsitektur CBOW bertujuan untuk menghasilkan keluaran sebuah kata dari berbagai konteks kata masukan, sedangkan *Skip-Gram* bertujuan untuk mendeteksi beberapa kata yang sesuai konteks dari kata yang diketik.

2.5.2. *FastText*

Fasttext adalah library untuk pembelajaran mesin yang lebih efisien dalam mengklasifikasikan kalimat dan merepresentasikan kata. Berbeda dengan *word2vec*, algoritma ini memiliki struktur hierarki dan direpresentasikan sebagai vektor padat. Salah satu kelebihan *fasttext* adalah dapat mengatasi distribusi data yang tidak seimbang (Nurdin et al., 2020).

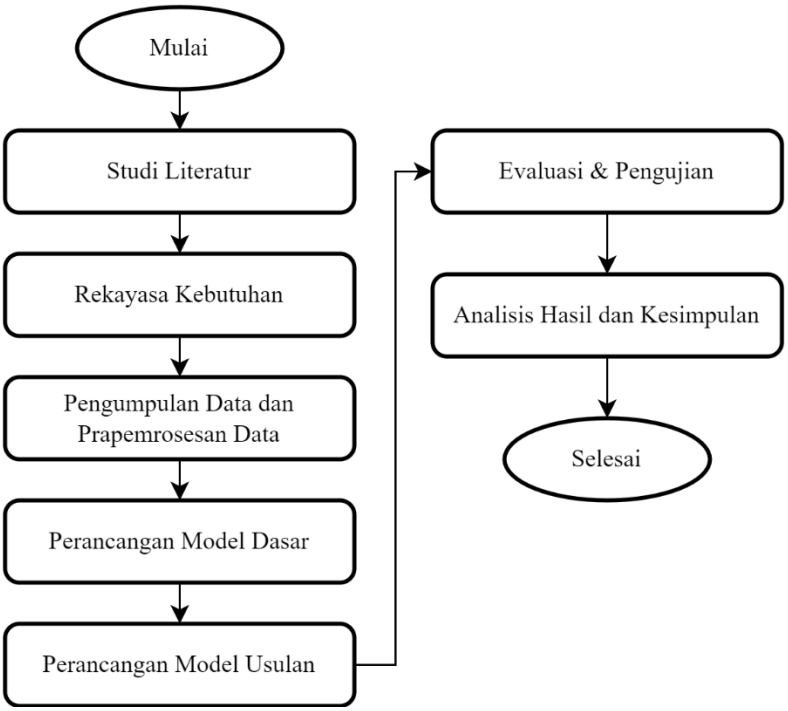
BAB III

METODE PENELITIAN

Pada bab ini akan dijelaskan tentang analisa dan perancangan sistem dari penelitian ini. Terdapat beberapa tahapan, yaitu tahapan penelitian yang dilakukan, kebutuhan yang akan dibuat dan penyelesaian masalah dalam klasifikasi terjemahan Al-Qur'an menggunakan varian algoritma *Long Short-Term Memory* (LSTM).

3.1. Desain Penelitian

Dalam melakukan penelitian ini, untuk mempermudah dalam penelitian maka dijabarkan langkah-langkah apa saja yang akan diambil dalam melakukan penelitian ini. Alur dari penelitian ini direpresentasikan pada Gambar 3.1.



Gambar 3.1 Desain Penelitian

3.2. Studi Literatur

Tahapan studi literatur melibatkan tindakan mengeksplorasi dan mengumpulkan literatur dan wawasan mutakhir saat melakukan penelitian (*state-of-the-art*). Beberapa teori terkait yang digunakan dalam penelitian ini di antaranya yaitu teknik klasifikasi *multi-label*, yang menggunakan *feature extraction* dengan *word2vec* dan *fastText*. Varian algoritma *Long Short-Term Memory* (LSTM) dan teori yang terkait dengan proses pengolahan data lainya yang sesuai dengan penelitian ini dengan diambil dari beberapa referensi, termasuk artikel jurnal ilmiah, buku, dan sumber lain yang relevan, digunakan untuk membantu penelitian ini.

3.3. Rekayasa Kebutuhan

Tahapan rekayasa kebutuhan melibatkan analisis semua elemen yang diperlukan untuk melakukan penelitian, seperti komponen perangkat keras dan perangkat lunak. Adapun perangkat keras yang digunakan pada penelitian ini memiliki spesifikasi *processor Core* I5 Gen 3, memori 8 GB, kapasitas *harddisk* 500 GB dan berjalan pada *windows* 10. Bahasa pemrograman yang digunakan untuk melakukan pengolahan bahan penelitian adalah dengan menggunakan bahasa pemrograman *Python* 3 yang dijalankan pada *Google Colaboratory*, dengan spesifikasi *memory* 12 GB dan *disk* 80 GB. *Library* yang digunakan pada penelitian ini adalah *scikit-learn*, *tensorflow*, *pandas* dan *library* pendukung lainnya.

3.4. Pengumpulan Data & Prapemrosesan Data

Tahap pengumpulan data merupakan proses pencarian data secara sistematis untuk mengumpulkan kumpulan data yang digunakan dalam penelitian. *Preprocessing* data dilakukan setelah selesai pengumpulan data untuk menghilangkan atau menyamakan keselarasan terjemahan Al-Quran.

3.4.1. Pengumpulan Data

Pada penelitian ini untuk keperluan klasifikasi teks maka diperlukan objek berupa teks. Data penelitian ini berasal terjemahan Al-Qur'an Kementerian Agama Republik Indonesia pada tahun 2022 berbahasa Indonesia yang dapat diakses pada situs https://quran.kemenag.go.id.

3.4.2. Pelabelan Data

Setelah data penelitian ini diperoleh, nantinya akan diberi label oleh pakar ahli secara manual. Sedangkan label yang ditentukan pada penelitian ini dibagi menjadi 4 label, yaitu, Tauhid, Ibadah, Akhlaq, dan Sejarah. Sehingga pakar akan memberikan label sesuai dengan ketentuan dari label penelitian pada setiap ayat terjemahan Al-Qur'an. Berikut deskripsi label yang digunakan pada penelitian kali ini :

1. **Tauhid,** label tauhid ini menggambarkan teks tersebut menyatakan tentang keesaan Allah Subhanahu Wa Ta'ala.
2. **Ibadah,** label ibadah ini menggambarkan teks yang mempunyai makna tentang tunduk dan patuh dalam melaksanakan dan menajuhi segala larangan Allah Subhanahu Wa Ta'ala.

3. **Akhlaq,** label akhlak ini menggambarkan teks yang menyatakan budi pekerti dan sifat yang dimiliki seseorang yang melahirkan perbuatan baik dan buruk.
4. **Sejarah / Tarikh,** label sejarah ini menggambarkan teks yang mengandung kisah ataupun peristiwa masa lampau yang menceritakan sejarah umat terdahulu untuk diambil pelajaran bagi umat sesudahnya.

3.4.3. *Text Preprocessing*

Tujuan utama dari *text preprocessing* adalah untuk meningkatkan data dengan membuatnya lebih terstruktur. Langkah *text preprocessing* diperlukan untuk mengubah kumpulan data sentimen menjadi teks terstruktur sehingga data siap untuk diproses ke tahapan selanjutnya. Selain itu, proses ini dapat meningkatkan performa dan akurasi model klasifikasi (Bessou & Aberkane, 2019). Berdasarkan Gambar 3.2, tahapan *text preprocessing* adalah *cleansing*, *case folding*, *tokenizing*. Berikut ilustrasi dari tahapan *text preprocessing* :

a. *Cleansing*

Proses penghapusan atau pembersihan pada data terjemahan yang berisi angka, delimeter seperti tanda koma (,), tanda petik (“) dan tanda titik (.) dan yang lainya.

Tabel 3.1 Ilustrasi *Cleansing*

	Teks terjemahan Al-Quran
Sebelum	Wahai orang-orang yang beriman, ingatlah nikmat Allah (yang dianugerahkan) kepadamu ketika suatu kaum bermaksud hendak

	menyerangmu dengan tangannya, lalu Dia menahan tangan (mencegah) mereka dari kamu. Bertakwalah kepada Allah dan hanya kepada Allahlah hendaknya orang-orang mukmin itu bertawakal.
Sesudah	Wahai orang-orang yang beriman ingatlah nikmat Allah yang dianugerahkan kepadamu ketika suatu kaum bermaksud hendak menyerangmu dengan tangannya lalu Dia menahan tangan mencegah mereka dari kamu Bertakwalah kepada Allah dan hanya kepada Allahlah hendaknya orang-orang mukmin itu bertawakal

b. *Case Folding*

Pada proses ini digunakan untuk mengubah semua karakter menjadi huruf yang sejenis. Dalam penelitian ini terjemahan Al-Qur'an di ubah yang awalnya huruf besar menjadi huruf kecil, agar kata yang sama akan tetapi penulisan hurufnya berbeda tidak menjadi data ganda. (Amalia et al., 2018).

Tabel 3.2 Ilustrasi *Case Folding*

	Teks terjemahan Al-Quran
Sebelum	Wahai orang-orang yang beriman ingatlah nikmat Allah yang dianugerahkan kepadamu ketika suatu kaum bermaksud hendak menyerangmu dengan tangannya lalu Dia menahan tangan mencegah mereka dari kamu Bertakwalah kepada Allah dan hanya kepada Allahlah hendaknya orang-orang mukmin itu bertawakal

Sesudah	wahai orang-orang yang beriman ingatlah nikmat allah yang dianugerahkan kepadamu ketika suatu kaum bermaksud hendak menyerangmu dengan tangannya lalu dia menahan tangan mencegah mereka dari kamu Bertakwalah kepada allah dan hanya kepada allahlah hendaknya orang-orang mukmin itu bertawakal

c. *Tokenizing*

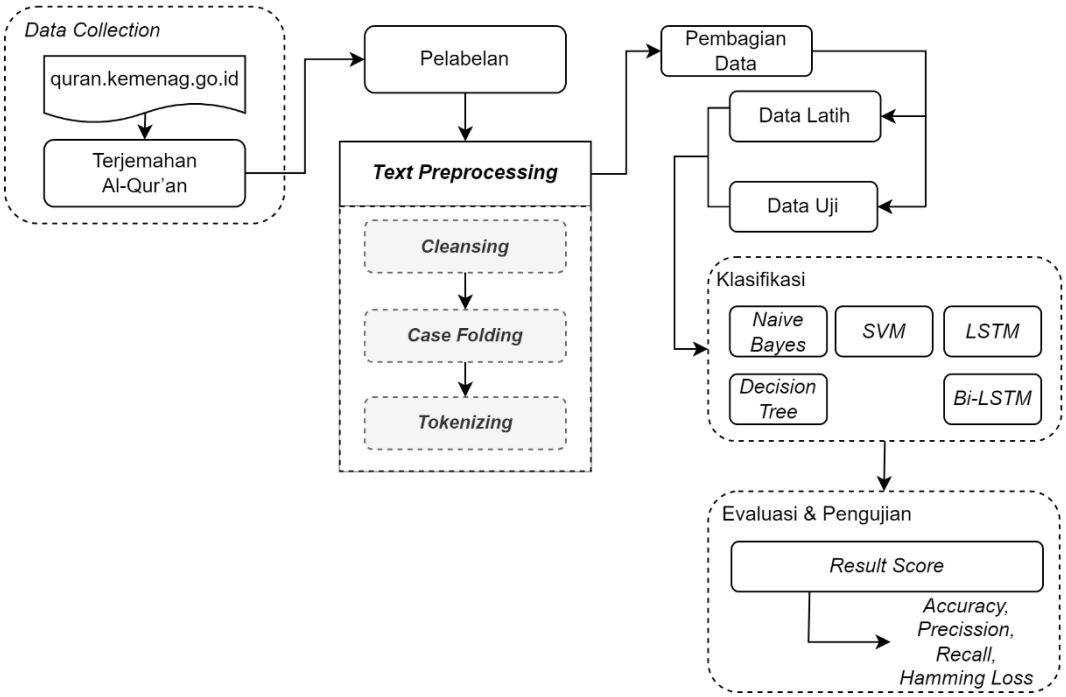
Proses langkah demi langkah untuk memecah kalimat, paragraf, atau dokumen menjadi beberapa bagian yang disebut token. Sebagai contoh ilustratif penggunaan token (kata) (Sari et al., 2019), perbedaan antara kumpulan data terakhir dan kumpulan data sebelumnya dapat dilihat pada Tabel 3.3.

Tabel 3.3 Ilustrasi *Tokenizing*

	Teks terjemahan Al-Quran
Sebelum	wahai orang-orang yang beriman ingatlah nikmat allah yang dianugerahkan kepadamu ketika suatu kaum bermaksud hendak menyerangmu dengan tangannya lalu dia menahan tangan mencegah mereka dari kamu Bertakwalah kepada allah dan hanya kepada allahlah hendaknya orang-orang mukmin itu bertawakal
Sesudah	'wahai', 'orang-orang', 'yang', 'beriman', 'ingatlah', 'nikmat', 'allah', 'yang', 'dianugerahkan', 'kepadamu', 'ketika', 'suatu', 'kaum', 'bermaksud', 'hendak', 'menyerangmu', 'dengan', 'tangannya', 'lalu', 'dia', 'menahan', 'tangan', 'mencegah', 'mereka', 'dari', 'kamu', 'Bertakwalah', 'kepada', 'allah', 'dan', 'hanya', 'kepada', 'allahlah', 'hendaknya', 'orang-orang', 'mukmin', 'itu', 'bertawakal'

3.5. Perancangan Model Dasar

Tahapan perancangan model dasar merupakan proses membangun model klasifikasi *multi-label* terjemahan Al-Quran yang akan digunakan sebagai tolak ukur model usulan. Perancangan model dasar terdiri dari dua proses utama, yaitu yang pertama pelatihan model *machine learning* dengan algoritma *Multinomial Naive Bayes*, *Decision Tree*, dan *Support Vector Machine*. Kedua pelatihan model varian algoritma *Long Short-Term Memory* (LSTM) dengan algoritma LSTM itu sendiri dengan pembacaan teks dalam satu arah dan *Bidirectional Long Short-Term Memory* (Bi-LSTM) yang membaca urutan teks dari dua arah yaitu kiri ke kanan dan kanan ke kiri.



Gambar 3.2 Perancangan model dasar.

3.5.1. Pembagian Data

Setelah melakukan beberapa langkah sebelumnya, pada bagian ini dataset penelitian dibagi menjadi data latih dan data uji. Data latih digunakan sebagai data latih dalam pengujian model, selanjutnya data uji adalah data yang digunakan untuk pengujian, yang belum pernah digunakan atau tidak digunakan sebagai data latih.

3.5.2. *Naïve Bayes Classifier*

Naive Bayes Classifier adalah algoritma klasifikasi berbasis teorema Bayes. Ketika ada banyak masukkan, metode kategorisasi ini tepat. Karena kecepatan dan kesederhanaannya, klasifikasi ini disukai. Tujuan dari algoritma ini adalah untuk menghitung probabilitas berdasarkan kategori data pelatihan. Sistem klasifikasi *Naive Bayes* memadukan informasi lama dan baru (Kim & Kang, 2020). Proses klasifikasi *Naïve Bayes* dapat dilihat pada Persamaan (3.1).

$$P\ (W_i \mid C_j) = \frac{N_{cw}+1}{N_c+V} \quad (3.1)$$

Persamaan (3.1) dimana nilai $P\ (W_i \mid C_j)$ merupakan peluang kategori j ketika terdapat kemunculan kata i. $N_{cw} + 1$merupakan jumlah kemunculan dari kata uji yang muncul dalam kategori *cw* ditambah 1 untuk menghindari nilai zero. N_c merupakan jumlah kemunculan seluruh kata dan V adalah jumlah seluruh kata unik yang ada pada seluruh kelas kategori.

3.5.3. *Decision Tree*

Decision tree (pohon keputusan) adalah pohon yang disusun dari sekumpulan atribut yang dapat diperiksa untuk memprediksi suatu hasil. Ketika

setiap *node* internal menampilkan pengujian tentang suatu atribut, hasil pengujian diwakili oleh setiap cabang dan label kelas yang dimiliki oleh setiap *node*. Dalam pohon keputusan, simpul teratas adalah simpul akar. Menentukan akar pohon menggunakan *gain* tertinggi dari setiap atribut atau berdasarkan nilai indeks *entropi* terendah. Dengan cara mencari terlebih dahulu nilai entropinya menggunakan rumus dari Persamaan (3.2). Selanjutnya menghitung nilai gain menggunakan rumus pada Persamaan (3.3).

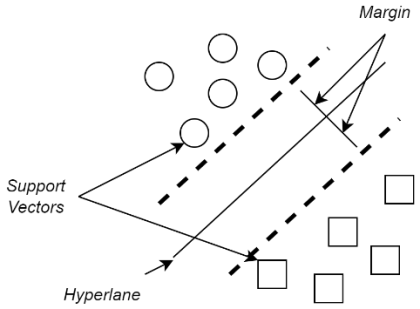
$$Entropy\ (S) = \sum_{i=0}^{n} - \pi *\ log^2\ \pi \tag{3.2}$$

$$Gain\ (S, A) = Entropy(S) = \ \sum_{i=1}^{n} \frac{|Si|}{|S|} * Entropy\ (Si) \tag{3.3}$$

Persamaan (3.2) dan Persamaan (3.3) dapat diketahui *Gain* (S,A) adalah *information gain* yang ingin kita hitung. Ini mengukur sejauh mana pemilihan atribut (A) akan mengurangi ketidakpastian (entropi) dalam *dataset* (S) ketika dibagi menjadi *subset* (Si). *Entropy* (S) mengukur tingkat ketidakpastian atau kekacauan dalam dataset. Semakin tinggi entropi, semakin banyak ketidakpastian. Σ (i=1 to n) digunakan untuk menjumlahkan kontribusi dari setiap *subset* (Si) yang dihasilkan saat kita membagi *dataset* (S) dengan menggunakan atribut (A). |Si| / |S| merupakan fraksi atau rasio ukuran *subset* (Si) terhadap ukuran *dataset* awal (S). Ini mengukur sejauh mana subset ini menyumbang terhadap *dataset* keseluruhan. *Entropi* (Si) merupakan *entropi* dari *subset* (Si) yang dihasilkan saat kita membagi dataset menggunakan atribut (A). *Entropi* ini mengukur ketidakpastian dalam *subset* tersebut setelah pembagian.

3.5.4. *Support Vector Machine*

Support Vector Machine adalah metode pembelajaran mesin dan termasuk dalam pembelajaran yang diawasi. *Support Vector Machine* bekerja dengan menentukan *hyperplane* terbaik dengan memaksimalkan margin antar lapisan yang berdekatan (Taufiqurrahman et al., 2021). Pada Gambar 2, Anda dapat melihat ilustrasi metode *Support Vector Machine*.



Gambar 3.3 Ilustrasi *Support Vector Machine*.

Hyperplane adalah fungsi yang memisahkan label kelas dalam data. *Hyperplane* terbaik dapat ditemukan dengan menghitung *margin* antar lapisan dan mencari nilai maksimumnya. Untuk menghitung *hyperplane* dapat dijelaskan berikut ini (Taufiqurrahman et al., 2021):

Sampel pelatihannya berbentuk $\{xi\ yi\}$, …, $\{xn, yn\}$ yang berarti $xi \in R\ m$, khusus atributnya, dan $yi \in \{-1,1\}$, khusus kelas labelnya. Untuk menghitung *hyperplane* dapat menggunakan persamaan $w^T x + b = 0$, dimana $w \in R\ m$ dan b adalah skalar. Untuk kasus dimana data dapat dipisahkan secara *linier* dengan baik, *hyperplane* optimal akan dihasilkan menggunakan Persamaan (3.4).

$$y_i\ (w^T x_i + b\ \geq 1, i = 1, \dots, N) \tag{3.4}$$

Dimana Persamaan (3.4) adalah :

- y_i : Merupakan label kelas untuk sampel ke-i. Biasanya, nilai y_i adalah -1 atau 1, yang mewakili label.
- w : Merupakan vektor bobot dari hyperplane.
- x_i : Merupakan vektor fitur dari sampel ke-i.
- b : Merupakan bias.
- N : Menunjukkan jumlah total sampel atau instance dalam dataset.

Persamaan *Support Vector Machine* (SVM) dalam konteks klasifikasi *multi-label* tertuang pada Persamaan (3.5).

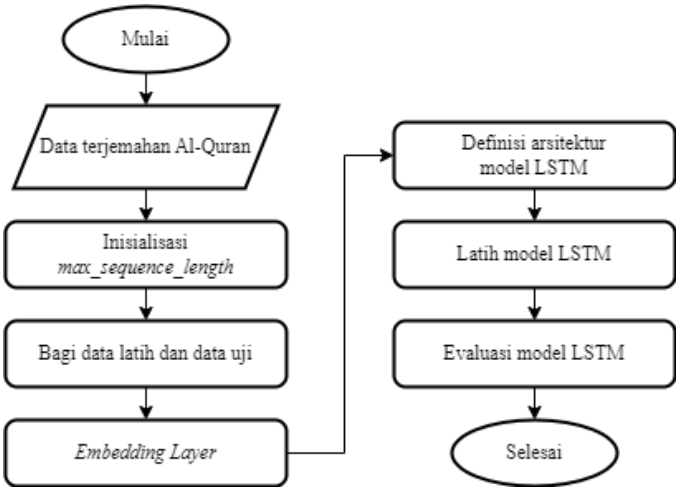
$$f(x) = sign\ (\sum_{i=1}^{n} a_i y_i\ [x, x_i] + b) \quad (3.5)$$

Dimana Persamaan (3.5) :

- f : fungsi keputusan yang memberikan prediksi kelas untuk suatu data x.
- $sign$: fungsi tanda yang menghasilkan tanda dari hasil perkalian dalam tanda kurung.
- $\sum_{i=1}^{n} a_i y_i\ [x, x_i]$: merupakan simbol untuk penjumlahan dari $i = 1$ hingga n, dengan n adalah jumlah sampel dalam dataset. a_i koefisien dari *support vector* ke-i. y_i adalah label kelas ke-i. Biasanya y_i bisa bernilai -1 atau 1 dalam kasus klasifikasi biner. $[x, x_i]$ adalah *dot product* antara vektor fitur x , dan vektor fitur x_i dari vektor pendukung ke-i.
- b : bias (offset) yang merupakan konstanta yang disesuaikan dalam proses pelatihan SVM.

