Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS184853

KLASIFIKASI INTENT UNTUK TASK-ORIENTED DIALOG SYSTEM MENGGUNAKAN ARSITEKTUR LONG SHORT-TERM MEMORY (STUDI KASUS : E-COMMERCE)

INTENT CLASSIFICATION FOR TASK-ORIENTED DIALOG SYSTEM USING LONG SHORT-TERM MEMORY ARCHITECTURE (CASE STUDY : E-COMMERCE)

MUHAMMAD AZZAM
NRP 05211540000022

Dosen Pembimbing
Renny Pradina K., S.T., M.T., SCJP

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR - IS184853

KLASIFIKASI INTENT UNTUK TASK-ORIENTED DIALOG SYSTEM MENGGUNAKAN ARSITEKTUR LONG SHORT-TERM MEMORY (STUDI KASUS : E-COMMERCE)

MUHAMMAD AZZAM
NRP 05211540000022

Dosen Pembimbing
Renny Pradina K., S.T., M.T., SCJP

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

UNDERGRADUATE THESIS - IS184853

INTENT CLASSIFICATION FOR TASK-ORIENTED DIALOG SYSTEM USING LONG SHORT-TERM MEMORY ARCHITECTURE (CASE STUDY : E-COMMERCE)

MUHAMMAD AZZAM
NRP 05211540000022

Supervisor
Renny Pradina K., S.T., M.T., SCJP

INFORMATION SYSTEM DEPARTMENT
Information Technology and Communication Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2019

LEMBAR PENGESAHAN

KLASIFIKASI INTENT UNTUK TASK-ORIENTED DIALOG SYSTEM MENGGUNAKAN ARSITEKTUR LONG SHORT-TERM MEMORY (STUDI KASUS : E-COMMERCE)

TUGAS AKHIR

Disusun Sebagai Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

MUHAMMAD AZZAM
NRP. 0521 15 4000 0022

Surabaya, Juni 2019

KEPALA
DEPARTEMEN SISTEM INFORMASI

Mahendrawati ER, ST, M.Sc, Ph.D
NIP 19761011 200604 2 001

LEMBAR PERSETUJUAN

KLASIFIKASI INTENT UNTUK TASK-ORIENTED DIALOG SYSTEM MENGGUNAKAN ARSITEKTUR LONG SHORT-TERM MEMORY (STUDI KASUS : ECOMMERCE)

Disusun Sebagai Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

MUHAMMAD AZZAM
NRP. 0521 15 4000 0022

Disetujui Tim Penguji: Tanggal Ujian: Juli 2019
Periode Wisuda: September 2019

Renny Pradina K., S.T, M.T., SCJP **(Pembimbing I)**

Faizal Johan Atletiko, S.Kom., M.T **(Penguji I)**

Nur Aini Rakhmawati S.Kom, M.Sc.Eng, Ph.D **(Penguji II)**

KLASIFIKASI INTENT UNTUK TASK-ORIENTED DIALOG SYSTEM MENGGUNAKAN ARSITEKTUR LONG SHORT-TERM MEMORY (STUDI KASUS : E-COMMERCE)

Nama Mahasiswa	**: Muhammad Azzam**
NRP	**: 05211540000022**
Departemen	**: Sistem Informasi FTIK-ITS**
Pembimbing I	**: Renny Pradina K., S.T., M.T., SCJP**

ABSTRAK

Penggunaan internet di Indonesia telah mengalami perkembangan yang cukup pesat setiap tahunnya. Sebagian besar layanan internet yang digunakan oleh masyarakat Indonesia adalah aplikasi chatting. Aplikasi chatting saat ini diminati oleh berbagai kalangan, yang memungkinkan penggunanya untuk dapat bertukar pesan secara bebas. Perusahaan e-commerce dapat menggunakan aplikasi chatting sebagai sarana bertukar pesan dengan pelanggan. Kesigapan penjual dalam menanggapi pesan pelanggan dapat menumbuhkan persepsi positif terhadap kualitas pelayanan yang diberikan. Namun, masalah muncul ketika jumlah pesan yang masuk dalam sehari terlampau banyak. Dengan sumber daya manusia yang terbatas, membalas satu persatu pesan dapat menyita banyak waktu.

Task-oriented dialog agents adalah program yang dapat berkomunikasi dengan pengguna dengan bahasa alami dan didesain khusus untuk menyelesaikan tugas tertentu. Sehingga penggunaannya memiliki potensi besar bagi penjual atau pemilik usaha untuk meningkatkan responsifitas mereka dalam membalas pesan pelanggan pada aplikasi chatting. Arsitektur task-oriented dialog terdiri dari modul Natural Language Understanding (NLU), Dialog Manager, dan Natural Language Generation (NLG). Modul NLU digunakan untuk mengubah masukan pengguna ke dalam bentuk semantik.

Long Short-Term Memory (LSTM) merupakan salah satu arsitektur Recurrent Neural Network (RNN) yang terbukti bekerja dengan baik pada tugas klasifikasi intent pada modul NLU. Sehingga penelitian ini menggunakan arsitektur tersebut untuk melakukan klasifikasi intent pada percakapan bahasa Indonesia dengan domain e-commerce. Dalam penerapannya LSTM membutuhkan masukan berupa vektor kata agar proses training untuk menjadi suatu model dapat dilakukan. Sehingga dalam penelitian ini menggunakan pretrained word embedding model untuk melakukan representasi vektor kata. Hasil yang diharapkan dalam penelitian ini adalah model klasifikasi intent dengan nilai akurasi tertinggi untuk percakapan bahasa Indonesia dengan domain e-commerce.

Kata Kunci: Intent Classification, Long Short-Term Memory, Natural Language Understanding, Task-Oriented Dialog System, Word Embedding

INTENT CLASSIFICATION FOR TASK-ORIENTED DIALOG SYSTEM USING LONG SHORT-TERM MEMORY ARCHITECTURE (CASE STUDY : E-COMMERCE)

Name	**: Muhammad Azzam**
NRP	**: 05211540000022**
Department	**: Information System FTIK-ITS**
Supervisor	**: Renny Pradina K., S.T., M.T., SCJP**

ABSTRACT

Internet usage in Indonesia has grown quite rapidly every year. Most of internet services used by Indonesian are chat applications. Chat applications are currently used by various group of people, which allow them to be able to exchange messages seamlessly. E-commerce companies can use chat applications to communicate with their customers. Responsiveness of sellers in replying to customer messages can have an effect on feedback regarding the quality of services provided. Problems arise when there are too many messages received by the seller in a day. With limited human resources, replying to those messages one by one can take a lot of time.

Task-oriented dialog agents are programs that can communicate with users using natural languages and are specifically designed to complete certain tasks. So that its use has great potential for sellers or business owners to increase their responsiveness in replying to customer messages on chat applications. The task-oriented dialog architecture consists of Natural Language Understanding (NLU), Dialog Manager, and Natural Language Generation (NLG) modules. The NLU module is used to convert user input into semantic form.

Long Short-Term Memory (LSTM) is one of the Recurrent Neural Network (RNN) architectures that has proven to work well in the intent classification task of the NLU module. So that this study uses LSTM to classify the intent in Indonesian

conversations within the e-commerce domain. In its implementation, LSTM requires input in the form of a word vector so that it can be processed for model training. To do a word vector representations, this study uses a pretrained word embedding model. The expected results in this study are the intent classification model with the highest accuracy for Indonesian conversations within the e-commerce domain.

Keywords: Intent Classification, Long Short-Term Memory, Natural Language Understanding, Task-Oriented Dialog System, Word Embedding

KATA PENGANTAR

Dengan mengucapkan rasa syukur kepada Tuhan Yang Maha Pengasih dan Maha Penyayang atas izin-Nya penulis dapat menyelesaikan buku yang sederhana ini dengan judul Klasifikasi Intent Untuk Task-Oriented Dialog System Menggunakan Arsitektur Long Short-Term Memory. Dalam penyelesaian Tugas Akhir ini, penulis diiringi oleh pihak-pihak yang selalu memberi dukungan, saran, dan doa sehingga penelitian berlangsung dengan lancar. Secara khusus penulis mengucapkan terima kasih dari lubuk hati terdalam kepada:

1. Ibu Mahendrawathi ER. S.T., M.Sc., Ph.D. selaku Ketua Departemen Sistem Informasi ITS Surabaya.
2. Ibu Renny Pradina K., S.T., M.T., SCJP, selaku dosen pembimbing yang telah mencurahkan segenap tenaga, waktu dan pikiran dalam penelitian ini, serta memberikan motivasi yang membangun.
3. Bapak Faizal Johan Atletiko, S.Kom., M.T., dan Ibu Irmasari Hafidz, S.Kom., M.Sc., selaku dosen penguji yang telah memberikan kritik dan saran yang membuat kualitas penelitian ini lebih baik lagi.
4. Segenap dosen dan karyawan Departemen Sistem Informasi.
5. Bapak Hendro Prasetyo dan Ibu Salma selaku kedua orang tua penulis, serta Imtiyaz Nurul Qonita, Thoriq Izzurrahman, dan Rizky Qurrota Ainy selaku saudara kandung dari penulis yang tiada henti memberikan doa, dukungan, dan semangat.
6. Hipzul, Farchan, dan Benny yang telah menemani penulis disaat penulis merasa jenuh.
7. Ardo, Gilang, dan Kevin sebagai teman perjuangan penulis dalam suka dan duka.
8. Indra, Habib, Edwin, Wahyu, Aldi, Dani, dan Salim selaku teman satu atap penulis selama dua tahun merantau di Kota Surabaya.

9. Pihak lainnya yang berkontribusi dalam tugas akhir yang belum dapat penulis sebutkan satu per satu.

Penyusunan tugas akhir ini masih jauh dari kata sempurna, untuk itu penulis menerima segala kritik dan saran yang membangun sebagai upaya menjadi lebih baik lagi ke depannya. Semoga buku tugas akhir ini dapat memberikan manfaat untuk pembaca.

Surabaya, 3 Juli 2019

Penulis

Muhammad Azzam

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan.

DAFTAR TABEL

Halaman ini sengaja dikosongkan.

DAFTAR KODE

Halaman ini sengaja dikosongkan.

BAB I
PENDAHULUAN

Bab ini akan menjelaskan tentang pendahuluan pengerjaan tugas akhir yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, dan manfaat yang akan diperoleh dari penelitian tugas akhir ini.

1.1 Latar Belakang

Indonesia merupakan salah satu negara dengan jumlah pengguna internet terbesar di dunia dengan total 143,26 juta jiwa. Angka tersebut menunjukkan bahwa 54,68% dari total penduduk Indonesia yang bejumlah 262 juta jiwa telah menggunakan internet dan jumlahnya terus bertambah setiap tahun [1]. Hal ini menjadi peluang besar bagi pedagang untuk menjual dan memasarkan barang dagangannya secara daring, dengan istilah yang dikenal *e-commerce*. Laporan PPRO, perusahaan layanan pembayaran terkemuka di dunia tentang pembayaran dan perdagangan *online* tahun 2018, menyatakan Indonesia memiliki pertumbuhan belanja *online* tertinggi mencapai 78% per tahun, disusul oleh Meksiko 59%, Filipina 51%, Kolombia 45%, dan Uni Emirat Arab (UEA) 33% [2]. Tingginya pertumbuhan belanja *online* ini berpotensi menguntungkan bagi ekonomi dan mendorong kesetaraan sosial di Indonesia. Dalam laporan yang berjudul *The Digital Archipelago*, McKinsey menyebutkan perekonomian Indonesia dapat diuntungkan dalam empat hal. Pertama, perdagangan *online* mendorong konsumsi, dimana sekitar 30% dari perdagangan *online* merupakan belanja tambahan atau diluar rutin yang nilainya mencapai Rp 42 triliun pada tahun 2017 dan berpotensi meningkat menjadi Rp 308 triliun pada tahun 2022. Kedua, membuka akses ekspor bagi barang industri kreatif. Dengan melibatkan distributor luar negeri, diperkirakan kanal perdagangan *online* menyumbang ekspor sebesar Rp 364 triliun pada tahun 2022. Ketiga, mendorong terciptanya lapangan pekerjaan. McKinsey memproyeksikan perdagangan *online* dapat menyokong 26 juta pekerjaan pada 2022. Keempat, meningkatkan kesetaraan sosial. Hal ini dikarenakan konsumen

di daerah terpencil di luar pulau Jawa bisa mendapatkan berbagai pilihan produk yang lebih beragam dengan harga yang lebih murah. Tercatat bahwa rata-rata harga barang yang diperdagangkan lewat *online* lebih rendah 11-25% dibanding toko ritel tradisional [3].

Indonesia telah memiliki ekosistem yang cukup baik dalam mendukung tumbuh kembangnya *e-commerce*. Maraknya situs marketplace seperti Tokopedia, Bukalapak, Shopee, telah membantu pemilik usaha dalam menjual produk miliknya. Sedangkan dalam kegiatan *marketing* dan menjalin hubungan dengan pelanggan, pemilik usaha terbantu oleh media sosial seperti Facebook, Instagram, LINE, dan lain-lain. Istilah ini biasa dikenal dengan *digital marketing*. Dari hasil survey APJII 2017, layanan media sosial merupakan layanan kedua paling banyak diakses oleh pengguna internet di Indonesia dengan persentase mencapai 87,13% [1]. Oleh karenanya, *digital marketing* melalui media sosial dianggap menjadi salah satu cara yang efektif dalam menarik pelanggan dan menjaga hubungan dengan pelanggan.

Jika media sosial digunakan sebagai media *marketing*, aplikasi *chatting* biasanya digunakan sebagai media interaksi antara penjual dengan pelanggan. Berdasarkan survey APJII 2017, layanan *chatting* adalah layanan yang paling banyak diakses oleh pengguna internet di Indonesia dengan persentase 89,35%, sedikit lebih banyak dibanding media sosial [1]. Pelanggan biasanya menanyakan hal-hal yang berkaitan dengan spesifikasi produk, ketersediaan, pengiriman, layanan purna jual, dan lain-lain melalui aplikasi *chatting*. Kesigapan penjual dalam menanggapi pesan pelanggan dapat menumbuhkan persepsi positif terhadap kualitas pelayanan yang diberikan. Masalah timbul ketika penjual memiliki ribuan pesan pelanggan dalam sehari. Dengan sumber daya manusia yang terbatas, membalas satu persatu pesan pelanggan dapat memakan banyak waktu. Sehingga berakibat munculnya persepsi negatif dari pelanggan akan kualitas pelayanan.

Task-oriented dialog agents adalah program yang dapat berkomunikasi dengan pengguna dengan bahasa alami dan didesain khusus untuk menyelesaikan tugas tertentu. Melalui percakapan singkat yang dilakukan, agen menggali informasi dari pengguna yang dibutuhkan dalam penyelesaian tugas. Contoh *task-oriented dialog agents* yang sudah banyak diterapkan adalah asisten digital pada smartphone (Siri, Google Now, Cortana, dan lain-lain) yang dapat membantu pengguna mencari restoran, membuat panggilan telepon, dan mengirim pesan singkat [4]. *Task-oriented dialog agents* memiliki potensi besar bagi pemilik usaha dalam membantu melayani pesan pelanggan pada layanan *chatting*. Pertanyaan-pertanyaan yang umum ditanyakan hingga pemesanan produk dapat dijawab dan dilayani oleh agen. Sehingga pemilik usaha tidak perlu menambah sumber daya manusia. Walau begitu, pertanyaan-pertanyaan khusus yang belum dilihat dan dipelajari agen tetap harus dialihkan ke *customer service*.

Dalam penerapannya *task-oriented dialog agents* menggunakan arsitektur yang terdiri dari modul *Natural Language Understanding* (NLU), *Dialog Manager*, dan *Natural Language Generation*. NLU merupakan modul yang bertugas untuk mengubah ujaran pengguna ke dalam bentuk semantik. NLU memiliki tiga proses utama di dalamnya, yaitu klasifikasi domain, *intent*, dan *slot-filling*.

Recurrent Neural Network (RNN) merupakan salah satu arsitektur pemrosesan yang dapat diterapkan dalam modul NLU. Penggunaan RNN terbukti bekerja lebih efektif dibanding metode statistika tradisional [5]. Penelitian terkini menunjukkan bahwa arsitektur *Long Short-Term Memory* (LSTM) merupakan pengembangan dari RNN yang telah terbukti bekerja lebih baik untuk melakukan klasifikasi *intent* pada sebuah ujaran dalam dataset Cortana dan ATIS, karena dapat mengatasi masalah gradien yang hilang [6].

Dengan adanya tugas akhir ini diharapkan dapat melakukan klasifikasi *intent* percakapan bahasa Indonesia mengenai layanan atau produk *e-commerce* menggunakan arsitektur

LSTM dan membandingkannya dengan arsitektur neural network yang lain. Sehingga mampu memberikan hasil klasifikasi *intent* dengan akurasi terbaik yang dapat digunakan sebagai input pada modul selanjutnya dalam sistem *task-oriented dialog*.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang yang telah disampaikan, berikut merupakan perumusan masalah pada tugas akhir ini.

1. Bagaimana menentukan kelas *intent* yang sesuai untuk domain percakapan *e-commerce*?
2. Bagaimana menerapkan arsitektur *Long Short-Term Memory* untuk tugas klasifikasi *intent*?
3. Bagaimana performa model *Long Short-Term Memory* untuk tugas klasifikasi *intent* jika dibandingkan dengan arsitektur *recurrent neural network* yang lain?
4. Bagaimana pengaruh *word embedding* terhadap performa model *Long Short-Term Memory*?

1.3 Batasan Permasalahan

Berikut ini Batasan masalah tugas akhir ini, berdasarkan latar belakang dan perumusan masalah yang telah dijelaskan.

1. Model *word embedding* yang digunakan pada penelitian ini merupakan model *pretrained* yang bersumber dari penelitian sebelumnya.
2. *Dataset* yang digunakan pada algoritme *Long Short-Term Memory* merupakan *dataset* percakapan bahasa Indonesia yang berlabel seputar topik jual beli *smartphone* pada *e-commerce*.
3. Setiap ujaran dalam *dataset* hanya berisi maksimal satu *intent*.

1.4 Tujuan

Berdasarkan perumusan masalah yang disebutkan sebelumnya, tujuan yang akan dicapai melalui tugas akhir ini adalah:

1. Menentukan apa saja kelas *intent* yang sesuai berdasarkan dataset percakapan dalam domain *e-commerce*.
2. Menerapkan arsitektur *Long Short-Term Memory* untuk mengklasifikasikan ujaran dalam percakapan ke kelas *intent* yang sesuai.
3. Membandingkan performa arsitektur *Long Short-Term Memory* dengan arsitektur *recurrent neural network* lain.
4. Menerapkan *word embedding* untuk meningkatkan performa model *Long-Short Term Memory*.

1.5 Manfaat

Berikut merupakan manfaat yang diharapkan dari penelitian yang dilakukan:

1. Mengetahui dan memahami arsitektur *Long Short-Term Memory*, serta pengaruh *word embedding* terhadap performanya untuk melakukan tugas klasifikasi *intent* pada sekumpulan *dataset* percakapan bahasa Indonesia dalam domain *e-commerce*.
2. Sebagai bentuk penelitian awal yang memungkinkan untuk dilanjutkan dengan penelitian-penelitian selanjutnya dan dapat diterapkan ke dalam rancang bangun sistem *task-oriented dialog* untuk membantu *customer service* dari berbagai bidang bisnis berinteraksi dengan pelanggannya.

1.6 Relevansi

Relevansi tugas akhir ini terhadap laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI) adalah karena tugas akhir ini berkaitan dengan mata kuliah bidang keilmuan laboratorium ADDI. Mata kuliah tersebut antara lain Sistem Cerdas, Pengolahan Bahasa Alami, dan Visualisasi Informasi.

Halaman ini sengaja dikosongkan.

BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka berisi penelitian sebelumnya yang dijadikan acuan dalam pengerjaan tugas akhir dan juga berisi dasar teori untuk menunjang penelitian pada tugas akhir.

2.1 Penelitian Sebelumnya

Terdapat beberapa penelitian yang memiliki topik yang hampir serupa dengan penelitian ini, diantaranya akan dijelaskan pada Tabel 2.1 hingga Tabel 2.3.

Tabel 2.1 Literatur 1

Judul	*A Comparative Study of Neural Network Models for Lexical Intent Classification* [6]
Nama, Tahun	Suman Ravuri, Andreas Stolcke, 2015
Gambaran umum penelitian	Penelitian ini bertujuan untuk menjawab pertanyaan model dengan algoritme apa yang bekerja paling baik dalam tugas klasifikasi domain dan *intent*. Peneliti menggunakan *dataset Automatic Terminal Information Service* (ATIS) untuk tugas klasifikasi *intent*, dan *dataset* Cortana untuk tugas klasifikasi domain. Percobaan *training* model pada kedua *dataset* dilakukan menggunakan beberapa parameter salah satunya adalah *optimizer*. *Optimizer Stochastic Gradient Descent* (SGD) terbukti menghasilkan model yang baik pada *dataset* Cortana. Sedangakan *optimizer Momentum* terbukti menghasilkan model yang baik pada *dataset* ATIS. Hasil dari penelitian menunjukkan secara konsisten *gated models* yaitu *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) bekerja paling baik, disusul oleh *standard recurrent model*, dan *feedforward model*.
Keterkaitan penelitian	Literatur ini merupakan rujukan utama penulis. Arsitektur LSTM akan digunakan untuk tugas klasifikasi *intent* pada *dataset* percakapan bahasa Indonesia seputar jual beli *smartphone* melalui *e-commerce*. Performa model LSTM juga akan

	dibandingkan dengan GRU karena kedua arsitektur bekerja lebih baik pada dataset yang berbeda. Selain kedua model tersebut, model dengan arsitektur *standard Recurrent Neural Network* (RNN) juga akan dicoba sebagai pembanding. *Optimizer* SGD dan *Momentum* akan diuji coba dalam proses *training* setiap model pada penelitian yang akan dilakukan penulis. Perbedaan utama dari literatur ini adalah tugas klasifikasi yang dilakukan merupakan tugas klasifikasi biner, sedangkan klasifikasi *intent* yang dilakukan penulis merupakan klasifikasi multinomial. *Pretrained word embedding model* dari hasil dari pelatihan Word2Vec menggunakan data yang diperoleh dari media sosial Twitter.

Tabel 2.2 Literatur 2

Judul	*A Comparative Study of Recurrent Neural Network Models for Lexical Domain Classification* [5]
Nama, Tahun	Suman Ravuri, Andreas Stolcke, 2016
Gambaran umum penelitian	Penelitian ini bertujuan untuk mencari pendekatan dengan performa terbaik dalam klasifikasi domain dan *intent* dan menggeneralisasikannya ke *n-way classification*. Pendekatan tersebut melibatkan *neural network sequence classifiers*, yang dikombinasikan dengan *language models* tradisional. Penelitian ini menggunakan *dataset* aplikasi asisten pribadi Cortana. Hasil penelitian menunjukkan arsitektur LSTM dan GRU memiliki performa yang terbaik, dan mengalahkan sistem *baseline* yang berbasis fitur n-gram.
Keterkaitan penelitian	Pada tugas akhir ini, penulis mengadaptasi arsitektur LSTM untuk tugas klasifikasi *intent* pada dataset percakapan bahasa Indonesia seputar jual beli smartphone melalui *e-commerce*. Selain itu arsitektur *recurrent neural network* lain seperti GRU juga diadaptasi untuk dibandingkan performanya dengan arsitektur LSTM.