3.5.5. *Long Short-Term Memory (LSTM)*

Model dasar ke-empat merupakan model dari *deep learning*, yaitu algoritma *Long Short-Term Memo*ry (LSTM).



Gambar 3.4 Diagram alir model klasifikasi LSTM.

Diagram alir pelatihan algoritma *Long Short-Term Memo*ry (LSTM) untuk klasifikasi *multi-label* terjemahan Al-Qur'an ditunjukkan pada Gambar 3.4 dan dijelaskan sebagai berikut :

1. Masukkan data terjemahan Al-Qur'an yang telah melewati tahapan *text preprocessing*.
2. Masukkan *word vectors* yang diperoleh dari pelatihan model *word embedding*. *Word vectors* terdiri dari kosa kata dan kosa kata vektor yang digunakan untuk membuat *embedding layer* pada tahap selanjutnya.
3. Tentukan panjang maksimal dari urutan kata dan simpan pada variabel max_seq_length. Panjang maksimal urutan kata didapatkan dengan menghitung distribusi panjang setiap kalimat pada data.

4. Langkah berikutnya adalah mempersiapkan *embedding layer*. Pertama, lakukan tokenisasi pada level kata untuk membangun *vocabulary*. Kemudian tentukan frekuensi kemunculan kata dan simpan pada variabel num_words. Hanya $num\_words - 1$ yang akan disimpan pada *vocabulary*. Sedangkan kata dengan kemunculan $\leq num\_words$ tidak digunakan. Hitung jumlah kata unik pada *vocabulary* dan simpan pada variabel $vocab\_size$. Berikutnya adalah mengubah token menjadi indeks bilangan bulat berdasarkan indeks dari *vocabulary*. Tambahkan token <unk> untuk mengganti kata yang tidak terdaftar pada *vocabulary*. Berikutnya tambahkan token <pad> untuk menambahkan padding ke urutan indeks token jika jumlahnya kurang dari max_seq_length dan lakukan pemotongan jika panjang urutan indeks token lebih dari max_seq_length. Langkah terakhir adalah membuat *embedding matrix* dari *word vectors* hasil pelatihan *word embedding*.
5. Bagi data terjemahan Al-Qur'an untuk data latih dan data uji.
6. Definisikan arsitektur model *Long short-term memory* (LSTM).
7. Latih model dengan inisialisasi parameter pelatihan yang telah ditentukan.
8. Lakukan evaluasi model menggunakan metrik *hamming loss*, *accuracy*, *precision*, *recall* dan *F1_Score*.

Selanjutnya melakukan perhitungan menggunakan *Long Short-Term Memory* (LSTM) dengan menggabungkan empat gerbang yang berbeda, secara khusus disebut sebagai *forget gates*, *input gates*, *cell gates*, dan *output gates*.

Perhitungan manual yang akan dilakukan ini telah menetapkan nilai bobot (W_x , W_h) dan nilai bias (b) dengan value telah diasumsikan sebagai berikut :

$$\begin{bmatrix} W_f & W_f & b_f \\ W_i & W_i & b_i \\ W_c & W_c & b_c \\ W_0 & W_0 & b_0 \end{bmatrix} = \begin{bmatrix} 0{,}7 & 0{,}15 & 0{,}2 \\ 0{,}95 & 0{,}8 & 0{,}65 \\ 0{,}45 & 0{,}15 & 0{,}2 \\ 0{,}6 & 0{,}25 & 0{,}1 \end{bmatrix}$$

Setelah itu kita telah memiliki nilai vektor yang diketahui $X_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, dan $X_2 = \begin{bmatrix} 0{,}5 \\ 3 \end{bmatrix}$ dimana nilai input tersebut adalah hasil dari proses *word embedding* atau layer pertama. Matriks W_h dan *b* memilik ukuran 1 x 1 dimana pada ukuran W_h didapat dari *hidden neuron* dikalikan jumlah *neuron* dan pada b memiliki ukuran 1 dikali *hidden neuron* sedangkan matriks W_x merupakan bobot dari *x* sehingga memiliki ukuran yang sama dengan *x*. Selain parameter diatas juga dibutuhkan nilai h_{t-1} dan C_{t-1}. Pada perhitungan ini, $h_{t-1} = [0]$ dan $C_{t-1} = [0]$ dikarenakan belum ada proses dan nilai sebelumnya atau perhitungan ini merupakan *t* = 1. Setelah parameter yang dibutuhkan terpenuhi, berikut merupakan contoh perhitungan metode LSTM :

1. *Forget Gates*

$$f_t = \sigma\,(W_f . [h_{t-1} . x_t] + b_f)$$

$$f_t = \sigma\,([0{,}7\ 0{,}45] . \begin{bmatrix} 1 \\ 2 \end{bmatrix} + [0.1] . [0] + [0{,}15])$$

$$f_t = \sigma\,([1{,}6] + [0{,}15])$$

$$f_t = \sigma\,([1{,}75])$$

$$f_t = \frac{1}{(1 + e^{-1{,}75})}$$

$f_t = [0{,}85195]$

2. *Input Gates*

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

$i_t = \sigma\,([0{,}95\ 0{,}8]\,.\,\begin{bmatrix}1\\2\end{bmatrix} + [0{,}8]\,.\,[0] + [0{,}65])$

$i_t = \sigma\,([2{,}55] + [0{,}65])$

$i_t = \sigma\,([3{,}2])$

$i_t = \dfrac{1}{(1 + e^{-3{,}2})}$

$i_t = [0{,}96083]$

Kemudian menghitung fungsi aktivasi tanh dimana hasil dari perhitungan aktivasi tanh akan digunakan untuk perhitungan *cell gates*.

$C_t = \tanh\,(W_c\,.\,[h_{t-1}\,.x_t] + b_c)$

$C_t = tanh\,([0{,}45\ 0{,}25]\,.\,\begin{bmatrix}1\\2\end{bmatrix} + [0{,}15]\,.\,[0] + [0{,}2])$

$C_t = tanh\,([0{,}95] + [0{,}2])$

$C_t = tanh\,([1{,}15])$

$C_t = \dfrac{2}{(1 + e^{-2{,}021})} - 1$

$C_t = [0{,}81775]$

3. *Cell Gates*

$C_t = f_t * C_{t-1} + i_t * C_t$

$C_t = [0{,}85195] * [0] + [0{,}96083] * [0{,}81775])$

$C_t = [0{,}78572]$

4. *Output Gates*

$$O_t = \sigma\,(W_o\,.\,[h_{t-1}\,.\,x_t] + b_0)$$

$$O_t = \sigma\,([0{,}6\ 0{,}4]\,.\,\begin{bmatrix}1\\2\end{bmatrix} + [0{,}25]\,.\,[0] + [0{,}1])$$

$$O_t = \sigma\,([1{,}4] + [0{,}1])$$

$$O_t = \sigma\,([1{,}5])$$

$$O_t = \frac{1}{(1 + e^{-0{,}12})}$$

$$O_t = [0{,}81757]$$

Pada perhitungan diatas di dapatkan *output* $h_t = [0{,}53631]$ dan $C_t = [0{,}78572]$, kedua *output* tersebut akan digunakan untuk *time step* berikutnya sampai dengan *time step* maksimal.

3.5.6. *Bidirectional Long Short-Term Memory (Bi-LSTM)*

Bidirectional Long Short-Term Memory (Bi-LSTM) merupakan salah satu varian dari algoritma LSTM. Seperti penjelasan sebelumnya, Bi-LSTM terdiri dari dua lapisan LSTM, yaitu LSTM maju (*forward* LSTM) untuk memodelkan konteks sebelumnya dan LSTM mundur (*backward* LSTM). Artinya model membaca string teks dalam dua arah, yaitu dari kiri ke kanan dan dari kanan ke kiri. Berikut contoh perhitungan dari *Bidirectional Long Short-Term Memory* (Bi-LSTM) dengan ditetapkan nilai dari bobot sebagai berikut :

$W_{N\vec{h}}$ = [0,45 0,25]	$W_{N\overleftarrow{h}}$ = [0,3 0,1]	$U_{N\vec{h}}$ = [0,15]	$U_{N\overleftarrow{h}}$ = [0,3]
$W_{i\vec{h}}$ = [0,95 0,8]	$W_{i\overleftarrow{h}}$ = [0,1 0,7]	$U_{i\vec{h}}$ = [0,8]	$U_{i\overleftarrow{h}}$ = [0,45]
$W_{f\vec{h}}$ = [0,7 0,45]	$W_{f\overleftarrow{h}}$ = [0,4 0,35]	$U_{f\vec{h}}$ = [0,1]	$U_{f\overleftarrow{h}}$ = [0,15]
$W_{o\vec{h}}$ = [0,6 0,4]	$W_{o\overleftarrow{h}}$ = [0,25 0,5]	$U_{o\vec{h}}$ = [0,25]	$U_{o\overleftarrow{h}}$ = [0,6]
$W_{y\vec{h}}$ = [0,5 0,7]	$W_{y\overleftarrow{h}}$ = [0,2]		

$b_{N\vec{h}}$ = [0,2] $b_{N\overleftarrow{h}}$ = [0,4] b_y = [0,3]

$b_{i\vec{h}}$ = [0,65] $b_{i\overleftarrow{h}}$ = [0,1] $x_0 = \begin{bmatrix}1\\2\end{bmatrix}$ $x_1 = \begin{bmatrix}0{,}5\\3\end{bmatrix}$

$b_{f\vec{h}}$ = [0,5] $b_{f\overleftarrow{h}}$ = [0,2]

$b_{o\vec{h}}$ = [0,1] $b_{o\overleftarrow{h}}$ = [0,6]

***Forward* LSTM dengan *t* = 0.**

$$\vec{f}_0 = \sigma\,(W_{f\vec{h}} \,.\, x_t + U_{f\vec{h}} \,.\, \vec{h}_{t-1} + b_{f\vec{h}})$$

$$= \sigma\,([0{,}7 \quad 0{,}45] \,.\, \begin{bmatrix}1\\2\end{bmatrix} + [0{,}1].[0] + [0{,}5]) = \sigma(2{,}1) = 0{,}8909 \approx 1$$

$$\vec{i}_0 = \sigma\,(W_{i\vec{h}} \,.\, x_t + U_{i\vec{h}} \,.\, \vec{h}_{t-1} + b_{i\vec{h}})$$

$$= \sigma\,([0{,}95 \;\; 0{,}8] \,.\, \begin{bmatrix}1\\2\end{bmatrix} + [0{,}8].[0] + [0{,}65]) = \sigma(3{,}2) = 0{,}9608 \approx 1$$

$$\vec{O}_0 = \sigma\,(W_{o\vec{h}} \,.\, x_t + U_{o\vec{h}} \,.\, \vec{h}_{t-1} + b_{o\vec{h}})$$

$$= \sigma\,([0{,}6 \quad 0{,}4] \,.\, \begin{bmatrix}1\\2\end{bmatrix} + [0{,}25].[0] + [0{,}1]) = \sigma(1{,}5) = 0{,}8175 \approx 1$$

$$\vec{N}_0 = tanh\,(W_{N\vec{h}} \,.\, x_t + U_{N\vec{h}} \,.\, \vec{h}_{t-1} + b_{N\vec{h}})$$

$$= tanh\,([0{,}45 \quad 0{,}25] \,.\, \begin{bmatrix}1\\2\end{bmatrix} + [0{,}15].[0] + [0{,}2])$$

$$= tanh\,(1{,}15) = 0{,}8177$$

$$\vec{C}_0 = N_{\vec{t}} \,.\, i_t + f_{\vec{t}} \,.\, C_{\vec{t}}$$

$$= (0{,}8177)(1) + (1)(0) = 0{,}8177$$

$$\vec{h}_0 = O_{\vec{t}} \,.\, tanh\,(C_t)$$

$$= (1) \,.\, tanh\,(0{,}8177) = 0{,}6738$$

***Forward* LSTM dengan *t* = 1.**

$$\vec{f}_1 = \sigma\left(W_{f\vec{h}} . x_t + U_{f\vec{h}} . \vec{h}_{t-1} + b_{f\vec{h}}\right)$$

$$= \sigma\left([0{,}7 \quad 0{,}45] . \begin{bmatrix}1{,}5\\4\end{bmatrix} + [0{,}1] . [0{,}6378] + [0{,}5]\right) = \sigma(3{,}42) = 0{,}9683 \approx 1$$

$$\vec{i}_1 = \sigma\left(W_{i\vec{h}} . x_t + U_{i\vec{h}} . \vec{h}_{t-1} + b_{i\vec{h}}\right)$$

$$= \sigma\left([0{,}95 \quad 0{,}8] . \begin{bmatrix}1{,}5\\4\end{bmatrix} + [0{,}8] . [0{,}6378] + [0{,}65]\right) = \sigma(5{,}814) = 0{,}997 \approx 1$$

$$\vec{O}_1 = \sigma\left(W_{O\vec{h}} . x_t + U_{O\vec{h}} . \vec{h}_{t-1} + b_{O\vec{h}}\right)$$

$$= \sigma\left([0{,}6 \quad 0{,}4] . \begin{bmatrix}1{,}5\\4\end{bmatrix} + [0{,}25] . [0{,}6378] + [0{,}1]\right) = \sigma(2{,}768) = 0{,}9409 \approx 1$$

$$\vec{N}_1 = tanh\left(W_{N\vec{h}} . x_t + U_{N\vec{h}} . \vec{h}_{t-1} + b_{N\vec{h}}\right)$$

$$= tanh\left([0{,}45 \quad 0{,}25] . \begin{bmatrix}1{,}5\\4\end{bmatrix} + [0{,}15] . [0{,}6378] + [0{,}2]\right)$$

$$= tanh\,(1{,}875) = 0{,}954$$

$$\vec{C}_1 = N_{\vec{t}} . i_t + f_{\vec{t}} . C_{\vec{t}}$$

$$= (0{,}954)(1) + (1)(\,0{,}8177) = 1{,}7717$$

$$\vec{h}_1 = O_{\vec{t}} . tanh\,(C_t)$$

$$= (1) . tanh\,(1{,}7717) = 0{,}9438$$

***Backward* LSTM dengan *t* = 1.**

$$\overleftarrow{f}_1 = \sigma\left(W_{f\overleftarrow{h}} . x_t + U_{f\overleftarrow{h}} . \overleftarrow{h}_{t-1} + b_{f\overleftarrow{h}}\right)$$

$$= \sigma\left([0{,}4 \quad 0{,}35] . \begin{bmatrix}1{,}5\\4\end{bmatrix} + [0{,}15] . [0] + [0{,}2]\right) = \sigma(2{,}2) = 0{,}9002 \approx 1$$

$$\overleftarrow{i}_1 = \sigma\left(W_{i\overleftarrow{h}} . x_t + U_{i\overleftarrow{h}} . \overleftarrow{h}_{t-1} + b_{i\overleftarrow{h}}\right)$$

$$= \sigma\left([0{,}1 \quad 0{,}7] \cdot \begin{bmatrix}1{,}5\\4\end{bmatrix} + [0{,}45].[0] + [0{,}1]\right) = \sigma(3{,}05) = 0{,}9547 \approx 1$$

$$\overleftarrow{O}_1 = \sigma\left(W_{O\overleftarrow{h}} . x_t + U_{O\overleftarrow{h}} . \overleftarrow{h}_{t-1} + b_{O\overleftarrow{h}}\right)$$

$$= \sigma\left([0{,}25\ 0{,}5] \cdot \begin{bmatrix}1{,}5\\4\end{bmatrix} + [0{,}6].[0] + [0{,}6]\right) = \sigma(2{,}975) = 0{,}9514 \approx 1$$

$$\overleftarrow{N}_1 = tanh\left(W_{N\overleftarrow{h}} . x_t + U_{N\overleftarrow{h}} . \vec{h}_{t-1} + b_{N\overleftarrow{h}}\right)$$

$$= tanh\left([0{,}3 \quad 0{,}1] \cdot \begin{bmatrix}1{,}5\\4\end{bmatrix} + [0{,}3].[0] + [0{,}4]\right)$$

$$= tanh\,(1{,}25) = 0{,}8482$$

$$\overleftarrow{C}_1 = N_{\overleftarrow{t}} . i_t + f_{\overleftarrow{t}} . C_{\overleftarrow{t}}$$

$$= (0{,}8482)(1) + (1)(0) = 0{,}8482$$

$$\vec{h}_1 = O_{\vec{t}} . tanh\,(C_t)$$

$$= (1) . tanh\,(0{,}8482) = 0{,}6901$$

***Backward* LSTM dengan *t* = 0.**

$$\overleftarrow{f}_1 = \sigma\left(W_{f\overleftarrow{h}} . x_t + U_{f\overleftarrow{h}} . \overleftarrow{h}_{t-1} + b_{f\overleftarrow{h}}\right)$$

$$= \sigma\left([0{,}4 \quad 0{,}35] \cdot \begin{bmatrix}1\\2\end{bmatrix} + [0{,}15].[0{,}6901] + [0{,}2]\right) = \sigma(1{,}4035) = 0{,}8027$$

$$\overleftarrow{\imath}_1 = \sigma\left(W_{i\overleftarrow{h}} . x_t + U_{i\overleftarrow{h}} . \overleftarrow{h}_{t-1} + b_{i\overleftarrow{h}}\right)$$

$$= \sigma\left([0{,}1 \quad 0{,}7] \cdot \begin{bmatrix}1\\2\end{bmatrix} + [0{,}45].[0{,}6901] + [0{,}1]\right) = \sigma(1{,}9105) = 0{,}871$$

$$\overleftarrow{O}_1 = \sigma\left(W_{O\overleftarrow{h}} . x_t + U_{O\overleftarrow{h}} . \overleftarrow{h}_{t-1} + b_{O\overleftarrow{h}}\right)$$

$$= \sigma\left([0{,}25\ 0{,}5] \cdot \begin{bmatrix}1\\2\end{bmatrix} + [0{,}6].[0{,}6901] + [0{,}6]\right) = \sigma(2{,}264) = 0{,}9058$$

$$\overleftarrow{N}_1 = tanh\left(W_{N\overleftarrow{h}} . x_t + U_{N\overleftarrow{h}} . \vec{h}_{t-1} + b_{N\overleftarrow{h}}\right)$$

$$= tanh\,([0{,}3 \quad 0{,}1] \,.\, \begin{bmatrix}1\\2\end{bmatrix} + [0{,}3].[0{,}6901] + [0{,}4])$$

$$= tanh\,(1{,}107) = 0{,}7515$$

$$\overleftarrow{C_1} = N_{\overleftarrow{t}} \,.\, i_t + f_{\overleftarrow{t}} \,.\, C_{\overleftarrow{t}}$$

$$= (0{,}7515)(1) + (1)(0{,}8482) = 1{,}5997$$

$$\vec{h}_1 = O_{\vec{t}} \,.\, tanh\,(C_t)$$

$$= (1) \,.\, tanh\,(1{,}5997) = 0{,}9216$$

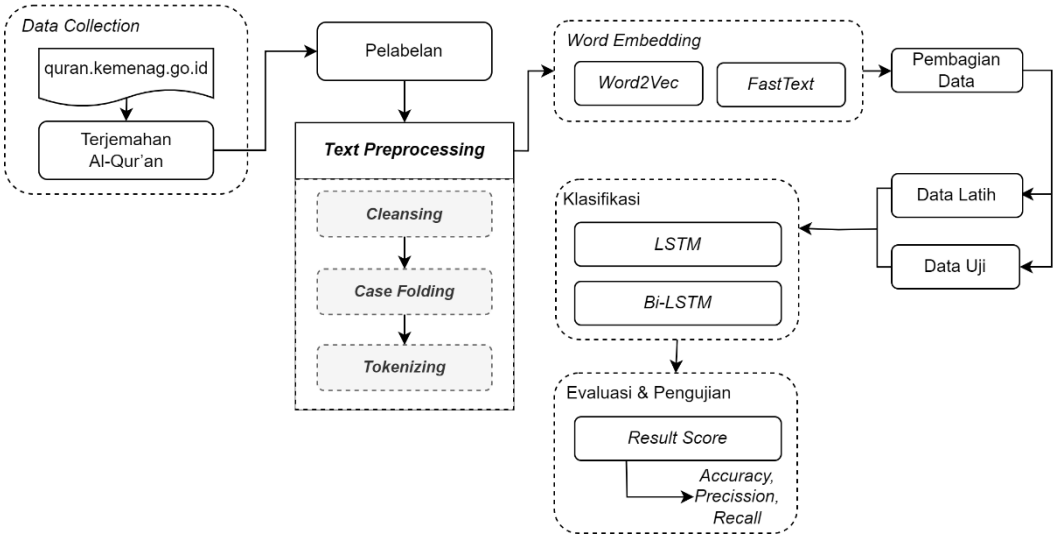
Maka, nilai dari $y_1$yaitu :

$$y_1 = W_{\vec{h}y}\vec{h}_1 + W_{\overleftarrow{h}y}\overleftarrow{h}_1 + b_y$$

$$= [0{,}5]\,[0{,}9438] + [0{,}2]\,[0{,}6901] + [0{,}3] = 0{,}9092$$

3.6. Perancangan Model Usulan

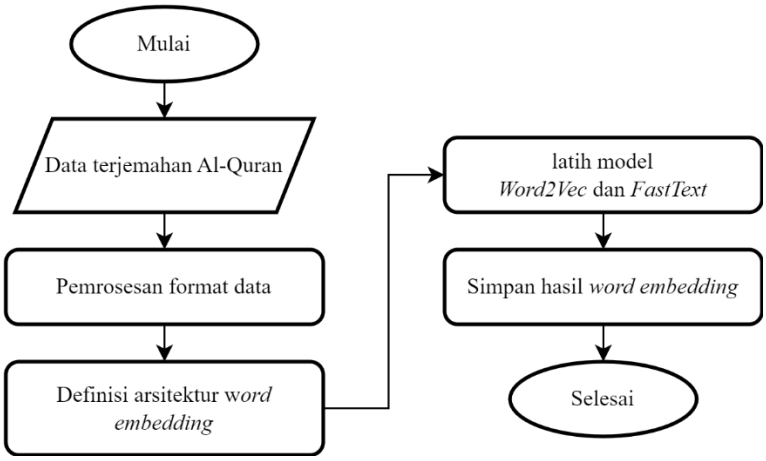
Pada tahapan ini dipaparkan arsitektur model dalam penelitian yang dilakukan, dalam penelitian ini menggunakan varian dari algoritma LSTM yang teridir dari *Long Short-Term Memory* (LSTM) dan *Bidirectional Long Short-Term Memory* (Bi-LSTM) untuk klasifikasi *multi-label* terjemahan Al-Quran berbahasa Indonesia yang direpresentasikan pada Gambar 3.5. Gambar 3.5 merupakan tujuan dari penelitian ini, dari gambar tersebut nantinya akan menjadi empat skema pengujian, yaitu LSTM + Word2Vec, LSTM + FastText, Bi-LSTM + Word2Vec dan yang terakhir Bi-LSTM + FastText.