Tabel 2.3 Literatur 3

Judul	*Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks* [7]
Nama, Tahun	Ji Young Lee, Franck Dernoncourt, 2016
Gambaran umum penelitian	Penelitian ini bertujuan memperkenalkan model berbasis *Recurrent Neural Networks* (RNNs) dan *Convolutional Neural Networks* (CNNs) untuk klasifikasi teks singkat berurut (*sequential short-text*), dan mengevaluasinya pada tugas *dialog act classification*. Model tersebut melakukan karakterisasi ujaran pada sebuah dialog berdasarkan kombinasi dari kriteria pragmatis, semantik, dan sintaksis. Hasil percobaan menunjukkan jika dengan menambahkan informasi berurut (*sequential*) pada tugas klasifikasi *dialog act*, kualitas dari prediksi dapat ditingkatkan.
Keterkaitan penelitian	Penulis mengadaptasi arsitektur LSTM sebagai pendekatan klasifikasi teks sekuensial untuk melakukan tugas klasifikasi *intent*. Representasi vektor menggunakan model *word embedding* juga menjadi rujukan penulis. Penelitian ini menggunakan *dataset* Google News untuk melatih *word embedding*, sedangkan penulis menggunakan *pretrained word embedding model* dari hasil dari pelatihan Word2Vec menggunakan data yang diperoleh dari media sosial Twitter.

2.2 Dasar Teori

Dasar teori berisikan mengenai teori-teori yang digunakan untuk melakukan penelitian ini.

2.2.1 *Machine Learning*

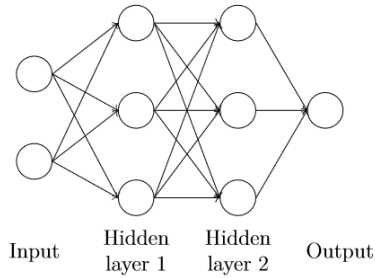
Machine learning adalah sebuah pemrograman komputer digital yang dapat bekerja seperti apa yang dilakukan manusia atau hewan, yaitu dengan melibatkan proses pembelajaran [8]. Ada tiga tipe algoritme pembelajaran pada *machine learning* yang dibedakan berdasarkan *feedback* untuk belajar. Tiga tipe

algoritme tersebut antara lain *supervised learning*, *unsupervised learning*, dan *reinforcement learning*.

Dalam *supervised learning*, agen mengamati beberapa contoh pasangan *input-output* yang diberikan dan mempelajari fungsi yang memetakan dari *input* ke *output* tersebut [9]. Beberapa tugas yang dapat diselesaikan oleh *supervised learning* antara lain klasifikasi dan regresi. Sebuah contoh penerapan algoritme *supervised learning* untuk sebuah tugas klasifikasi adalah program *spam filter* pada *email*. Tugasnya adalah memprediksi apakah sebuah *email* masuk merupakan *spam* atau bukan. Untuk dapat melakukan pembelajaran dibutuhkan sebuah *dataset*. *Dataset* tersebut berisi contoh *email* yang dikategorikan *spam* dan *email* yang dikategorikan bukan *spam*. Performa dari agen dapat diukur menggunakan rasio *email* yang dikelompokkan dengan benar [10].

2.2.2 *Neural Network*

Merupakan sebuah area *machine learning* yang popularitasnya meningkat dari tahun ke tahun sejak 1940. *Neural networks* terinspirasi dari cara kerja jaringan saraf biologis dalam otak dan mencoba untuk menirukannya. *Neural networks* terdiri dari sebuah *input*, satu atau lebih *hidden layers*, plus *output layer* [10]. Istilah *N-layer neural network* digunakan untuk menunjukkan ada berapa *hidden layer* plus *output layer* pada *neural network* tersebut. Contoh dari *three-layer neural network* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Contoh *Neural Network* Dengan 2 *Hidden Layer* [10]

2.2.3 *Natural Language Processing*

Natural Language Processing (NLP) adalah sebuah pendekatan terkomputerisasi untuk menganalisis dan merepresentasikan teks untuk tujuan mencapai pemrosesan bahasa yang menyerupai manusia untuk berbagai macam tugas [11]. NLP telah diterapkan pada beberapa tugas pembelajaran mesin seperti *automatic summarization*, analisis sentimen, klasifikasi teks, dan *question answering*.

Tujuan dari NLP adalah untuk memproses Bahasa semirip mungkin dengan manusia. Sebelumnya NLP lebih dikenal dengan istilah *Natural Language Understanding* (NLU). Sebuah NLP dapat dikatakan sebagai NLP jika sistemnya dapat melakukan [11] :

- Parafrase dari *input* teks
- Menerjemahkan teks tersebut ke Bahasa lain
- Menjawab pertanyaan dari teks
- Mengambil inti dari teks

Dalam menampilkan apa yang sebenarnya terjadi, sistem NLP memiliki pendekatan dengan beberapa level yang secara psikolinguistik diproses secara dinamis dan antar level saling terkait dalam berbagai macam urutan [11].

Phonology. Level ini berkaitan dengan interpretasi kata-kata berdasarkan bagaimana kata tersebut diucapkan. Ada 3 aturan dalam level ini antara lain *phonetic rules* untuk membedakan suara setiap kata, *phonemic rules* untuk mendeteksi variasi pengucapan apabila beberapa kata diucapkan secara bersama dan *prosodic rules* untuk mendeteksi fluktuasi penekanan intonasi dalam suatu kalimat [11].

Morphology. Level ini berkaitan dengan analisis *morphemes* (unit makna terkecil) dari setiap kata. Sebagai contoh, kata ‘memakan’ dapat dianalisis secara morfologis dengan memisah antara awalan ‘me’ dengan kata dasar ‘makan’. Dalam NLP istilah yang umum digunakan untuk memisahkan *morphem* adalah *stemming* [11].

Lexical. Pada level ini, baik manusia maupun sistem NLP menginterpretasikan definisi dari setiap kata. Beberapa tipe pemrosesan yang berkontribusi terhadap pemahaman tingkat kata dapat dilakukan salah satunya *part-of-speech tagging*. Pada pemrosesan ini, kata-kata yang memiliki lebih dari satu *part-of-speech tag* diberikan *tag* yang paling sesuai berdasarkan konteks dimana kata tersebut berada [11].

Syntactic. Level ini berfokus pada analisis kata dalam kalimat untuk mendeteksi grammar (tata bahasa) dari kalimat tersebut. *Output* dari level ini adalah struktural dependensi dari hubungan antar kata [11].

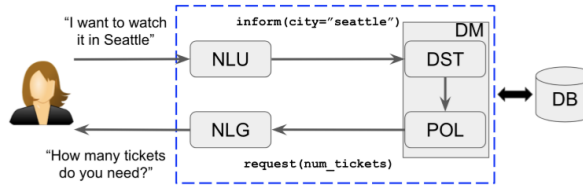
Semantic. Level ini berkaitan dengan proses menentukan makna berdasarkan mayoritas pemikiran manusia. Pemrosesan semantik menentukan makna yang mungkin dari suatu kalimat dengan memfokuskan pada katerkaitan antar kata dalam kalimat tersebut [11].

Discourse. Berbeda dengan *syntax* dan *semantic*, level ini bekerja pada unit teks yang lebih panjang dari kalimat. Teks tidak dianggap sebagai gabungan kalimat, yang masing-masing dapat ditafsirkan secara tunggal. Sebaliknya, discourse berfokus pada properti teks secara keseluruhan yang menyampaikan makna dengan membuat hubungan antara kalimat-kalimat komponen [11].

Pragmatic. Level ini berkaitan dengan penggunaan bahasa dengan tujuan tertentu dalam konteks isi teks untuk pemahaman. Tujuannya adalah untuk menjelaskan makna tambahan di dalam teks tanpa benar-benar dikodekan di dalamnya. Level ini membutuhkan banyak pengetahuan dunia, termasuk pemahaman tentang *intent*, *plan*, dan *goal* [11].

2.2.4 *Task-Oriented Dialog System*

Task-oriented dialog system merupakan salah satu kategori dari *spoken dialog systems* yang dirancang untuk membantu pengguna dalam mencapai tujuan spesifik. Paling sering, tujuannya adalah untuk mendapatkan informasi (mencari kamar hotel, tiket bioskop, jadwal bus) [12].



Gambar 2.2 Arsitektur *Task-Oriented Dialog System* [13]

Untuk membuat *task-oriented dialog system*, arsitektur yang digambarkan pada Gambar 2.2 sering digunakan dalam praktiknya. Sistem ini terdari modul-modul berikut [13].

- *Natural Language Understanding* (NLU)
  Modul ini mengambil ujaran pengguna sebagi *input* dan mengubahnya menjadi bentuk semantik dari *dialog acts*.
- *Dialog Manager* (DM)
  Modul ini adalah pengontrol pusat dari *dialog system*. Biasanya terdiri dari sub-modul *Dialog State Tracking* (DST) yang berfungsi untuk melacak *dialog state* saat ini. Sub-modul lainnya yaitu *Dialog Policy* (POL) berfungsi memilih tindakan berdasarkan keadaan internal yang disediakan DST. Tindakan dapat berupa respons ke pengguna, atau operasi pada *backend database*.
- *Natural Language Generation* (NLG)
  Jika POL memilih untuk menanggapi pengguna, modul ini akan mengubah *dialog act* menjadi bentuk bahasa alami.

Task-oriented dialog system yang telah digunakan oleh banyak industri. Apple Siri, Microsoft Cortana, Google Now, Amazon Echo, Baidu DuerOS, dan Facebook M telah memfokuskan pada pengembangan teknik *Natural Language Understanding* (NLU) untuk mengurai ucapan pengguna ke dalam slot semantik yang telah ditentukan sebelumnya [14].

2.2.5 *Intent Classification*

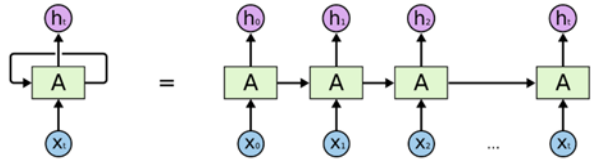
Intent classification adalah salah satu komponen dari *Natural Language Understanding* (NLU) yang penting dalam tahap pra-pemrosesan untuk banyak *dialog systems* yang menginterpretasikan *input* ucapan. Sebagai contoh, seorang

pengguna bertanya kepada Siri atau Cortana “Jelaskan padaku mengenai cuaca hari ini” maka ucapan tersebut harus diklasifikasikan sebagai *weather-query* sehingga *query* dapat diarahkan ke subsistem *natural understanding* yang tepat [6].

Beberapa model dapat digunakan untuk melakukan klasifikasi *intent*, seperti *feedforward*, *recurrent*, dan *gated*. Dalam penelitian yang dilakukan oleh Suman Ravuri dkk, secara konsisten *gated models* yaitu LSTM dan GRU menunjukkan performa paling baik, diikuti oleh *standard recurrent model* dan *feedforward model* [6].

2.2.6 *Recurrent Neural Network*

Jaringan saraf berulang atau *Recurrent Neural Network* (RNN) merupakan jenis arsitektur *artificial neural network* dimana pemrosesannya dipanggil berulang-ulang untuk memproses *input* yang biasanya berupa data sekuensial. RNN termasuk kategori *deep learning* karena pemrosesannya dilakukan menggunakan banyak *layer*. RNN telah mengalami kemajuan dengan pesat dan diterapkan pada bidang-bidang seperti *Natural Language Processing* (NLP), *voice recognition*, *video analytic*, dan lain-lain.



Gambar 2.3 Arsitektur RNN [15]

Feedforward neural network (*linear*, CNN) hanya melihat lima hingga sepuluh kata sebelumnya ketika hendak memprediksi kata berikutnya. Padahal diketahui jika manusia dapat memahami konteks yang lebih panjang dengan keberhasilan yang besar. Sementara itu *Recurrent Neural Networks* (RNNs) tidak membatasi besaran konteks. Dengan menggunakan koneksi berulang (memori), informasi dapat berputar disamping jaringan-jaringan ini dalam jangka waktu yang lama [16]. Informasi ini digunakan untuk memproses setiap sampel,

sehingga setiap sampel diproses dengan mempertimbangkan sampel-sampel sebelumnya bahkan yang berjarak jauh sekalipun. Arsitektur RNN dapat dijabarkan seperti yang ditunjukkan pada Gambar 2.3.

Pentingnya memori pada RNN dapat ditunjukkan melalui contoh berikut.

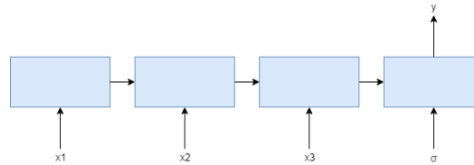
"Adam yang biasanya selalu memakai kacamata pergi ke sekolah."

Yang pergi ke sekolah adalah Adam, bukan kacamata, walaupun kata "kacamata" lebih dekat dengan kata "pergi".

Ada beberapa jenis pemrosesan RNN yang dibedakan berdasarkan jumlah *input* yang diproses dan jumlah *output* yang dihasilkan, yaitu *many-to-many*, *many-to-one*, dan *one-to-many*. Untuk tugas klasifikasi *intent*, jenis pemrosesan yang digunakan adalah *many-to-one*. Dimana model memroses banyak *input* dan menghasilkan satu *output*. Misalnya ujaran pada percakapan jual beli *smartphone* berikut.

"Untuk versi 128gb harganya berapa?"

Input kalimat yang terdiri dari banyak kata tersebut akan diproses dan menghasilkan *output intent* yaitu "harga".

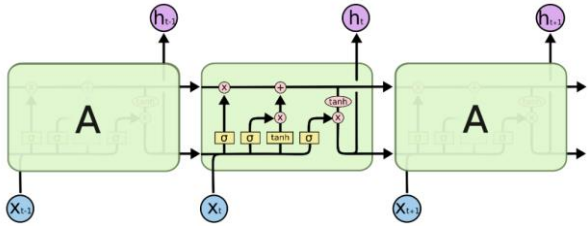


Gambar 2.4 ***Many-to-one RNN***

Namun sering diakui bahwa mempelajari *long-term dependencies* dengan *Stochastic Gradient Descent* (SGD) dapat menjadi sangat sulit dan menimbulkan masalah seperti gradien yang menghilang [16]. Untuk mengatasi masalah ini adalah dengan menggunakan unit RNN yang lain seperti *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU).

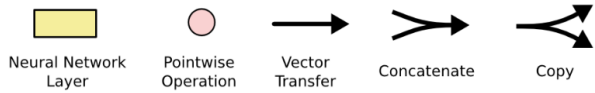
2.2.7 *Long Short-Term Memory*

Long Short-Term Memory (LSTM) adalah salah satu arsitektur *recurrent network* dengan pembelajaran berbasis gradien yang ditemukan oleh Sepp Hochreiter dan Jurgen Schmidhuber pada tahun 1997 [17]. LSTM dirancang untuk mengatasi masalah gradien yang menghilang (*vanishing gradient*) pada RNN dengan memisahkan memori dengan representasi *output*. LSTM telah terbukti bekerja sangat baik pada berbagai kasus, dan saat ini banyak digunakan.



Gambar 2.5 Arsitektur LSTM [15]

LSTM menggunakan tiga *gates* yaitu *input*, *forget*, dan *output* untuk mengatur berapa banyak representasi saat ini, sebelumnya, dan keluaran yang harus dilibatkan dalam catatan waktu saat ini [5]. Cara kerja arsitektur LSTM pada Gambar 2.5. Selanjutnya akan dijelaskan dengan pedoman notasi pada Gambar 2.6.



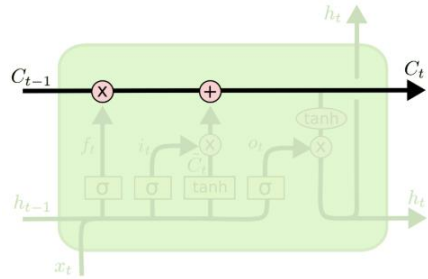
Gambar 2.6 Notasi Yang Dipakai [15]

Setiap garis pada notasi membawa seluruh vektor dari *output* suatu node ke *input* yang lain. Lingkaran merah muda mewakili operasi elemen, seperti perkalian atau penjumlahan antar elemen vektor. Sedangkan kotak kuning merupakan *neural network layer* (mengandung parameter dan bias) yang memiliki kemampuan belajar. Dua garis yang bergabung menandakan penggabungan dua matriks/vektor. Sementara garis berpisah

menandakan kontennya disalin dan salinannya pergi menuju *node* yang berbeda [15].

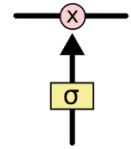
2.2.7.1 Mekanisme Kunci LSTM

Ide kunci dari LSTM adalah adanya jalur horizontal yang menghubungkan konteks lama (C_{t-1}) ke konteks baru (C_t) di bagian atas model LSTM, seperti Gambar 2.7.



Gambar 2.7 *Cell State* [15]

C_t disebut juga *cell state* atau *memory cell*. LSTM memiliki kemampuan untuk menghapus atau menambahkan informasi ke dalam *cell state*, diatur dengan hati-hati oleh struktur yang disebut *gates*.

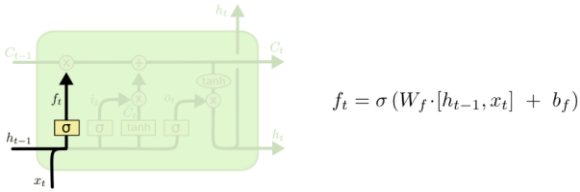


Gambar 2.8 Gerbang Sigmoid [15]

Gates adalah gerbang masuknya informasi secara opsional. Gerbang ini terdiri dari lapisan jaringan saraf sigmoid dan operasi multiplikasi yang searah (*pointwise operation*). *Output* dari lapisan jaringan saraf sigmoid ini bernilai antara nol dan satu, yang mendeskripsikan seberapa banyak komponen yang bisa lewat. Nilai nol berarti "tidak ada komponen yang bisa lewat", sedangkan nilai satu berarti "biarkan semuanya lewat". LSTM memiliki tiga *gates* berikut untuk mengontrol *cell state*.

2.2.7.2 *Forget Gate*

Langkah pertama dalam LSTM adalah memutuskan informasi apa yang akan dibuang dari *cell state*. Keputusan ini dibuat oleh lapisan sigmoid yang disebut "gerbang lupa" (*forget gate*, f_t) yang dapat dilihat pada Gambar 2.9. Gerbang ini membaca nilai h_{t-1} dan x_t, dan menghasilkan angka antara 0 dan 1 untuk setiap elemen dalam C_{t-1}. Nilai 1 artinya "benar-benar jaga elemen ini" sedangkan nilai 0 artinya "benar-benar singkirkan elemen ini."



Gambar 2.9 *Forget Gate* [15]

Notasi $[h_{t-1}, x_t]$ merupakan operasi konkatenasi; menambahkan baris dari h_{t-1} dengan baris dari x_t. Pada contoh model bahasa yang mencoba memprediksi kata berikutnya berdasarkan kata-kata sebelumnya.

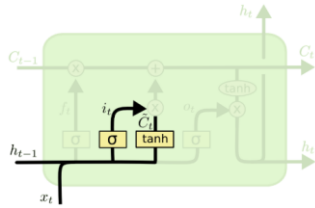
"George always helps everyone. Everyone likes ___."

Informasi gender subjek saat ini mungkin disimpan oleh suatu elemen dalam C_{t-1} sehingga kata ganti subjek yang benar dapat digunakan. Dalam contoh tersebut kata ganti subjek yang tepat adalah "*him*" karena George adalah nama pria. Ketika melihat subjek baru, maka elemen lama dalam C_{t-1} bisa dilupakan pada gerbang ini.

2.2.7.3 *Input Gate*

Langkah selanjutnya adalah memutuskan informasi baru apa yang akan disimpan ke *cell state*. Langkah ini memiliki dua bagian. Pertama, gerbang sigmoid yang disebut "gerbang masukan" (*input gate*, i_t) memutuskan nilai mana yang akan diperbarui (Gambar 2.10). Selanjutnya, *layer tanh* membuat

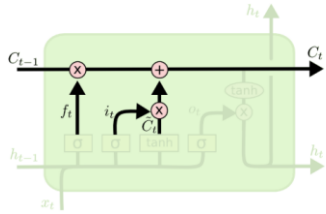
kandidat vektor konteks baru, $\tilde{C}_t$. Selanjutnya, kedua bagian digabungkan untuk membuat pembaruan konteks ke *cell state*.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Gambar 2.10 *Input Gate* [15]

Dalam contoh model bahasa sebelumnya, subjek dengan jenis kelamin baru ditambahkan ke konteks, untuk menggantikan elemen lama yang telah dilupakan pada *forget gate*.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

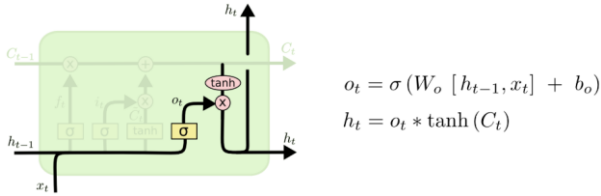
Gambar 2.11 Proses Memperbarui *Cell State* [15]

Kemudian saatnya untuk memperbarui konteks lama C_{t-1} ke konteks baru C_t. Kalikan konteks lama dengan f_t untuk melupakan hal-hal yang diputuskan untuk dilupakan. Kemudian kandidat konteks baru $\tilde{C}_t$ dikalikan dengan i_t untuk memutuskan seberapa banyak kandidat konteks baru akan disertakan. Lalu tambahkan keduanya.

2.2.7.4 *Output Gate*

Terakhir, apa yang menjadi *output* akan diputuskan. *Output* didasarkan pada nilai dalam konteks (*cell state*) yang dilewatkan terlebih dahulu ke suatu *filter*. Pertama, gerbang sigmoid yang dinamakan "gerbang luaran" (*output gate*, o_t) dijalankan untuk menentukan bagian-bagian apa dari konteks yang akan dijadikan *output*. Kemudian, konteks (*cell state*) dilewatkan melalui *tanh* untuk menjadikan nilainya antara −1

dan 1, dan mengalikannya dengan gerbang luaran (o_t) tadi, sehingga *output* hanya dihasilkan dari bagian yang diputuskan.



Gambar 2.12 *Output Gate* [15]

Pada contoh model bahasa sebelumnya, ketika subjek baru ditemukan, mungkin informasi yang relevan dari subjek tersebut harus disimpan untuk menghasilkan kata berikutnya. Misalnya apakah subjek tersebut tunggal atau jamak, informasi jenis kelaminnya, dan sebagainya.

2.2.8 *Word Embedding*

Word embedding adalah proses mengubah teks menjadi vektor angka dengan representasi numerik yang berbeda antar teks. Dengan *word embedding*, Kata-kata serupa diharapkan saling dekat di dalam suatu ruang vektor. Sebagai contoh pada kalimat *"The kid said he would grow up to be superman"* dan *"The child said he would grow up to be superman"*. Berdasarkan konteks, *"kid"* dan *"child"* merupakan dua kata yang berbeda namun memiliki makna yang sama. Sehingga, dua kata tersebut akan memiliki vektor kata yang menyerupai. Dengan *word embedding*, komputer dapat mengerti kata *"apple"* dalam kalimat *"apple is a tasty fruit"* merupakan buah apel yang bisa dimakan, bukan sebuah perusahaan teknologi ternama asal Amerika Serikat.

Peran *word embedding* diperlukan karena algoritme *deep learning* dan hampir semua aristektur *deep learning* hanya dapat memproses angka sebagai masukan untuk tugas-tugas seperti regresi, klasifikasi, dan lain-lain. Sehingga *dataset* teks yang

berbentuk *string* harus diubah terlebih dahulu menjadi vektor angka agar dapat diproses.

Sebelumnya telah ada metode vektorisasi lain seperti count vector dan TF-IDF untuk mengubah teks berupa *string* menjadi angka. Namun metode ini belum dapat merepresentasikan hubungan nilai semantik antar kata (representasi kontinu). Sehingga kemudian para peneliti mengembangkan berbagai tipe model *word embedding* untuk memberikan solusi representasi kata secara kontinu.

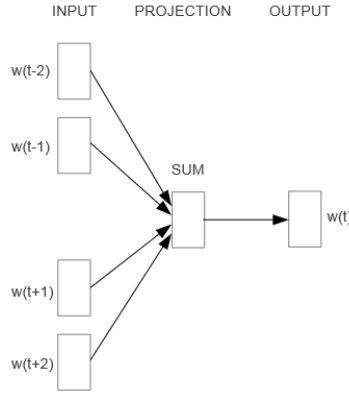
2.2.9 Word2Vec

Word2Vec merupakan proyek yang mengimplementasikan *word embedding* yang terbukti bekerja dengan menggunakan *dataset* besar dengan jutaan kata di dalam kamus. Proyek ini dikerjakan oleh Tomas Mikolov, dkk. Beberapa tipe model telah diajukan sebelumnya untuk merepresentasikan kata-kata secara kontinu, termasuk *Latent Semantic Analysis* (LSA) dan *Latent Dirichlet Allocation* (LDA). Word2Vec berfokus pada representasi terdistribusi dari kata-kata yang dilatih menggunakan *neural network*s, dimana sebelumnya telah terbukti bekerja lebih baik dibanding LSA dalam menjaga keteraturan linear antar kata dan lebih baik dari LDA dalam komputasi *dataset* yang besar [18].

Terdapat dua arsitektur model *log-linear* dalam Word2Vec untuk mempelajari representasi kata-kata terdistribusi yang mencoba meminimalisir kerumitan komputasi. Kerumitan sebelumnya disebabkan oleh *non-linear hidden layer* di dalam model *neural network*s. Oleh karena itu Mikolov dkk mencoba untuk mengeksplorasi model yang lebih sederhana, yang mungkin tidak dapat merepresentasikan data setepat *neural network*s, namun bekerja lebih efisien pada banyak data [18].

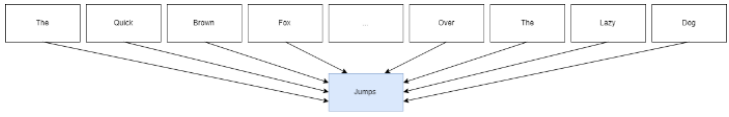
2.2.9.1 *Continuous Bag of Words* (CBOW) *Model*

Arsitektur ini mirip dengan *feedforward* NNLM, dimana *non-linear hidden layer* dihilangkan dan *projection layer* dibagi untuk semua kata. Dengan demikian, semua kata dapat diproyeksikan ke posisi yang sama dengan merata-ratakan nilai vektornya. Urutan kata-kata tidak mempengaruhi nilai proyeksi. Tidak seperti model *bag-of-words* standar, CBOW menggunakan representasi konteks yang didistribusikan secara kontinu [18].



Gambar 2.13 Arsitektur CBOW

Arsitektur CBOW memprediksi kata saat ini w(t) berdasarkan konteksnya. Cara kerja dari model arsitektur ini ditunjukkan pada Gambar 2.13. Kalimat *"The Quick Brown Fox X Over the Lazy Dog"* yang mana X merupakan kata target, kemungkinan adalah kata kerja yang berhubungan secara semantik.

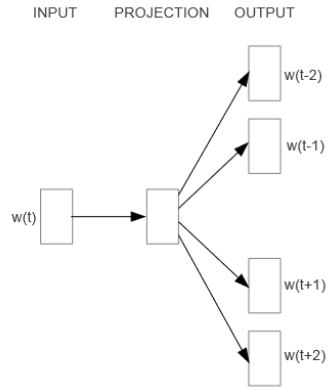


Gambar 2.14 Cara Kerja Model Arsitektur CBOW

Kemudian X diprediksi sebagai *"Jumps"* berdasarkan konteks yang diperoleh dari kata-kata disekitarnya.

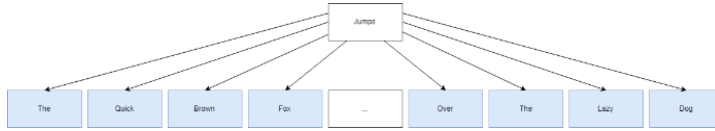
2.2.9.2 *Continuous Skip-gram Model*

Arsitektur kedua mirip dengan CBOW, namun alih-alih memprediksi kata saat ini berdasarkan konteksnya, *continuous skip-gram* mencoba memaksimalkan klasifikasi kata berdasarkan kata lain dalam kalimat yang sama. Setiap kata saat ini w(t) dijadikan *input* untuk klasifikasi log-linear menggunakan lapisan proyeksi kontinu (*continuous projection layer*), dan memprediksi kata-kata dalam rentang tertentu sebelum w(t-1, t-2, …, t-n) dan sesudah w(t+1, t+2, …, t+n) kata saat ini. Mikolov menemukan bahwa meningkatkan rentang kata dapat meningkatkan kualitas vektor yang dihasilkan, tetapi juga meningkatkan kompleksitas komputasi [18].



Gambar 2.15 Arsitektur *Continuous Skip-gram*

Skip-gram memprediksi kata-kata di sekitar kata saat ini w(t). Cara kerja dari model arsitektur ini dapat dilihat pada Gambar 2.15. Kata *"Jumps"* yang merupakan kata target/saat ini w(t) digunakan untuk memprediksi kata-kata konteks di sekitarnya.



Gambar 2.16 Cara Kerja Model Arsitektur *Continuous Skip-gram*

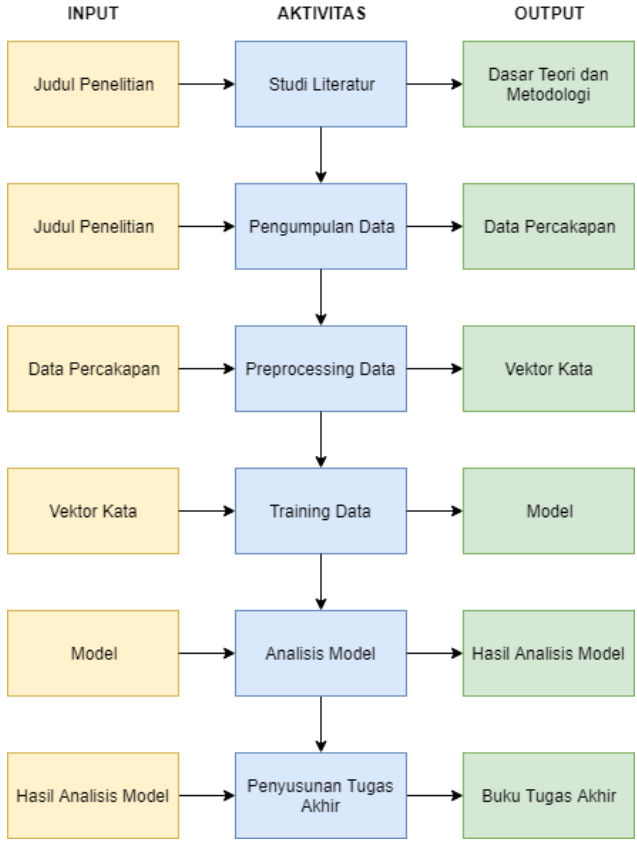
Halaman ini sengaja dikosongkan.

BAB III
METODOLOGI

Pada bagian ini dijelaskan metodologi yang akan digunakan sebagai panduan untuk menyelesaikan penelitian tugas akhir ini.

3.1 Diagram Metodologi

Pada sub bab ini akan dijelaskan mengenai tahapan yang dilakukan dalam penelitian sesuai Gambar 3.1.



Gambar 3.1 Metodologi Penelitian

3.1.1 Studi Literatur

Tahap ini dilakukan dengan tujuan memahami semua konsep, metode, dan algoritma yang digunakan di dalam penelitian. Pada tahap ini pula dilakukan penggalian kebutuhan terkait studi kasus yang akan diambil sebagai bekal dalam pengerjaan selanjutnya. *Paper* ilmiah yang dijadikan referensi utama merupakan *paper-paper* yang berasal dari luar negeri dan merupakan pengembangan terkini. Jurnal yang diambil adalah penelitian seputar *Long Short-Term Memory* (LSTM). Jurnal utama yang dijadikan rujukan tentang LSTM adalah *"A Comparative Study of Neural Network Models for Lexical Intent Classification"* oleh Suman Ravuri. Jurnal tersebut membahas mengenai cara klasifikasi *intent* menggunakan arsitektur *Long Short-Term Memory* (LSTM) dan membandingkan performanya dengan arsitektur *neural network* lain.

3.1.2 Pengumpulan Data

Penelitian ini menggunakan kumpulan data representasi vektor kata berbahasa Indonesia yang diekstraksi dari *pretrained word embedding model*. *Pretrained model* didapatkan dari arsip tugas akhir milik lab Akuisisi Data dan Diseminasi Informasi (ADDI) Departemen Sistem Informasi ITS. Model tersebut dihasilkan dari pelatihan model Word2Vec menggunakan data yang diperoleh dari Twitter dengan algoritme *Skip-gram*.

Selain data representasi vektor kata, penelitian ini menggunakan *dataset* percakapan berbahasa Indonesia untuk pelatihan model LSTM dan model *neural network* lain. Data yang digunakan merupakan data yang telah dikumpulkan dari arsip mata kuliah Pengolahan Bahasa Alami tahun 2018. Data yang dikumpulkan merupakan hasil simulasi percakapan antara penjual dan pembeli *smartphone* pada *e-commece*. Simulasi percakapan dibuat oleh setidaknya 30 mahasiswa yang pernah menggunakan fitur *chat* pada aplikasi *e-commerce*.

Pada *dataset* percakapan akan dipilih data ujaran berupa sapaan atau pertanyaan yang termasuk dalam topik untuk dimasukkan ke dalam *dataset* pelabelan. Sedangkan ujaran yang tidak

berupa sapaan atau pertanyaan dan yang tidak mengandung topik tidak dipilih. Tahap ini dilakukan untuk memudahkan pembuatan daftar kelas *intent* pada tahap selanjutnya. Contoh ujaran yang dimasukkan ke dalam *dataset* pelabelan dapat dilihat pada kolom kuning pada Tabel 3.1.

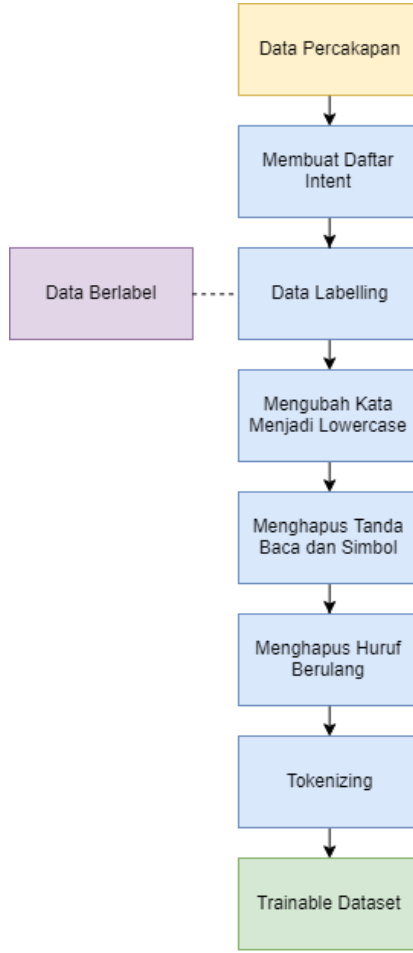
Tabel 3.1 Contoh Ujaran Pelabelan

Marker	Start
Pembeli	Pengiriman bisa lewat apa aja?
Penjual	Pos Indonesia dan JNE.
Pembeli	Oke, segera saya order.
Marker	End

Data percakapan yang telah dikumpulkan terbagi ke dalam beberapa *file* Excel dengan format tabel dan penulisan yang berbeda-beda. Nantinya, *file-file* tersebut akan disimpan ke dalam satu *file* Excel dengan format tabel dan penulisan yang seragam.

3.1.3 Pra-Pemrosesan Data

Data percakapan yang telah diperoleh akan masuk ke tahapan pra-pemrosesan data. Beberapa tahap pra-pemrosesan dilakukan satu per satu seperti pada Gambar 3.2.



Gambar 3.2 Tahap Pra-Pemrosesan Data

3.1.3.1 Membuat Daftar Kelas *Intent*

Pada tahap ini daftar *intent* ditentukan berdasarkan *dataset* percakapan. Daftar ini nantinya akan dijadikan acuan dalam proses pelabelan data.

3.1.3.2 *Data Labeling*

Tahap selanjutnya yaitu memberikan label *intent* pada setiap data dalam *dataset* pelabelan. Proses ini *menggunakan Human Intelligence Task* (HIT). Setiap HIT akan dilakukan oleh satu orang *annotator*. Dimana *annotator* melabeli ujaran dengan salah satu kelas *intent* dari daftar kelas *intent* yang telah dibuat sebelumnya. Berikut merupakan contoh proses pelabelan yang dapat dilihat pada Tabel 3.2.

Tabel 3.2 Contoh Hasil Pelabelan

Ujaran	Label *Intent*
Ukuran layarnya berapa ya?	Spesifikasi
Ini barang baru atau second?	Kondisi
Kira-kira baterainya tahan berapa jam?	Spesifikasi

3.1.3.3 Mengubah Kata Menjadi *Lowercase*

Setiap kata pada teks dalam *dataset* percakapan diubah menjadi *lowercase*/huruf kecil. Sehingga nantinya pada tahap pemrosesan data tidak terjadi ambiguitas antar kata yang sama namun berbeda penggunaan huruf kapital.

3.1.3.4 Menghapus Tanda Baca dan Simbol

Pada tahap ini, semua tanda baca dan simbol-simbol pada setiap teks dalam *dataset* percakapan yang tidak memiliki arti akan dihapus, dan sisanya akan dipisahkan dari kata yang mendahuluinya. Sehingga menghapus peluang adanya ambiguitas yang disebabkan kata-kata yang memiliki makna sama namun berbeda penulisan karena penambahan tanda baca.

3.1.3.5 Menghapus Huruf Berulang

Pada percakapan sehari-hari Bahasa Indonesia, sering ditemukan penulisan kata dengan huruf yang berulang seperti “okeeee”. Kata dengan huruf berulang tersebut akan diubah dengan batasan perulangan menjadi maksimal dua huruf. Sehingga kata “okeeee” akan diubah menjadi “okee”. Hal ini

dilakukan untuk mengatasi ambiguitas kata-kata dengan makna sama namun penulisan berbeda karena adanya huruf berulang.

3.1.3.6 *Tokenizing*

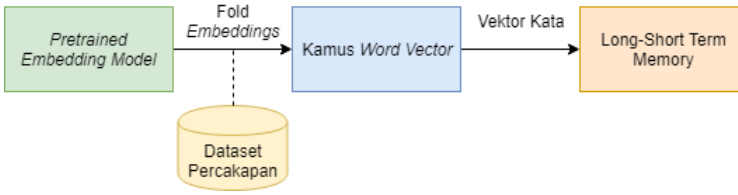
Tokenizing adalah proses membuat *token*/memisahkan kata per kata. *Token-token* tersebut akan digunakan untuk merepresentasikan sebuah kata ke dalam ruang vektor menggunakan model Word2Vec.

3.1.4 *Data Splitting*

Pada tahap ini *dataset* yang telah dilakukan tahap pra-pemrosesan dibagi menjadi 3 bagian yaitu *train set, dev set* dan *test set*. *Train set* digunakan untuk pelatihan model, *dev set* digunakan untuk mencari model terbaik, sedangkan *test set* digunakan untuk menguji kualitas dan performa dari model.

3.1.5 Implementasi Model *Word Embedding*

Pada tahap ini dilakukan implementasi *pretrained word embedding model* untuk data yang telah melewati tahap pra-pemrosesan. Implementasi bertujuan untuk mengubah setiap kata pada *dataset* percakapan menjadi nilai vektor agar dapat diproses dalam arsitektur LSTM dan arsitektur *neural network* lainnya.



Gambar 3.3 Proses *Word Embedding*

Word embedding dilakukan dengan memanfaatkan *library* Gensim yang diimplementasikan menggunakan Python.

3.1.6 Pembuatan Model LSTM dan Model *Neural Network* Lain

Pada tahap ini akan dilakukan pelatihan model *Long Short-Term Memory* (LSTM) dan model *neural network* lain pada *train set*

menggunakan *library* Pytorch. Pada setiap *epoch*, akurasi model diuji pada *dev set* menghasilkan *dev accuracy*. Model dengan *dev accuracy* tertinggi untuk setiap percobaan akan disimpan.

3.1.7 Analisis dan Pengujian Performa Model LSTM dan Model *Neural Network* Lain

Model-model LSTM dan *neural network* lain yang telah dibuat kemudian diuji dan dievaluasi pada *test set* yang belum pernah dilihat dan dipelajari sebelumnya. Metrik yang digunakan untuk menilai performa model yaitu *test accuracy*, *precision, recall*, dan *F1-score*. Nilai pada metrik tersebut berkisar antara 0-1. Semakin besar nilai *test accuracy*, *precision, recall,* dan *F1-score*, berarti kualitas model yang dihasilkan akan semakin baik.

3.1.8 Penyusunan Laporan Tugas Akhir

Tahapan terakhir adalah penyusunan laporan tugas akhir sesuai format yang telah ditentukan sebagai bentuk dokumentasi atas terlaksananya tugas akhir. Laporan diharapkan dapat bermanfaat sebagai referensi untuk pengerjaan penelitian lain, serta sebagai acuan untuk pengembangan lebih lanjut terhadap topik penelitian yang serupa.

Halaman ini sengaja dikosongkan.

BAB IV
PERANCANGAN

Pada bab ini akan dijelaskan mengenai rancangan untuk setiap proses yang dilakukan untuk mendapatkan luaran penelitian tugas akhir ini. Perancangan yang dibuat antara lain rancangan pengumpulan data, rancangan pra-pemrosesan data, rancangan *data splitting*, rancangan implementasi model *word embedding*, rancangan pembuatan model LSTM dan *neural network* lain, dan rancangan analisis dan pengujian performa model LSTM dan *neural network* lain.

4.1 Perancangan Pengumpulan Data

Model LSTM dan *neural network* lain yang akan dibuat bertujuan untuk melakukan klasifikasi *intent* pada ujaran pembeli. Sehingga, data yang akan digunakan untuk proses pelatihan dan evaluasi model hanya data teks ujaran yang berasal dari pembeli dan berupa sapaan atau pertanyaan. Untuk memilah data tersebut dari tumpukan percakapan, dilakukan pembuatan kode Python sederhana dengan memanfaatkan fungsi standar seperti *startswith* dan memanfaatkan *library* NumPy dan Pandas. NumPy adalah salah satu *library* Python yang populer untuk pemrosesan angka, sedangkan Pandas dapat mengubah *file* CSV atau Excel menjadi tabel yang disebut *dataframe* dan melakukan pengoperasian pada baris dan kolom pada *dataframe* tersebut. Contoh ujaran yang digunakan pada penelitian dapat dilihat pada kolom kuning pada Tabel 4.1.

Tabel 4.1 Contoh Ujaran Yang Digunakan Pada Penelitian

Marker;START
Pembeli;mas hp ready?
Penjual;ready kak
Pembeli;hijau lumutnya ada ?
Penjual;ada mas
Pembeli;dikirim hari ini bisa ?
Penjual;bisa kak kalau pembayaran sebelum jam 1 siang
Pembeli;oke
Penjual;siap

MARKER;END
MARKER;START
Pembeli;Mas redmi 5a masih ada kah?
Penjual;masih
Pembeli;garansinya TAM kan mas?
Penjual;iya
Pembeli;Oh iya harganya sesuai dengan iklan itu yaa?
Penjual;iya
Pembeli;bisa cod mas?
Penjual;bisa
Pembeli;cod di juanda mas
Penjual;sekitar jam 3 sore ya
Pembeli;oke mas di tunggu
Penjual;terimakasih
MARKER;END

Data percakapan berbahasa Indonesia yang dikumpulkan terbagi menjadi beberapa *file* dengan format yang berbeda-beda. Oleh karena itu untuk memudahkan proses selanjutnya, dilakukan penyeragaman format data pada setiap *file*. Penyeragaman format data akan dilakukan menggunakan kode Python sederhana dengan memanfaatkan *library* NumPy dan Pandas. Contoh beberapa format data dapat dilihat pada Tabel 4.2 dan Tabel 4.3.

Tabel 4.2 Contoh Format Data

Pembeli;warna hitam masih ada gan ?
Penjual;ready kak
Pembeli;garansi resmi TAM ya?
Penjual;TAM
Pembeli;ongkir brapa mas?
Penjual;bisa langsung cek di websitenya mas
Pembeli;oke gan
Penjual;oke
MARKER;END

Tabel 4.2 merupakan salah satu contoh format data sebelum dilakukan penyeragaman. Tabel tersebut hanya memiliki satu kolom. *Marker* penanda awal percakapan, akhir percakapan,

dan pemberi ujaran berada pada awal ujaran dengan menambahkan tanda *semicolon*. Sehingga agar formatnya sama, *marker* di awal setiap kalimat akan dihapus.