Gambar 3.5 Perancangan model penelitian

3.6.1. Representasi *Word Embedding*

Setelah melakukan proses *text preprocessing*, proses selanjutnya adalah melatih model *word embedding*. Secara sederhana, *word embedding* adalah teknik untuk mengubah teks menjadi vektor agar dapat di proses oleh algoritma tertentu. Penelitian ini membandingkan dua tipe *word embedding* berdasarkan model *Word2Vec* dan *FastText* berbasis arsitektur *Continous Bag of Word* (CBOW). CBOW menghasilkan kinerja yang lebih baik karena memiliki akurasi yang sedikit lebih tinggi pada kata-kata yang sering muncul, dan pada kumpulan data pencarian ini terdapat banyak kata yang sering muncul. (Af'idah et al., 2021).



Gambar 3.6 Diagram alir pelatihan model *word embedding*.

Gambar 3.6 menggambarkan diagram alir untuk pelatihan teknik *word embedding* dan dijelaskan sebagai berikut.

1. Masukkan data terjemahan Al-Qur'an yang telah melewati tahapan *text preprocessing*.
2. Lakukan pemrosesan data sehingga sesuai dengan format masukkan arsitektur.
3. Definisikan arsitektur model *word embedding*.
4. Latih model dengan inisialisasi parameter pelatihan.
5. Simpan matiks kosakata hasil pelatihan model.

Perbedaan utama antara *Word2Vec* dan *FastText* berkaitan dengan penggabungan $n{-}gram$. *Word2Vec* memperoleh vektor kata lengkap yang ada dalam leksikon. *FastText* mampu memperoleh pengetahuan tentang vektor $n{-}gram$ yang ada di setiap entri kamus (Nurdin et al., 2020).

3.6.2. Evaluasi & Pengujian

Tahapan pengujian merupakan proses melakukan pengujian terhadap model pada penelitian ini. Pengujian pertama dilakukan untuk mengetahui kinerja teknik *word embedding word2Vec* pada arsitektur model *Long short-term memory* (LSTM). Pengujian kedua mengujikan *word embedding FastText* pada arsitektur model yang sama. Untuk mengevaluasi metode yang diusulkan, menggunakan *Confusion Matrix* merupakan metode pengukuran dan evaluasi keakuratan model. Pada tabel 3.4 merupakan *Confusion Matrix* yang digunakan pada penelitian ini.

Tabel 3.4 *Confusion Matrix*.

	Kelas Aktual	
Kelas Prediksi	TP (*True Positive*)	FN (*False Negative*)
	FP (False Positive)	TN (*True Negative*)

Kinerja model diukur menggunakan metrik *accuracy*, *precision*, *recall* dan *Hamming loss* merupakan salah satu metode yang dapat digunakan untuk mengevaluasi kinerja. Metode ini bertujuan untuk menghitung jumlah kesalahan klasifikasi pada data yang diuji. Semakin kecil nilai yang diperoleh maka kinerja klasifikasi semakin baik dan sebaliknya. Persamaan untuk menghitung kerugian *Haming* dapat dilihat pada Persamaan (3.1).

$$HammingLoss(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{q} \mid h(x_i)\Delta y_i \mid + \tag{3.1}$$

Dimana p adalah jumlah data yang akan digunakan dalam klasifikasi, Q adalah jumlah label kelas yang ada, dan $|h(x)\triangle Y|$ adalah jumlah kesalahan yang terjadi pada saat klasifikasi.

Accuracy digunakan untuk mengukur sejauh mana model dapat mengklasifikasikan dengan benar seluruh label pada data uji. Persamaan untuk menghitung *Accuracy* dapat dilihat pada Persamaan (3.2).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.2)$$

Dimana persamaan (3.2) sebagai berikut :

- TP : Jumlah kasus positif yang diprediksi dengan benar oleh model.
- TN : Jumlah kasus negatif yang diprediksi dengan benar oleh model
- FP : Jumlah kasus negatif yang salah diprediksi sebagai positif oleh model.
- FN : Jumlah kasus positif yang salah diprediksi sebagai negatif oleh model.

Dalam persamaan *accuracy* TP + TN merupakan jumlah total prediksi yang benar yang terdiri dari prediksi positif yang benar (TP) dan prediksi negatif yang benar (TN). TP + TN + FP + FN merupakan total jumlah data yang diuji oleh model.

Precision mendeskripsikan dalam mengukur sejauh mana prediksi positif yang dilakukan oleh model dengan benar. Persamaan untuk menghitung *Precision* dapat dilihat pada Persamaan (3.3).

$$Precision = \frac{TP}{TP+FP} \quad (3.3)$$

Dimana :

- TP : Jumlah contoh positif yang benar-benar diklasifikasikan dengan benar oleh model.

- FP : Jumlah contoh negatif yang salah diklasifikasikan sebagai positif oleh model.

Recall menentukan kapasitas model untuk mendeteksi atau mengidentifikasi label yang sesuai dalam kumpulan data. Persamaan untuk menghitung *Recall* dapat dilihat pada Persamaan (3.4).

$$Recall = \frac{TP}{TP+FN} \tag{3.4}$$

Dimana :

- TP : Jumlah instance positif yang secara tepat diprediksi sebagai positif oleh model.
- FN : Jumlah instance positif yang seharusnya diprediksi sebagai positif, namun salah prediksi sebagai negatif oleh model.

Pengujian terakhir adalah pengujian *hyperparameter* untuk mencari parameter yang optimal agar kinerja tuning pada model bisa maksimal. Parameter yang diuji adalah *learning rate*, *batch size*, dan jumlah *epoch*.

3.6.3. *Hyperparameter Tuning*

Penerapan *hyperparameter* dilakukan untuk menguji model yang diusulkan agar mendapatkan hasil yang optimal atau akurasi terbaik. Berikut adalah skenario pengujiannya :

1. Melakukan *tuning* pada *epoch*

Epoch adalah *hyperparameter* yang menentukan berapa kali algoritma pembelajaran mendalam berjalan di seluruh kumpulan data baik secara *forward* maupun *backward*.

Dalam istilah yang lebih sederhana, Suatu *Epoch* dicapai ketika semua batch berhasil disebarkan satu kali melalui jaringan saraf. Dalam contoh kasus di atas, 1 *epoch* dicapai ketika 10 batch sampel data pelatihan telah diproses. Dalam penelitian ini penerapan *epoch* menggunakan kisaran 10 – 100 *epoch*.

2. Melakukan *tuning* pada *batch size*

   Batch size adalah jumlah sampel data yang biasanya melewati jaringan saraf pada waktu yang sama. *batch size* menentukan berapa banyak sampel yang perlu diproses sebelum memperbarui parameter internal model. Bayangkan *batch size* sebagai iterasi *for-loop* melalui satu atau lebih sampel dan membuat prediksi. Di akhir *batch size*, hasil prediksi dibandingkan dengan variabel keluaran sebenarnya untuk menghitung nilai kesalahan (kerugian). Dari kesalahan ini, algoritma diperbarui untuk memperbaiki model. Sederhananya, *batch size* adalah jumlah sampel pelatihan yang digunakan dalam satu *batch* untuk satu iterasi. Pengujian penelitian *Batch size* dikisaran 16 – 128.

3. Melakukan *tuning* pada *learning rate*

   Skenario ketiga melibatkan penyesuaian *learning rate*. Khususnya dengan menentukan *learning rate* pada *optimizer*. *Learning rate* dalam penelitian ini kisaran 0.00001 (1e-5) sampai dengan 0.1 (1e-1).

3.7. Analisis Hasil dan Kesimpulan

Tahap analisis hasil merupakan proses menganalisis hasil pengujian. Sedangkan penarikan kesimpulan merupakan proses akhir dari kegiatan analisis. Kesimpulan menggambarkan hasil dan temuan yang telah dikumpulkan selama penelitian. Kesimpulan akan membahas dan menjawab rumusan masalah yang telah dikemukakan dalam penelitian ini.

BAB IV

HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan hasil penelitian dan analisis yang diperoleh selama penelitian klasifikasi terjemahan Al-Qur'an menggunakan varian dari algoritma *Long Short-Term Memory* (LSTM).

4.1. Data Terjemahan Al-Quran

Data terjemahan Al-Qur'an yang digunakan dalam penelitian ini berasal dari *database* Kementerian Agama Republik Indonesia pada tahun 2022 berbahasa Indonesia yang dapat diakses pada situs https://quran.kemenag.go.id.

Tabel 4.1 Dataset Terjemahan Al-Qur'an

Surah	Ayat	Terjemahan
4	1	Hai sekalian manusia, bertakwalah kepada Tuhan-mu yang telah menciptakan kamu dari seorang diri, dan dari padanya Allah menciptakan isterinya; dan dari pada keduanya Allah memperkembang biakkan laki-laki dan perempuan yang banyak. Dan bertakwalah kepada Allah yang dengan (mempergunakan) nama-Nya kamu saling meminta satu sama lain, dan (peliharalah) hubungan silaturrahim. Sesungguhnya Allah selalu menjaga dan mengawasi kamu.
4	2	Dan berikanlah kepada anak-anak yatim (yang sudah balig) harta mereka, jangan kamu menukar yang baik dengan yang buruk dan jangan kamu makan harta mereka bersama hartamu. Sesungguhnya tindakan-tindakan (menukar dan memakan) itu, adalah dosa yang besar.
4	3	Dan jika kamu takut tidak akan dapat berlaku adil terhadap (hak-hak) perempuan yang yatim (bilamana kamu mengawininya), maka kawinilah wanita-wanita (lain) yang kamu senangi: dua, tiga atau empat. Kemudian jika kamu takut tidak akan dapat berlaku adil, maka (kawinilah) seorang saja, atau budak-budak yang kamu miliki. Yang demikian itu adalah lebih dekat kepada tidak berbuat

Surah	Ayat	Terjemahan
		aniaya.
4	4	Berikanlah maskawin (mahar) kepada wanita (yang kamu nikahi) sebagai pemberian dengan penuh kerelaan. Kemudian jika mereka menyerahkan kepada kamu sebagian dari maskawin itu dengan senang hati, maka makanlah (ambillah) pemberian itu (sebagai makanan) yang sedap lagi baik akibatnya.
4	5	Dan janganlah kamu serahkan kepada orang-orang yang belum sempurna akalnya, harta (mereka yang ada dalam kekuasaanmu) yang dijadikan Allah sebagai pokok kehidupan. Berilah mereka belanja dan pakaian (dari hasil harta itu) dan ucapkanlah kepada mereka kata-kata yang baik.
4	6	Dan ujilah anak yatim itu sampai mereka cukup umur untuk kawin. Kemudian jika menurut pendapatmu mereka telah cerdas (pandai memelihara harta), maka serahkanlah kepada mereka harta-hartanya. Dan janganlah kamu makan harta anak yatim lebih dari batas kepatutan dan (janganlah kamu) tergesa-gesa (membelanjakannya) sebelum mereka dewasa. Barang siapa (di antara pemelihara itu) mampu, maka hendaklah ia menahan diri (dari memakan harta anak yatim itu) dan barangsiapa yang miskin, maka bolehlah ia makan harta itu menurut yang patut. Kemudian apabila kamu menyerahkan harta kepada mereka, maka hendaklah kamu adakan saksi-saksi (tentang penyerahan itu) bagi mereka. Dan cukuplah Allah sebagai Pengawas (atas persaksian itu).
…	…	…
6	165	Dan Dialah yang menjadikan kamu penguasa-penguasa di bumi dan Dia meninggikan sebahagian kamu atas sebahagian (yang lain) beberapa derajat, untuk mengujimu tentang apa yang diberikan-Nya kepadamu. Sesungguhnya Tuhanmu amat cepat siksaan-Nya dan sesungguhnya Dia Maha Pengampun lagi Maha Penyayang.
6	162	Katakanlah: sesungguhnya sembahyangku, ibadatku, hidupku dan matiku hanyalah untuk Allah, Tuhan semesta alam.
6	163	Tiada sekutu bagi-Nya; dan demikian itulah yang diperintahkan kepadaku dan aku adalah orang yang pertama-tama menyerahkan diri (kepada Allah)".
6	164	Katakanlah: "Apakah aku akan mencari Tuhan selain Allah, padahal Dia adalah Tuhan bagi segala sesuatu. Dan tidaklah seorang membuat dosa melainkan kemudharatannya kembali

Surah	Ayat	Terjemahan
		kepada dirinya sendiri; dan seorang yang berdosa tidak akan memikul dosa orang lain. Kemudian kepada Tuhanmulah kamu kembali, dan akan diberitakan-Nya kepadamu apa yang kamu perselisihkan".
6	165	Dan Dialah yang menjadikan kamu penguasa-penguasa di bumi dan Dia meninggikan sebahagian kamu atas sebahagian (yang lain) beberapa derajat, untuk mengujimu tentang apa yang diberikan-Nya kepadamu. Sesungguhnya Tuhanmu amat cepat siksaan-Nya dan sesungguhnya Dia Maha Pengampun lagi Maha Penyayang.

Tabel 4.1 menunjukkan data terjemahan Al-Qur'an yang digunakan dalam penelitian ini, dengan memberikan informasi surah, ayat dan teks terjemahan berbahasa Indonesia. Total dataset terjemahan Al-Qur'an sebanyak 6236 ayat. Jumlah data yang diperoleh dan digunakan dalam penelitian ini dari juz 4 - 8 diantaranya, surah An-Nisa', surah Al-Maidah dan surah Al-An'am dengan total 461 teks terjemahan ayat Al-Qur'an.

4.2. Pemberian label *dataset*

Proses pemberian label dilakukan secara manual yang dilakukan oleh pakar ahli dengan pemberian label Tauhid, Ibadah, Akhlaq, dan Sejarah pada setiap ayat terjemahan Al-Qur'an. Tabel 4.2 merupakan hasil pelabelan yang telah dilakukan.

Tabel 4.2 Hasil pelabelan *dataset*

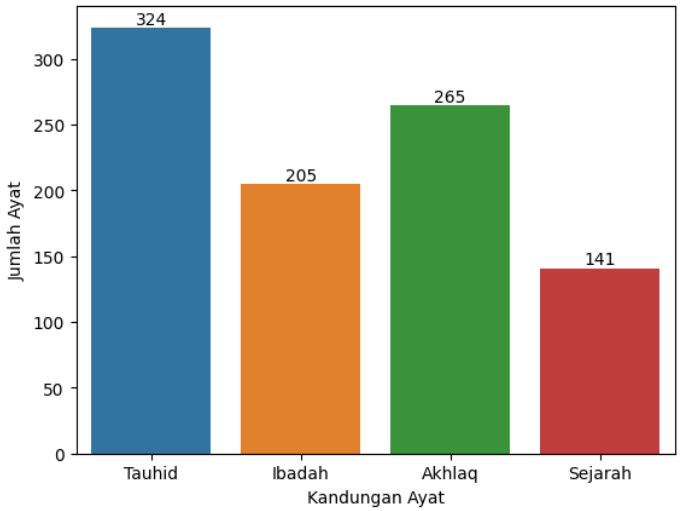
No	Juz	Surah	Terjemahan	Tauhid	Ibadah	Akhlaq	Sejarah
1	6	5	Allah berfirman: "(Jika demikian), maka sesungguhnya negeri itu diharamkan atas mereka selama empat puluh tahun, (selama itu) mereka akan berputar-putar kebingungan di bumi (padang Tiih) itu. Maka janganlah kamu bersedih hati	0	0	1	1

No	Juz	Surah	Terjemahan	Tauhid	Ibadah	Akhlaq	Sejarah
			(memikirkan nasib) orang-orang yang fasik itu".				
2	6	5	Ceritakanlah kepada mereka kisah kedua putera Adam (Habil dan Qabil) menurut yang sebenarnya, ketika keduanya mempersembahkan korban, maka diterima dari salah seorang dari mereka berdua (Habil) dan tidak diterima dari yang lain (Qabil). Ia berkata (Qabil): "Aku pasti membunuhmu!". Berkata Habil: "Sesungguhnya Allah hanya menerima (korban) dari orang-orang yang bertakwa".	1	1	1	1
3	6	5	Sungguh kalau kamu menggerakkan tanganmu kepadaku untuk membunuhku, aku sekali-kali tidak akan menggerakkan tanganku kepadamu untuk membunuhmu. Sesungguhnya aku takut kepada Allah, Tuhan seru sekalian alam.	1	0	1	1
4	6	5	Sesungguhnya aku ingin agar kamu kembali dengan (membawa) dosa (membunuh)ku dan dosamu sendiri, maka kamu akan menjadi penghuni neraka, dan yang demikian itulah pembalasan bagi orang-orang yang zalim.	1	0	0	1
5	6	5	Maka hawa nafsu Qabil menjadikannya menganggap mudah membunuh saudaranya, sebab itu dibunuhnyalah, maka jadilah ia seorang diantara orang-orang yang merugi.	0	0	1	1
	…	…	…	…	…	…	…
458	8	6	Sesungguhnya orang-orang yang memecah belah agama-	1	0	1	0

No	Juz	Surah	Terjemahan	Tauhid	Ibadah	Akhlaq	Sejarah
			Nya dan mereka menjadi bergolongan, tidak ada sedikitpun tanggung jawabmu kepada mereka. Sesungguhnya urusan mereka hanyalah terserah kepada Allah, kemudian Allah akan memberitahukan kepada mereka apa yang telah mereka perbuat.				
459	8	6	Barangsiapa membawa amal yang baik, maka baginya (pahala) sepuluh kali lipat amalnya; dan barangsiapa yang membawa perbuatan jahat maka dia tidak diberi pembalasan melainkan seimbang dengan kejahatannya, sedang mereka sedikitpun tidak dianiaya (dirugikan).	1	1	0	0
460	8	6	Katakanlah: "Sesungguhnya aku telah ditunjuki oleh Tuhanku kepada jalan yang lurus, (yaitu) agama yang benar, agama Ibrahim yang lurus, dan Ibrahim itu bukanlah termasuk orang-orang musyrik".	1	0	1	1
461	8	6	Katakanlah: sesungguhnya sembahyangku, ibadatku, hidupku dan matiku hanyalah untuk Allah, Tuhan semesta alam.	1	1	1	0

Dari hasil pelabelan yang berada di tabel 4.2, terdapat informasi yakni nama label Tauhid, Ibadah, Akhlaq, dan Sejarah. Pada tiap label diberi nilai 1 jika terjemahan tersebut termasuk kategori pada label, dan nilai 0 jika terjemahan tidak termasuk label. Sehingga distribusi label dapat ditinjau pada Gambar 4.1 dengan jumlah label Tauhid sebanyak 324 label, Ibadah 205 label, Akhlaq 265 dan

Sejarah 141 label. Sehingga tabel 4.2 diatas, dapat diuraikan bahwa juz 6 dalam surah ke-5 baris 1 mempunyai klasifikasi terjemahan akhlaq dan Sejarah, dikarenakan pakar memberikan label pada kolom Akhlaq dan Sejarah. Begitu juga pada baris ke tiga pakar memberikan klasifikasi terjemahan ayat tersebut masuk dalam klasifikasi Tauhid, Akhlaq dan Sejarah.



Gambar 4.1 Jumlah distribusi label setelah pelabelan.

4.3. *Text Preprocessing*

Data yang digunakan dalam penelitian ini mewakili 461 data terjemahan Al-Qur'an bahasa Indonesia. Pembersihan data dilakukan dengan menghapus beberapa pesan duplikat, simbol, tanda baca, spasi, angka, karakter khusus, yang disebut *Cleansing*. Selanjutnya proses *case folding* untuk mengubah seluruh huruf menjadi huruf kecil sehingga kata yang sama namun ditulis dengan huruf berbeda tidak menjadi data duplikat. Kemudian memecah sebuah kalimat, paragraf atau dokumen menjadi bagian-bagian yang disebut token atau bahkan disebut

Tokenization. Proses *preprocessing* menghasilkan data terjemahan Al-Qur'an yang bersih, yang kemudian digunakan sebagai data pelatihan dan pengujian.

Tabel 4.3 Kode program *preprocessing text*.

Preprocessing Text	
1	`def text_cleaning(text):`
2	`  text = text.lower()`
3	`  text = re.sub(r'[-+]?[0-9]+', '', text)`
4	`  text = re.sub(r'[^\w\s]','', text)`
5	`  text = re.sub(r'\s+', ' ', text).strip()`
6	
7	`  return text`
8	

Penjelasan potongan kode program pada Tabel 4.3 diuraikan sebagai berikut:

Baris 1 Mendefinisikan nama fungsi *text_cleaning*, digunakan untuk membersihkan teks atau dokumen teks dengan serangkaian langkah pemrosesan.

Baris 3-8 Tahapan yang dilakukan oleh fungsi text_*cleaning*. *Preprocessing* yang dilakukan mengubah semua karakter dalam teks menjadi huruf kecil (*lowercase*), menghapus semua angka (termasuk angka negatif dan bilangan bulat) dari teks, membersihkan teks dari tanda baca yang mungkin tidak relevan dalam analisis teks dan menggantikan dua atau lebih spasi berturut-turut dengan satu spasi tunggal dan menghapus spasi yang berlebihan di awal dan akhir teks.

Tabel 4.4 menyajikan contoh hasil implementasi *text preprocessing*. Pada penelitian ini, tahapan *stemming*, menghapus kata sambung tidak dilakukan untuk menjaga konteks setiap terjemahan Al-Quran.

Tabel 4.4 Hasil *preprocessing text*.

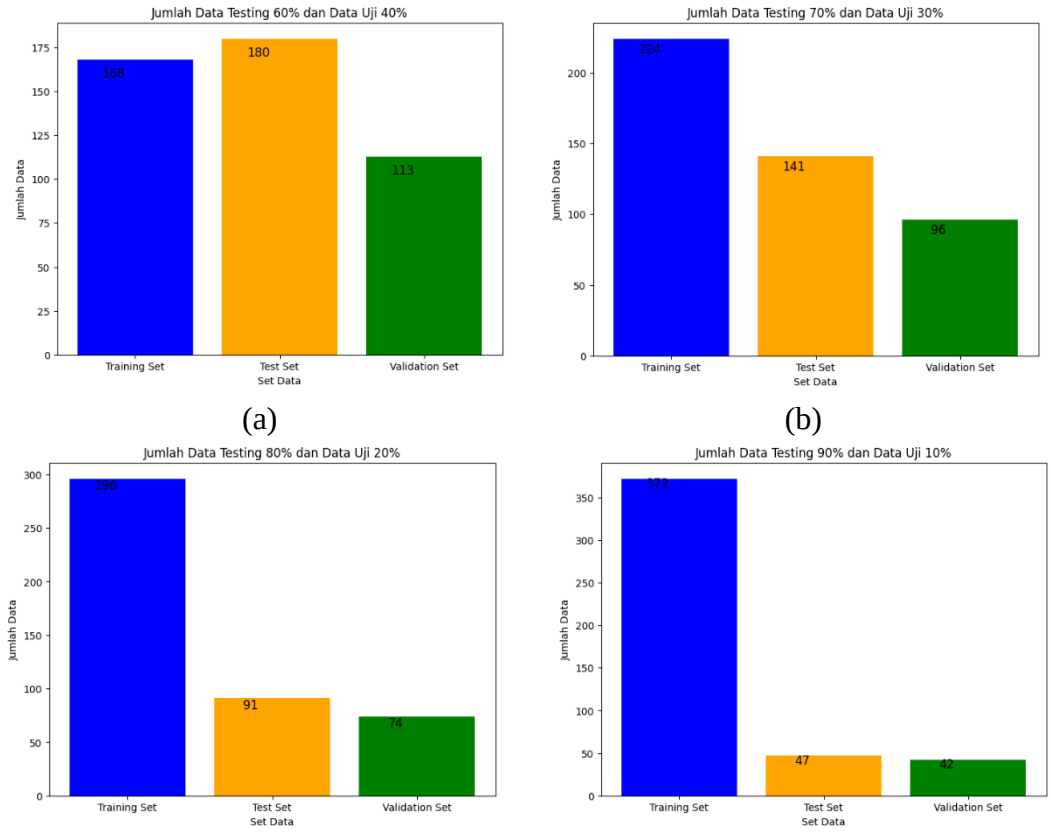
Sebelum *Text Preprocessing*	Sesudah *Text Preprocessing*
Hai sekalian manusia, bertakwalah kepada Tuhan-mu yang telah menciptakan kamu dari seorang diri,	hai sekalian manusia bertakwalah kepada tuhanmu yang telah menciptakan kamu dari seorang diri dan

dan dari padanya Allah menciptakan isterinya; dan dari pada keduanya Allah memperkembang biakkan laki-laki dan perempuan yang banyak. Dan bertakwalah kepada Allah yang dengan (mempergunakan) nama-Nya kamu saling meminta satu sama lain, dan (peliharalah) hubungan silaturrahim. Sesungguhnya Allah selalu menjaga dan mengawasi kamu.	dari padanya allah menciptakan isterinya dan dari pada keduanya allah memperkembang biakkan lakilaki dan perempuan yang banyak dan bertakwalah kepada allah yang dengan mempergunakan namanya kamu saling meminta satu sama lain dan peliharalah hubungan silaturrahim sesungguhnya allah selalu menjaga dan mengawasi kamu
Berikanlah maskawin (mahar) kepada wanita (yang kamu nikahi) sebagai pemberian dengan penuh kerelaan. Kemudian jika mereka menyerahkan kepada kamu sebagian dari maskawin itu dengan senang hati, maka makanlah (ambillah) pemberian itu (sebagai makanan) yang sedap lagi baik akibatnya.	berikanlah maskawin mahar kepada wanita yang kamu nikahi sebagai pemberian dengan penuh kerelaan kemudian jika mereka menyerahkan kepada kamu sebagian dari maskawin itu dengan senang hati maka makanlah ambillah pemberian itu sebagai makanan yang sedap lagi baik akibatnya
Dan mereka (orang-orang munafik) mengatakan: "(Kewajiban kami hanyalah) taat". Tetapi apabila mereka telah pergi dari sisimu, sebahagian dari mereka mengatur siasat di malam hari (mengambil keputusan) lain dari yang telah mereka katakan tadi. Allah menulis siasat yang mereka atur di malam hari itu, maka berpalinglah kamu dari mereka dan tawakallah kepada Allah. Cukuplah Allah menjadi Pelindung.	dan mereka orangorang munafik mengatakan kewajiban kami hanyalah taat tetapi apabila mereka telah pergi dari sisimu sebahagian dari mereka mengatur siasat di malam hari mengambil keputusan lain dari yang telah mereka katakan tadi allah menulis siasat yang mereka atur di malam hari itu maka berpalinglah kamu dari mereka dan tawakallah kepada allah cukuplah allah menjadi pelindung

4.4. Pembagian Data latih dan Data Uji

Memisahkan data latih dan data uji pada klasifikasi terjemahan Al - Quran bahasa Indonesia sangat penting untuk menguji performa model secara akurat. Biasanya, pembagian data ini memungkinkan model untuk belajar dari data yang

ada (data latih), selanjutnya data latih dilakukan evaluasi model selama proses pelatihan dan penyetelan parameter untuk menghindari *overfitting* (data validasi). dan kemudian menguji data yang belum pernah dilihat sebelumnya (data uji), sehingga memungkinkan mereka mengevaluasi performa model pada data baru. Pada penelitian ini melakukan pengujian model dengan beberapa variasi data latih dan data uji pada ukuran 60% : 40%, 70% : 30%, 80% : 20% dan 90% : 10%.



Gambar 4.2 Visualisasi Pembagian Data Latih dan Uji.

Gambar 4.2, terdapat empat skenario berbagi data untuk melatih, memvalidasi, dan menguji model klasifikasi. Pada skenario pertama (a), terdapat pemisahan 60% data untuk pelatihan dan 40% untuk pengujian. Jumlah data yang

tersedia adalah 168 data untuk pelatihan, 113 data untuk validasi, dan 180 data untuk pengujian. Pada skenario kedua (b), pembagian datanya adalah 70% untuk pelatihan dan 30% untuk pengujian. Jumlah data yang digunakan adalah 224 data untuk pelatihan, 96 data untuk validasi, dan 141 data untuk pengujian. Skenario ketiga (c) membagi 80% untuk pelatihan dan 20% untuk pengujian. Jumlah data yang tersedia adalah 296 data untuk pelatihan, 74 data untuk validasi, dan 91 data untuk pengujian. Terakhir pada skenario keempat (d), pembagian datanya adalah 90% untuk pelatihan dan 10% untuk pengujian. Jumlah data yang digunakan adalah 372 data untuk pelatihan, 42 data untuk validasi, dan 47 data untuk pengujian.

Dalam setiap skenario, penggunaan data untuk pelatihan, validasi, dan pengujian terjadi pada tingkat yang berbeda. Rasio ini mempengaruhi jumlah data yang digunakan pada setiap langkah pembuatan model, seperti melatih model pada data pelatihan, menetapkan parameter pada data validasi, dan menguji performa model pada data pengujian.

4.5. Implementasi Mode Dasar *(Baseline Model)*

Subbab ini menjelaskan implementasi yang dilakukan pada perancangan model dasar. Model dasar ini dilakukan sebagai hasil pembanding dengan model usulan pada penelitian ini. Berdasarkan Gambar 3.2 Halaman 35 terdapat beberapa model dasar yang digunakan pada penelitian ini menggunakan algoritma *multinomial naïve bayes*, *decision tree*, dan *support vector machine*. Dan varian algoritma *long short-term memory* yang tidak menggunakan *word embedding*.

4.5.1. Model dasar tahapan pertama

Tahapan pertama model dasar mengimplementasikan algoritma *Multinomial Naïve Bayes*, *Decision Tree*, dan *Support Vector Machine*. Pada tahapan ini jumlah pembagian data dibagi menjadi data *training* dan data *testing* dengan memiliki variasi 60% : 40%, 70% : 30%, 80% : 20% dan 90% : 10%.

Tabel 4.5 Kode program pembagian data model dasar.

Split data training dan testing	
1	`X = df['clean_terjemahan']`
2	`y = df[['Tauhid', 'Ibadah', 'Akhlaq', 'Sejarah']]`
3	
4	`TEST_SIZE = 0.3`
5	`X_train, X_test, y_train, y_test =`
6	`multilabel_train_test_split(X, y, stratify=y,`
7	`test_size=TEST_SIZE, shuffle=True)`
8	`X_train.shape, X_test.shape, y_train.shape, y_test.shape`

Penjelasan potongan kode program pada Tabel 4.5 diuraikan sebagai berikut:

Baris 1-2 Mendefinisikan dua variabel dengan nama X dan Y. Variabel X berisi kolom "clean_terjemahan" dari DataFrame df. Ini adalah teks yang akan digunakan sebagai fitur untuk pelatihan dan pengujian model. Variabel y berisi empat kolom kategorikal dari DataFrame df, yaitu "Tauhid," "Ibadah," "Akhlaq," dan "Sejarah." Ini adalah label yang akan digunakan untuk melakukan prediksi.

Baris 4 Menentukan nilai konstanta TEST_SIZE yang mengindikasikan seberapa besar proporsi data yang akan dijadikan data pengujian.

Baris 5-8 Fungsi multilabel_train_test_split digunakan untuk membagi data menjadi dua set utama. X_train dan y_train adalah set data pelatihan yang akan digunakan untuk melatih model. X_test dan y_test adalah set data pengujian yang akan digunakan untuk menguji kinerja model.

Setelah memperoleh jumlah data *training* dan data *testing*, selanjutnya dilakukan mengubah teks menjadi representasi vektor numerik yang

menggambarkan bobot relatif kata-kata dalam dokumen dengan menggungkan TF-IDF (*Term Frequency-Inverse Document Frequency*).

Tabel 4.6 Kode program TF-IDF *Vectorization*

Proses TF-IDF	
1	`df_kd = pd.read_csv('data-kata-baseline/kamus-kata-`
2	`dasar.csv')`
3	`stop_words_id = df_kd['kata_dasar'].tolist()`
4	
5	`vectorizer = TfidfVectorizer(max_features=1000,`
6	`stop_words=stop_words_id)`
7	`X_train_tfidf = vectorizer.fit_transform(X_train)`
8	`X_test_tfidf = vectorizer.transform(X_test)`

Penjelasan potongan kode program pada Tabel 4.6 diuraikan sebagai berikut:

Baris 1 Membaca sebuah file CSV yang berisi daftar kata dasar bahasa Indonesia (kamus kata dasar). Hasil pembacaan file CSV disimpan dalam DataFrame dengan nama **df_kd**.

Baris 3 Mengambil kolom "kata_dasar" dari DataFrame df_kd dan mengonversinya ke dalam bentuk variabel dengan nama stop_words_id. Daftar ini akan berisi kata-kata dasar bahasa Indonesia yang akan dianggap sebagai *stop words*.

Baris 5 Membangun objek TfidfVectorizer yang akan digunakan untuk mengubah teks ke dalam representasi TF-IDF. Parameter max_features diatur ke 1000, yang berarti hanya 1000 kata teratas dengan nilai TF-IDF tertinggi yang akan diambil sebagai fitur. Parameter stop_words diatur dengan stop_words_id, yang mengidentifikasi kata-kata yang harus diabaikan dalam perhitungan TF-IDF.

Baris 7-8 Data pelatihan X_train (yang berisi teks) diubah menjadi representasi TF-IDF menggunakan vektorizer yang telah dibangun sebelumnya. Hasilnya disimpan dalam X_train_tfidf, yang akan digunakan dalam pelatihan model. Dan data pengujian X_test juga diubah menjadi representasi TF-IDF menggunakan vektorizer yang

sama. Hasilnya disimpan dalam X_test_tfidf, yang akan digunakan untuk menguji model.

Proses selanjutnya pelatihan menggunakan algoritma model dasar, yang dipaparkan pada Tabel 4.7 berikut ini

Tabel 4.7 Kode program Inisialisasi model dasar

Pelatihan model dasar	
1	`# Multinomial Naive Bayes`
2	`nb_classifier = MultiOutputClassifier(MultinomialNB())`
3	`nb_classifier.fit(X_train_tfidf, y_train)`
4	`nb_predictions = nb_classifier.predict(X_test_tfidf)`
5	
6	`# Decision Tree`
7	`dt_classifier = DecisionTreeClassifier()`
8	`dt_classifier.fit(X_train_tfidf, y_train)`
9	`dt_predictions = dt_classifier.predict(X_test_tfidf)`
10	
11	`# Support Vector Machine`
12	`svm_classifier = OneVsRestClassifier(SVC())`
13	`svm_classifier.fit(X_train_tfidf, y_train)`
14	`svm_predictions = svm_classifier.predict(X_test_tfidf)`

Penjelasan potongan kode program pada Tabel 4.7 diuraikan sebagai berikut:

Baris 1-4 Merupakan pelatihan model menggunakan *multinomial naïve bayes*. Baris 2 - *MultiOutputClassifier* digunakan untuk menangani *multi-label classification*, yang berarti satu sampel teks dapat terkait dengan beberapa label. Baris 3 - melatih model *Naive Bayes* pada data latih yang telah di- TF-IDF sebelumya. Di mana X_test_tfidf adalah data teks yang telah di- TF-IDF. Hasil prediksi disimpan dalam variabel nb_predictions.

Baris 6-9 dt_classifier merupakan inisiasi model *Decision Tree*, yang akan digunakan untuk klasifikasi multi-label. Model ini akan mencoba membangun pohon keputusan untuk memisahkan sampel-sampel ke dalam kategori-kategori label yang sesuai. Hasil prediksi disimpan dalam variabel dt_predictions.

Baris 11-14 svm_classifier merupakan inisiasi model *Support Vector Machine* (SVM) dengan algoritma *One-Versus-Rest*. Model ini memungkinkan SVM digunakan dalam konteks *multi-label classification*. Algoritma SVM berusaha untuk menemukan hiperplane terbaik yang memisahkan sampel-sampel dalam beberapa kategori label. Hasil prediksi disimpan dalam variabel svm_predictions

Langkah selanjutnya mengevaluasi model dasar untuk menentukan nilai akurasi, *precision*, *recall* dan *hamming loss*.

Tabel 4.8 Kode program evaluasi model

Evaluasi model dasar	
1	`def evaluate_model(predictions, true_labels):`
2	`    accuracy = accuracy_score(true_labels, predictions)`
3	`    precision = precision_score(true_labels,`
4	`     predictions, average='micro')`
5	`    recall = recall_score(true_labels, predictions,`
6	`     average='micro')`
7	`    hamming = hamming_loss(true_labels, predictions)`
8	`    f1 = f1_score(true_labels, predictions,`
9	`     average='micro')`
10	`    cm = multilabel_confusion_matrix(true_labels,`
11	`     predictions)`
12	`    return accuracy, precision, recall, hamming, f1, cm`
13	
14	`nb_accuracy, nb_precision, nb_recall, nb_hamming, nb_f1,`
15	`  nb_cm = evaluate_model(nb_predictions, y_test)`
16	`dt_accuracy, dt_precision, dt_recall, dt_hamming, dt_f1,`
17	`  dt_cm = evaluate_model(dt_predictions, y_test)`
18	`svm_accuracy, svm_precision, svm_recall, svm_hamming,`
19	`  svm_f1, svm_cm = evaluate_model(svm_predictions,`
	`  y_test)`

Dari kode program evaluasi model yang dipaparkan pada Tabel 4.8, menghasillkan nilai evaluasi yang dijelaskan pada Tabel 4.9. Dengan nilai akurasi

tertinggi dari tiga model dasar tahap pertama didapatkan algoritma *naïve bayes* memiliki akurasi tertinggi, dan *hamming loss* terendah.

Tabel 4.9 Hasil evaluasi model dasar

Algoritma	**Evaluasi Model**			
	Accuracy	***Precision***	***Recall***	***Hamming Loss***
Naïve Bayes	0,21	0,75	0,69	0,32
Decision Tree	0,19	0,62	0,56	0,40
SVM	0,19	0,68	0,66	0,33

4.5.2. Model dasar tahap kedua

Berdasarkan Gambar 3.2 Halaman 35 tahapan kedua model dasar mengimplementasikan algoritma *Long Short-Term Memory* (LSTM) dan *Bidirectional Long Short-Term Memory* (Bi-LSTM). Pada tahapan ini merupakan pengujian klasifikasi terjemahan Al-Quran yang menggunakan model LSTM akan tetapi pada model ini tidak mengimplementasikan *word embedding*. Pada tahapan ini dengan pengujian berbasis LSTM langkah awal dilakukan dengan *preprocessing* seperti yang dilakukan pada model dasar tahap pertama. Langkah selanjutnya adalah menentukan panjang maksimal rangkaian token kata yang dapat diakomodasi oleh model dengan menghitung distribusi panjang token kata untuk setiap kalimat dalam data, seperti yang ditunjukkan pada Tabel 4.10.

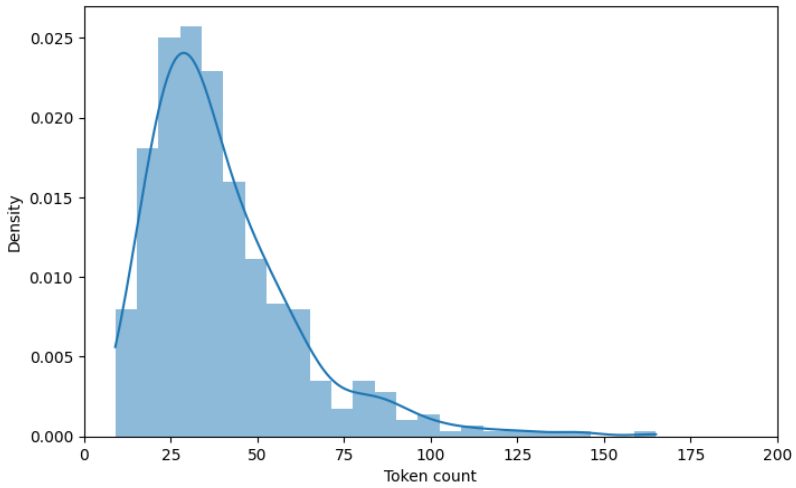
Tabel 4.10 Kode program implementasi menentukan panjang token kata

Maximum Length of the Sequence

```
1  token_lens = []
2
3  for text in df['clean_terjemahan']:
4      tokens = text.split()
5      token_lens.append(len(tokens))
```

```
6
7   plt.figure(figsize=(8, 5))
8   sns.histplot(token_lens, kde=True, stat='density',
9   linewidth=0)
10  plt.xlim([0, 200]);
11  plt.xlabel('Token count');
```

Hasil implementasi kode program pada Tabel 4.10 adalah histogram yang menunjukkan distribusi jumlah token per kalimat di seluruh data, seperti yang ditunjukkan pada Gambar 4.3. Dari Gambar 4.3, panjang token terbesar sekitar 20 hingga 50 token, sedangkan panjang token terkecil sekitar 125 hingga 150 token. Oleh karena itu, nilai max_seq_length pada model dasar tahap kedua ditentukan sebesar 150.



Gambar 4.3 Distribusi token kata dalam data.

untuk menentukan panjang maksimum string token kata.

Proses selanjutnya adalah menyiapkan lapisan *embedding* untuk mengubah data teks menjadi format masukan yang sesuai dengan model klasifikasi seperti terlihat pada kode program Tabel 4.11.

Tabel 4.11 Kode program Embedding Layer

	Preparing Data for Embedding Layer
1	`NUM_WORDS = 10000`
2	`OOV_TOKEN = '<unk>'`
3	`tokenizer = Tokenizer(num_words=NUM_WORDS,`
4	`oov_token=OOV_TOKEN, lower=True, split=' ')`
5	`tokenizer.fit_on_texts(df['clean_terjemahan'])`
6	
7	`tokenizer.word_index['<pad>'] = 0`
8	`tokenizer.index_word[0] = '<pad>'`
9	
10	`WV_DICTIONARY = tokenizer.word_index`
11	`WV_DICTIONARY_SIZE = len(WV_DICTIONARY)`
12	
13	`data_seq =`
14	`  tokenizer.texts_to_sequences(df['clean_terjemahan'])`
15	
16	`PADDING = 'post'`
17	`TRUNCATING = 'post'`
18	`data_pad = pad_sequences(data_seq,`
19	`   maxlen=MAX_SEQ_LENGTH, padding=PADDING,`
20	`   truncating=TRUNCATING)`

Penjelasan potongan kode program pada Tabel 4.11 diuraikan sebagai berikut:

Baris 1 Inisialisasi variabel num_words untuk menentukan frekuensi kemunculan kata di seluruh data. Hanya $num\_words - 1$ yang akan disimpan dalam kamus kosakata. Sedangkan kata yang muncul $\leq num\_words$ tidak digunakan.

Baris 2 Inisialisasi variabel oov_token untuk menambahkan token jika kata tersebut tidak ada dalam kamus kosakata

Baris 3-8 Tentukan fungsi $tokenizer$ untuk melakukan tokenisasi tingkat kata pada semua data. Fungsi ini menerima parameter masukan berupa variabel num_words dan oov_token. Selain itu, selama tokenisasi, semua teks akan diubah menjadi huruf kecil.

Baris 10-14 Atur variabel $wv\_dictionary$ untuk menyimpan kamus kosakata yang menampilkan pasangan kata dan indeksnya. Tentukan variabel $wv\_dictionary\_size$ untuk menampilkan jumlah kata dalam

kamus dari hasil fungsi $tokenizer$. Mendefinisikan fungsi untuk mengonversi string token kata di setiap baris data menjadi string indeks bilangan bulat berdasarkan indeks kamus leksikal.

Baris 16-20 Tentukan fungsi yang menambahkan token ke string indeks token kata jika jumlahnya kurang dari max_seq_length. Sedangkan jika string indeks token kata melebihi max_seq_length maka dilakukan pemotongan sehingga total panjang string indeks token kata = max_seq_length.

Proses selanjutnya adalah membagi data latih dan data uji dengan perbandingan 70:30. Setelah memperoleh data uji, prosedur selanjutnya adalah membagi data latih dan data validasi dengan perbandingan yang sama. Tabel 4.12 menunjukkan kode program untuk melakukan pembagian data latih, data validasi, dan data uji pada model dasar.

Tabel 4.12 Kode program implementasi pembagian data.