Tabel 4.3 Contoh Format Data

marker	start
Pembeli	Misi gan,hp Oppo F1 ready?
Penjual	ready kak, mau yang warna apa?
Pembeli	yang silver metallic ada gan?
Penjual	Ada kak, apakah sekalian beli case nya?
Pembeli	Iya gan
Penjual	silakan order saja kak
marker	end

Tabel 4.3 merupakan contoh format data lain sebelum dilakukan penyeragaman. Berbeda dengan contoh pada Tabel 4.2, format data ini memiliki dua kolom yaitu *marker* untuk penanda awal percakapan, akhir percakapan, dan siapa yang memberi ujaran. Data yang dibutuhkan dalam proses *training* hanya teks ujarannya saja sehingga kolom *marker* akan dihapus.

Setelah setiap data memiliki format yang sama, data perlu digabung ke dalam satu *file* Excel. Penggabungan dilakukan secara manual menggunakan *copy-paste* pada *clipboard.* Hasil akhir yang diharapkan pada tahap ini adalah satu file Excel yang berisi seluruh data ujaran pembeli.

4.2 Perancangan Pra-Pemrosesan Data

Sebelum diproses, data harus terlebih dahulu mengalami pra-pemrosesan. Ada beberapa tipe proses dalam tahap pra-pemrosesan data yang bertujuan untuk mengubah data mentah ke dalam format yang lebih mudah dan efektif untuk diolah saat pembuatan dan evaluasi model pada langkah selanjutnya. Untuk melakukan seluruh tipe proses tersebut, dibuat kode Python dengan memanfaatkan *library* Pandas dan *regular expression*. Berikut merupakan perancangan yang dilakukan pada tahap pra-pemrosesan data.

4.2.1 Perancangan Daftar Kelas *Intent*

Sebelum melakukan *data labeling*, perlu dilakukan pembuatan daftar kelas *intent* apa saja yang terdapat pada *dataset* percakapan. Daftar kelas *intent* yang dibuat nantinya akan dijadikan patokan dalam proses *data labeling*. Sehingga label yang boleh diberikan hanyalah label kelas *intent* yang terdapat pada daftar. Daftar kelas *intent* juga perlu dievaluasi lagi pada tahap pengujian model. Jika terdapat kelas dengan performa sangat rendah, *class merging* dapat dilakukan.

4.2.2 Perancangan *Data Labeling*

Pada tahap ini dilakukan pemberian label pada seluruh data teks ujaran dalam *dataset* percakapan. Untuk melakukannya, satu kolom ditambahkan pada *dataset* percakapan untuk diisi dengan label kelas *intent* dari setiap teks ujaran. *Data labeling* dilakukan menggunakan daftar kelas *intent* yang dibuat pada sub bab 4.2.1.

4.2.3 Perancangan Pengubahan Kata Menjadi *Lowercase*

Untuk mengubah setiap kata pada *dataset* menjadi *lowercase*, perlu dibuat kode Python yang memanfaatkan fungsi *lower*. Fungsi ini diterapkan pada setiap baris data menggunakan library Pandas.

4.2.4 Perancangan Penghapusan Huruf Berulang

Untuk melakukan penghapusan huruf berulang, perlu dibuat kode Python sederhana yang memanfaatkan fungsi *sub* milik *library regular rxpression*. Dengan fungsi tersebut, kata dapat dibatasi maksimal memiliki dua huruf berulang.

4.2.5 Perancangan Penghapusan Tanda Baca dan Simbol

Dalam teks percakapan antara penjual dan pembeli di *e-commerce*, kemungkinan mengandung berbagai simbol, tanda baca, dan angka yang menempel pada kata. Beberapa simbol, tanda baca, dan angka tersebut tidak memiliki arti atau kurang berpengaruh dalam proses *training* sehingga perlu dihapus. Sedangkan beberapa simbol lainnya seperti titik, koma, dan

tanda tanya hanya perlu dipisahkan dari kata yang mendahuluinya, sehingga selanjutnya pada proses tokenisasi tanda baca dan simbol tersebut akan menjadi token sendiri.

Untuk menghapus atau memisahkan tanda baca, simbol, dan angka, dilakukan proses *substitute* menggunakan *regular expression* dalam Python. Simbol-simbol yang akan dihilangkan antara lain.

- Menghapus semua karakter ASCII, seperti “$”, ”~”, ”@”
- Menghapus semua karakter non-ASCII, seperti “é”,”ö”, “ĉ”
- Menghapus semua karakter angka, seperti 1, 2, 3, 4, 5, 6, 7, 8, 9, 0

4.2.6 Perancangan *Tokenizing*

Model LSTM dan model *neural network* lain memproses kalimat *input token* per *token*. Sehingga, perlu dilakukan *tokenizing* setiap kalimat pada *dataset* percakapan. Kode Python yang dibuat menggunakan fungsi *split* dengan memanfaatkan *library* Pandas untuk menerapkan fungsi tersebut pada setiap baris data.

4.3 Perancangan *Data Splitting*

Pembagian *dataset* menjadi *train set*, *dev set*, dan *test set* dilakukan dengan rasio 80:10:10. Selain karena rasio ini umum digunakan, 80% *dataset* untuk *train set* dilihat cukup baik jika menimbang jumlah data yang sedikit. Untuk melakukan pembagian tersebut dibuat kode Python yang memanfaatkan *library* Pandas, *random*, dan Torchtext. Pandas berfungsi untuk melakukan pemrosesan-pemrosesan umum pada *dataset*. *Random* digunakan untuk melakukan *shuffle* pada *dataset* menggunakan *seed*. Sedangkan Torchtext digunakan untuk menyimpan *train set*, *dev set*, dan *test set* akhir.

4.4 Perancangan Implementasi Model *Word Embedding*

Pre-trained word embedding model yang diperoleh dari penelitian sebelumnya digunakan untuk membuat kamus *word vector* dari seluruh kata yang terdapat pada *dataset* percakapan. Tidak semua kata dari *dataset* percakapan diketahui oleh model

word embedding. Sehingga kata yang tidak diketahui tersebut akan diberi nilai vektor secara *random*. Kamus *word vector* akan dijadikan referensi *initial weights* untuk setiap input kata dalam pembuatan model *Long Short-Term Memory*.

4.5 Perancangan Pembuatan Model LSTM dan Model *Neural Network* Lain

Train set dan *initial weights* yang telah diekstrak dari *pre-trained embedding* dijadikan masukan untuk *training* model *Long Short-Term Memory* (LSTM) dan model *neural network* lain. Model *neural network* lain yang akan dibuat pada penelitian ini yaitu *standard Recurrent Neural Network* (RNN), *Gated Recurrent Unit* (GRU) dan *Bidirectional Long Short-Term Memory* (BiLSTM). Pembuatan keempat model untuk tugas klasifikasi *intent* akan memanfaatkan *library* Pytorch. *Library* ini memiliki kelebihan yaitu prosesnya dapat dijalankan di GPU sehingga mempercepat durasi *training* dengan signifikan.

Ada beberapa skenario percobaan yang akan dilakukan dengan berbagai kombinasi yang meliputi jenis *optimizer*, nilai *learning rate* dan nilai *hyperparameter*. Percobaan dengan beberapa skenario ini dilakukan untuk mengetahui kombinasi *optimizer*, *learning rate*, dan *hyperparameter* apa yang paling cocok untuk setiap jenis arsitektur model.

Beberapa *optimizer algorithm* yang akan dicoba yaitu *Stochastic Gradient Descent* (SGD) dan *momentum* yang telah teruji coba bekerja dengan baik pada *dataset* Cortana dan ATIS [6]. Selain itu, algoritme terbaru seperti *Adaptive Moment Estimation* (ADAM) juga akan diuji coba.

Sedangkan untuk nilai *learning rate* yang akan dicoba antara lain:

- 0,001
- 0,01
- 0,015

Hyperparameter yang tepat dapat memberi dampak yang signifikan terhadap performa model. Berikut *hyperparameter* yang dimiliki keempat model.

Tabel 4.4 *Hyperparameter* Model

Parameter	Definisi
Input_size	Jumlah *expected features* dalam input x
Hidden_size	Jumlah fitur dalam *hidden state* h
Num_layers	Jumlah *recurrent layers*. Contoh, konfigurasi num_layers=2 berarti dua LSTM ditumpuk untuk membentuk *stacked* LSTM, dengan LSTM kedua mengambil output dari LSTM pertama untuk menghitung hasil akhir. Default: 1
Bias	Jika *false*, maka layer tidak menggunakan *bias weights b_ih* dan *b_hh*. Default: *true*
Batch_first	Jika *true*, maka tensor input dan output disajikan dengan bentuk (batch, seq, feature). Default: *false*
Dropout	Nilai indeks untuk proses *dropout* saat *training* model
Bidirectional	Jika *true*, model akan menjadi *bidirectional* LSTM

Pada penelitian ini, parameter yang diuji coba adalah *num layers* dengan nilai sebagai berikut:

- 1
- 2
- 3

Kombinasi jenis *optimizer*, nilai *learning rate*, dan jumlah *num layers* untuk setiap arsitektur menghasilkan 108 skenario pembuatan model. Proses *training* seluruh skenario akan dilakukan dengan 150 *epoch* dan menyimpan model dengan *dev accuracy* tertinggi. *train accuracy*, *dev accuracy*, dan durasi *training* disimpan agar dapat dianalisis pada tahap selanjutnya.

4.6 Perancangan Analisis dan Pengujian Performa Model LSTM dan Model *Neural Network* Lain

Tahap ini dilakukan untuk menguji dan mengevaluasi kualitas model. Setiap model dari hasil *training* seluruh skenario diuji pada *test set*. Untuk melakukan pengukuran performa setiap model, digunakan tiga metrik berbeda yaitu *precision*, *recall*, dan *F1-score*. Metrik tersebut dapat memberikan pandangan mengenai performa dari setiap *intent* pada model, maupun performa model secara keseluruhan.

Precision mengidentifikasi frekuensi prediksi yang benar, pada setiap prediksi *intent* A. *Precision* menjawab pertanyaan "Dari semua prediksi *intent* A, berapa banyak yang benar?". Berikut rumus dari *precision*.

$$\text{Precision} \triangleq \frac{tp}{tp + fp}$$

Gambar 4.1 Rumus precision

Recall mengidentifikasi berapa kali model dapat mengidentifikasi *intent* A dengan benar, dari seluruh data test dengan *actual intent* A. Berikut rumus dari *recall*.

$$\text{Recall} \triangleq \frac{tp}{tp + fn}$$

Gambar 4.2 Rumus recall

Sedangkan *F1-score* menghitung rata-rata harmonis dari *precision* dan *recall*. *F1-score* membantu menjawab pertanyaan "bagaimana performa prediksi model yang berkaitan dengan *intent* A?". Berikut rumus dari *F1-score*.

$$\text{F1-score} \triangleq 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Gambar 4.3 Rumus F1-score

Karena *intent* berjumlah 16, maka untuk mendapatkan nilai *precision*, *recall*, dan *F1-score* dari setiap model secara global, perlu dilakukan perhitungan *weighted average* pada setiap metrik.

Selain metrik tersebut, performa model juga diukur menggunakan nilai akurasi. Namun, akurasi bisa menyesatkan karena tidak menceritakan keseluruhan keadaan, terutama ketika jumlah data pada setiap kelas *intent* relatif tidak seimbang. Jika 80% dari *test set* adalah data dengan *intent* A, dan model dapat memprediksi seluruh data dengan *intent* tersebut dengan benar, maka nilai akurasi adalah 0,8. Nilai tersebut tentunya tidak mencerminkan kenyataan.

Dari hasil pengujian seluruh model dapat dilakukan beberapa analisis berikut:

- *Optimizer*, *learning rate*, dan *num layer* terbaik untuk setiap arsitektur model.
- Arsitektur dengan performa terbaik pada data percakapan yang dimiliki.
- Pengaruh jumlah data dan jumlah *word vocab* terhadap performa kelas *intent.*
- Pengaruh *word embedding* terhadap performa model LSTM dan model *neural network* lain.

Halaman ini sengaja dikosongkan.

BAB V
IMPLEMENTASI

Pada bab ini, akan dijelaskan mengenai implementasi dari perancangan yang telah dilakukan sesuai dengan metode pengembangan yang dibuat. Bagian yang dijelaskan yaitu lingkungan implementasi, pembuatan fitur-fitur aplikasi dalam bentuk kode, serta pengujian aplikasi.

5.1 Lingkungan Implementasi

Pengerjaan penelitian ini menggunakan 2 komputer dengan spesifikasi seperti yang tertera pada Tabel 5.1 dan Tabel 5.2.

Tabel 5.1 Spesifikasi *Hardware* Komputer 1

Processor	Intel i5-8250U
Memory	8 GB DDR4 *Memory*
GPU	Nvidia GeForce MX150
Sistem Operasi	Windows 10 Home
Arsitektur Sistem	64-bit *Operating System*, x64-*based processor*

Tabel 5.2 Spesifikasi *Hardware* Komputer 2

Processor	Intel i5-6500
Memory	16 GB DDR4 *Memory*
GPU	Nvidia GeForce GTX 1060 6GB
Sistem Operasi	Windows 10 Home
Arsitektur Sistem	64-bit *Operating System*, x64-*based processor*

Aplikasi dikembangkan dengan menggunakan beberapa library, bahasa pemrograman, dan perangkat lunak yang disajikan pada Tabel 5.3.

Tabel 5.3 *Library* Yang Digunakan

Bahasa Pemrograman	Python
Computational Environment	Jupyter Notebook
Virtual Environment	Anaconda
Library	• Pytorch • Pandas • Numpy • Gensim • Sklearn • Random • Time • Math • Torchtext • Copy

5.2 Pengumpulan Data

Tahap ini memiliki tujuan untuk menggabungkan *dataset* percakapan dari beberapa *file* ke dalam satu *file* dengan format yang sama. Library yang digunakan pada tahap ini yaitu NumPy dan Pandas.

Setiap file memiliki format yang berbeda-beda sehingga perlu disimpan ke dalam format yang sama sebelum digabung menjadi satu. Kode-kode pada tahap ini menjelaskan proses transformasi setiap file menjadi format yang sesuai.

```
1.  df = pd.read_excel('D:/Tugas Akhir/Conversations/Nor
    malized/5-
    normalized.xlsx', index_col=None, header=None)
2.
3.  def remove_sender(x):
4.      if x.startswith('Pembeli;'):
5.          return x[8:]
6.      elif x.startswith('Pembeli'):
7.          return x[7:]
8.      else:
9.          return x
```

```
10.
11. def clean_str(string):
12.     string = re.sub(r"\t", "", string)
13.     string = re.sub(r";", "", string)
14.     return string
15.
16. df[0] = df[0].apply(remove_sender).apply(clean_str)

17. df = df[~df[0].astype(str).str.startswith(('Marker',
     'Penjual'))]
18. df.dropna(inplace=True)
19.
20. #save
21. df.to_excel('D:/Tugas Akhir/Conversations/Normalized
    /new_5-normalized.xlsx', index=False, header=False)
```

Kode 5.1 Penghapusan *Sender Marker* Pada Kalimat Ujaran

Pada beberapa *file*, kalimat ujaran diawali dengan *sender marker* untuk menandai siapa yang memberi ujaran tersebut. *Marker* tersebut harus dihapus sebelum ujaran digunakan baik untuk pelatihan maupun evaluasi model. Kode 5.1 menjelaskan proses penghapusan *marker* pembeli dan penjual pada setiap ujaran. Dengan potongan kode di atas, ujaran "Pembeli;Kalau garansinya langsung dari *brand* atau distributor sih?" akan ditransformasi menjadi "Kalau garansinya langsung dari *brand* atau distributor sih?".

Setelah *marker* dihapus, seluruh ujaran dari pembeli diambil untuk dimasukkan ke tabel baru menggunakan kode pada baris ke 17. Selanjutnya, tabel tersebut disimpan ke dalam *file* Excel.

```
1.  df = pd.read_excel('D:/Tugas Akhir/Conversations/Nor
    malized/10-
    normalized.xlsx', index_col=None, header=None)
2.
3.  df_new = df.copy()
4.  df_new = df_new[df_new[0] == 'Pembeli']
5.  df_new = df_new.loc[:, 1]
6.
7.  #save
```

```
8.  df_new.to_excel('D:/Tugas Akhir/Conversations/Normal
    ized/new_10-
    normalized.xlsx', index=False, header=False)
```

Kode 5.2 Pengambilan Ujaran Dengan Marker Pembeli

Pada *file* lain, *marker* ‘penjual’ dan ‘pembeli’ terdapat pada kolom tabel yang berbeda. Sehingga, *marker* tersebut tidak perlu dihapus karena tidak menjadi satu dengan kalimat ujaran. Kode 5.2 menjelaskan proses pengambilan seluruh ujaran dari pembeli dan memasukkannya ke tabel baru. Selanjutnya, tabel tersebut disimpan ke dalam *file* Excel.

5.3 Pra-Pemrosesan Data

Tahap pra-pemrosesan data dimulai dari membuat daftar kelas *intent*, memberi label setiap data dengan satu kelas *intent* dari daftar yang telah dibuat. Setelah data telah berlabel, seluruh teks ujaran diubah menjadi *lowercase*. Huruf berulang, tanda baca, dan simbol yang terdapat pada teks dihapus atau dipisahkan dari kata yang mendahuluinya. Lalu dilakukan *tokenizing* untuk mengubah teks ujaran menjadi *token-token* sehingga dapat diproses pada tahap selanjutnya. Berikut penjelasan proses implementasi pra-pemrosesan data secara detail.

5.3.1 Pembuatan Daftar Kelas *Intent*

Karena tugas klasifikasi *intent* merupakan tugas yang termasuk dalam *supervised learning*, daftar kelas *intent* dibuat terlebih dahulu secara manual dengan kecerdasan manusia. Untuk membuat daftar tersebut, penulis membaca seluruh teks ujaran pada *dataset*. Sembari membaca, kandidat kelas *intent* yang memungkinkan dicatat pada daftar. Setelah proses pembacaan *dataset* selesai, setiap kelas *intent* pada daftar diberi deskripsi, termasuk ujaran apa saja yang dapat dikategorikan ke dalam kelas tersebut. Deskripsi ini penting untuk memudahkan proses *data labeling* pada tahap selanjutnya.

5.3.2 *Data Labeling*

Pada tahap ini, setiap ujaran pembeli diberi label kelas *intent* yang sesuai. Penentuan label mengacu pada daftar kelas *intent*

yang dibuat pada sub bab 5.3.1. Pemberian label dilakukan oleh penulis secara manual dengan menimbang kecocokan setiap ujaran dengan deskripsi pada daftar kelas *intent*.

5.3.3 Pengubahan Kata Menjadi *Lowercase*

Pada tahap ini dilakukan proses *case folding* yaitu mengubah setiap huruf pada ujaran menjadi *lowercase*.

```
1. df = pd.read_excel('D:/Tugas Akhir/Conversations/Normalized/merge_normalized_1800_new.xlsx', index_col=None, header=None)
2. df.columns = ['text','tag']
3.
4. df['text'] = df['text'].str.lower()
```

Kode 5.3 *Case Folding*

Kode 5.3 merupakan potongan kode yang digunakan pada tahap ini. *Dataset* yang berbentuk *file* Excel diimpor dan diubah ke dalam *dataframe* Pandas. Kemudian kedua kolom diberi nama *text* dan *tag*. *text* adalah kolom yang berisi teks ujaran pembeli, sedangkan *tag* merupakan label *intent* dari teks tersebut. Dengan memanfaatkan fungsi *lower*, seluruh ujaran pada kolom *text* diubah menjadi *lowercase*.

5.3.4 Penghapusan Huruf Berulang

Huruf berulang pada ujaran dibatasi maksimal hanya 2 huruf. Untuk melakukannya, dibuat fungsi Python dengan memanfaatkan *library regular expression*.

```
1. def replace_mult_occurences(string):
2.     return re.sub(r'(.)\1{2,}', r'\1\1', string)
3.
4. df['text'] = df['text'].apply(replace_mult_occurences)
```

Kode 5.4 Penghapusan Huruf Berulang

Kode 5.4 menjelaskan tentang proses penghapusan huruf berulang. Fungsi *replace_mult_occurences* pada baris ke 1-2 mengambil argument ujaran untuk diproses menggunakan *regular expression*. Karakter yang berulang lebih dari dua pada

ujaran diganti/*substitute* dengan dua karakter saja. Fungsi *replace_mult_occurences* diterapkan pada setiap ujaran pada kolom *text* menggunakan fungsi *apply* milik Pandas, seperti pada baris ke 4. Disini salah satu kelebihan *library* Pandas terlihat, yaitu untuk menerapkan fungsi pada setiap baris data tidak perlu dilakukan perulangan menggunakan *for each*.

5.3.5 Penghapusan Tanda Baca dan Simbol

Pada tahap ini dilakukan proses penghapusan tanda baca, simbol, dan angka pada setiap kata dalam ujaran. Proses ini dilakukan agar kamus kata tidak berisi kata-kata yang sama dengan tanda baca berbeda. Untuk melakukannya, dibuat fungsi yang memanfaatkan *library regular expression*.

```
1.  def clean_str(string):
2.      string = re.sub(r"\.", " . ", string)
3.      string = re.sub(r",", " , ", string)
4.      string = re.sub(r":", " : ", string)
5.      string = re.sub(r";", " ; ", string)
6.      string = re.sub(r"!", " ! ", string)
7.      string = re.sub(r"\?", " ? ", string)
8.      string = re.sub(r"\(", " ( ", string)
9.      string = re.sub(r"\)", " ) ", string)
10.     string = re.sub(r"#", " <hash_tag> ", string)
11.     string = re.sub(r"\[", " [ ", string)
12.     string = re.sub(r"\]", " ] ", string)
13.     string = re.sub(r"[^A-Za-
    z09().,<_>!?\'`]", " ", string)
14.     string = re.sub(r"\s{2,}", " ", string)
15.     return string
16.
17. df['text'] = df['text'].apply(clean_str)
```

Kode 5.5 Penghapusan Tanda Baca dan Simbol

Kode 5.5 menjelaskan tentang proses penghapusan tanda baca dan simbol. Dibuat fungsi *clean_str* yang berisi tanda baca dan simbol yang perlu dihapus. Pengecualian untuk tanda baca/simbol yang menempel pada kata, tanda baca/simbol tersebut tidak dihapus melainkan dipisahkan dari kata yang menempel, seperti pada baris ke 2-12. Untuk memisahkannya, setiap tanda baca/simbol diganti/*substitute* dengan *string* tanda

baca/simbol tersebut yang ditambahkan spasi di depan dan belakangnya. Fungsi *clean_str* diterapkan pada seluruh ujaran dalam kolom *text* menggunakan fungsi *apply* milik Pandas.

5.3.6 *Tokenizing*

Agar dapat diproses baik dalam pelatihan maupun evaluasi model, setiap ujaran harus diubah menjadi *token-token* terlebih dahulu. Untuk melakukannya, dibuat kode Python yang dapat dilihat pada Kode 5.6.

```
def split_text(x):
    return x.split()

df['text_splitted'] = df['text'].apply(split_text)
```

Kode 5.6 *Tokenizing*

Kode di atas hanya memanfaatkan fungsi *split* Python yang diterapkan pada setiap ujaran dalam kolom *text*. Hasil ujaran yang telah berupa *token-token* kemudian disimpan pada kolom baru bernama *text_splitted*.

5.4 *Data Splitting*

Pada tahap ini, *dataset* dibagi menjadi tiga bagian/*set* yaitu *train set*, *dev set*, dan *test set*. Pembagian dilakukan menggunakan metode *stratified sampling*, dimana rasio tersebut diterapkan untuk setiap kelas *intent*. Metode ini dipilih agar setiap set memiliki data dengan porsi *intent* yang berimbang.

```
# build list per intent
all_set = []
intent = {'ketersediaan':[],
          'harga': [],
          'pengiriman':[],
          'spesifikasi':[],
          'garansi':[],
          'sapaan':[],
          'pembayaran':[],
          'tipe':[],
          'kelengkapan':[],
          'kondisi':[],
          'aksesori':[],
```

```
            'keaslian':[],
            'toko':[],
            'transaksi':[],
            'pengembalian':[],
            'service':[]
        }

random.seed(4)

for key, value in intent.items():
    temp = new_df[new_df['tag'] == key]
    temp['tag_list'] = temp['tag'].apply(to_list)

    for index, row in temp.iterrows():
        value.append((row['text_splitted'], row['tag
_list']))

    random.shuffle(value)
    all_set+=value

    print("{0} length:".format(key),len(value))
```

Kode 5.7 Pengelompokkan Ujaran Berdasarkan Kelas *Intent*

Kode 5.7 menjelaskan tahap awal dalam proses *stratified sampling* yaitu mengelompokkan setiap ujaran berdasarkan kelas *intent*. Setelah setiap ujaran dikelompokkan ke dalam *list intent* masing-masing, *list* tersebut diacak menggunakan fungsi *shuffle* milik *library* random pada python. *Shuffle* dilakukan menggunakan *seed* dengan agar urutan ujaran dalam *list* konsisten setiap kali kode dijalankan.

```
# split data
train_ratio = 0.8
test_ratio = 0.2

train_size = int(train_ratio * len(all_set))
test_size = len(all_set) - train_size

train_set = []
dev_set = []
test_set = []

for key, value in intent.items():
```

```
13.     train_size = int(train_ratio * len(intent[key]))
14.     test_size = int((len(intent[key]) - train_size)/
    2)
15.
16.     train_data = intent[key][:train_size]
17.     train_set += train_data
18.
19.     test_data = intent[key][train_size:]
20.
21.     dev_data = test_data[:test_size]
22.     dev_set += dev_data
23.
24.     del test_data[:test_size]
25.     test_set += test_data
```

Kode 5.8 *Split Data*

Setelah ujaran dikelompokkan ke dalam *list* sesuai dengan kelas *intent* masing-masing, setiap ujaran dalam *list* tersebut dibagi menjadi *train set*, *dev set*, dan *test set* dengan rasio 80:10:10. *Train set* dan *dev set* akan digunakan pada proses *training* model untuk mendapatkan model dengan akurasi terbaik. Sedangkan *test set* akan digunakan pada proses evaluasi model untuk menguji seberapa baik performa model pada data ujaran yang belum pernah dilihat sebelumnya.

```
1.  # Create instructions for processing the text
2.  text_field = torchtext.data.Field()
3.  label_field = torchtext.data.Field(sequential=False)
4.
5.  fields = [('text', text_field), ('label', label_fiel
    d)]
6.
7.  examples = [torchtext.data.Example.fromlist([text, l
    abel], fields)
8.               for text, label in all_set]
9.
10. train_examples = [torchtext.data.Example.fromlist([t
    ext, label], fields)
11.              for text, label in train_set]
12.
```

```
dev_examples = [torchtext.data.Example.fromlist([text, label], fields)
                for text, label in dev_set]

test_examples = [torchtext.data.Example.fromlist([text, label], fields)
                for text, label in test_set]

all_set = torchtext.data.Dataset(examples, fields=fields)
train_set = torchtext.data.Dataset(examples=train_examples, fields=fields)
dev_set = torchtext.data.Dataset(examples=dev_examples, fields=fields)
test_set = torchtext.data.Dataset(examples=test_examples, fields=fields)
```

Kode 5.9 Pembuatan Torchtext *Dataset*

Kode 5.9 menjelaskan proses pembuatan Torchtext *dataset*. Pada kode ini, *train set*, *dev set*, dan *test set* yang masih berupa *list* diubah ke dalam *torchtext dataset* untuk memudahkan pemanggilan teks dan label dari setiap data pada tahap berikutnya.

5.5 Implementasi Model *Word Embedding*

Model *neural network* hanya dapat memproses *input* berupa vektor. Sehingga, untuk menerjemahkan kalimat ujaran ke dalam bentuk vektor perlu dilakukan pembuatan kamus *word vector* terlebih dahulu. Kamus *word vector* berisi seluruh *token* kata dalam *dataset* beserta nilai vektornya.

Pada penelitian ini, pembuatan kamus *word vector* memanfaatkan *pre-trained word embedding* yang diperoleh dari arsip penelitian Lab ADDI.

```
# Building the vocabs
text_field.build_vocab(all_set)
label_field.build_vocab(all_set)

# Embeddings initial values
vocab_to_idx = dict(text_field.vocab.stoi)
```

```
idx_to_vocab = {v: k for k, v in vocab_to_idx.items()}

label_to_idx = dict(label_field.vocab.stoi)
idx_to_label = {v: k for k, v in label_to_idx.items()}
```

Kode 5.10 Pembuatan Kamus Kata dan Kamus Label

Kode 5.10 menjelaskan tentang proses pembuatan kamus kata dan kamus label dari *dataset* percakapan. Proses ini memanfaatkan *library* Torchtext pada Python. Kamus kata menyimpan seluruh *token* kata yang terdapat pada *dataset* percakapan sebelum diterjemahkan ke dalam bentuk vektor. Sedangkan kamus label menyimpan seluruh label *intent* yang terdapat pada *dataset* beserta representasi kategorikal dari label tersebut.

```
def create_fold_embeddings(embeddings_dim, key_vector):

    emb_init_values = []
    unk = []

    a = 0
    b = 0

    for i in range(idx_to_vocab.__len__()):
        word = idx_to_vocab.get(i)
        if word == '<unk>':
            emb_init_values.append(np.random.uniform(-0.25, 0.25, embeddings_dim).astype('float32'))

        elif word == '<pad>':
            emb_init_values.append(np.zeros(embeddings_dim).astype('float32'))

        elif word in key_vector.wv.vocab:
            emb_init_values.append(key_vector.wv.word_vec(word))
            b = b+1
        else:
            emb_init_values.append(np.random.uniform(-0.25, 0.25, embeddings_dim).astype('float32'))
```

```
            a = a+1
            unk.append(word)

    known_word = b
    unknown_word = a

    return known_word, unknown_word, emb_init_values
```

Kode 5.11 Proses Inisialisasi Vektor Kata

Kode 5.11 digunakan untuk proses inisialisasi vektor kata dari *dataset* untuk proses pelatihan model LSTM dan model *neural network* lainnya dengan melihat kesesuaian kata pada kamus kata yang telah dibuat pada tahap sebelumnya. Jika kata pada kamus kata terdapat pada *pretrained word embedding model*, maka nilai vektor dari kata tersebut akan menyesuaikan dengan nilai vektor pada model. Namun, jika kata tidak terdapat pada *pretrained word embedding model* maka nilai vektor dari kata tersebut akan diambil secara *random* dengan besar dimensi disesuaikan dengan model *pretrained word embedding model* yang digunakan, seperti pada baris ke 21.

```
#Loading pre-trained embeddings
print("loading word2vec...")
start = time.time()
word_vectors = KeyedVectors.load_word2vec_format("D:
/Tugas Akhir/Word Embedding/pretrainedembedding.bin"
,
                                                  bin
ary="True", unicode_errors='ignore')
end = time.time()
print("word2vec loading done in {} seconds".format(e
nd-start))

word2vec = word_vectors
embed_dim = 300

known_word, unknown_word, emb_init_values = create_f
old_embeddings(embed_dim, word_vectors)
emb_init_values = np.array(emb_init_values)
```

Kode 5.12 *Load Pre-trained Word Embedding*

Kode 5.12 menjelaskan tentang proses *loading pretrained word embedding model* untuk digunakan pada inisialisasi vektor kata. Dimensi *embedding* yang digunakan berjumlah 300 menyesuaikan dengan dimensi pada *pretrained model* yang akan digunakan. Dimensi dan kamus vektor kata milik *pretrained word embedding model* selanjutnya dijadikan parameter untuk menjalankan fungsi *create_fold_embeddings*.

5.6 Pembuatan Model LSTM dan Model *Neural Network* Lain

Pada tahap ini model LSTM dan model *neural network* lain dibuat menggunakan *library* Pytorch. *Library* tersebut dipilih karena dapat memanfaatkan kemampuan komputasi arsitektur CUDA pada GPU untuk melakukan komputasi secara paralel dengan performa lebih cepat dibandingkan dengan komputasi pada CPU.

```
1.  class LSTMTagger(nn.Module):
2.
3.      def __init__(self, weights_matrix, hidden_size,
    output_size, n_layers=1):
4.          super(LSTMTagger, self).__init__()
5.          self.hidden_size = hidden_size
6.          self.n_layers = n_layers
7.
8.          self.embedding = nn.Embedding.from_pretraine
    d(torch.tensor(weights_matrix, dtype=torch.float))
9.          self.lstm = nn.LSTM(hidden_size, hidden_size
    , n_layers)
10.         self.fc = nn.Linear(hidden_size, output_size
    )
11.
12.     def forward(self, sentence):
13.         # Note: we run this all at once (over the wh
    ole input sequence)
14.
15.         # input = B x S . size(0) = B
16.         batch_size = sentence.size(0)
17.
18.         # input:  B x S  -- (transpose) --> S x B
19.         sentence = sentence.t()
20.
```

```
21.         # Embedding S x B -
    > S x B x I (embedding size)
22.         embedded = self.embedding(sentence)
23.
24.         output, (ht, ct) = self.lstm(embedded)
25.
26.         # Use the last layer output as FC's input
27.         # No need to unpack, since we are going to u
    se hidden
28.
29.         fc_output = self.fc(ht[-1])
30.         tag_scores = F.log_softmax(fc_output, dim=1)

31.
32.         return tag_scores
```

Kode 5.13 Kelas Model LSTM

Pembuatan model LSTM diawali dengan inisialisasi *hidden size*, jumlah *recurrent layers*, *embedding, num layer*, dan *fully-connected layer* pada baris ke 5-10. *Embedding* yang digunakan diperoleh dari model *pretrained* yang sebelumnya telah ditransformasi ke dalam kamus vektor kata. *Embedding* dibutuhkan untuk *initial weight* dari masing-masing kata sebelum *training* dilakukan.

Setelah inisialisasi, terdapat fungsi *forward* pada baris ke 12 yang digunakan untuk proses pembelajaran pada model yang telah dibuat. Fungsi ini dimulai dengan mendefinisikan *batch size*, lalu melakukan *transpose* pada input. Selanjutnya, dilakukan proses *embedding* kata. Proses ini akan melakukan perubahan setiap kata pada input menjadi vektor. Karena *embedding* dari model *pretrained* memiliki dimensi sebesar 300, maka setiap kata pada input yang masuk akan direpresentasikan dengan vektor 300 dimensi. Setelah proses *embedding*, maka dihasilkan *feature maps* dengan dimensi S x B x I. Dimana S merupakan banyaknya kata dalam kalimat, B merupakan *batch size*, dan I adalah besaran *embedding*.

Selanjutnya *feature maps* diproses menggunakan *layer* LSTM. Proses tersebut menghasilkan *output*, *hidden*, dan *context*, sesuai dengan teori cara kerja LSTM. *Hidden layer* yang

terakhir kemudian diambil untuk diproses di dalam *fully-connected layer*. Lalu dengan perhitungan softmax, didapatkan tag scores sebagai prediksi kelas *intent* dari kalimat *input*.

Selain kelas model LSTM, dibuat juga kelas untuk model *neural network* lain. Model yang akan dibuat antara lain *standard Recurrent Neural Network* (RNN), *Gated Recurrent Unit* (GRU), *Bidirectional Long Short-Term Memory* (LSTM).

```
1.  class RNNTagger(nn.Module):
2.
3.      def __init__(self, weights_matrix, hidden_size,
    output_size, n_layers=1):
4.          super(RNNTagger, self).__init__()
5.          self.hidden_size = hidden_size
6.          self.n_layers = n_layers
7.
8.          self.embedding = nn.Embedding.from_pretraine
    d(torch.tensor(weights_matrix, dtype=torch.float))
9.          self.rnn = nn.RNN(hidden_size, hidden_size,
    n_layers)
10.         self.fc = nn.Linear(hidden_size, output_size
    )
11.
12.     def forward(self, sentence):
13.         # Note: we run this all at once (over the wh
    ole input sequence)
14.
15.         # input = B x S . size(0) = B
16.         batch_size = sentence.size(0)
17.
18.         # input:  B x S  -- (transpose) --> S x B
19.         sentence = sentence.t()
20.
21.         # Embedding S x B -
    > S x B x I (embedding size)
22.         embedded = self.embedding(sentence).to(devic
    e)
23.
24.         # Make a hidden
25.         hidden = self._init_hidden(batch_size).to(de
    vice)
26.
27.         output, hidden = self.rnn(embedded, hidden)
```

```
28.
29.         # Use the last layer output as FC's input
30.         # No need to unpack, since we are going to u
    se hidden
31.         fc_output = self.fc(hidden[-1])
32.         tag_scores = F.log_softmax(fc_output, dim=1)

33.
34.         return tag_scores
35.
36.     def _init_hidden(self, batch_size):
37.         hidden = torch.zeros(self.n_layers, batch_si
    ze, self.hidden_size)
38.         return Variable(hidden)
```

Kode 5.14 Kelas Model RNN

Kode 5.14 menjelaskan tentang kelas model RNN. Konstruksi model tersebut hampir sama dengan model LSTM, hanya saja pada bagian inisialisasi, *layer* LSTM diganti dengan *layer* RNN.

Pada fungsi *forward*, vektor *input* diproses ke dalam RNN sehingga menghasilkan *output* dan *hidden*. RNN tidak memiliki *layer context*, karena itu tingkat kompleksitasnya lebih rendah dibanding LSTM.

```
1.  class GRUTagger(nn.Module):
2.
3.      def __init__(self, weights_matrix, hidden_size,
    output_size, n_layers=1):
4.          super(GRUTagger, self).__init__()
5.          self.hidden_size = hidden_size
6.          self.n_layers = n_layers
7.
8.          self.embedding = nn.Embedding.from_pretraine
    d(torch.tensor(weights_matrix, dtype=torch.float))
9.          self.gru = nn.GRU(hidden_size, hidden_size,
    n_layers)
10.         self.fc = nn.Linear(hidden_size, output_size
    )
11.
12.     def forward(self, sentence):
13.         # Note: we run this all at once (over the wh
    ole input sequence)
14.
```

```
15.         # input = B x S . size(0) = B
16.         batch_size = sentence.size(0)
17.
18.         # input:  B x S  -- (transpose) --> S x B
19.         sentence = sentence.t()
20.
21.         # Embedding S x B -
    > S x B x I (embedding size)
22.         embedded = self.embedding(sentence).to(devic
    e)
23.
24.         # Make a hidden
25.         hidden = self._init_hidden(batch_size).to(de
    vice)
26.
27.         output, hidden = self.gru(embedded, hidden)

28.
29.         # Use the last layer output as FC's input
30.         # No need to unpack, since we are going to u
    se hidden
31.         fc_output = self.fc(hidden[-1])
32.         tag_scores = F.log_softmax(fc_output, dim=1)

33.
34.         return tag_scores
35.
36.     def _init_hidden(self, batch_size):
37.         hidden = torch.zeros(self.n_layers, batch_si
    ze, self.hidden_size)
38.         return Variable(hidden)
```

Kode 5.15 Kelas Model GRU

Kode 5.15 menjelaskan tentang kelas model GRU. Konstruksi dari model ini sama seperti model-model *recurrent neural network* sebelumnya, yang berbeda hanya pada bagian inisialisasi, *recurrent layer* yang digunakan memanfaatkan fungsi nn.GRU milik Pytorch.

Pada fungsi *forward*, GRU juga tidak memiliki *layer context*. Setiap proses pada GRU hanya menghasilkan *output* dan *hidden*. *Hidden layer* paling terakhir diambil untuk diproses dengan *fully-connected layer*, kemudian menggunakan *softmax*

untuk menghitung *tag scores* sebagai hasil prediksi kelas *intent* dari *input* kalimat.

```
class BiLSTMTagger(nn.Module):

    def __init__(self, weights_matrix, hidden_size, output_size, n_layers=1):
        super(BiLSTMTagger, self).__init__()
        self.hidden_size = hidden_size
        self.n_layers = n_layers

        self.embedding = nn.Embedding.from_pretrained(torch.tensor(weights_matrix, dtype=torch.float))
        self.lstm = nn.LSTM(hidden_size, hidden_size, n_layers, bidirectional=True)
        self.fc = nn.Linear(hidden_size * 2, output_size)

    def forward(self, sentence):
        # Note: we run this all at once (over the whole input sequence)

        # input = B x S . size(0) = B
        batch_size = sentence.size(0)

        # input:  B x S  -- (transpose) --> S x B
        sentence = sentence.t()

        # Embedding S x B -
> S x B x I (embedding size)
        embedded = self.embedding(sentence).to(device)

        # Set initial states
        h0 = torch.zeros(self.n_layers*2, batch_size, self.hidden_size).to(device) # 2 for bidirection

        c0 = torch.zeros(self.n_layers*2, batch_size, self.hidden_size).to(device)

        output, _ = self.lstm(embedded, (h0, c0))

        # Use the last layer output as FC's input
        # No need to unpack, since we are going to use hidden
```

```
32.
33.         fc_output = self.fc(output[-1])
34.         tag_scores = F.log_softmax(fc_output, dim=1)

35.
36.         return tag_scores
```

Kode 5.16 Kelas Model BiLSTM

Model BiLSTM pada dasarnya adalah model LSTM dengan parameter *bidirectional* bernilai *true* seperti pada baris ke 9. Secara *default*, *parameter* tersebut bernilai *false*. Karena BiLSTM bekerja dua arah, pada bagian inisialisasi *fully-connected layer*, parameter *hidden size* perlu dikalikan dua.

Pada fungsi *forward*, model ini hampir sama dengan LSTM hanya saja *initial hidden layer* dan *initial context layer* menggunakan parameter *num layers* yang dikalikan dua.

Setelah seluruh kelas model dibuat, dilakukan *training* model LSTM dan model *neural network* lain pada *train set*. *Training* dilakukan sesuai skenario yang telah dirancang pada sub bab 4.5, yaitu melakukan kombinasi jenis *optimizer*, nilai *learning rate*, dan jumlah *num layer*. Untuk setiap arsitektur, kombinasi tersebut menghasilkan 27 model. Sehingga 4 arsitektur menghasilkan 108 model. Proses *training* akan dilakukan 2 kali menggunakan komputer 1 dan komputer 2.

```
1.  scenario = {'SC1': 0.001,
2.               'SC2': 0.01,
3.               'SC3': 0.015}
4.  train_dur_list = []
5.  for key, value in scenario.items():
6.      print("\n=== {} ===".format(key))
7.      for i in range(1, 4):
8.          HIDDEN_SIZE = 300
9.          model = LSTMTagger(emb_init_values, HIDDEN_S
    IZE, len(label_to_idx), n_layers=i)
10.         # use GPU
11.         model = model.to(device)
12.         loss_function = nn.NLLLoss()
13.         optimizer = optim.SGD(model.parameters(), lr
    =value)
```

```
14.         #optimizer = torch.optim.Adam(model.paramete
    rs(), lr=value)
15.
16.         train_accuracies = []
17.         dev_accuracies = []
18.         loss_list = []
19.
20.         model.train()
21.
22.         best_accuracy = 0
23.         start = time.time()
24.         print("\n=== TRAINING {} LAYER STARTED ===".
    format(i))
25.
26.         for epoch in range(150):
27.             go = time.time()
28.             print("\nEpoch:",epoch+1)
29.
30.             for example in train_set:
31.                 model.zero_grad()
32.
33.                 sentence_in = prepare_sequence(examp
    le.text, vocab_to_idx)
34.                 targets = prepare_sequence(example.l
    abel, label_to_idx)
35.
36.                 # use GPU
37.                 sentence_in = sentence_in.to(device)

38.                 targets = targets.to(device)
39.
40.                 fc_output = model(sentence_in)
41.
42.                 loss = loss_function(fc_output, targ
    ets.view(-1))
43.                 loss.backward()
44.                 optimizer.step()
45.
46.             # Mencatat loss
47.             loss_list.append(loss.data)
48.
49.             # Menghitung Train Accuracy
50.             _, _, train_accuracy = getAccuracy(train
    _set)
```

```
51.             train_accuracies.append(train_accuracy)
52.             print("Training accuracy is %d%%" % (tra
    in_accuracy*100))
53.
54.             # Menghitung Dev Accuracy
55.             _, _, dev_accuracy = getAccuracy(dev_set
    )
56.             dev_accuracies.append(dev_accuracy)
57.             print("Dev accuracy is %d%%" % (dev_accu
    racy*100))
58.
59.             if(dev_accuracy >= best_accuracy):
60.                 best_accuracy = dev_accuracy
61.                 best_model = copy.deepcopy(model)
62.
63.             finish = time.time()
64.             dur = finish - go
65.             print("duration:", dur)
66.
67.         end = time.time()
68.         train_dur = end - start
69.         train_dur_list.append(train_dur)
70.         torch.save(best_model.state_dict(), 'D:/Tuga
    s Akhir/Model/Final/ALLDATA_LSTM_SGD_{0}LAYER_{1}.pt
    '.format(i,key))
71.         print("=== TRAINING {0} LAYER DONE IN {1} SE
    CONDS ===".format(i,train_dur))
72.         print("Best accuracy:", best_accuracy)
73.
74.         df = pd.DataFrame({'loss': loss_list,
75.                            'train acc': train_accuracies
    ,
76.                            'dev acc': dev_accuracies})
77.
78.         df.to_excel('D:/Tugas Akhir/Model/Final/ALLD
    ATA_LSTM_SGD_{0}LAYER_{1}.xlsx'.format(i,key))
79. print("=== TRAINING ALL 9 SCENARIOS AND LAYERS DONE
    ===")
80. print("training duration per layer",train_dur_list)

81. %notify -m "Training done."
```

Kode 5.17 *Training* Model

Kode 5.17 menjelaskan proses *training* model. Proses training diawali dengan menyimpan nilai *learning rate* yang akan digunakan ke dalam kamus bernama *scenario*, seperti pada baris ke 1-3. Pada penelitian kali ini menggunakan *learning rate* 0.001, 0.01, dan 0.015. Selanjutnya pada baris ke 5 dilakukan perulangan sebanyak jumlah skenario *learning rate* dan di dalamnya dilakukan perulangan lagi sebanyak jumlah skenario jumlah *num layer*. Jumlah *num layer* yang akan dicoba pada proses *training* yaitu 1, 2, dan 3 *layer*.