Data Splitting

```
1  TEST_SIZE = 0.3
2
3  X_train, X_test, y_train, y_test =
4    multilabel_train_test_split(X, y, stratify=y,
5    test_size=TEST_SIZE, shuffle=True, random_state=SEED)
6  X_train, X_val, y_train, y_val =
7    multilabel_train_test_split(X_train, y_train,
8    test_size=TEST_SIZE, random_state=SEED)
9
10 X_train.shape, X_test.shape, X_val.shape, y_train.shape,
11   y_test.shape, y_val.shape
```

Setelah melakukan pembagian data, proses selanjutnya adalah mendefinisikan model klasifikasi multi-label untuk terjemahan Al-Quran. Tabel 4.13 menyajikan potongan kode program untuk implementasi arsitektur *Long*

Short-Term Memory (LSTM). Sedangkan Tabel 4.14 potongan kode program untuk arsitektur *Bidirectional Long Short-Term Memory* (Bi-LSTM).

Tabel 4.13 Kode program implementasi arsitektur *LSTM*

Long Short-Term Memory Acrchitecture	
1	`UNITS = 128`
2	
3	`model = Sequential()`
4	`model.add(Embedding(`
5	`    input_dim = WV_DICTIONARY_SIZE,`
6	`    output_dim = 300,`
7	`    input_length = MAX_SEQ_LENGTH,`
8	`  ))`
9	`model.add(LSTM(UNITS*2, dropout=0.1,`
10	`  recurrent_regularizer='l1_l2',`
11	`  return_sequences=False))`
12	`model.add(Dense(UNITS, activation='relu'))`
13	`model.add(Dropout(0.5))`
14	`model.add(Dense(len(LABEL_NAME), activation='sigmoid'))`

Tabel 4.14 Kode program implementasi arsitektur *Bi-LSTM*

Bidirectional Long Short-Term Memory Acrchitecture	
1	`UNITS = 128`
2	
3	`model = Sequential()`
4	`model.add(Embedding(`
5	`    input_dim = WV_DICTIONARY_SIZE,`
6	`    output_dim = 300,`
7	`    input_length = MAX_SEQ_LENGTH,`
8	`  ))`
9	`model.add(Bidirectional(LSTM(UNITS*2, dropout=0.1,`
10	`  recurrent_regularizer='l1_l2',`
11	`  return_sequences=False)))`
12	`model.add(Dense(UNITS, activation='relu'))`
13	`model.add(Dropout(0.5))`
14	`model.add(Dense(len(LABEL_NAME), activation='sigmoid'))`

Penjelasan potongan kode program pada Tabel 4.13 dan Tabel 4.14 diuraikan sebagai berikut:

Baris 1 Variabel awal untuk menentukan ukuran keluaran kelas model yang akan dibangun.

Baris 3 Menentukan arsitektur model awal yang akan dibangun sesuai dengan urutan lapisan yang telah dirancang sebelumnya.

Baris 4-8 Definisi Lapisan *embedding* menerima urutan indeks token dari data dengan panjang max_seq_length. Ukuran masukan lapisan penyematan adalah $wv\_dictionary\_size$. Sedangkan ukuran output dari lapisan *embedding* adalah 300.

Baris 9-11 Definisi lapisan LSTM dan Bi-LSTM dengan memberikan dropout 0.1.

Baris 12-13 Definisi lapisan *fully connected* ke fungsi aktivasi ReLU. Kelas ini juga memberikan *dropout* sebesar 0,5 untuk menghindari *overfitting* dan mempercepat proses latihan.

Baris 14 Tentukan lapisan keluaran menggunakan fungsi aktivasi sigmoid untuk menghitung probabilitas label.

Setelah arsitektur model diimplementasikan, proses selanjutnya adalah melatih model. Tabel 4.15 menyajikan kode program untuk melatih model *Long Short-Term Memory* (LSTM) dan *Bidirectional Long Short-Term Memory* (Bi-LSTM).

Tabel 4.15 Kode program implementasi pelatihan model dasar (2)

Training Model	
1	`EPOCHS = 50`
2	`BATCH_SIZE = 64`
3	`LEARNING_RATE = 1e-4`
4	
5	`OPTIMIZER = Adam(learning_rate=LEARNING_RATE)`
6	`LOSS_FUNCTION =`
7	`  BinaryCrossentropy(name='binary_crossentropy')`
8	
9	`model.compile(loss=LOSS_FUNCTION, optimizer=OPTIMIZER,`
10	`  metrics=METRICS)`
11	

```
12 model_history = model.fit(
13     X_train,
14     y_train,
15     epochs=EPOCHS,
16     batch_size=BATCH_SIZE,
17     validation_data=(X_val, y_val)
18 )
```

Penjelasan potongan kode program pada Tabel 4.15 diuraikan sebagai berikut:

Baris 1-7 Mulai parameter pelatihan model sesuai dengan desain model dasar.

Baris 9-10 Tentukan fungsi yang mengkompilasi arsitektur model yang dibangun dengan parameter pelatihan.

Baris 12-18 Definisi fungsi untuk memulai pelatihan model.

Proses terakhir melibatkan pelaksanaan pengujian model untuk menentukan kinerja model. Tabel 4.16 menunjukkan kode program untuk melakukan pengecekan model dasar.

Tabel 4.16 Kode program implementasi pengujian model dasar (2)

Testing Model

```
1  y_pred = np.round(model.predict(X_test))
2  y_true = y_test
3
4  test_loss, test_accuracy, test_precision, test_recall,
5  test_hamming_loss, test_F1_score =
6  model.evaluate(X_test, y_test)
7
8  cm = multilabel_confusion_matrix(y_true, y_pred)
9  print(classification_report(y_pred, y_true,
10    target_names=LABEL_NAME))
```

Penjelasan potongan kode program pada Tabel 4.16 diuraikan sebagai berikut:

Baris 1-2 Variabel respon keluaran (y) diprediksi oleh model. Tentukan

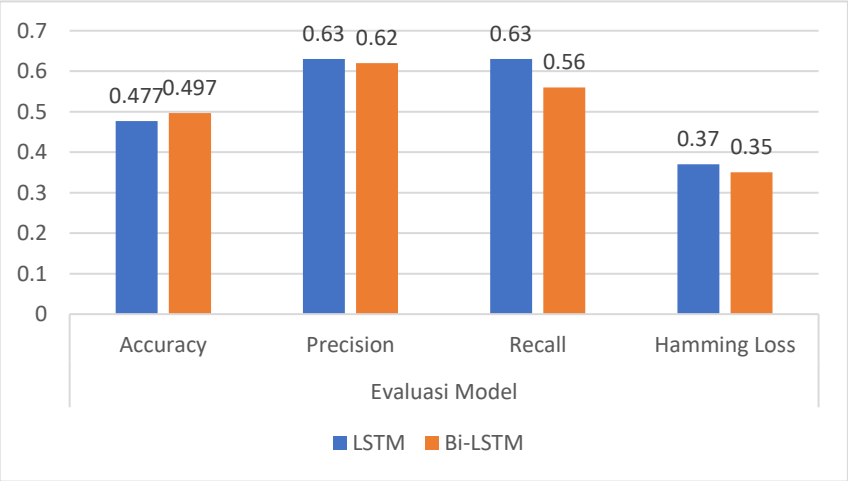
variabel respon sebenarnya (y).

Baris 4-6 Fungsi didefinisikan untuk menentukan kinerja model melalui metrik akurasi, presisi, recall, F1-Score dan hamming_loss serta nilai kerugian model selama pengujian.

Baris 8 Fungsi untuk menampilkan *confusion matrix*.

Baris 9-10 Fungsi untuk menampilkan *classification report*

Berdasarkan hasil pengujian pada model dasar tahap kedua, disampaikan pada Gambar 4.4. Pada Gambar 4.4 Algoritma *Bidirectional Long Short-Term Memory* (Bi-LSTM) lebih unggul dibandingkan algoritma *Long Short-Term Memory* (LSTM).



Gambar 4.4 Hasil evaluasi model dasar tahap kedua.

4.6. Implementasi model *word embedding*

Pada tahapan ini yang digambarkan pada Gambar 3.5 Halaman 49 menerapkan representasi kata-kata sebagai vektor atau disebut juga dengan *word embedding*. Dalam penelitian ini penggunaan *word embedding* diharapkan mampu meningkatkan nilai akurasi dari model dasar yang memiliki akurasi lebih baik dari model dasar lainnya. Dalam model dasar yang sudah dilakukan algoritma

Bidirectional Long Short-Term Memory (Bi-LSTM) memiliki hasil yang terbaik dibandingkan algoritma yang ada pada model dasar. Oleh karena itu implementasi dari *word embedding* diterapkan pada algoritma tersebut.

Pada pembahasan sebelumnya, implementasi *word embedding* pada penelitian ini menggunakan *Word2Vec* dan *FastText* dengan arsitketur *Continous Bag of Word* (CBOW). Penyajian kode program *word2Vec* terdapat pada tabel 4.17, sedangkan tabel 4.18 penyajian dari *word embedding* model *FastText*.

Tabel 4.17 Kode program implementasi *word embedding* model *word2vec*.

Word2vec CBOW model

```
import gensim
from gensim.models import Word2Vec

word2vec_model = Word2Vec(
    sentences,
    vector_size=EMBEDDING_SIZE,
    sg=SG,
    min_count=MIN_WORD,
    window=WINDOW_SIZE,
    epochs=EPOCH,
    workers=CPU_CORES-1,
    negative=NEGATIVE,
    hs=HS,
    seed=SEED
  )

word2vec_model.save('model-word-embedding/id-quran-
  word2vec-cbow-Dim300/id-quran-word2vec-cbow-Dim300-
  epoch-30.model')
word2vec_model.save('model-word-embedding/id-quran-
  word2vec-cbow-Dim300/id-quran-word2vec-cbow-Dim300-
  epoch-30.bin')

word2vec_word_vectors = word2vec_model.wv
word2vec_word_vectors.save('model-word-embedding/id-
  quran-word2vec-cbow-Dim300/id-quran-word2vec-cbow-
  Dim300-epoch-30.wordvectors')
```

Tabel 4.18 Kode program implementasi *word embedding* model *FastText*.

FastText CBOW model

```
1  import gensim
2  from gensim.models import FastText
3
4  fasttext_model = FastText(
5      sentences,
6      vector_size=EMBEDDING_SIZE,
7      sg=SG,
8      min_count=MIN_WORD,
9      window=WINDOW_SIZE,
10     epochs=EPOCH,
11     workers=CPU_CORES-1,
12     negative=NEGATIVE,
13     hs=HS,
14     seed=SEED
15   )
16
17 fasttext_model.save('model-word-embedding/id-quran-
18   fastText-cbow-Dim300/id-quran-fastText-cbow-Dim300-
19   epoch-30.model')
20 fasttext_model.save('model-word-embedding/id-quran-
21   fastText-cbow-Dim300/id-quran-fastText-cbow-Dim300-
22   epoch-30.bin')
23
24 fasttext_word_vectors = fasttext_model.wv
25 fasttext_word_vectors.save('model-word-embedding/id-
26   quran-fastText-cbow-Dim300/id-quran-fastText-cbow-
27   Dim300-epoch-30.wordvectors')
```

Penjelasan potongan kode program pada Tabel 4.17 dan Tabel 4.18 diuraikan sebagai berikut:

Baris 1-2 Definisi library Python yang digunakan

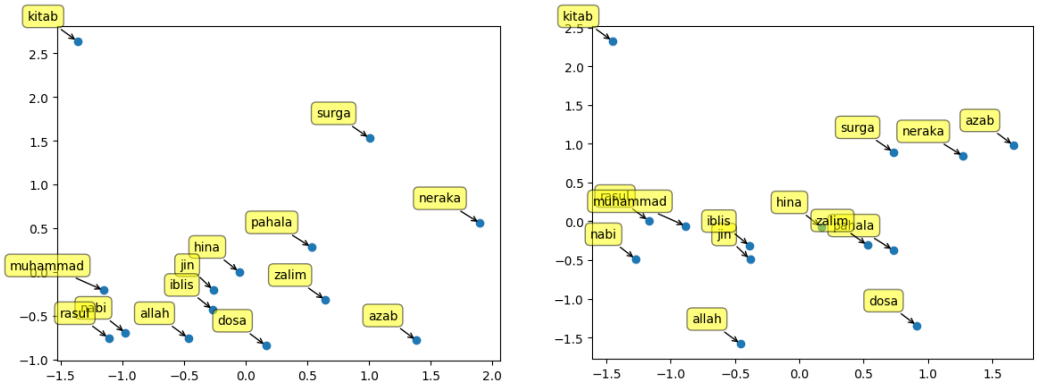
Baris 4-15 Definisi inisiasi parameter pelatihan model *word embedding*.

Baris 17-27 Proses untuk menyimpan hasil pelatihan *word embedding* dalam bentuk vektor kosakata (.wordvectors).

Hasil pelatihan *word embedding* dapat dilihat pada ruang *embedding*. Visualisasi dilakukan untuk mengetahui jarak kontekstual antar kata. Jika dua kata memiliki konteks yang sama, keduanya harus berdekatan dalam ruang penyematan. Sedangkan kata-kata yang berbeda konteks akan mempunyai jarak yang lebih jauh. Pada tahapan pelatihan *word embedding* dalam penelitian ini memiliki skema pengujian yang disajikan pada Tabel 4.19. Tabel 4.19 merupakan skema pengujian parameter pada model *word embedding Word2Vec* dan *FastText*. Di dalam tabel tersebut, ada dua faktor utama yang diuji: dimensi dari *embedding* (100, 200, 300) dan jumlah *epoch* (iterasi melalui seluruh dataset) yang digunakan dalam pelatihan (10, 20, 30, 50, 100, 1000). Contohnya, pada tabel tersebut, untuk *Word2Vec* dengan dimensi *embedding* 100, pengujian dilakukan dengan jumlah *epoch* 10, 20, 30, 50, 100, dan 1000. Hal yang sama berlaku untuk dimensi 200 dan 300 pada model *Word2Vec*, serta pada model *FastText* dengan dimensi 100, 200, dan 300.

Tabel 4.19 Skema pengujian *word embedding*.

Model *Word Embedding*	**Parameter**	
	Dimensi Size	***Epoch***
Word2Vec	100	10, 20, 30, 50, 100, 1000
	200	10, 20, 30, 50, 100, 1000
	300	10, 20, 30, 50, 100, 1000
FastText	100	10, 20, 30, 50, 100, 1000
	200	10, 20, 30, 50, 100, 1000
	300	10, 20, 30, 50, 100, 1000



(a) (b)

Gambar 4.5 Visualisasi dua dimensi hasil pelatihan *word embedding*. (a) Model *word2vec* dimensi 300 *epoch* 30. (b) Model *FastText* dimensi 300 *epoch* 30.

Hasil dari *word embedding* divisualisasikan pada Gambar 4.5 (a) dimana visualisasi *word2vec* memvisualisasikan penilaian karakter dalam Al Quran dengan rentang nilai dari -2.5 hingga 2.5 secara vertikal, mencerminkan keseimbangan karakter dalam relasi dengan Allah dan umat. Skala nilai negatif menandakan dosa, sementara nilai positif mengindikasikan pahala atau kebaikan. Secara horizontal, karakter seperti Muhammad, Rasulullah, Allah, Jin, dan lainnya direpresentasikan dengan tanda grafis yang mencerminkan tingkat keseimbangan atau keuletan mereka dalam Al Quran. Dengan visualisasi ini, dapat dipahami bagaimana karakter-karakter tersebut dinilai dalam hubungannya dengan nilai-nilai moral dan spiritual yang terkandung dalam teks Al Quran.

Sedangkan visualisasi *FastText* pada Gambar 4.5 (b) memvisualisasikan distribusi jarak antar bahan al-haram dalam peringkat Kitab al-Haramayn. Skala x-aksi menunjukkan jarak antara bahan al-haram yang memiliki kategori yang sama, sementara skala y-aksi menggambarkan jarak antara bahan al-haram yang memiliki kategori berbeda. Dalam pengkategorian umum, bahan al-haram dibagi

menjadi jenis-jenis seperti iblis (ke-22, 32, dan 54), muhammad (habi), dosa, dan azab. Setiap kategori bahan al-haram tersebut dipasangkan dengan yang lain, dan kategori yang berdekatan secara horizontal memiliki jarak yang sama antara keduanya, begitu pula dengan yang berdekatan secara vertikal. Misalnya, antara al-Faras dan al-Qar'a memiliki jarak 1.5 secara horizontal, sementara antara al-Faras dan al-Maqsurah memiliki jarak 0.5 secara vertikal.

4.7. Pengujian *Long Short-Term Memory* dengan *Word Embedding*

Tahapan sub-bab ini mengimplementasikan hasil dari *word embedding* yang terdiri dari model *word2vec* dan *fastText* dengan model yang berbasis *Long Short-Term Memory*. Proses integrasi antara *word embedding* dengan *Long Short-Term Memory* tereletak setelah proses *text preprocessing* dengan menambahkan kode program yang disajikan pada Tabel 4.20.

Tabel 4.20 Kode program integrasi *word embedding* dengan LSTM.

Load Word Embedding	
1	`WORD_EMBEDDING_PATH = word-vectors-path'`
2	
3	`word2vec = KeyedVectors.load(WORD_EMBEDDING_PATH,`
4	`  mmap='r')`
5	`vocabulary = word2vec.index_to_key`
6	
7	`VOCAB_SIZE = len(vocabulary)`
8	`EMBEDDING_SIZE = word2vec.vector_size`
9	
10	`word2vec_dict = {}`
11	
12	`for word in vocabulary:`
13	`    word2vec_dict[word] = word2vec.get_vector(word)`

Penjelasan potongan kode program pada Tabel 4.20 diuraikan sebagai berikut:

Baris 1-5 Definisi file (.*wordvectors*). dan menyimpan daftar kata-kata dalam *vocabulary*.

Baris 7 Tentukan variabel $vocab\_size$ untuk mewakili jumlah total kata dalam kamus kosakata.

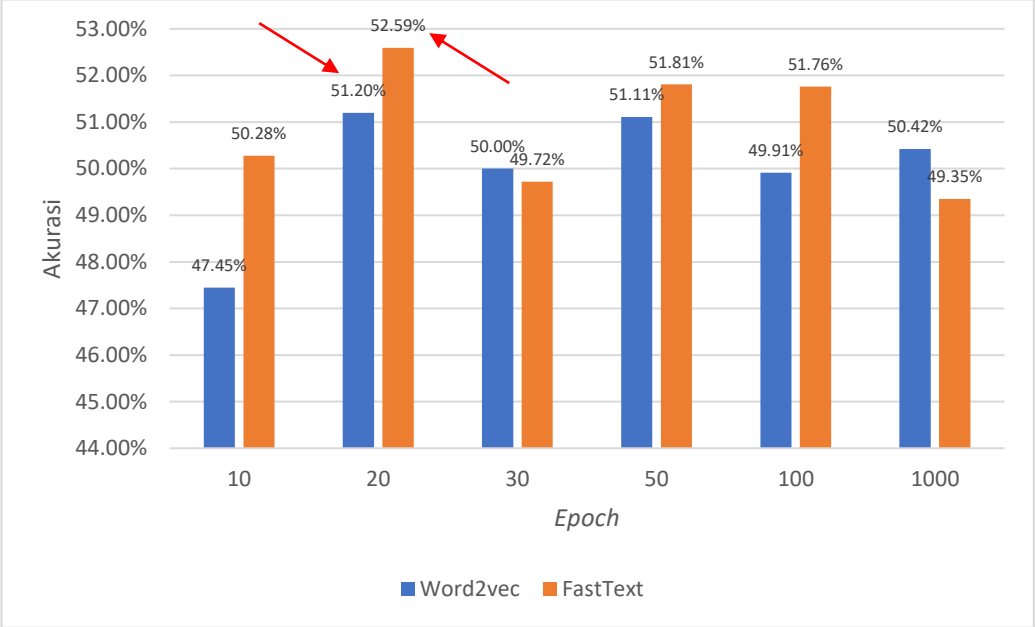
Baris 8 Definisikan variabel $embedding\_size$ yang mewakili arah vektor kosakata.

Baris 10-13 Atur variabel $word2vec\_dict$ untuk membuat kamus yang menampilkan pasangan kata dan vektornya berdasarkan hasil pelatihan penyematan kata.

4.7.1. Hasil pengujian *word embedding* dimensi 100

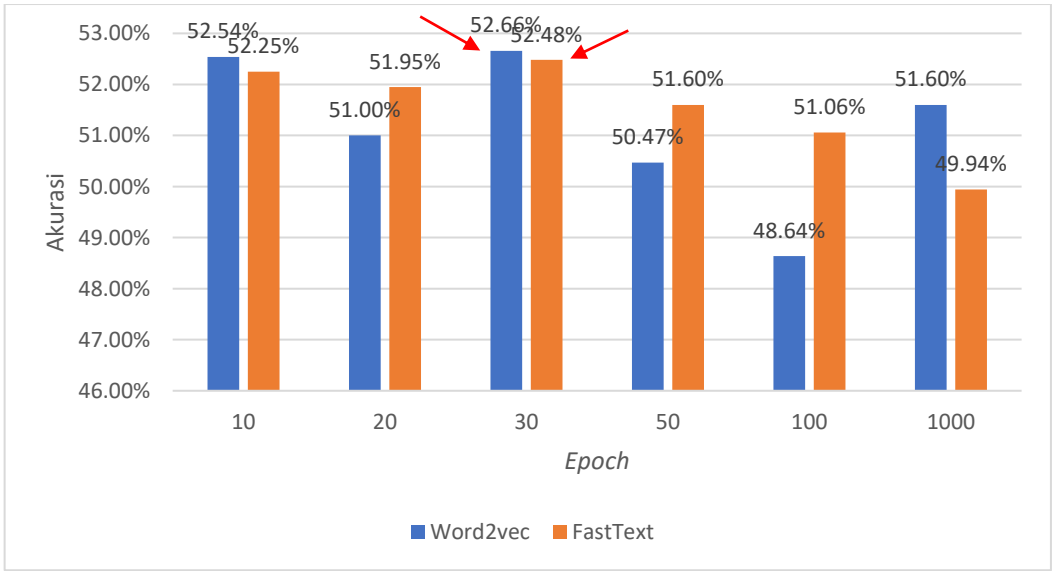
Penelitian ini melibatkan penggunaan model *word embedding* 100 dimensi menggunakan *word2vec* dan *fastText*. Berbagai percobaan dilakukan terhadap dua jenis model dengan jumlah epoch yang berbeda yaitu 10, 20, 30, (Handayani et al., 2022) 50, 100 dan 1000 untuk word2vec dan fastText. Setiap *word embedding* yang dihasilkan memberikan representasi vektor 100 dimensi untuk setiap kata dalam kumpulan data. Penelitian ini juga menerapkan model *baseline* terbaik yaitu Bi-LSTM untuk menguji performa penyematan kata yang dihasilkan.