Pada setiap perulangan, dilakukan pembuatan objek dari kelas model LSTM/RNN/GRU/BiLSTM. Selanjutnya dilakukan inisialisasi *loss function* dan *optimizer*. *Optimizer* yang dicoba pada penelitian ini adalah SGD, *momentum*, dan ADAM. Sebelum menjalankan proses *training* juga dilakukan inisialisasi *train_accuracies*, *dev_accuracies*, dan *lost_list* pada baris ke 16-18 untuk menyimpan nilai akurasi pada data *train*, data *dev*, dan *loss* per *epoch*.

Proses *training* model dilakukan dengan 150 *epoch*. Pada setiap *epoch*, model melakukan pembelajaran dari *train set*. *Train accuracy* dan *dev accuracy* dihitung pada setiap *epoch* lalu disimpan pada *list train_accuracies* dan *dev_accuracies*. Pada baris ke 70, model dengan *dev accuracy* tertinggi disimpan untuk diuji dan dievaluasi pada tahap selanjutnya.

```
1.  # Get tag prediction for every data in test set
2.  def predict(dataset):
3.      """Gets a list of indices of training_data, and
    returns a list of predicted lists of tags"""
4.      for example in dataset:
5.          inputs = prepare_sequence(example.text, voca
    b_to_idx).to(device)
6.          tag_scores = model(inputs)
7.          values, target = torch.max(tag_scores, 1)
8.          yield target.cpu().numpy()
9.
10. # Get true tag for every data in test set
11. def true_tag_to_ix(dataset):
12.     "gets a list of indices of training_data, and re
    turns a list of tag_to_ix true tags"
13.     for example in dataset:
```

```
        true_tag = prepare_sequence(example.label, l
abel_to_idx)
        yield true_tag

# Calculate Accuracy
def getAccuracy(dataset):
    y_pred = list(predict(dataset))
    y_true = list(true_tag_to_ix(dataset))
    accuracy = accuracy_score(y_pred, y_true)
    return y_pred, y_true, accuracy
```

Kode 5.18 Perhitungan Akurasi

Akurasi adalah salah satu metrik untuk mengukur seberapa baik performa dari model yang dibuat. Perhitungan akurasi dapat dilakukan pada *train set*, *dev set*, yang sebelumnya sudah diimplementasi saat proses *training*. Selain itu perhitungan akurasi juga dapat dilakukan pada *test set* yang akan digunakan saat proses evaluasi model.

Kode 5.18 menjelaskan tentang proses perhitungan akurasi. Proses dimulai dengan melakukan prediksi *intent* dari setiap contoh pada *dataset* lalu memasukkannya ke dalam list *y_pred*. Kemudian *intent* sebenarnya dari setiap contoh tersebut dimasukkan ke dalam list *y_true*. Selanjutnya *y_pred* dan *y_true* dijadikan *parameter* untuk fungsi *accuracy_score* milik *library* sklearn. Fungsi tersebut mengembalikan nilai akurasi dari model.

5.7 Analisis dan Pengujian Performa Model LSTM dan Model *Neural Network* Lain

Pada tahap ini, setiap model diuji pada *test set* untuk mengevaluasi performa masing-masing dan membandingkannya satu sama lain. Metrik pengukuran yang digunakan sesuai pada bab 5.4. yaitu akurasi, *precision*, *recall*, dan *F1-score*.

```
HIDDEN_SIZE = 300
model = BiLSTMTagger(emb_init_values, HIDDEN_SIZE, l
en(label_to_idx), n_layers=i).to(device)
```

```
3.  model.load_state_dict(torch.load('D:/Tugas Akhir/Mod
    el/Repeat 1/Model/BiLSTM - SGD/ALLDATA_BiLSTM_SGD_1L
    AYER_SC1.pt'.format(i,key)))
4.  model.eval()
5.
6.  # Menghitung Test Accuracy
7.  y_pred, y_true, test_accuracy = getAccuracy(test_set
    )
8.  print("Test accuracy is {}".format(str(round(test_ac
    curacy * 100, 2))))
```

Kode 5.19 Perhitungan *Test Accuracy*

Kode 5.19 menjelaskan tentang proses perhitungan *test accuracy*. Tahap pertama yang dilakukan adalah melakukan *load* model dari *file directory*. Pada baris ke 7, akurasi pada *test set* dihitung menggunakan fungsi *getAccuracy* yang telah dibuat sebelumnya. Fungsi tersebut juga mengembalikan *list y_true* dan *y_pred* yang merupakan daftar *actual intent* dan *predicted intent* dari *test set*.

```
1.  # mengubah seluruh id dalam list menjadi string labe
    lnya
2.  def id_to_label(seq):
3.      labels = [idx_to_label[w.squeeze().tolist()] for
     w in seq]
4.      return labels
5.
6.  # Membuat dataframe untuk prediction tag and actual
    tag dari data test
7.  y_pred = id_to_label(y_pred)
8.  y_true = id_to_label(y_true)
9.  df = pd.DataFrame({'prediction': y_pred,
10.                    'actual': y_true})
11.
12. # Create confusion matrix
13. prediction = df['prediction']
14. actual = df['actual']
15. confusion_matrix = pd.crosstab(actual, prediction)
16. confusion_matrix
17.
18. # Save prediction and actual dataframe
19. df.to_excel('D:/Tugas Akhir/Model/Test/PREDACTUAL_AL
    LDATA_LSTM_SGD_1LAYER_SC1.xlsx', index=False)
```

```
20.
21. # Save confusion matrix
22. confusion_matrix.to_excel('D:/Tugas Akhir/Model/Test
    /CONFUSION_ALLDATA_LSTM_SGD_1LAYER_SC1.xlsx')
```

Kode 5.20 Pembuatan *Confusion Matrix*

Kode 5.20 menjelaskan tentang proses pembuatan *confusion matrix*. Tahap pertama yang dilakukan pada proses ini adalah membuat fungsi *id_to_label* untuk mengubah id label menjadi *string* label tersebut. Dengan fungsi ini label dengan id sama dengan 1 akan diubah menjadi 'ketersediaan', label dengan id sama dengan 2 akan diubah menjadi 'harga', dan seterusnya. Fungsi *id_to_label* digunakan untuk mengubah seluruh id label pada *list y_pred* dan *y_true* yang didapatkan dari Kode 5.19 menjadi *string* label tersebut.

Confusion matrix dibuat pada baris ke 15 dengan melakukan operasi *crosstab* pada *y_pred* dan *y_true*. Tabel yang berisi kolom *y_pred* dan *y_true* disimpan ke dalam *file* Excel, begitu juga dengan *confusion matrix*.

```
1.  # Function untuk menghitung precision, recall, dan f
    measure
2.  def prec_rec_fm(df, label):
3.      new_df = df[(df['prediction'] == label) | (df['a
    ctual'] == label)]
4.      actual_label_count = new_df[new_df['actual'] ==
    label].actual.count()
5.
6.      tp = 0
7.      fp = 0
8.      fn = 0
9.      for index, row in new_df.iterrows():
10.         if row[0] == row[1]:
11.             tp+=1
12.         elif row[0] == label:
13.             fp+=1
14.         elif row[1] == label:
15.             fn+=1
16.
17.     precision = tp / (tp + fp) if (tp + fp) > 0 else
     0
```

```
18.     recall = tp / (tp + fn) if (tp + fn) > 0 else 0
19.     fmeasure = 2 * precision * recall / (precision +
     recall) if (precision + recall) > 0 else 0
20.
21.     print("\n{0} precision:".format(label), precisio
    n)
22.     print("{0} recall:".format(label), recall)
23.     print("{0} fmeasure:".format(label), fmeasure)
24.
25.     return precision, recall, fmeasure, actual_label
    _count
```

Kode 5.21 Fungsi Perhitungan *Precision, Recall, F1-score*

Kode 5.21 menjelaskan tentang fungsi *prec_rec_fm* untuk perhitungan performa model pada *test set* menggunakan metrik *precision*, *recall*, dan *F1-score* (biasa disebut juga dengan *f-measure*). Fungsi *prec_rec_fm* membutuhkan dua *argument* yaitu *dataframe* pandas dengan kolom *actual* dan *predicted* yang didapat dari Kode 5.20, dan label kelas *intent* yang akan diukur performanya. Pada baris ke 3, setiap baris dalam *dataframe* dengan label *actual* atau *predicted* sama dengan label kelas pada *argument* diambil dan disimpan ke dalam *dataframe* baru bernama *new_df*. Selanjutnya pada baris ke 9 dilakukan perulangan pada setiap baris pada *new_df* untuk menghitung nilai *true positive*, *false positive*, dan *false negative* yang disimpan pada *variable* tp, fp, dan fn. *Variable* tersebut digunakan untuk melakukan perhitungan *precision*, *recall*, dan *F1-score* sesuai dengan rumus pada sub bab 4.6. Fungsi *prec_rec_fm* mengembalikan ketiga nilai metrik tersebut dan *actual_label_count*. *Actual_label_count* adalah jumlah kalimat ujaran pada *test set* dengan label *actual* sama dengan label pada *argument*.

```
1.  # Hitung precision, recall, dan fmeasure dari masing
    -masing label pada test set
2.  print("=== INTENT PERFORMANCE ON TEST SET ===")
3.  print("Test accuracy: {}".format(test_accuracy))
4.  ketersediaan_pr, ketersediaan_rc, ketersediaan_fm, k
    etersediaan_count = prec_rec_fm(df, 'ketersediaan')
```

```
harga_pr, harga_rc, harga_fm, harga_count = prec_rec_fm(df, 'harga')
pengiriman_pr, pengiriman_rc, pengiriman_fm, pengiriman_count = prec_rec_fm(df, 'pengiriman')
spesifikasi_pr, spesifikasi_rc, spesifikasi_fm, spesifikasi_count = prec_rec_fm(df, 'spesifikasi')
garansi_pr, garansi_rc, garansi_fm, garansi_count = prec_rec_fm(df, 'garansi')
sapaan_pr, sapaan_rc, sapaan_fm, sapaan_count = prec_rec_fm(df, 'sapaan')
pembayaran_pr, pembayaran_rc, pembayaran_fm, pembayaran_count = prec_rec_fm(df, 'pembayaran')
tipe_pr, tipe_rc, tipe_fm, tipe_count = prec_rec_fm(df, 'tipe')
kelengkapan_pr, kelengkapan_rc, kelengkapan_fm, kelengkapan_count = prec_rec_fm(df, 'kelengkapan')
kondisi_pr, kondisi_rc, kondisi_fm, kondisi_count = prec_rec_fm(df, 'kondisi')
aksesori_pr, aksesori_rc, aksesori_fm, aksesori_count = prec_rec_fm(df, 'aksesori')
keaslian_pr, keaslian_rc, keaslian_fm, keaslian_count = prec_rec_fm(df, 'keaslian')
toko_pr, toko_rc, toko_fm, toko_count = prec_rec_fm(df, 'toko')
transaksi_pr, transaksi_rc, transaksi_fm, transaksi_count = prec_rec_fm(df, 'transaksi')
pengembalian_pr, pengembalian_rc, pengembalian_fm, pengembalian_count = prec_rec_fm(df, 'pengembalian')

service_pr, service_rc, service_fm, service_count = prec_rec_fm(df, 'service')
```

Kode 5.22 Perhitungan *Precision, Recall, F1-score* Setiap Kelas Intent

Fungsi *prec_rec_fm* dipanggil sebanyak 16 kali sesuai dengan jumlah jenis *intent* dan nilai pengembaliannya disimpan ke dalam *variable* dengan format [nama_*intent*]_pr untuk *precision*, [nama_*intent*]_rc untuk *recall*, [nama_*intent*]_fm untuk *F1-score*, dan [nama_*intent*]_count untuk *actual_label_count*.

```
# Hitung weighted average
avg_pr = (ketersediaan_pr * ketersediaan_count + harga_pr * harga_count + pengiriman_pr * pengiriman_count +
          spesifikasi_pr * spesifikasi_count + garansi_pr * garansi_count + sapaan_pr * sapaan_count +
          pembayaran_pr * pembayaran_count + tipe_pr * tipe_count + kelengkapan_pr * kelengkapan_count +
          kondisi_pr * kondisi_count + aksesori_pr * aksesori_count + keaslian_pr * keaslian_count +
          toko_pr * toko_count + transaksi_pr * transaksi_count + pengembalian_pr * pengembalian_count +
          service_pr * service_count)/len(test_set)

avg_rc = (ketersediaan_rc * ketersediaan_count + harga_rc * harga_count + pengiriman_rc * pengiriman_count +
          spesifikasi_rc * spesifikasi_count + garansi_rc * garansi_count + sapaan_rc * sapaan_count +
          pembayaran_rc * pembayaran_count + tipe_rc * tipe_count + kelengkapan_rc * kelengkapan_count +
          kondisi_rc * kondisi_count + aksesori_rc * aksesori_count + keaslian_rc * keaslian_count +
          toko_rc * toko_count + transaksi_rc * transaksi_count + pengembalian_rc * pengembalian_count +
          service_rc * service_count)/len(test_set)

avg_fm = (ketersediaan_fm * ketersediaan_count + harga_fm * harga_count + pengiriman_fm * pengiriman_count +
          spesifikasi_fm * spesifikasi_count + garansi_fm * garansi_count + sapaan_fm * sapaan_count +
          pembayaran_fm * pembayaran_count + tipe_fm * tipe_count + kelengkapan_fm * kelengkapan_count +
```

```
            kondisi_fm * kondisi_count + aksesori_fm *
     aksesori_count + keaslian_fm * keaslian_count +
            toko_fm * toko_count + transaksi_fm * tran
    saksi_count + pengembalian_fm * pengembalian_count +

            service_fm * service_count)/len(test_set)
```

Kode 5.23 Perhitungan *Weighted Average Precision, Recall, F1-score*

Untuk mengevaluasi performa model secara global, perlu dilakukan perhitungan *precision*, *recall*, dan *F1-score* menggunakan *weighted average*. Nilai *precision*/*recall*/ *F1-score* setiap kelas *intent* dikalikan *actual_label_count intent* tersebut. Selanjutnya, hasil perkalian tersebut dijumlahkan untuk seluruh kelas *intent*, lalu dibagi dengan jumlah data pada *test set*. Proses perhitungan *weighted average* dapat dilihat pada Kode 5.23.

```
# Membuat tabel hasil pengukuran
pr_list = [ketersediaan_pr, harga_pr, pengiriman_pr,
 spesifikasi_pr, garansi_pr, sapaan_pr, pembayaran_p
r, tipe_pr,
            kelengkapan_pr, kondisi_pr, aksesori_pr,
keaslian_pr, toko_pr, transaksi_pr, pengembalian_pr,
 service_fm, avg_pr]
rc_list = [ketersediaan_rc, harga_rc, pengiriman_rc,
 spesifikasi_rc, garansi_rc, sapaan_rc, pembayaran_r
c, tipe_rc,
            kelengkapan_rc, kondisi_rc, aksesori_rc,
keaslian_rc, toko_rc, transaksi_rc, pengembalian_rc,
 service_fm, avg_rc]
fm_list = [ketersediaan_fm, harga_fm, pengiriman_fm,
 spesifikasi_fm, garansi_fm, sapaan_fm, pembayaran_f
m, tipe_fm,
            kelengkapan_fm, kondisi_fm, aksesori_fm,
keaslian_fm, toko_fm, transaksi_fm, pengembalian_fm,
 service_fm, avg_fm]
index = ['ketersediaan', 'harga', 'pengiriman', 'spe
sifikasi', 'garansi', 'sapaan', 'pembayaran', 'tipe'
,
             'kelengkapan', 'kondisi', 'aksesori',
'keaslian', 'toko', 'transaksi', 'pengembalian', 'se
rvice', 'average']

```

```
df = pd.DataFrame({'precision': pr_list,
                   'recall': rc_list,
                   'fmeasure': fm_list},
                   index = index)
# Save hasil pengukuran
df.to_excel('D:/Tugas Akhir/Model/Test/MEASUREMENT_ALLDATA16_BiLSTM_SGD_1LAYER_SC3.xlsx')
```

Kode 5.24 Pembuatan Tabel Hasil Pengukuran Performa

Untuk memudahkan pembacaan hasil pengukuran performa yang terdiri dari *precision*, *recall*, dan *F1-score* setiap *intent*, dan *weighted average* dari ketiganya, dilakukan pembuatan tabel menggunakan *dataframe* Pandas. *Dataframe* selanjutnya disimpan ke dalam *file* Excel.

Karena percobaan dilakukan dua kali masing-masing pada komputer 1 dan komputer 2, maka hasil akhir pengukuran performa pada *test set* adalah hasil rata-rata dari dua percobaan tersebut. Ini berlaku untuk semua metrik pengukuran yaitu akurasi, *precision*, *recall*, dan *F1-score*.

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini, akan dijelaskan mengenai hasil dan analisis terhadap hasil yang diperoleh dari proses implementasi penelitian.

6.1 Hasil Pengumpulan Data

Proses pengumpulan data percakapan menghasilkan 1.806 baris ujaran pembeli seputar jual beli *smartphone* pada *e-commerce*, dengan jumlah *token* sebanyak 12.359. Dari kumpulan data tersebut, ujaran memiliki rata-rata 6,84 *token*, min 1 *token*, dan max 24 *token*. Ujaran yang dikumpulkan hanya ujaran berupa sapaan dan pertanyaan. Seluruh data ujaran tersebut diubah menjadi format yang seragam dan digabung ke dalam satu *file* sehingga siap untuk digunakan tahap selanjutnya.

Tabel 6.1 Hasil Penggabungan Data

Misi gan,hp Oppo F1 ready?
yang silver metallic ada gan?
mba untuk hp harga 1.5 an yang udah 4g ada gak ?
memori berapa ya ?
itu baru mbak ?
Barang ready gk gan?
Kira2 jika pesan hari ini sampe kapan?
Barang masih ada ?
garansi apa mas?
Kamera silent kan?
yg ready warna apa aja?
Fullset gan?

Tabel 6.1 merupakan contoh hasil pengumpulan data. Ujaran pada tabel di atas telah bersih dari *marker* penanda awal percakapan, akhir percakapan, dan pemberi ujaran.

6.2 Hasil Pra-Pemrosesan Data

Hasil pra-pemrosesan data merupakan *dataset* ujaran berlabel yang telah bersih dan diubah menjadi *token-token* yang siap

untuk diproses untuk pembuatan dan evaluasi model. Berikut hasil pra-pemrosesan data tahap demi tahap.

6.2.1 Hasil Pembuatan Daftar Kelas *Intent*

Pada tahap ini dilakukan pembuatan daftar kelas *intent* yang terdapat pada *dataset*. Dari implementasi yang dilakukan, diperoleh 16 kandidat kelas *intent* pada *dataset*. Berikut daftar kelas *intent* beserta deskripsinya.

Tabel 6.2 Daftar Kandidat Kelas *Intent*

Kelas Intent	Keterangan
Ketersediaan	Kelas ini berisi pertanyaan seputar info ketersediaan *smartphone*, termasuk tanggal *restock*, dan jumlah barang *ready stock*. Pertanyaan mengenai ketersediaan dapat berdasarkan tipe, warna, maupun spesifikasi *smartphone*. Contoh: Mas Samsung S9 ram untuk 6GB ada gak ?
Harga	Kelas ini berisi pertanyaan seputar harga *smartphone*, termasuk nego harga, total harga + ongkos kirim, harga grosir, hingga diskon dan promo. Contoh: Kak iphone *plus* 256GB *second gold* masih bisa nego harga tidak kak?
Pengiriman	Kelas ini berisi pertanyaan seputar pengiriman barang, termasuk jenis kurir, estimasi durasi pengiriman, *packing* barang, dan lain-lain. Contoh: kalo yes pesen sekarang masih bisa ikut pengiriman hari ini?
Spesifikasi	Kelas ini berisi pertanyaan seputar spesifikasi *smartphone*, mulai dari RAM, *chipset*, Memori internal, kamera, dan fitur-fitur lain. Contoh: ini s9+ *single* sim atau *dual* sim?
Garansi	Kelas ini berisi pertanyaan seputar garansi *smartphone*, termasuk jenis garansi, penyedia

Kelas Intent	Keterangan
	garansi, durasi garansi, hingga prosedur klaim garansi. Contoh: Garansi resmi atau internasional kak?
Sapaan	Kelas ini berisi sapaan sehari-hari. Contoh: Halo, selamat pagi.
Pembayaran	Kelas ini berisi pertanyaan seputar pembayaran, termasuk metode pembayaran, cicilan, dan nomor rekening tujuan. Contoh: klou misalnya cicil bos.. berapa berbulan dibayar
Tipe	Kelas ini berisi pertanyaan seputar tipe *smartphone*, termasuk tipe ROM *smartphone*, tipe *smartphone* berdasarkan harga, spesifikasi, dan performa. Contoh: Gan HP ram 4 gb, kamera 5 *megapixel* dapet apa ya?
Kelengkapan	Kelas ini berisi pertanyaan seputar kelengkapan dalam paket penjualan *smartphone*. Contoh: setiap pembelian iphone apakah tetep dapet *earphone* biasa?
Kondisi	Kelas ini berisi pertanyaan seputar kondisi *smartphone*, termasuk kondisi fisik barang, segel pada box, dan durasi pemakaian. Contoh: Permisi, iphone 6 *gold second* udah dipake berapa lama?
Aksesori	Kelas ini berisi pertanyaan seputar aksesori *smartphone* seperti *case*, *screen guard*, baterai, *charger*, dan lain-lain. Contoh: Bisa sekalian beli tamper glassnya?
Keaslian	Kelas ini berisi pertanyaan seputar keaslian barang. Contoh: ini brg ori bukan *refurbish*?

Kelas Intent	Keterangan
Toko	Kelas ini berisi pertanyaan seputar toko fisik dari penjual, termasuk alamat toko, cabang toko, hingga jam buka toko. Contoh: Punya toko fisik ga bang? Kalau ada dimana ya?
Transaksi	Kelas ini berisi pertanyaan seputar transaksi, termasuk prosedur pemesanan, dan pembatalan pemesanan. Contoh: Kak maaf kalau saya mau *cancel* pembelian bisa tidak kak?
Pengembalian	Kelas ini berisi pertanyaan seputar pengembalian barang yang telah diterima, baik karena cacat atau ketidaksesuaian dengan iklan. Pertanyaan termasuk prosedur, dan syarat dan ketentuan. Contoh: Gan untuk pembelian hape redmi 4x ada cacat dibagian layarnya, apakah bisa diretur?
Service	Kelas ini berisi pertanyaan seputar *service*, termasuk lokasi *service center*. Contoh: Bisa dimana aja ya kak kalau mau *service* misalkan HP nya ada kerusakan?