Skema pengujian yang digunakan meliputi perbandingan data pelatihan dan pengujian yaitu 60:40, 70:30, 80:20 dan 90:10. Skema pengujian *word embedding* dicatat dan disajikan pada Tabel 4.19, yang menunjukkan kinerja dan efisiensi penyematan kata dalam model Bi-LSTM dengan skema berbagi data pelatihan dan pengujian yang berbeda. Hal ini memberikan gambaran yang jelas tentang bagaimana representasi kata dalam penyematan kata memengaruhi performa model pada tugas tertentu.



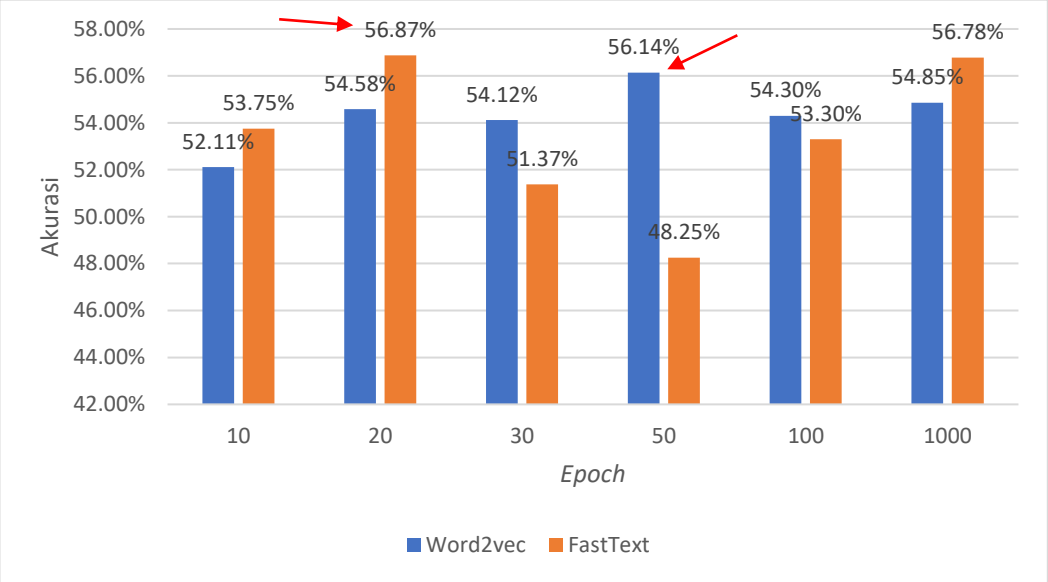
Gambar 4.6 Hasil *Word Embedding* Dimensi 100 Skema 60 : 40.

Gambar 4.6 menyajikan hasil nilai akurasi pengujian model *Bidirectional Long Short-Term Memory* (Bi-LSTM) dengan *word2vec* dan *fastText* dengan skema data latih 60% dan data uji 40%. Hasil tersebut menunjukkan *fastText* lebih unggul pada pengujian *epoch* 20 dengan nilai akurasi 52,59%. Sedangkan *word2vec* tertinggi di *epoch* 20 dengan nilai 51,20%.



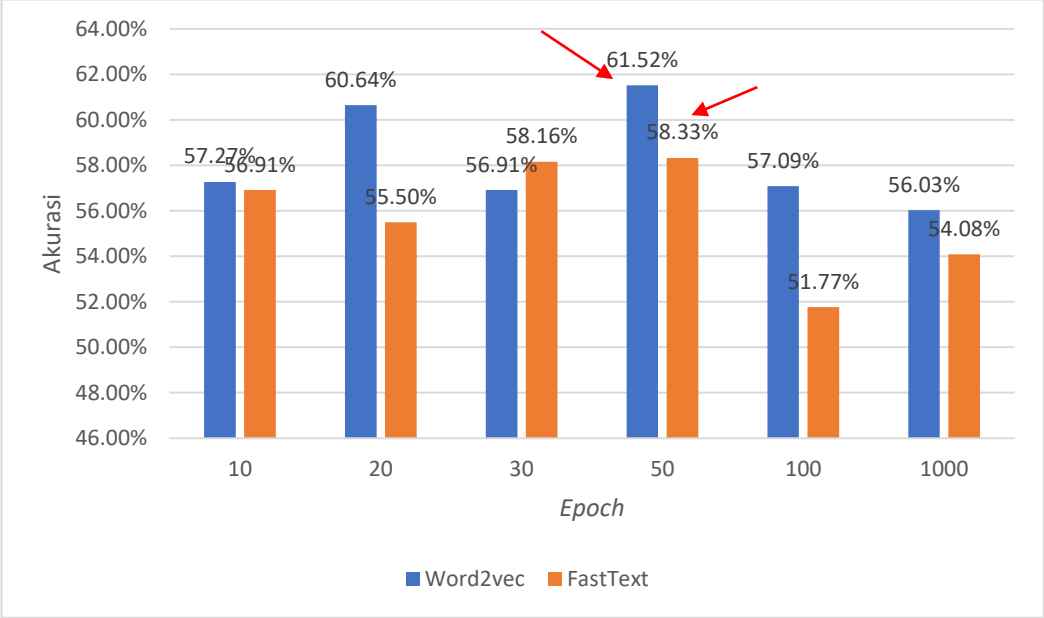
Gambar 4.7 Hasil *Word Embedding* Dimensi 100 Skema 70 : 30.

Pengujian model *Bidirectional Long Short-Term Memory* (Bi-LSTM). dengan skema data latih 70% dan data uji 30% pada dimensi 100 *word embedding* yang ditunjukkan pada Gambar 4.7. *Word2vec* lebih unggul dibandingkan dengan *FastText*, *word2vec* mendapatkan nilai akurasi 52,66%. Sedangkan FastText memiliki akurasi tertinggi 52,48% pada epoch 30.



Gambar 4.8 Hasil *Word Embedding* Dimensi 100 Skema 80 : 20.

Hasil *word embedding* Gambar 4.8 dengan pengujian *Bidirectional Long Short-Term Memory* (Bi-LSTM) skema data latih 80% dan data uji 20%. *FastText* unggul dalam pengujian parameter *FastText epoch* 20 dengan nilai akurasi 56,87%. Sedangkan *word2vec* mendapat nilai tertinggi pada *epoch* 50 dengan nilai 56,14%.



Gambar 4.9 Hasil *Word Embedding* Dimensi 100 Skema 90 : 10.

Gambar 4.9 menyajikan hasil dari pengujian *word embedding* dengan model *Bidirectional Long Short-Term Memory* (Bi-LSTM) skema 90 : 10 pada data latih dan data uji. Dari skema ini, *word2Vec* lebih unggul dari setiap *epoch* pengujian yang dilakukan dibandingkan dengan *FastText*. Nilai tertinggi pada pengujian ini 61,52% pada *epoch* 50 *word2vec*. Pada *epoch* yang sama *FastText* juga memiliki akurasi tertinggi dengan nilai 58,33%.

Tabel 4.21 Hasil pengujian *word embedding* dimensi 100.

Epoch	*Word2Vec*				*FastText*			
	40:60	70:30	80:20	90:10	40:60	70:30	80:20	90:10
10	47.45%	52.54%	52.11%	57.27%	50.28%	52.25%	53.75%	56.91%
20	51.20%	51.00%	54.58%	60.64%	52.59%	51.95%	56.87%	55.50%
30	50.00%	52.66%	54.12%	56.91%	49.72%	52.48%	51.37%	58.16%
50	51.11%	50.47%	56.14%	61.52%	51.81%	51.60%	48.25%	58.33%
100	49.91%	48.64%	54.30%	57.09%	51.76%	51.06%	53.30%	51.77%
1000	50.42%	51.60%	54.85%	56.03%	49.35%	49.94%	56.78%	54.08%

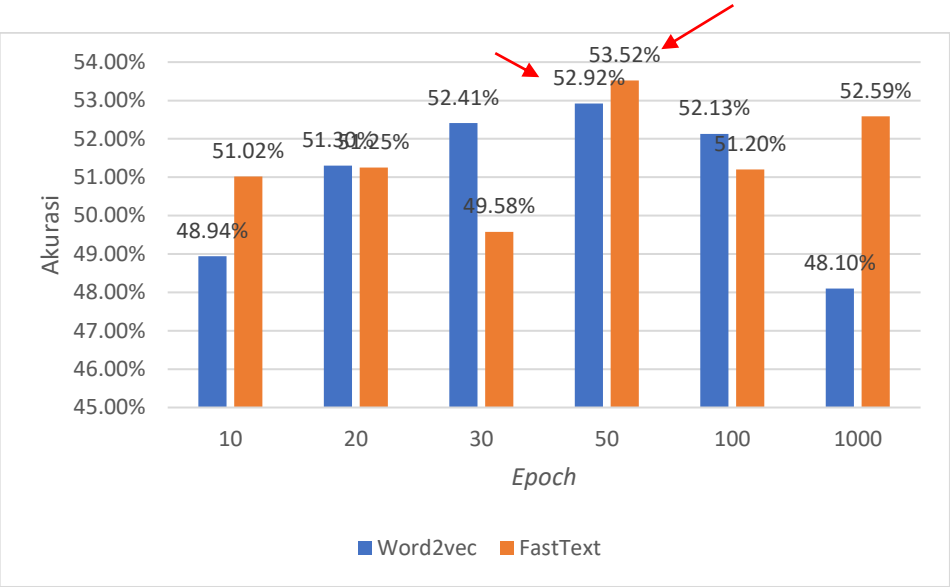
Berdasarkan hasil pengujian menggunakan skema pembagian data latih dan uji yang berbeda (60:40, 70:30, 80:20, dan 90:10) dengan dimensi 100 pada *word*

embedding, dapat disimpulkan berdasarkan Tabel 4.21 bahwa kinerja model *Bidirectional Long Short-Term Memory* (Bi-LSTM) sangat dipengaruhi oleh kombinasi antara jenis *word embedding* (*Word2vec* dan *FastText*) pada hal ini di wakilkan dengan dimensi 100, serta pembagian proporsi data latih dan uji. Pada skema 60:40, *FastText* menunjukkan unggulannya pada *epoch* 20 dengan akurasi 52,59% dibandingkan dengan *Word2vec* yang mencapai 51,20%. Namun, pada skema 70:30, 80:20, dan 90:10, *Word2vec* berhasil melampaui FastText pada beberapa titik iterasi (10, 30, dan 50), mencapai nilai akurasi tertinggi seperti 52,66%, 56,14%, dan bahkan 61,52% pada masing-masing skema. Dengan demikian, konsistensi performa lebih tinggi dari *Word2vec* pada skema data latih dan uji yang lebih besar menunjukkan potensi keunggulan dalam menangkap pola dan informasi yang relevan dari data, meskipun tetap perlu diperhatikan bahwa performa ini sangat tergantung pada kombinasi parameter serta proporsi data latih dan uji yang digunakan.

4.7.2. Hasil pengujian *word embedding* dimensi 200

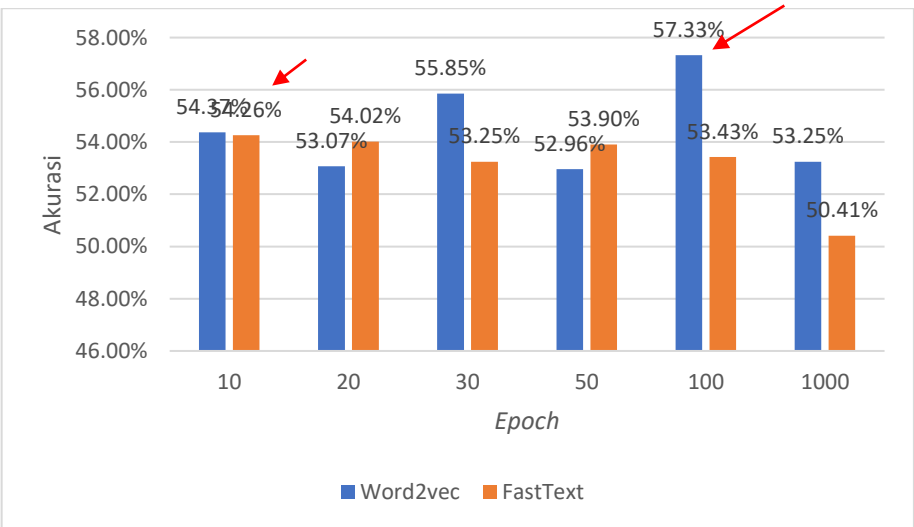
Sama halnya dengan dimensi 100, pada skema pengujian dengan dimensi 200 dalam penelitian ini, menggunakan *word2vec* dan *fastText*. Berbagai eksperimen dilakukan pada kedua jenis model dengan variasi jumlah *epoch*, yaitu 10, 20, 30, 50, 100, dan 1000 untuk *word2vec* dan *fastText*. Setiap model *word embedding* menghasilkan representasi vektor 100 dimensi untuk setiap kata dalam dataset. Studi ini juga menerapkan model *baseline* terbaik, yaitu Bi-LSTM, untuk mengevaluasi performa dari representasi kata yang dihasilkan. Pengujian

dilakukan dengan skema berbeda dalam pembagian data latih dan uji, meliputi perbandingan 60:40, 70:30, 80:20, dan 90:10.



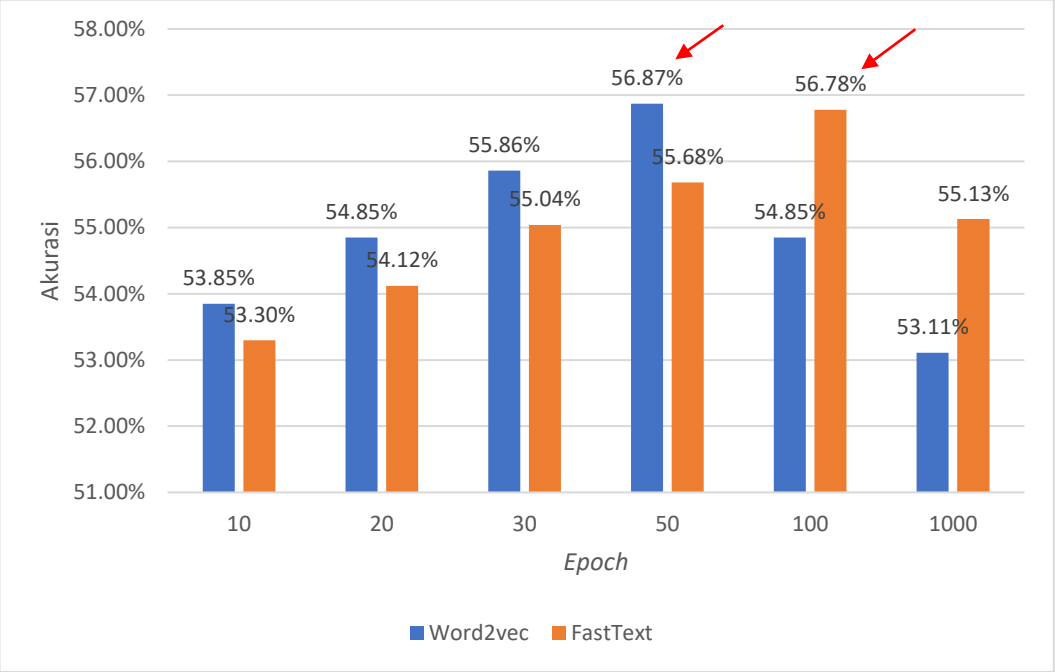
Gambar 4.10 Hasil *Word Embedding* Dimensi 200 Skema 60 : 40.

Pada Gambar 4.10 menyajikan hasil pengujian model dengan *word embedding* dimensi 200 dengan skema data latih 60% dan data uji 40%. Dari gambar tersebut *fastText* lebih unggul dibandingkan dengan *word2Vec* dengan nilai akurasi tertinggi di 53,52%. Sedangkan *word2Vec* tertinggi berada di nilai 52,92%.



Gambar 4.11 Hasil *Word Embedding* Dimensi 200 Skema 70 : 30.

Gambar 4.11 menyajikan hasil skema pengujian data latih 70% dan data uji 30% dengan model *Bidirectional Long Short-Term Memory* (Bi-LSTM) dan word embedding (word2Vec dan fastText).Hasil yang didapatkan word2vec mendapat nilai terbaik dengan akurasi 57,33% pada epoch 100. Sedangkan fastText terbaik dengan nilai 54,26% pada epoch 10.

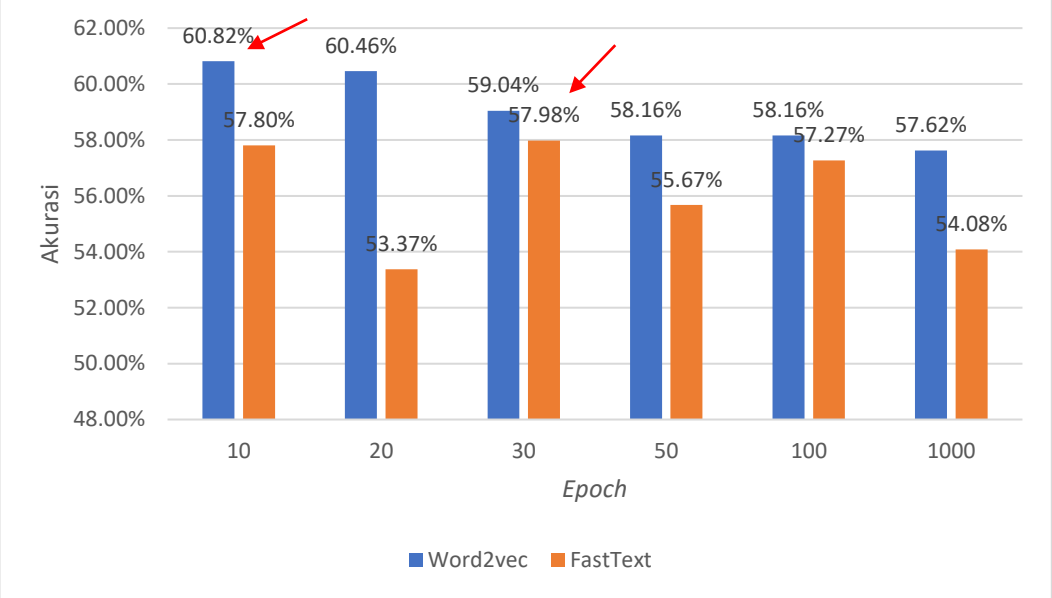


Gambar 4.12 Hasil *Word Embedding* Dimensi 200 Skema 80 : 20.

Pengujian dengan skema data latih 80% dan data uji 20% pada dimensi 200 disajikan pada Gambar 4.12. Hasil yang di dapatkan word2Vec lebih unggul pada epoch 10, 20, 30 dan 50. Akan tetapi mengalami penurunan performa pada epoch 100 dan 1000. Nilai akurasi tertinggi pada skema pengujian ini diperoleh pada word2vec dengan 56,87% pada epoch 50. Sedangkan fastText pada epoch 100 mendapat nilai terbaik 56,78%.

Pada Gambar 4.13 pengujian dengan skema 90 : 10 untuk data latih dan data uji dengan dimensi 200. Hasil dari skema ini, word2vec lebih unggul dari semua

pengujian epoch. Nilai terbaik dari word2Vec berada di epoch 10 dengan nilai 60,82%.



Gambar 4.13 Hasil *Word Embedding* Dimensi 200 Skema 90 : 10.

Berdasarkan hasil percobaan menggunakan *word embedding* dengan ukuran 200 dan skema distribusi data pelatihan dan pengujian yang berbeda (60:40, 70:30, 80:20, dan 90:10), terlihat pada Tabel 4.22 bahwa kinerja Bi-LSTM sangat dipengaruhi oleh kombinasi jenis kata, *embeddings* (*Word2vec* dan *FastText*), dan rasio berbagi data. Pada diagram 60:40, FastText menunjukkan keunggulan akurasi tertinggi sebesar 53,52% dibandingkan *Word2vec* yang mencapai 52,92%. Namun pada skema 70:30, *Word2vec* mencapai skor tertinggi pada *epoch* 100 dengan akurasi 57,33%, sedangkan *FastText* mencapai 54,26% pada *epoch* 10. Pada skema 80:20, *Word2vec* mengungguli *FastText* pada iterasi pertama (10, 20, 30 dan 50), meskipun turun pada *epoch* 100 dan 1000. Nilai presisi tertinggi pada plot ini dimiliki oleh *Word2vec* pada *epoch* 50 sebesar 56,87%.

Tabel 4.22 Hasil pengujian *word embedding* dimensi 200.

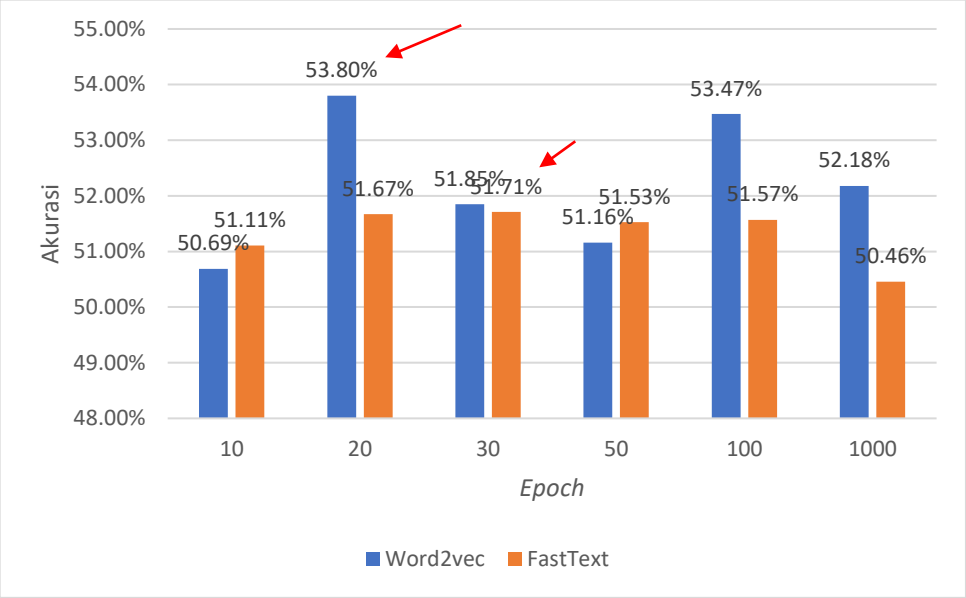
Epoch	*Word2Vec*				*FastText*			
	40:60	70:30	80:20	90:10	40:60	70:30	80:20	90:10
10	48.94%	54.37%	53.85%	60.82%	51.02%	54.26%	53.30%	57.80%
20	51.30%	53.07%	54.85%	60.46%	51.25%	54.02%	54.12%	53.37%
30	52.41%	55.85%	55.86%	59.04%	49.58%	53.25%	55.04%	57.98%
50	52.92%	52.96%	56.87%	58.16%	53.52%	53.90%	55.68%	55.67%
100	52.13%	57.33%	54.85%	58.16%	51.20%	53.43%	56.78%	57.27%
1000	48.10%	53.25%	53.11%	57.62%	52.59%	50.41%	55.13%	54.08%

Sedangkan *FastText* mencapai 56,78% pada *epoch* 100. Pada skema 90:10, *Word2vec* secara konsisten mengungguli semua iterasi, dengan nilai tertinggi pada *epoch* 10 mencapai 60,82%. Hal ini menunjukkan bahwa kinerja relatif *Word2vec* lebih stabil dan lebih baik dalam program pengujian dengan proporsi data pelatihan yang lebih besar, menunjukkan potensi manfaat dalam menangkap pola dan informasi penting dari kumpulan data yang lebih penting.

4.7.3. Hasil pengujian *word embedding* dimensi 300

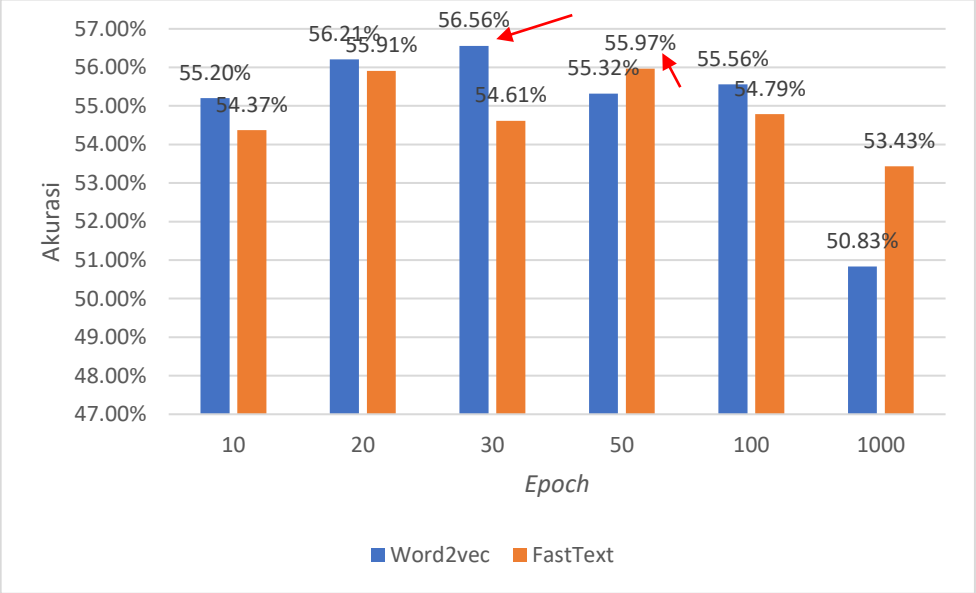
Selanjutnya eksperimen dilakukan pada *Word2vec* dan *FastText* dengan jumlah epoch yang berbeda (10, 20, 30, 50, 100, dan 1000), menghasilkan representasi vektor 300 dimensi untuk setiap kata dalam dataset. Evaluasi dilakukan dengan menggunakan model Bi-LSTM sebagai tolak ukur untuk mengevaluasi kualitas representasi dari hasil. Pengujian dilakukan pada model distribusi data pelatihan dan pengujian yang berbeda, khususnya 60:40, 70:30, 80:20, dan 90:10. Hasil eksperimen menunjukkan bahwa performa model penyematan dua kata berbeda bergantung pada kombinasi jumlah *epoch* dan skema pembagian data.

Berdasarkan Gambar 4.14 yang merupakan penyajian hasil dari pengujian model dengan dimensi embedding 300 dengan skema pengujian model 60:40 untuk data latih dan data uji. Nilai terbaik dihasilkan oleh word2vec dengan nilai 53,80% pada epoch 20. Sedangkan fastText mendapat nilai terbaik 51,71% pada epoch 30.

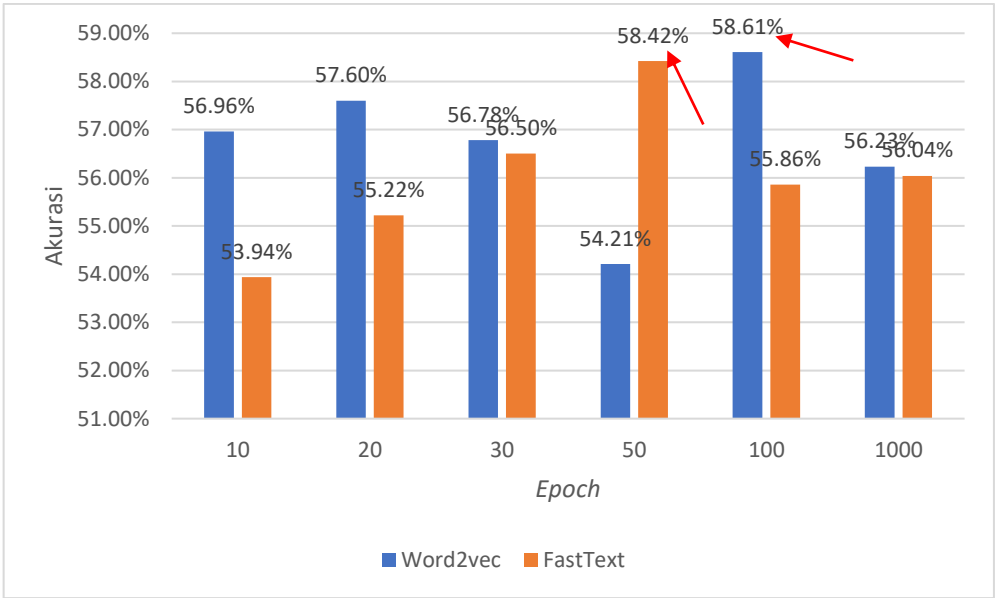


Gambar 4.14 Hasil *Word Embedding* Dimensi 300 Skema 60 : 40.

Selanjutnya dengan skema 70:30 data latih dan data uji pada dimensi 300 disajikan pada Gambar 4.15. Hasil yang diperoleh dari pengujian tersebut, *word2vec* unggul pada epoch 10, 20, 30 dan 100 dengan nilai terbaik pada *epoch* 30 mencapai 56,56%. Sedangkan *fastText* nilai terbaik pada *epoch* 50 mencapai nilai 55,97%.

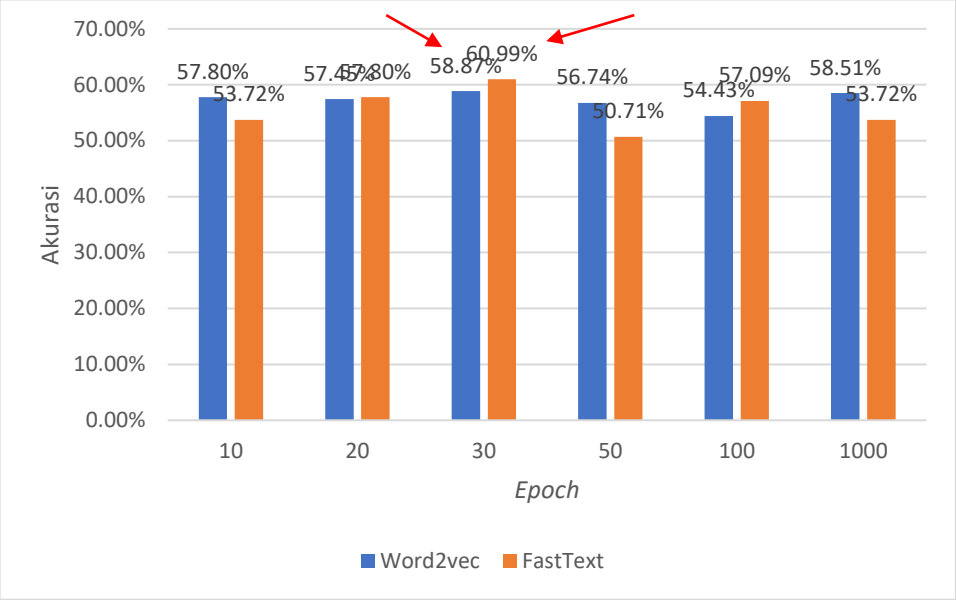


Gambar 4.15 Hasil *Word Embedding* Dimensi 300 Skema 70 : 30.



Gambar 4.16 Hasil *Word Embedding* Dimensi 300 Skema 80 : 20.

Berdasarkan Gambar 4.16 diatas, yang merupakan pengujian dengan skema 80 : 20 data latih dan data uji dengan dimensi embedding 300. Nilai word2Vec unggul pada epoch 100 mencapai nilai 58,61%. Sedangkan fastText unggul pada epoch 50 dengan nilai akurasi 58,42%.



Gambar 4.17 Hasil *Word Embedding* Dimensi 300 Skema 90 : 10.

Dari Gambar 4.17 yang disajikan, terlihat bahwa model *Word2vec* dan *FastText* dievaluasi menggunakan metrik akurasi pada berbagai epoch (10, 20, 30, 50, 100, dan 1000). Ada variasi yang signifikan dalam performa keduanya seiring dengan peningkatan jumlah iterasi. Pada sebagian besar iterasi, *Word2vec* cenderung menunjukkan performa yang lebih tinggi dibandingkan dengan *FastText*. Namun, pola ini tidak konsisten sepanjang iterasi, dengan variasi performa yang terlihat pada titik-titik tertentu. Misalnya, pada beberapa iterasi awal, *FastText* mungkin menunjukkan performa serupa atau sedikit lebih baik sebelum *Word2vec* unggul pada iterasi selanjutnya. Hasil terbaik dari dimensi 300 terdapat pada *epoch* 30 dari setiap *embedding*, dengan mencapai nilai 58,87% untuk wordi2vec dan 60,99% untuk *fastText*.

Tabel 4.23 Hasil pengujian *word embedding* dimensi 300.

Epoch	*Word2Vec*				*FastText*			
	40:60	70:30	80:20	90:10	40:60	70:30	80:20	90:10
10	50.69%	55.20%	56.96%	57.80%	51.11%	54.37%	53.94%	53.72%
20	53.80%	56.21%	57.60%	57.45%	51.67%	55.91%	55.22%	57.80%
30	51.85%	56.56%	56.78%	58.87%	51.71%	54.61%	56.50%	60.99%
50	51.16%	55.32%	54.21%	56.74%	51.53%	55.97%	58.42%	50.71%
100	53.47%	55.56%	58.61%	54.43%	51.57%	54.79%	55.86%	57.09%
1000	52.18%	50.83%	56.23%	58.51%	50.46%	53.43%	56.04%	53.72%

Berdasarkan skema pengujian model *word embedding* 300 dimensi dan skema pemisahan data yang berbeda (60:40, 70:30, dan 80:20), *Word2vec* dan *FastText* menunjukkan variasi kinerja yang signifikan pada waktu yang iterasi yang berbeda (10, 20, 30, 50, 100 dan 1000). Hasil Secara keseluruhan ditampilkan pada Tabel 4.23, *Word2vec* cenderung berkinerja lebih baik, namun modelnya tidak konsisten di seluruh iterasi, dengan perbedaan kinerja terkadang terlihat jelas. Hasil terbaik dicapai dengan *embedding* 300, dimana pada epoch 30, *Word2vec* mencapai 58,87% sedangkan *FastText* mencapai 60,99%. Hal ini menunjukkan bahwa performa terbaik kedua model terjadi pada titik iterasi yang berbeda, namun *FastText* mencapai akurasi tertinggi dengan mengintegrasikan 300 pada *epoch* 30.

4.7.4. Analisa pengujian skema word embedding

Berdasarkan serangkaian percobaan dengan variasi *word embedding* (dimensi 100, 200, dan 300) dan skema pemisahan data yang berbeda (60:40, 70:30, 80:20, dan 90:10), terlihat bahwa kinerja Model Bi-LSTM sangat bergantung pada jenis penyematan kata yang digunakan serta rasio berbagi data pelatihan dan pengujian. Pada 100 dimensi, *Word2vec* menunjukkan konsistensi

kinerja yang lebih besar di seluruh sampel data pelatihan dan pengujian yang lebih besar, menunjukkan kemampuannya untuk menangkap pola dan wawasan dari kumpulan data dengan lebih baik. Namun, pada 200 dimensi, *FastText* mencapai akurasi yang lebih tinggi di beberapa plot, menunjukkan perbedaan kinerja antara dua jenis penyematan kata pada skala data yang berbeda. Di sisi lain, pada dimensi 300, terlihat bahwa performa terbaik dari kedua model dicapai pada titik iterasi yang berbeda, dengan *FastText* mencapai akurasi tertinggi pada *epoch* 30. Hal ini menyoroti kompleksitas dalam menentukan ukuran penyematan kata yang optimal dan pentingnya peran penskalaan data dalam mempengaruhi kinerja model.

Meskipun *Word2vec* cenderung menunjukkan kinerja yang lebih konsisten pada beberapa program pengujian, perbedaan kinerja antara *Word2vec* dan *FastText* bergantung pada berbagai faktor seperti ukuran penyematan dan penskalaan data. Dengan kata lain, tidak ada solusi yang bisa diterapkan untuk semua orang, dan memilih perujuk yang tepat harus mempertimbangkan faktor-faktor seperti dimensi, skema data, dan karakteristik kumpulan data yang digunakan. Ringkasnya, untuk memaksimalkan kinerja model Bi-LSTM dalam konteks klasifikasi seperti ini, perlu dilakukan eksplorasi lebih dalam terhadap parameter penyematan kata dan strategi berbagi data yang sesuai dengan tujuan klasifikasi jenis yang diinginkan.

4.8. Penyetelan *Hyperparameter*

Untuk melatih model hasil dari implementasi *word embedding* dengan lebih baik, penting untuk menentukan *hyperparameter*. *Hyperparameter* digunakan

untuk mencari nilai optimal pada suatu model dan untuk memperbaiki performa model. *Hyperparameter* dalam model yang diusulkan mencakup *epoch*, *batch size*, dan *learning rate*. *Hyperparameter* terbaik akan diterapkan pada pengujian model dasar yang terbaik yang sudah terintegrasikan dengan *word embedding*, agar akurasi yang didapatkan menjadi lebih baik. Potongan kode program penyetelan *hyperparameter* disajikan pada Tabel 4.24.

Tabel 4.24 Kode program *hyperparameter*.

Hyperparameter yang akan dioptimasi	
1	`LEARNING_RATE = trial.suggest_float('learning_rate', 1e-`
2	`  5, 1e-1)`
3	`BATCH_SIZE = trial.suggest_int('batch_size', 16, 128)`
4	`EPOCHS = trial.suggest_int('epochs', 10, 100)`

Setiap model *word embedding* baik itu *word2vec* dan *FastText* akan dilakukan berdasarkan pengujian *hyperparameter* yang ada pada Tabel 4.24. Tabel 4.24 melakukan pengujian parameter *learning rate* dengan range 0.00001 (1e-5) sampai dengan 0.1 (1e-1). *Batch size* dengan range 16 – 128 dan *epochs* dengan range 10 – 100 pengujian.

Tabel 4.25 Hasil penyetelan *hyperparameter*

Model *Word Embedding*	**Penyetelan *Hyperparameter***		
	Learning Rate	***Batch Size***	***Epochs***
Word2Vec	0,092857763	107	96
FastText	0,090495983	51	89

Hasil yang didapatkan selama fase *hyperparameter* disajikan pada Tabel 4.25. Model *word2Vec* menghasilkan *learning rate* 0,092857763, *batch size* 107 dan *epoch* 96. Sedangkan *FastText* mendapatkan nilai *learning rate* 0,090495983, *batch size* 51 dan *epoch* 89.

4.9. Analisa Hasil Pengujian

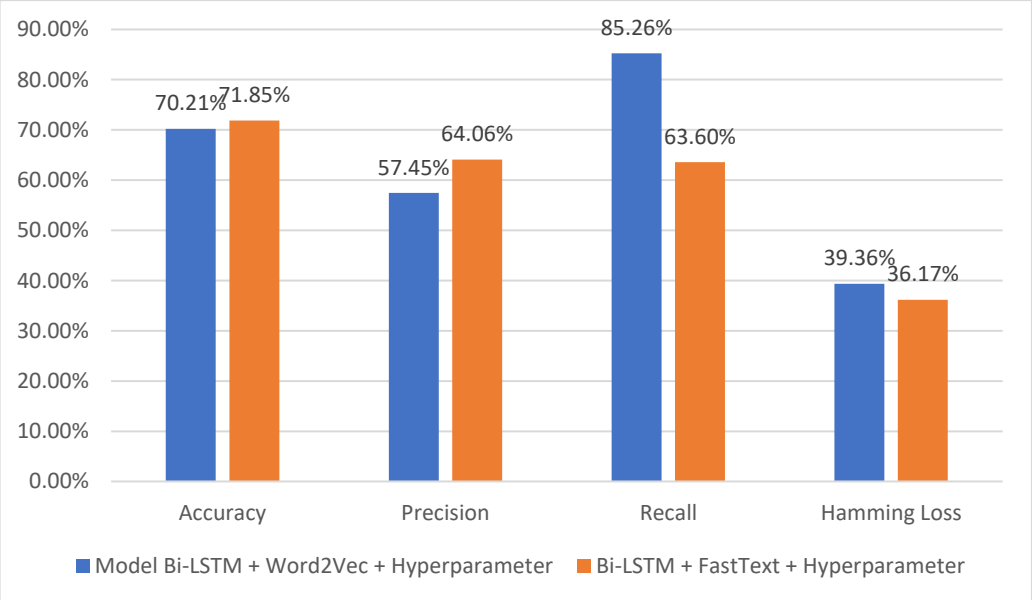
Hasil dari beberapa skema pengujian yang telah dilakukan pada penelitian ini, mendapatkan hasil yang cukup baik. Dari pengujian klasifikasi terjemahan Al-Qur'an yang menggunakan model *Long Short-Term Memory* (LSTM) hingga pengujian implementasi *word embedding* dan *hyperparameter* pada model *Long Short-Term Memory* (LSTM).

Pada pengujian yang menggunakan basis algoritma *Long Short-Term Memory* (LSTM) mendapatkan algoritma *Bidirectional Long Short-Term Memory* (Bi-LSTM) lebih unggul yang ditunjukkan pada Gambar 4.3, dengan nilai *accuracy* 50%, *precision* 62%, *recall* 56% dan *hamming loss* 35%. Hal inilah yang menjadi modal dasar untuk melanjutkan implementasi berikutnya dengan menggunakan *word embedding*.

Penggunaan *word embedding* yang sudah disampaikan pada pembahasan sebelumnya menggunakan tiga model dimensi *embedding*, yaitu 100, 200 dan 300 dimensi baik model *word2vec* ataupun *fastText*. Serta jumlah *epoch* yang berbeda dengan variasi *epoch* 10, 20, 30 (Handayani et al., 2022) pada masing-masing dimensi. Hasil pelatihan *word embeddings* yang menggunakan model *word2Vec* dengan 300 dimensi dan *epoch* 30 mempunyai nilai tertinggi. Dengan hasil akurasi sebesar 57%. Sedangkan model *fastText* dengan dimensi 300 dan *epoch* 20 merupakan hasil yang terbaik dengan hasil akurasi 56%.

Penyetelan *hyperparameter* dilakukan berdasarkan hasil dari *word embedding* yang terbaik pada masing-masing model. Parameter yang di setel dengan *hyperparameter* meliputi *epoch*, *batch size*, dan *learning rate* yang sudah

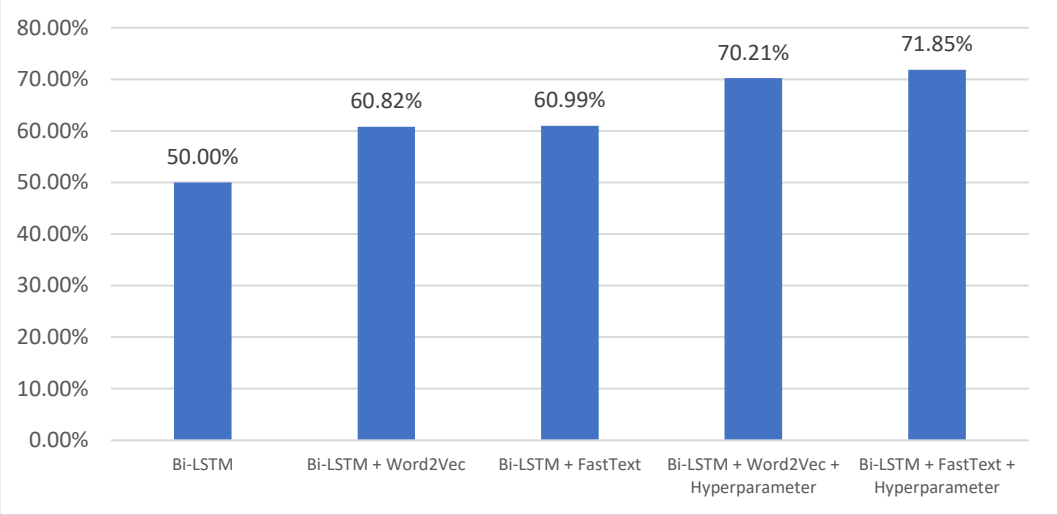
dijelaskan pada pembahasan sebelumnya. Berdasarkan Gambar 4.18, hasil pengujian *hyperparameter* dengan parameter terbaik sebanyak 96 *epoch*, *batch size* 107, dan *learning rate* sebesar 0,09285776391334735 dari pengujian model *Bi-Directional Long Short-Term Memory* (Bi-LSTM) dengan *Word2Vec* menunjukkan nilai akurasi sebesar 70,21%, *precision* 57,45%, *recall* 85,26% dan *hamming loss* 39,36%. Sedangkan pengujian dengan *FastText* mendapatkan parameter terbaik sebanyak 89 *epoch*, *batch size* 51 dan *learning rate* 0.09049598310368184 dan menghasilkan akurasi sebesar 71,63%, *precision* 64,06%, *recall* 63,60% dan *hamming loss* 36,17%.