Sebelum kelas *intent* pada Tabel 6.2 digunakan pada 108 percobaan seperti yang telah dirancang pada sub bab 3.1.6, dilakukan pengujian terlebih dahulu. Hal ini penting untuk mengidentifikasi kelas *intent* mana yang memiliki performa paling buruk sehingga kelas tersebut perlu diperbaiki sebelum percobaan lain dilakukan. Proses pengujian melewati implementasi sub bab 4.2.2 hingga sub bab 4.6. Distribusi data berdasarkan label setelah melalui implementasi *data splitting* dapat dilihat pada Tabel 6.3.

Tabel 6.3 Hasil Distribusi Data Berdasarkan Label Setelah *Splitting*

	***Train set* (80%)**	***Dev set* (10%)**	***Test set* (10%)**	**Total**
Ketersediaan	390	49	49	488
Harga	174	22	22	218
Pengiriman	167	21	21	209
Spesifikasi	144	18	19	181
Garansi	120	15	15	150
Sapaan	93	12	12	117
Pembayaran	66	8	9	83
Tipe	62	8	8	78
Kelengkapan	61	8	8	77
Kondisi	52	6	7	65
Aksesori	30	4	4	38
Keaslian	22	3	3	28
Toko	20	2	3	25
Transaksi	17	2	3	22
Pengembalian	14	2	2	18
Service	7	1	1	9
Total	1439	181	186	1806

Untuk menguji performa setiap kelas *intent*, dilakukan *training* 4 model dengan arsitektur yang berbeda yaitu LSTM, RNN, GRU, dan BiLSTM menggunakan parameter terbaik berdasarkan Tabel 6.41. Berikut parameter ke-4 model yang digunakan dalam proses *training*.

Tabel 6.4 Parameter Model LSTM Untuk Pengujian Performa *Intent*

Parameter	Values
Embedding Dimension	300
Batch Size	1
Num Layers	1
Optimizer	ADAM
Learning Rate	0,001

Tabel 6.5 Parameter Model RNN Untuk Pengujian Performa *Intent*

Parameter	Values
Embedding Dimension	300
Batch Size	1

Num Layers	3
Optimizer	SGD
Learning Rate	0,001

Tabel 6.6 Parameter Model GRU Untuk Pengujian Performa *Intent*

Parameter	Values
Embedding Dimension	300
Batch Size	1
Num Layers	1
Optimizer	ADAM
Learning Rate	0,001

Tabel 6.7 Parameter Model BiLSTM Untuk Pengujian Performa *Intent*

Parameter	Values
Embedding Dimension	300
Batch Size	1
Num Layers	1
Optimizer	*Momentum*
Learning Rate	0,01

Hasil pembuatan ke-4 model tersebut kemudian diuji pada *test set* untuk mendapatkan nilai *precision*, *recall*, dan *F1-score* setiap kelas *intent*. Berikut hasil dari pengujian masing-masing model pada *test set*.

Tabel 6.8 Hasil Pengujian Kelas *Intent* Model LSTM

	Precision (%)	Recall (%)	F1-score (%)
Ketersediaan	91,84	91,84	91,84
Harga	90,91	90,91	90,91
Pengiriman	83,33	95,24	88,89
Spesifikasi	73,68	73,68	73,68
Garansi	92,31	80	85,71
Sapaan	92,31	100	96
Pembayaran	69,23	100	81,82
Tipe	50	37,5	42,86
Kelengkapan	80	100	88,89
Kondisi	50	57,14	53,33
Aksesori	100	25	40
Keaslian	100	100	100

Toko	100	66,67	80
Transaksi	0	0	0
Pengembalian	100	50	66,67
Service	0	0	0

Pada pengujian model LSTM, kelas *intent* Transaksi dan *Service* memiliki performa paling buruk. *Precision*, *recall*, dan *F1-score* dari kedua kelas tersebut bernilai 0. Ini berarti prediksi *intent* pada ujaran dengan *actual intent* Transaksi dan *Service* seluruhnya salah.

Tabel 6.9 Hasil Pengujian Kelas *Intent* Model RNN

	Precision (%)	**Recall (%)**	**F1-score (%)**
Ketersediaan	91,67	89,8	90,72
Harga	73,91	77,27	75,56
Pengiriman	76	90,48	82,61
Spesifikasi	70,59	63,16	66,67
Garansi	92,86	86,67	89,66
Sapaan	92,31	100	96
Pembayaran	81,82	100	90
Tipe	30	37,5	33,33
Kelengkapan	62,5	62,5	62,5
Kondisi	50	42,86	46,15
Aksesori	100	50	66,67
Keaslian	100	66,67	80
Toko	50	66,67	57,14
Transaksi	100	33,33	50
Pengembalian	100	100	100
Service	0	0	0

Sedangkan pada pengujian model RNN, hanya kelas *intent Service* yang memiliki performa paling buruk. *Precision*, *recall*, dan *F1-score* dari kelas tersebut bernilai 0. Ini berarti prediksi *intent* pada ujaran dengan *actual intent Service* seluruhnya salah.

Tabel 6.10 Hasil Pengujian Kelas *Intent* Model GRU

	Precision (%)	**Recall (%)**	**F1-score (%)**
Ketersediaan	92,31	97,96	95,05
Harga	91,3	95,45	93,33
Pengiriman	86,96	95,24	90,91
Spesifikasi	85,71	63,16	72,73
Garansi	92,86	86,67	89,66
Sapaan	100	100	100
Pembayaran	100	100	100
Tipe	83,33	62,5	71,43
Kelengkapan	72,73	100	84,21
Kondisi	66,67	57,14	61,54
Aksesori	50	75	60
Keaslian	100	100	100
Toko	75	100	85,71
Transaksi	100	33,33	50
Pengembalian	100	100	100
Service	0	0	0

Seperti model RNN, pada hasil pengujian kelas *intent* model GRU hanya kelas *intent Service* yang memiliki performa terburuk dengan *precision*, *recall*, dan *F1-score* bernilai 0. Ini berarti prediksi *intent* pada ujaran dengan *actual intent Service* seluruhnya salah.

Tabel 6.11 Hasil Pengujian Kelas *Intent* Model BiLSTM

	Precision (%)	**Recall (%)**	**F1-score (%)**
Ketersediaan	93.88	93.88	93.88
Harga	83.33	90.91	86.96
Pengiriman	76.92	95.24	85.11
Spesifikasi	81.25	68.42	74.29
Garansi	86.67	86.67	86.67
Sapaan	100	100	100
Pembayaran	88.89	88.89	88.89
Tipe	50	37.5	42.86
Kelengkapan	66.67	75	70.59
Kondisi	50	71.43	58.82
Aksesori	33.33	25	28.57
Keaslian	100	100	100

	Precision (%)	**Recall (%)**	**F1-score (%)**
Toko	50	33.33	40
Transaksi	0	0	0
Pengembalian	0	0	0
Service	0	0	0

Pada pengujian model BiLSTM, ada 3 kelas *intent* dengan performa paling buruk yaitu Transaksi, Pengembalian, dan *Service*. *Precision*, *recall*, dan *F1-score* dari ketiga kelas tersebut bernilai 0. Ini berarti prediksi *intent* pada ujaran dengan *actual intent* Transaksi, Pengembalian dan *Service* seluruhnya salah.

Berdasarkan hasil pengujian kelas *intent* ke-4 model, kelas *intent Service* selalu memiliki performa terburuk dengan *precision*, *recall*, dan *F1-score* bernilai 0. Hal ini mungkin disebabkan oleh data ujaran dengan *actual intent Service* hanya berjumlah 9. Untuk memperbaiki performa kelas *intent* tersebut, dapat dilakukan penambahan data. Namun karena waktu penelitian terbatas, penambahan data tidak dilakukan. Solusi lain untuk memperbaiki performa kelas *intent* adalah dengan melakukan *class merging* dengan kelas *intent* yang paling serupa. Berdasarkan Tabel 6.2, diputuskan bahwa kelas *Service* akan digabung dengan kelas Garansi menjadi kelas *Aftersales*.

Tabel 6.12 Daftar Kelas *Intent* Final

Kelas Intent	**Keterangan**
Ketersediaan	Kelas ini berisi pertanyaan seputar info ketersediaan *smartphone*, termasuk tanggal *restock*, dan jumlah barang *ready stock*. Pertanyaan mengenai ketersediaan dapat berdasarkan tipe, warna, maupun spesifikasi *smartphone*. Contoh: Mas Samsung S9 ram untuk 6GB ada gak ?
Harga	Kelas ini berisi pertanyaan seputar harga *smartphone*, termasuk nego harga, total harga + ongkos kirim, harga grosir, hingga diskon dan promo.

Kelas Intent	Keterangan
	Contoh: Kak iphone *plus* 256GB *second gold* masih bisa nego harga tidak kak?
Pengiriman	Kelas ini berisi pertanyaan seputar pengiriman barang, termasuk jenis kurir, estimasi durasi pengiriman, *packing* barang, dan lain-lain. Contoh: kalo yes pesen sekarang masih bisa ikut pengiriman hari ini?
Spesifikasi	Kelas ini berisi pertanyaan seputar spesifikasi *smartphone*, mulai dari RAM, *chipset*, Memori internal, kamera, dan fitur-fitur lain. Contoh: ini s9+ *single* sim atau *dual* sim?
Aftersales	Kelas ini berisi pertanyaan seputar garansi *smartphone*, termasuk jenis garansi, penyedia garansi, durasi garansi, prosedur klaim garansi, hingga *service center*. Contoh: Garansi resmi atau internasional kak?
Sapaan	Kelas ini berisi sapaan sehari-hari. Contoh: Halo, selamat pagi.
Pembayaran	Kelas ini berisi pertanyaan seputar pembayaran, termasuk metode pembayaran, cicilan, dan nomor rekening tujuan. Contoh: klou misalnya cicil bos.. berapa berbulan dibayar
Tipe	Kelas ini berisi pertanyaan seputar tipe *smartphone*, termasuk tipe ROM *smartphone*, tipe *smartphone* berdasarkan harga, spesifikasi, dan performa. Contoh: Gan HP ram 4 gb, kamera 5 *megapixel* dapet apa ya?
Kelengkapan	Kelas ini berisi pertanyaan seputar kelengkapan dalam paket penjualan *smartphone*.

Kelas Intent	Keterangan
	Contoh: setiap pembelian iphone apakah tetep dapet *earphone* biasa?
Kondisi	Kelas ini berisi pertanyaan seputar kondisi *smartphone*, termasuk kondisi fisik barang, segel pada box, dan durasi pemakaian. Contoh: Permisi, iphone 6 *gold second* udah dipake berapa lama?
Aksesori	Kelas ini berisi pertanyaan seputar aksesori *smartphone* seperti *case*, *screen guard*, baterai, *charger*, dan lain-lain. Contoh: Bisa sekalian beli tamper glassnya?
Keaslian	Kelas ini berisi pertanyaan seputar keaslian barang. Contoh: ini brg ori bukan *refurbish*?
Toko	Kelas ini berisi pertanyaan seputar toko fisik dari penjual, termasuk alamat toko, cabang toko, hingga jam buka toko. Contoh: Punya toko fisik ga bang? Kalau ada dimana ya?
Transaksi	Kelas ini berisi pertanyaan seputar transaksi, termasuk prosedur pemesanan, dan pembatalan pemesanan. Contoh: Kak maaf kalau saya mau *cancel* pembelian bisa tidak kak?
Pengembalian	Kelas ini berisi pertanyaan seputar pengembalian barang yang telah diterima, baik karena cacat atau ketidaksesuaian dengan iklan. Pertanyaan termasuk prosedur, dan syarat dan ketentuan. Contoh: Gan untuk pembelian hape redmi 4x ada cacat dibagian layarnya, apakah bisa diretur?

Dengan demikian, ke-108 percobaan yang akan dilakukan selanjutnya akan menggunakan daftar kelas *intent* pada Tabel 6.12.

6.2.2 Hasil *Data Labeling*

Pada tahap ini seluruh data yang telah dikumpulkan diberi label salah satu dari 15 kelas *intent* pada Tabel 6.12. Kolom label ditambahan di sebelah kolom teks ujaran pada *dataset* dan setiap barisnya diisi dengan label kelas *intent* dari ujaran pada kolom teks di baris yang sama. Berikut contoh hasil yang didapatkan dari proses pemberian label.

Tabel 6.13 Hasil Pemberian Label

Teks	Label
Misi gan,hp Oppo F1 ready?	Ketersediaan
yang silver metallic ada gan?	Ketersediaan
mba untuk hp harga 1.5 an yang udah 4g ada gak ?	Ketersediaan
memori berapa ya ?	Spesifikasi
itu baru mbak ?	Kondisi
Barang ready gk gan?	Ketersediaan
Kira2 jika pesan hari ini sampe kapan?	Pengiriman
Barang masih ada ?	Ketersediaan
garansi apa mas?	*Aftersales*
Kamera silent kan?	Spesifikasi
yg ready warna apa aja?	Ketersediaan
Fullset gan?	Kelengkapan

Setelah seluruh ujaran pada *dataset* diberi label, dilakukan perhitungan distribusi data berdasarkan label masing-masing. Hasil distribusinya dapat dilihat pada Tabel 6.14.

Tabel 6.14 Distribusi Data Berdasarkan Label

Label	Jumlah Data
Ketersediaan	488
Harga	218
Pengiriman	209
Spesifikasi	181
Aftersales	159
Sapaan	117
Pembayaran	83
Tipe	78

Label	Jumlah Data
Kelengkapan	77
Kondisi	65
Aksesori	38
Keaslian	28
Toko	25
Transaksi	22
Pengembalian	18

Berdasarkan Tabel 6.14, dapat dilihat bahwa *dataset* yang digunakan merupakan *dataset* dengan kategori *unbalanced* atau tidak seimbang, karena selisih jumlah data ujaran antar label cukup besar. Hal ini disebabkan oleh keterbatasan dari proses pengumpulan data percakapan.

6.2.3 Hasil Pembersihan *Dataset*

Sebelum *dataset* dapat digunakan, dilakukan proses pembersihan setiap data ujaran dengan tujuan meminimalisir *token* berbeda namun memiliki makna semantik yang sama. Proses pembersihan *dataset* terdiri dari *lowerize/case folding*, penghapusan huruf berulang, penghapusan/pemisahan tanda baca dan simbol, dan *tokenizing*. Berikut contoh hasil dari proses pembersihan *dataset*.

Tabel 6.15 Hasil Pembersihan *Dataset*

Aktivitas / Kondisi	Hasil
Ujaran Awal	Ooooo oke, harganya berapa ya mas?
Lowerize	ooooo oke, harganya berapa ya mas?
Menghapus Karakter Berulang	oo oke, harganya berapa ya mas?
Memisahkan Tanda Baca dan Simbol	oo oke , harganya berapa ya mas ?
Tokenizing	[“oo”, “oke”, “,”, “harganya”, “berapa”, “ya”, “mas”, “?”]

6.2.3.1 *Token Count* Per Kelas

Setelah proses *tokenizing*, jumlah kemunculan setiap *token* pada masing-masing kelas *intent* dihitung untuk melihat *token* apa saja yang menjadi ciri khas *intent* tersebut. Berikut 10 *token* dengan jumlah kemunculan terbanyak pada masing-masing kelas.

Tabel 6.16 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Ketersediaan

No.	*Token*	*Count*
1	?	462
2	ready	262
3	gan	166
4	ada	143
5	warna	121
6	,	84
7	yang	81
8	apa	69
9	masih	61
10	iphone	5

Tabel 6.17 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Harga

No.	*Token*	*Count*
1	?	195
2	berapa	68
3	harganya	58
4	gan	53
5	ya	47
6	bisa	34
7	harga	34
8	,	31
9	kak	28
10	ada	27

Tabel 6.18 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Pengiriman

No.	*Token*	*Count*
1	?	199
2	bisa	99
3	gan	56
4	kirim	51
5	hari	51
6	ya	44
7	,	41
8	ini	38
9	dikirim	32
10	kira	28

Tabel 6.19 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Spesifikasi

No.	*Token*	*Count*
1	?	172
2	ini	44
3	berapa	38
4	ya	34
5	gan	33
6	nya	31
7	sudah	28
8	apakah	26
9	apa	25
10	kak	21

Tabel 6.20 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas *Aftersales*

No.	*Token*	*Count*
1	?	162
2	garansi	114
3	ya	41
4	apa	31
5	gan	29
6	resmi	27
7	ada	25
8	kalau	21

No.	*Token*	*Count*
9	bisa	21
10	garansinya	20

Tabel 6.21 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Sapaan

No.	*Token*	*Count*
1	gan	34
2	selamat	28
3	kak	23
4	halo	22
5	pagi	21
6	siang	19
7	sore	16
8	.	16
9	,	14
10	malam	13

Tabel 6.22 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Pembayaran

No.	*Token*	*Count*
1	?	76
2	bisa	55
3	cod	21
4	ga	15
5	kredit	14
6	ya	13
7	gan	12
8	,	12
9	gak	10
10	kalau	10

Tabel 6.23 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Tipe

No.	*Token*	*Count*
1	?	67
2	yang	45
3	gan	35

No.	*Token*	*Count*
4	ya	34
5	apa	32
6	,	19
7	ini	18
8	global	17
9	hp	16
10	ada	11

Tabel 6.24 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Kelengkapan

No.	*Token*	*Count*
1	?	75
2	bonus	29
3	,	20
4	apa	19
5	gan	13
6	aja	12
7	dapet	11
8	ada	10
9	masih	10
10	apakah	9

Tabel 6.25 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Kondisi

No.	*Token*	*Count*
1	?	66
2	masih	14
3	gan	12
4	apa	10
5	baru	9
6	second	8
7	ya	8
8	ada	8
9	kak	8
10	Segel	7

Tabel 6.26 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Aksesori

No.	*Token*	*Count*
1	?	36
2	gan	17
3	ada	14
4	sekalian	10
5	jual	9
6	bisa	9
7	hp	9
8	ga	9
9	beli	7
10	nya	6

Tabel 6.27 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Keaslian

No.	*Token*	*Count*
1	?	27
2	ori	19
3	gan	11
4	ini	8
5	asli	6
6	kan	5
7	pembeli	5
8	atau	4
9	ya	3
10	harganya	3

Tabel 6.28 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Toko

No.	*Token*	*Count*
1	?	24
2	toko	11
3	dimana	9
4	ada	6
5	di	6
6	ya	6
7	gan	5
8	mana	5

No.	*Token*	*Count*
9	mas	5
10	tokonya	3

Tabel 6.29 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Transaksi

No.	*Token*	*Count*
1	?	20
2	saya	12
3	order	11
4	mau	10
5	bisa	10
6	gan	6
7	pre	6
8	kak	6
9	,	5
10	gimana	5

Tabel 6.30 *Token* Dengan Jumlah Kemunculan Terbanyak Pada Kelas Pengembalian

No.	*Token*	*Count*
1	bisa	12
2	?	12
3	gan	7
4	apakah	6
5	ya	5
6	barang	5
7	untuk	5
8	ga	4
9	pengembalian	4
10	kira	4

Berdasarkan tabel di atas, dapat dilihat bahwa beberapa kelas memiliki *token* khas yang jumlah kemunculannya lebih banyak dibanding *token* yang lain. Contohnya kelas Ketersediaan memiliki *token 'ready'* dengan kemunculan 262 kali, kelas Harga memiliki *token* 'harganya' dengan kemunculan 58 kali, kelas *aftersales* memiliki *token* 'garansi' dengan kemunculan

114 kali, kelas Sapaan memiliki *token* ‘selamat’ dengan kemunculan 28 kali, dan kelas pembayaran memiliki *token* ‘cod’ dengan kemunculan 21 kali.

6.3 Hasil *Data Splitting*

Dataset yang telah diberi label dibagi menjadi tiga bagian/set yaitu *train set*, *dev set*, dan *test set*. *Train set* akan digunakan untuk pembelajaran model. Model melihat seluruh sampel data dari *train set* dan belajar dari sana. *Dev set* digunakan untuk mengevaluasi model pada saat proses pembelajaran dilakukan. Sehingga, sistem dapat mengetahui model terbaik dan menyimpannya sebelum terjadi *overfitting* pada *train set*. Sedangkan *test set* digunakan untuk mengevaluasi model final setelah *training* dilakukan.

Split data dilakukan dengan rasio 80:10:10 untuk *train set*, *dev set*, dan *test set* secara berurutan. Rasio tersebut diterapkan untuk setiap kelas *intent*. Berikut hasil distribusi data berdasarkan kelas *intent* setelah dilakukan proses *splitting*.

Tabel 6.31 Hasil Distribusi Data Setelah *Data Splitting*

	Train Set (80%)	**Dev Set (10%)**	**Test Set (10%)**
Ketersediaan	390	49	49
Harga	174	22	22
Pengiriman	167	21	21
Spesifikasi	144	18	19
Aftersales	127	16	16
Sapaan	93	12	12
Pembayaran	66	8	9
Tipe	62	8	8
Kelengkapan	61	8	8
Kondisi	52	6	7
Aksesori	30	4	4
Keaslian	22	3	3
Toko	20	2	3
Transaksi	17	2	3
Pengembalian	14	2	2
TOTAL	1439	181	186

6.4 Hasil Implementasi Model *Word Embedding*

Implementasi *pretrained word embedding model* menghasilkan 1245 kata yang diketahui dan 55 kata yang tidak diketahui dalam *dataset*. Kata yang diketahui telah memiliki representasi vektor, sedangkan kata yang tidak diketahui diberi nilai vektor secara *random*. Sehingga total kata pada kamus *word vector* adalah 1300 kata.

6.5 Hasil Pembuatan Model LSTM dan Model *Neural Network* Lain

Skenario pembuatan model dalam penelitian ini berjumlah 108 seperti yang telah dijelaskan pada sub bab 4.5.

6.5.1 Konfigurasi Awal Model

Sebelum menjalankan proses *training*, harus dilakukan konfigurasi model terlebih dahulu. Konfigurasi akan sama untuk semua skenario agar hasilnya dapat dibandingkan satu sama lain. Berikut tabel terkait konfigurasi awal model.