Gambar 4.18 Hasil pengujian model dengan *hyperparameter*.

Berdasarkan pengujian model untuk klasifikasi *multi-label* terjemahan Al-Quran berbahasa Indonesia dapat ditampilkan pada Gambar 4.19. Gambar 4.19 menjelaskan hasil klasifikasi *multi-label* terjemahan Al-Quran berbahasa Indonesia tanpa menggunakan *word embedding* dan penyetelan *hyperparameter* memiliki akurasi yang rendah. Akurasi berangsur naik jika model *Bi-Directional*

Long Short-Term Memory (Bi-LSTM) dengan model *word embedding word2Vec*. Sedangkan hasil optimal didapatkan akurasi sebesar 71,63% dengan penyetelan *hyperparameter* menggunakan *embedding* model *FastText*.



Gambar 4.19 Hasil pengujian model.

Dengan hasil tersebut dapat diketahui, bahwa *word embedding* menggunakan *word2vec* dan *fastText* sangat berpengaruh untuk meningkatkan nilai akurasi. Sedangankan penyetelan *hyperparameter* dibutuhkan jika menginginkan hasil akurasi yang optimal dari suatu model.

Akan tetapi berdasarkan hasil percobaan terlihat bahwa *tuning hyperparameter* optimal pada model *Bi-directional Long Short-Term Memory* (Bi-LSTM) dengan penyematan kata *FastText* menghasilkan akurasi tertinggi sebesar 71, 63%., meningkat dibandingkan penggunaan *Word2Vec*. mencapai 70,21. %. Namun, akurasinya belum mencapai 100% dalam mengklasifikasikan beberapa label terjemahan Al-Quran Indonesia. Hal ini mungkin disebabkan oleh beberapa faktor. Pertama, alasannya adalah bahasa yang kompleks dan konteks Al-Quran yang sangat spesifik dapat menyebabkan variasi penafsiran yang sulit

diwakilkan sepenuhnya oleh model bahasa mesin. Kedua, adanya variasi gaya penulisan dan struktur kalimat yang dapat mempersulit pemodelan bahasa. Selain itu, keberhasilan klasifikasi *multi-label* juga bergantung pada ketersediaan dan kualitas kumpulan data yang digunakan untuk melatih model. Faktor-faktor ini dapat menyebabkan keterbatasan dalam prediksi sempurna, bahkan dengan pengaturan *hyperparameter* terbaik. Dalam konteks kompleks seperti ini, mencapai akurasi 100% bisa jadi sangat sulit karena kompleksitas bahasa dan variasi penafsiran yang mungkin timbul.

4.10. Integrasi Islam

Sangat penting bagi seluruh umat Islam dan umat manusia pada umumnya untuk memberikan perhatian maksimal pada pembelajaran dan pemahaman isi Al-Qur'an. Salah satu ibadah yang paling dianjurkan dalam Islam adalah mempelajari Al-Qur'an. Lebih jauh lagi, mempelajari ajaran-ajaran Al-Quran dapat membantu individu memahami ajaran-ajaran Islam secara lebih komprehensif, sehingga membantu mereka untuk mematuhi ajaran-ajaran tersebut secara akurat dan tegas. Hal ini Allah Subhanahu Wa Ta'ala berfirman pada Surat Fatir Ayat 29-30 yang berbunyi :

اِنَّ الَّذِيْنَ يَتْلُوْنَ كِتٰبَ اللّٰهِ وَاَقَامُوا الصَّلٰوةَ وَاَنْفَقُوْا مِمَّا رَزَقْنٰهُمْ سِرًّا وَّعَلَانِيَةً يَّرْجُوْنَ
تِجَارَةً لَّنْ تَبُوْرَ ۙ﴿٢٩﴾ لِيُوَفِّيَهُمْ اُجُوْرَهُمْ وَيَزِيْدَهُمْ مِّنْ فَضْلِهٖۗ اِنَّهٗ غَفُوْرٌ شَكُوْرٌ ﴿٣٠﴾

Artinya : "*Sesungguhnya orang-orang yang selalu membaca Kitab Allah (Al-Qur'an) dan melaksanakan salat dan menginfakkan sebagian rezeki yang Kami anugerahkan kepadanya dengan diam-diam dan terang-terangan, mereka itu mengharapkan perdagangan yang tidak akan rugi, agar Allah menyempurnakan*

pahalanya kepada mereka dan menambah karunia-Nya. Sungguh, Allah Maha Pengampun, Maha Mensyukuri" (QS. Fatir : 29 - 30).

Menurut Syaikh Abdurrahman bin Nashir as-Sa'di seorang pakar tafsir pada abad 14 H dalam buku tafsir as-Sa'di menjelaskan ayat tersebut "Sesungguhnya mereka itulah orang-orang yang senantiasa membaca Kitab Allah." Artinya, mereka memperhatikan perintah-perintah dan menunaikannya, serta larangan-larangannya, membenarkan dan mengimaninya, serta tidak mendahulukan segala sesuatu yang menurut pendapat manusia bertentangan dengannya. Dan mereka juga membaca kata-kata dengan belajar, membaca makna dengan perasaan dan penalaran.

Oleh karena itu, ketika membaca Al-Qur'an, penting untuk mencoba memahami semua ayat di dalamnya. Melalui perenungan, pemahaman dan pembacaan Al-Quran secara teratur, seseorang dapat merasakan kepuasan luar biasa yang dibawanya.

Dalam kitab Sahihnya, Imam Al-Bukhari meriwayatkan sebuah hadits dari Hajjaj bin Minhal dari Syu'bah dari Alqamah bin Martsad dari Sa'd bin Ubaidah dari Abu Abdirrahman As-Sulami dari Utsman bin Affan Radhiyallahu Anhu, semoga Rasulullah Shallallahu Alaihi wa Sallam bersabda :

خَيْرُكُمْ مَنْ تَعَلَّمَ الْقُرْآنَ وَعَلَّمَهُ

Artinya : "*Sebaik-baik kalian adalah orang yang belajar Al-Qur`an dan mengajarkannya*."

Masih dalam hadits riwayat Al-Bukhari dari Utsman bin Affan, namun dalam pembahasan yang sedikit berbeda, disebutkan bahwa Nabi Shallallahu Alaihi wa Sallam bersabda :

إِنَّ أَفْضَلَكُمْ مَنْ تَعَلَّمَ الْقُرْآنَ وَعَلَّمَهُ

Artinya : "*Sesungguhnya orang yang paling utama di antara kalian adalah yang belajar Al-Qur`an dan mengajarkannya*."

Dalam dua hadis di atas, ada dua amalan yang bisa menjadikan seorang muslim menjadi yang terbaik di antara sesama muslim, yaitu belajar Al-Quran dan mengajar Al-Quran. Tentu saja pembelajaran dan pengajaran yang dapat membantu seseorang menjadi yang terbaik di sini tidak lepas dari keutamaan Al-Quran itu sendiri. Al-Qur'an adalah kalam Allah, firman-Nya diturunkan kepada nabi-Nya melalui malaikat Jibril Alaihissalam. Al-Quran merupakan sumber pertama dan rujukan utama ajaran Islam. Karena keutamaan yang tinggi tersebut, maka Abu Abdirrahman As-Sulami salah satu perawi hadis ini ingin mempelajari dan mengajarkan Al-Quran sejak zaman Utsman bin Affan hingga zaman Al - Hajjaj bin Yusuf Ats-Tsaqafi.

Selain belajar dan mengajarkan Al-Quran seorang pengajar harus mengetahui kelompok-kelompok ayat atau klasifikasi ayat tersebut termasuk kategori dalam topik apa yang digunakan. Hal ini dijelaskan dalam Surat Al-Baqarah Ayat 197 yang berbunyi :

اَلْحَجُّ اَشْهُرٌ مَّعْلُوْمٰتٌ ۚ فَمَنْ فَرَضَ فِيْهِنَّ الْحَجَّ فَلَا رَفَثَ وَلَا فُسُوْقَ وَلَا جِدَالَ فِى الْحَجِّ ۗ وَمَا تَفْعَلُوْا مِنْ خَيْرٍ يَّعْلَمْهُ اللّٰهُ ۗ وَتَزَوَّدُوْا فَاِنَّ خَيْرَ الزَّادِ التَّقْوٰى ۖ وَاتَّقُوْنِ يٰٓاُولِى الْاَلْبَابِ ﴿١٩٧﴾

Artinya : *"(Musim) haji itu (pada) bulan-bulan yang telah dimaklumi. Barangsiapa mengerjakan (ibadah) haji dalam (bulan-bulan) itu, maka janganlah dia berkata jorok (rafats), berbuat maksiat dan bertengkar dalam (melakukan ibadah) haji. Segala yang baik yang kamu kerjakan, Allah mengetahuinya. Bawalah bekal, karena sesungguhnya sebaik-baik bekal adalah takwa. Dan bertakwalah kepada-Ku wahai orang-orang yang mempunyai akal sehat!*" (QS. Al Baqarah : 197).

Menurut Syaikh Prof. Dr. Umar bin Abdullah al-Muqbil, professor fakultas syari'ah Universitas Qashim, Saudi Arabia. Ayat ini menjelaskan petunjuk atau tata tertib yang berkaitan dengan ibadah haji dan umrah. Misalnya, aturan haji dan umrah disebutkan secara terpisah, yang menunjukkan pengelompokkan khusus terkait dengan setiap ibadah haji. Hal ini membantu untuk memahami bahwa meskipun keduanya melibatkan perjalanan ke Tanah Suci, terdapat perbedaan dalam prosedur dan waktu yang dibutuhkan. Hal ini menciptakan pengelompokan atau pembedaan dalam panduan ibadah untuk memenuhi kebutuhan dan kondisi individu.

BAB V

PENUTUP

5.1. Kesimpulan

Penelitian ini bertujuan untuk mengetahui kinerja dari *word embedding* menggunakan *word2Vec* dan *FastText* dalam mengoptimalkan varian algoritma *Long Short-Term Memory* (LSTM) dalam klasifikasi terjemahan Al-Qur'an. Algoritma LSTM dapat mengalami kesulitan ketika menemukan kata-kata langka yang tidak ada dalam kamus atau kumpulan data pelatihan. Hal ini dapat menjadi masalah jika terjemahan Al-Qur'an menggunakan kosakata yang kurang umum atau khusus. Dengan hasil pengujian yang didapatkan dalam penelitian ini hasil algoritma yang menggunakan basis LSTM mendapatkan nilai akurasi 47%, *precision* 63%, *recall* 63% dan *hamming loss* 37%. Pada Bi-LSTM mendapatkan hasil akurasi sebesar 49%, *precision* 62%, *recall* 56% dan *hamming loss* 35%. Dari hasil tersebut menunjukkan kinerja dari algortima LSTM sendiri masih belum optimal.

Pengoptimalan algoritma LSTM dapat dilakukan dengan mengimplementasikan model *word embedding* menggunakan *word2Vec* dan *fastText*, serta dilakukan penyetelan *hyperparameter*. Sehingga penelitian ini melakukan pengujian dengan skema algoritma Bi-LSTM + *word2Vec* dan Bi-LSTM + *FastText* dengan jumlah data dan parameter yang sama, hasilnya dapat diketahui akurasi Bi-LSTM + *word2Vec* sebesar 60,82% dengan dimensi 200 dan 10 *epoch* pada skema pengujian data latih 90% dan data uji 10%. Sedangkan Bi-

LSTM + *FastText* mendapatkan akurasi 60,99% dengan dimensi 300 dan 30 *epoch* dengan skema model pengujian data latih 90% dan data uji 10%..

Hasil yang optimal dari klasifikasi *multi-label* terjemahan Al-Qur'an didapatkan dengan penyetelan *hyperparameter*. Penulis melakukan penyetelan *hyperparameter* berdasarkan hasil dari model *word embedding* terbaik pada masing-masing model. Hasilnya dapat disimpulkan bahwa Bi-LSTM + *FastText* + *Hyperparameter* mengungguli dengan akurasi 71,85%, *precision* 64,06%, *recall* 63,60 dan *hamming loss* 36,17% berdasarkan parameter terbaik *epoch* 89, *batch size* 51 dan *learning rate* 0.09049598310368184.

5.2. Saran

Adapun saran untuk penelitian selanjutnya adalah sebagai berikut :

1. Penelitian dapat dilakukan dengan menerapkan arsitektur lainnya seperti GRU (*Gated Recurrent Unit*), model *Transformer* atau penggabungan beberapa model sehingga bisa mengklasifikasikan dengan menggunakan kumpulan data yang sama untuk mendapatkan performa yang lebih baik.
2. Melibatkan ahli Bahasa Indonesia dengan ahli studi Al-Quran untuk melakukan kolaborasi dalam memperdalam pemahaman atas konteks kultural dan bahasa yang sangat spesifik dari terjemahan Al-Quran
3. Menggunakan teknik penyeimbangan dataset menggunakan SMOTE..
4. Menggunakan *word embedding* yang lain seperti GloVe.
5. Hasil dari penelitian bisa diterapkan pada aplikasi nyata.

.

DAFTAR PUSTAKA

Abdullahi, A., Samsudin, N. A., Rahim, M. H. A., Khalid, S. K. A., & Efendi, R. (2021). Multi-label classification approach for Quranic verses labeling. *Indonesian Journal of Electrical Engineering and Computer Science*, *24*(1), 484–490. https://doi.org/10.11591/ijeecs.v24.i1.pp484-490

Af'idah, D. I., Dairoh, D., Handayani, S. F., & Pratiwi, R. W. (2021). Pengaruh Parameter Word2Vec terhadap Performa Deep Learning pada Klasifikasi Sentimen. *Jurnal Informatika: Jurnal Pengembangan IT*, *6*(3), 156–161. https://doi.org/10.30591/jpit.v6i3.3016

Af'idah, D. I., Dairoh, D., Handayani, S. F., Pratiwi, R. W., & Sari, S. I. (2022). Sentimen Ulasan Destinasi Wisata Pulau Bali Menggunakan Bidirectional Long Short Term Memory. *MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, *21*(3), 607–618. https://doi.org/10.30812/matrik.v21i3.1402

Aghaeiboorkheili, M., & Kawagle, J. G. (2022). The History of the Derivation of Euler's Number. *Journal of Applied Mathematics and Physics*, *10*(09), 2780–2795. https://doi.org/10.4236/jamp.2022.109185

Almuzaini, H. A., & Azmi, A. M. (2020). Impact of Stemming and Word Embedding on Deep Learning-Based Arabic Text Categorization. *IEEE Access*, *8*, 127913–127928. https://doi.org/10.1109/ACCESS.2020.3009217

Alsaleh, D., & Larabi-Marie-Sainte, S. (2021). Arabic Text Classification Using Convolutional Neural Network and Genetic Algorithms. *IEEE Access*, *9*, 91670–91685. https://doi.org/10.1109/ACCESS.2021.3091376

Alwehaibi, A., Bikdash, M., Albogmi, M., & Roy, K. (2021). A study of the performance of embedding methods for Arabic short-text sentiment analysis using deep learning approaches. *Journal of King Saud University - Computer and Information Sciences*, *xxxx*. https://doi.org/10.1016/j.jksuci.2021.07.011

Amalia, A., Oktinas, W., Aulia, I., & Rahmat, R. F. (2018). Determination of quality television programmes based on sentiment analysis on Twitter. *Journal of Physics: Conference Series*, *978*(1). https://doi.org/10.1088/1742-6596/978/1/012117

Anjana, S., Saruladha, K., & Sathyabama, K. (2019). Bidirectional and stacked LSTM for sleep disorders prediction. *Proceedings of the 3rd International Conference on Computing Methodologies and Communication, ICCMC 2019*, *Iccmc*, 912–916. https://doi.org/10.1109/ICCMC.2019.8819736

Antariksa, K., Purnomo WP, Y. S., & Ernawati, E. (2019). Klasifikasi Ujaran Kebencian pada Cuitan dalam Bahasa Indonesia. *Jurnal Buana Informatika*, *10*(2), 164. https://doi.org/10.24002/jbi.v10i2.2451

Bessou, S., & Aberkane, R. (2019). Subjective Sentiment Analysis for Arabic Newswire Comments. *Journal of Digital Information Management*, *17*(5), 289. https://doi.org/10.6025/jdim/2019/17/5/289-295

Ertugrul, A. M., & Karagoz, P. (2018). Movie Genre Classification from Plot Summaries Using Bidirectional LSTM. *Proceedings - 12th IEEE International Conference on Semantic Computing, ICSC 2018*, *2018-Janua*, 248–251. https://doi.org/10.1109/ICSC.2018.00043

Hana, K. M., Adiwijaya, Al Faraby, S., & Bramantoro, A. (2020). Multi-label Classification of Indonesian Hate Speech on Twitter Using Support Vector Machines. *2020 International Conference on Data Science and Its Applications, ICoDSA 2020*. https://doi.org/10.1109/ICoDSA50139.2020.9212992

Hanafi, A., Adiwijaya, A., & Astuti, W. (2020). Klasifikasi Multi Label pada Hadis Bukhari Terjemahan Bahasa Indonesia Menggunakan Mutual Information dan k-Nearest Neighbor. *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, *9*(3), 357–364. https://doi.org/10.32736/sisfokom.v9i3.980

Handayani, S. F., Pratiwi, R. W., Dairoh, D., & Af'idah, D. I. (2022). Analisis Sentimen pada Data Ulasan Twitter dengan Long-Short Term Memory. *JTERA (Jurnal Teknologi Rekayasa)*, *7*(1), 39. https://doi.org/10.31544/jtera.v7.i1.2022.39-46

Hasyim, Y. (2020). AKIDAH AKHLAK MTs KELAS VII. In *Akidah Akhlak*. Kementerian Agama Republik Indonesia.

Indrapurasih, R. D., Bijaksana, M. A., & Sardi, I. L. (2018). Implementasi dan Analisis Kesamaan Semantik Antar Kata Bahasa Indonesia Menggunakan Metode GloVe. *EProceedings of Engineering*, *5*(3), 7699–7706.

Isnain, A. R., Sihabuddin, A., & Suyanto, Y. (2020). Bidirectional Long Short Term Memory Method and Word2vec Extraction Approach for Hate Speech Detection. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *14*(2), 169. https://doi.org/10.22146/ijccs.51743

Kholifatullah, B. A. H., & Prihanto, A. (2023). Penerapan Metode Long Short Term Memory Untuk Klasifikasi Pada Hate Speech. *Journal of Informatics and Computer Science (JINACS)*, *04*, 292–297. https://doi.org/10.26740/jinacs.v4n03.p292-297

Kim, M., & Kang, K.-H. (2020). Comparison of Neural Network Techniques for Text Data Analysis. *International Journal of Advanced Culture Technology*, *8*(2), 231–238. https://doi.org/https://doi.org/10.17703/IJACT.2020.8.2.231

Le, X. H., Ho, H. V., Lee, G., & Jung, S. (2019). Application of Long Short-Term Memory (LSTM) neural network for flood forecasting. *Water (Switzerland)*, *11*(7). https://doi.org/10.3390/w11071387

Lim, E., Setiawan, E. I., & Santoso, J. (2019). Stance Classification Post

Kesehatan di Media Sosial Dengan FastText Embedding dan Deep Learning. *Journal of Intelligent System and Computation*, *1*(2), 65–73. https://doi.org/10.52985/insyst.v1i2.86

Luo, X. (2021). Efficient English text classification using selected Machine Learning Techniques. *Alexandria Engineering Journal*, *60*(3), 3401–3409. https://doi.org/10.1016/j.aej.2021.02.009

Nurdin, A., Anggo Seno Aji, B., Bustamin, A., & Abidin, Z. (2020). Perbandingan Kinerja Word Embedding Word2Vec, Glove, Dan Fasttext Pada Klasifikasi Teks. *Jurnal Tekno Kompak*, *14*(2), 74. https://doi.org/10.33365/jtk.v14i2.732

Pasaribu, D. J. M., Kusrini, K., & Sudarmawan, S. (2020). Peningkatan Akurasi Klasifikasi Sentimen Ulasan Makanan Amazon dengan Bidirectional LSTM dan Bert Embedding. *Inspiration: Jurnal Teknologi Informasi Dan Komunikasi*, *10*(1), 9–20. https://doi.org/10.35585/inspir.v10i1.2568

Prabowo, Y. D., Marselino, T. L., & Suryawiguna, M. (2019). Pembentukan Vector Space Model Bahasa Indonesia Menggunakan Metode Word to Vector. *Jurnal Buana Informatika*, *10*(1), 29. https://doi.org/10.24002/jbi.v10i1.2053

Sari, E. Y., Wierfi, A. D., & Setyanto, A. (2019). Sentiment Analysis of Customer Satisfaction on Transportation Network Company Using Naive Bayes Classifier. *2019 International Conference on Computer Engineering, Network, and Intelligent Multimedia, CENIM 2019 - Proceeding*, *2019-Novem*. https://doi.org/10.1109/CENIM48368.2019.8973262

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification. *Augmented Human Research*, *5*(1). https://doi.org/10.1007/s41133-020-00032-0

Taher, Y., Moussaoui, A., & Moussaoui, F. (2022). Automatic Fake News Detection based on Deep Learning, FastText and News Title. *International Journal of Advanced Computer Science and Applications*, *13*(1), 146–158. https://doi.org/10.14569/IJACSA.2022.0130118

Taradhita, D. A. N., & Putra, I. K. G. D. (2021). Hate speech classification in Indonesian language tweets by using convolutional neural network. *Journal of ICT Research and Applications*, *14*(3), 225–239. https://doi.org/10.5614/itbj.ict.res.appl.2021.14.3.2

Taufiqurrahman, F., Faraby, S. Al, & Purbolaksono, M. D. (2021). Klasifikasi Teks Multi Label pada Hadis Terjemahan Bahasa Indonesia Menggunakan Chi Square dan SVM. *E-Proceeding of Engineering*, *8*(5), 10650–10659.

Vu, M. T., Jardani, A., Massei, N., & Fournier, M. (2021). Reconstruction of missing groundwater level data by using Long Short-Term Memory (LSTM)

deep neural network. *Journal of Hydrology*, *597*. https://doi.org/10.1016/j.jhydrol.2020.125776

Yuslan, M., & Bakar, A. (2021). Klasifikasi Teks Hadis Bukhari Terjemahan Indonesia Menggunakan Recurrent Convolutional Neural Network (CRNN). *Jurnal Teknologi Informasi Dan Ilmu Komputer (JTIIK)*, *8*(5), 907–918. https://doi.org/10.25126/jtiik.202183750