Tabel 6.32 Konfigurasi Awal Model

Parameter	Nilai
Epoch	150
Batch	1
Pretrained Word Embedding Model	Word2Vec Skip-gram
Embedding Dimension	300
Bias	*True*
Batch_first	*False*
Dropout	1

6.5.2 Durasi *Training* Model

Training model LSTM, RNN, GRU, dan BiLSTM dilakukan menggunakan komputer 1 dan komputer 2. Durasi *training* setiap skenario pada komputer 1 dan komputer 2 dapat dilihat pada Tabel 6.33 dan Tabel 6.34**.**

Tabel 6.33 Durasi Training Model Pada Komputer 1 (Detik)

	Num Layer	SGD			*Momentum*			ADAM		
Learning Rate		**0.001**	**0.01**	**0.015**	**0.001**	**0.01**	**0.015**	**0.001**	**0.01**	**0.015**
RNN	**1**	1180	1080	1071	1149	1162	1155	1354	1371	1370
	2	1545	1511	1559	1602	1601	1603	1961	1976	1977
	3	1945	1950	1946	2087	2096	2093	2613	2614	2677
GRU	**1**	1278	1253	1252	1362	1348	1338	1545	1582	1586
	2	1824	1907	1910	2081	2086	2062	2494	2507	2474
	3	2545	2558	2540	2892	2886	2881	3449	3458	3480
LSTM	**1**	1261	1243	1266	1537	1649	1481	1697	1718	1720
	2	2042	2043	2039	2516	2425	2366	2800	2800	2807
	3	2902	2902	2896	3372	3320	3300	3946	3938	3943
BI-LSTM	**1**	2054	2021	2112	3862	3805	3855	2797	2809	2825
	2	3868	3825	3860	5898	5895	5890	5400	5409	5438
	3	5660	5621	5662	8403	8416	8432	8040	8057	8087

Tabel 6.34 Durasi Training Model Pada Komputer 2 (Detik)

	Num Layer	SGD			*Momentum*			ADAM		
Learning Rate		**0.001**	**0.01**	**0.015**	**0.001**	**0.01**	**0.015**	**0.001**	**0.01**	**0.015**
RNN	**1**	1140	1114	1116	1255	1247	1248	1561	1560	1539
	2	1447	1444	1440	1679	1675	1676	2176	2178	2144
	3	1787	1793	1792	2141	2125	2131	2820	2823	2783
GRU	**1**	1139	1138	1139	1269	1264	1267	1581	1580	1579
	2	1501	1500	1503	1737	1736	1740	2229	2227	2230
	3	1881	1880	1903	2216	2217	2219	2897	2898	2893
LSTM	**1**	1173	1171	1171	1279	1279	1287	1546	1556	1544
	2	1548	1549	1550	1765	1767	1769	2217	2222	2213
	3	1960	1963	1958	2270	2271	2274	2900	2901	2890
BI-LSTM	**1**	1612	1602	1598	1836	1837	1842	2278	2280	2275
	2	2419	2425	2422	2833	2834	2837	3660	3720	3659
	3	3211	3209	3211	3786	3782	3793	5040	4980	5012

Dapat dilihat bahwa secara keseluruhan, durasi *training* model pada komputer 2 lebih cepat dibandingkan dengan komputer 1. Hal ini disebabkan oleh spesifikasi komputer terutama CPU dan GPU. Selain spesifikasi komputer, parameter yang digunakan juga berpengaruh terhadap durasi *training* setiap model. Salah satu parameter yang memiliki pengaruh cukup besar yaitu *num layer*. Berdasarkan Tabel 6.33 dan Tabel 6.34, semakin banyak jumlah *num layer*, maka durasi *training* akan semakin lama. Parameter lain yang berpengaruh adalah jenis *optimizer*. Jenis *optimizer* dengan durasi *training* paling cepat ke paling lambat secara berurutan yaitu SGD, *Momentum*, ADAM. *Learning rate* tidak berpengaruh terhadap durasi *training* karena model-model dengan konfigurasi sama namun berbeda *learning rate* memiliki perbedaan durasi *training* yang tipis.

Di antara ke-4 arsitektur, arsitektur model dengan durasi paling cepat ke lambat secara berurutan adalah RNN, GRU, LSTM, BiLSTM. Ini disebabkan oleh kompleksitas arsitektur, dimana RNN merupakan model dengan arsitektur standar, GRU dan LSTM merupakan *gated* model dengan jumlah *gate* 2 dan 3, sedangkan BiLSTM merupakan LSTM dengan dua arah yang tentunya lebih kompleks.

6.6 Hasil Analisis dan Pengujian Performa Model LSTM dan *Neural Network* Lain

Pada tahap ini, 108 model yang telah dibuat diuji pada *test set* dan dievaluasi performanya. Metrik yang digunakan dalam pengujian yaitu akurasi, *precision*, *recall*, dan *F1-score*. Hasil pengujian pada komputer 1 dan komputer 2 dirata-rata untuk mendapatkan hasil akhir. Hasil akhir pengujian dari 108 model yang dibuat dapat dilihat pada Tabel 6.36 hingga Tabel 6.39. Keterangan warna dari keempat tabel tersebut dapat dilihat pada Tabel 6.35.

Tabel 6.35 Keterangan Warna

Warna	Keterangan
	Model terbaik per arsitektur pada setiap *optimizer*
	Model terbaik per arsitektur secara umum

Tabel 6.36 Hasil Test Accuracy Seluruh Model

	Num Layer	SGD (%)			*Momentum* (%)			ADAM (%)		
Learning Rate		**0,001**	**0,01**	**0,015**	**0,001**	**0,01**	**0,015**	**0,001**	**0,01**	**0,015**
RNN	**1**	70,43	22,31	13,44	27,42	11,02	12,91	29,15	8,34	6,18
	2	76,62	22,58	16,13	26,08	2,96	1,35	12,10	5,92	5,92
	3	76,61	19,90	12,37	17,48	4,03	8,34	16,67	4,57	5,92
GRU	**1**	68,55	82,53	80,92	81,46	34,68	23,39	87,10	47,05	42,47
	2	67,21	81,19	85,22	76,08	83,88	19,36	81,18	18,82	16,40
	3	51,62	80,38	82,80	78,50	37,37	8,61	70,43	5,11	5,11
LSTM	**1**	61,03	76,08	78,76	79,30	82,26	80,65	83,61	72,58	68,01
	2	41,40	79,30	80,65	79,84	80,92	81,19	83,07	24,20	17,47
	3	29,57	71,77	78,50	69,36	77,96	75,54	73,39	13,44	6,45
BI-LSTM	**1**	56,99	79,31	79,84	77,69	83,61	82,53	83,07	72,85	62,91
	2	41,40	79,57	79,57	79,03	80,92	81,18	77,96	13,98	10,48
	3	26,61	74,46	79,30	75,81	79,57	77,15	76,62	6,19	1,61

Tabel 6.37 Hasil Precision Seluruh Model

	Num Layer	SGD (%)			*Momentum* (%)			ADAM (%)		
Learning Rate		**0,001**	**0,01**	**0,015**	**0,001**	**0,01**	**0,015**	**0,001**	**0,01**	**0,015**
RNN	**1**	74,84	38,68	28,87	51,35	10,33	11,33	52,70	8,72	5,31
	2	77,32	39,53	33,56	49,77	4,49	3,24	16,60	5,05	5,05
	3	78,46	27,33	20,99	42,23	1,64	8,73	17,33	5,45	4,87
GRU	**1**	78,88	83,87	82,22	83,74	69,96	64,81	87,78	65,82	72,04
	2	75,37	82,02	86,70	79,22	82,83	45,66	81,90	61,77	49,00
	3	58,91	81,69	84,36	80,20	62,34	9,21	73,52	2,13	5,55
LSTM	**1**	75,04	78,20	79,52	80,76	83,84	80,96	85,40	79,48	78,36
	2	50,55	82,52	82,91	80,99	82,82	81,94	84,23	60,39	54,18
	3	30,36	75,99	78,77	71,26	77,85	76,09	75,91	54,50	9,97
BI-LSTM	**1**	73,56	80,63	82,28	79,04	84,56	83,53	83,80	77,33	74,17
	2	24,56	81,04	81,52	79,77	82,08	82,39	79,09	30,21	19,04
	3	12,40	77,74	80,92	77,58	78,84	77,53	79,93	5,68	0,03

Tabel 6.38 Hasil Recall Seluruh Model

	Num Layer	SGD (%)			*Momentum* (%)			ADAM (%)		
Learning Rate		**0,001**	**0,01**	**0,015**	**0,001**	**0,01**	**0,015**	**0,001**	**0,01**	**0,015**
RNN	**1**	70,43	22,31	13,44	27,42	11,02	12,91	29,30	8,34	6,18
	2	76,62	22,58	16,13	26,08	2,96	1,35	12,10	5,92	5,92
	3	76,61	19,90	12,37	17,48	4,03	8,34	16,67	4,57	5,92
GRU	**1**	68,55	82,53	80,92	81,46	34,68	23,39	87,10	47,05	42,47
	2	67,21	81,19	85,22	76,08	83,88	19,36	81,18	18,82	16,40
	3	51,62	80,38	82,80	78,50	37,37	8,61	70,43	5,11	5,11
LSTM	**1**	61,03	76,08	78,76	79,30	82,26	80,65	83,61	72,58	68,01
	2	41,40	79,30	80,65	79,84	80,92	81,19	83,07	24,20	17,47
	3	29,38	71,77	78,50	69,36	77,96	75,54	73,39	13,44	6,45
BI-LSTM	**1**	56,99	79,31	79,84	77,69	83,61	82,53	83,07	72,85	62,91
	2	41,40	79,57	79,57	79,03	80,92	81,18	77,96	13,98	10,48
	3	26,61	74,46	79,30	75,81	79,57	77,15	76,62	6,19	1,61

Tabel 6.39 Hasil F1-score Seluruh Model

	Num Layer	SGD (%)			*Momentum* (%)			ADAM (%)		
Learning Rate		**0,001**	**0,01**	**0,015**	**0,001**	**0,01**	**0,015**	**0,001**	**0,01**	**0,015**
RNN	**1**	71,44	23,62	12,99	31,17	8,42	11,09	31,54	7,92	5,10
	2	76,32	24,99	17,00	29,60	1,83	0,52	11,18	4,36	4,36
	3	76,83	21,03	12,93	20,67	2,06	6,72	13,28	3,70	4,22
GRU	**1**	71,61	82,51	80,82	81,90	40,32	28,83	86,67	50,16	45,41
	2	68,56	81,16	85,28	76,93	82,64	23,38	80,72	22,85	19,75
	3	51,52	80,23	82,68	78,22	39,42	7,76	70,68	2,73	3,55
LSTM	**1**	65,29	76,24	78,29	79,04	81,85	79,88	83,44	73,71	70,49
	2	39,83	79,99	81,00	79,53	80,41	80,73	83,03	27,20	20,12
	3	24,66	73,60	77,92	69,48	77,12	74,32	74,14	16,24	6,43
BI-LSTM	**1**	62,08	78,73	80,00	77,32	83,21	82,15	82,73	73,85	65,73
	2	30,11	79,83	79,65	78,60	80,40	80,49	77,64	15,31	10,14
	3	16,12	74,73	79,20	75,85	78,51	76,38	76,72	5,08	0,05

6.6.1 Konfigurasi Parameter Terbaik

Dari hasil pengujian, dilakukan analisis untuk mengetahui konfigurasi parameter terbaik setiap arsitektur. Analisis dilakukan dengan membandingkan hasil pengujian pada setiap konfigurasi. Metrik yang digunakan untuk analisis adalah akurasi dan *F1-score* karena kedua metrik tersebut cukup untuk merepresentasikan performa model.

Berdasarkan hasil pengujian dengan metrik akurasi pada Tabel 6.36, model RNN memiliki skor akurasi paling baik dengan *optimizer* SGD, 2 *num layer*, dan *learning rate* 0,001. *Gated* model yaitu GRU dan LSTM memiliki skor akurasi terbaik dengan *optimizer* ADAM, 1 *num layer*, dan *learning rate* 0,001. Sedangkan model BiLSTM memiliki skor akurasi terbaik dengan *optimizer Momentum*, 1 *num layer*, dan *learning rate* 0,01. Konfigurasi parameter terbaik berdasarkan akurasi dapat dilihat pada Tabel 6.40.

Tabel 6.40 Konfigurasi Parameter Terbaik Berdasarkan Akurasi

	Optimizer	*Num Layer*	*Learning Rate*	Akurasi (%)
RNN	SGD	2	0,001	76,72
GRU	ADAM	1	0,001	87,10
LSTM	ADAM	1	0,001	83,61
BiLSTM	*Momentum*	1	0,01	83,61

Berdasarkan hasil pengujian dengan metrik *F1-score* pada Tabel 6.39, perbedaan konfigurasi parameter terbaik dengan Tabel 6.40 hanya terdapat pada model RNN. Model RNN memiliki *F1-score* terbaik dengan 3 *num layer*, *optimizer* SGD, dan *learning rate* 0,001. Sedangkan model GRU, LSTM, dan BiLSTM memiliki *F1-score* terbaik dengan parameter yang sama seperti pada Tabel 6.40. Konfigurasi parameter terbaik berdasarkan *F1-score* dapat dilihat pada Tabel 6.41.

Tabel 6.41 Konfigurasi Parameter Terbaik Berdasarkan *F1-score*

	Optimizer	*Num Layer*	*Learning Rate*	*F1-score* (%)
RNN	SGD	3	0,001	76,83
GRU	ADAM	1	0,001	86,67
LSTM	ADAM	1	0,001	83,44
BiLSTM	*Momentum*	1	0,01	83,21

Karena terdapat perbedaan konfigurasi parameter terbaik pada Tabel 6.40 dengan Tabel 6.41, maka diputuskan bahwa konfigurasi parameter terbaik yang akan digunakan pada tahap selanjutnya menggunakan konfigurasi parameter pada Tabel 6.41. Ini dilakukan karena *F1-score* merupakan metrik pengukuran terbaik pada *dataset* dengan distribusi kelas tidak seimbang.

Selain konfigurasi parameter terbaik, hasil pengujian menunjukkan bahwa untuk melakukan *training* model RNN pada konfigurasi apapun, *learning rate* terbaik adalah 0,001. Sedangkan untuk *gated* model seperti GRU, LSTM, dan BiLSTM, *learning rate* terbaik bergantung pada *optimizer* yang digunakan. *Training* model GRU/LSTM/BiLSTM menggunakan *optimizer* SGD paling baik dilakukan dengan *learning rate* 0,015, *training* menggunakan *optimizer Momentum* paling baik dilakukan dengan *learning rate* 0,01, dan *training* menggunakan *optimizer* ADAM paling baik dilakukan dengan *learning rate* 0,001.

6.6.2 Arsitektur Model Terbaik

Berdasarkan pengujian ke-4 arsitektur model yang dilatih dengan konfigurasi parameter terbaik pada Tabel 6.41, diperoleh hasil pengukuran sebagai berikut.

Tabel 6.42 Hasil Pengujian Pada Model Dengan Konfigurasi Parameter Terbaik

	Akurasi	***Precision***	***Recall***	***F1-score***
RNN	76,61	78,46	76,61	76,83
GRU	87,10	87,78	87,10	86,67
LSTM	83,61	85,50	83,61	83,44

	Akurasi	***Precision***	***Recall***	***F1-score***
BiLSTM	83,61	83,80	83,61	83,21

Pada Tabel 6.42 dapat dilihat bahwa GRU merupakan arsitektur dengan performa klasifikasi *intent* terbaik pada *dataset* yang dimiliki. Disusul oleh arsitektur LSTM dan BiLSTM yang memiliki performa tidak jauh berbeda. Sementara RNN merupakan arsitektur dengan performa paling buruk dengan skor akurasi, *precision*, *recall*, dan *F1-score* kurang dari 80%.

Hasil ini mendukung beberapa penelitian yang dilakukan sebelumnya, antara lain penelitian Wenpeng Yin dkk yang menemukan bahwa GRU merupakan arsitektur dengan performa terbaik untuk tugas klasifikasi teks, mengalahkan LSTM [19]. Selain itu, penelitian Suman Ravuri dan Stoclke juga menyebutkan bahwa *gated* RNN (pada penelitian ini LSTM, GRU, dan BiLSTM) menghasilkan performa yang lebih baik jika dibandingkan dengan regular/basic RNN [6].

LSTM, GRU, dan BiLSTM dapat menghasilkan performa yang lebih baik dibandingkan RNN karena ketiga model tersebut memiliki *gates* untuk mengatasi masalah *vanishing gradient*. Masalah *vanishing gradient* sering terjadi pada teks dengan *sequence* yang panjang. Sehingga, *intent* dari beberapa ujaran dengan kalimat yang lebih panjang dapat diprediksi lebih baik oleh LSTM, GRU, dan BiLSTM.

6.6.3 Durasi Klasifikasi *Intent* Model Terbaik

Pada tahap ini dilakukan pengujian durasi model GRU dalam melakukan klasifikasi *intent* suatu ujaran. Pengujian dilakukan menggunakan tiga skenario berdasarkan panjang ujaran, yaitu ujaran dengan 5 *token*, ujaran dengan 10 *token*, dan ujaran dengan 20 *token*. Ketiga ujaran tersebut yaitu :

- selamat malam, mau tanya
- gan untuk hp dengan ram 4gb adanya apa ya?
- untuk iphone 7 plus warna hitam apa masih ada diskon gan? soalnya di gambar tertulis diskon sampai hari rabu

Pengujian diulang sebanyak 10 kali dan mengambil nilai rata-rata sebagai hasil akhirnya. Berikut hasil pengujian durasi klasifikasi *intent* model GRU pada ketiga skenario.

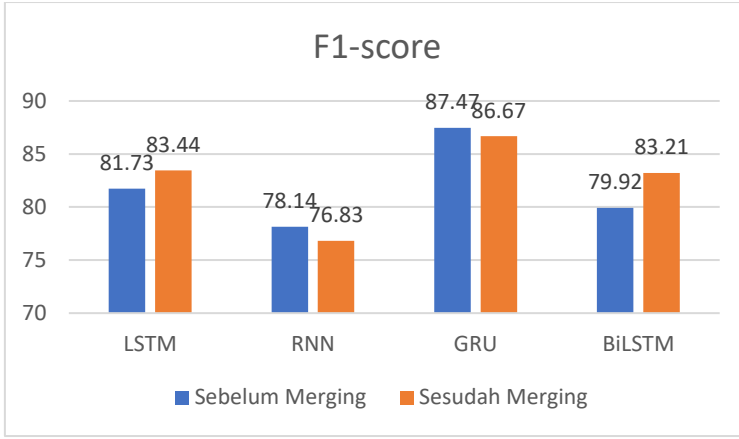
Tabel 6.43 Durasi Klasifikasi *Intent* Model GRU

No	Duration (Second)		
	5	**10**	**20**
1	0,0070	0,0070	0,0090
2	0,0070	0,0080	0,0100
3	0,0060	0,0080	0,0110
4	0,0080	0,0090	0,0100
5	0,0080	0,0090	0,0100
6	0,0080	0,0090	0,0090
7	0,0060	0,0079	0,0080
8	0,0060	0,0080	0,0090
9	0,0070	0,0080	0,0080
10	0,0060	0,0080	0,0090
AVG	0,0069	0,0082	0,0093

Berdasarkan Tabel 6.43, durasi klasifikasi *intent* model GRU bergantung pada jumlah *token* dalam suatu ujaran. Semakin banyak jumlah *token*, maka durasi klasifikasi *intent* akan semakin lama.

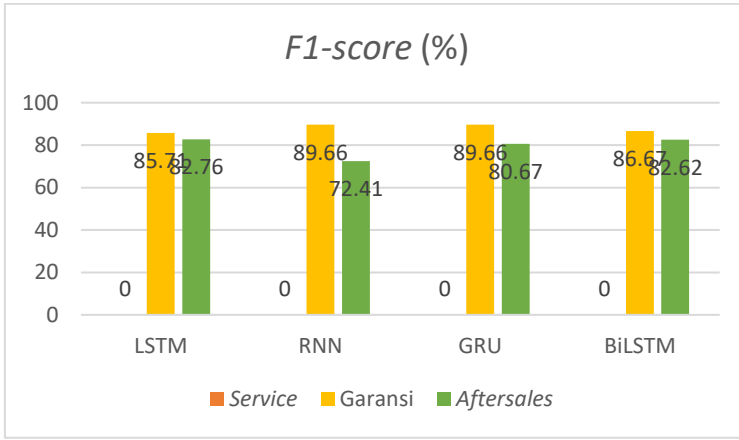
6.6.4 Analisis *Class Merging*

Pada sub bab 6.2.1 telah dilakukan *class merging* pada kelas *intent* Garansi dan *Service* menjadi *aftersales*. Dengan konfigurasi parameter yang sama, dilakukan perbandingan performa ke-4 arsitektur model sebelum *class merging* dan setelah *class merging*. Perbandingan dapat dilihat pada gambar berikut.



Gambar 6.1 *F1-score* Model Sebelum dan Sesudah *Class Merging*

Berdasarkan Gambar 6.1, *F1-score* model LSTM dan BiLSTM setelah *class merging* mengalami peningkatan performa global. Sedangkan, model RNN dan GRU mengalami penurunan performa global.



Gambar 6.2 *F1-score Intent Service*, Garansi, dan *Aftersales*

Pada Gambar 6.2 dilakukan perbandingan *F1-score* pada *intent Service*, Garansi sebelum dilakukan *class merging* dengan *intent Aftersales* setelah dilakukan *class merging*. Hasilnya *intent Aftersales* memiliki *F1-score* lebih rendah dibanding

dengan Garansi, namun jauh lebih tinggi dibanding *Service* yang selalu bernilai 0.

Dari perbandingan tersebut dapat diketahui bahwa *class merging* merupakan salah satu solusi untuk menghapus kelas *intent Service* dengan performa yang sangat buruk. Namun, hal ini berakibat pada rendahnya performa kelas *intent* hasil *class merging* yaitu *Aftersales* jika dibandingkan dengan Garansi. untuk meningkatkan performa kelas *intent Aftersales*, tetap perlu dilakukan penambahan data seputar *service* dan *service center*.

6.6.5 Performa Kelas *Intent*

Pada tahap ini dilakukan pengujian performa setiap kelas *intent* dari ke-4 model dengan konfigurasi parameter terbaik. Karena model dibuat pada 2 komputer, maka hasil akhir pengujian setiap kelas *intent* adalah rata-rata dari hasil pengujian model pada komputer 1 dan komputer 2. Berikut hasil pengujian setiap kelas *intent* pada ke-4 model.

Tabel 6.44 Hasil Pengujian Performa Kelas *Intent* Model LSTM

	Precision (%)	**Recall (%)**	**F1-score (%)**
Ketersediaan	95,92	93,88	94,88
Harga	88,18	84,09	86,04
Pengiriman	79,27	100	88,43
Spesifikasi	77,64	73,68	74,46
Aftersales	92,31	75	82,76
Sapaan	100	100	100
Pembayaran	79,22	77,78	77,5
Tipe	51,89	75	61,32
Kelengkapan	67,78	56,25	60,18
Kondisi	74,68	92,86	81,75
Aksesori	45,83	37,5	41,07
Keaslian	100	66,67	80
Toko	100	33,33	50
Transaksi	100	83,33	90
Pengembalian	75	50	58,33

Berdasarkan pengujian model LSTM pada Tabel 6.44, kelas *intent* Sapaan, Ketersediaan, dan Pengiriman merupakan tiga kelas dengan performa terbaik dengan. Sedangkan tiga kelas *intent* dengan performa terburuk adalah Aksesori, Toko, dan Pengembalian. Hal ini mungkin terjadi karena jumlah data ketiga kelas *intent* tersebut berdasarkan Tabel 6.14 sedikit, dengan jumlah di bawah 40 data.

Tabel 6.45 Hasil Pengujian Performa Kelas *Intent* Model RNN

	Precision (%)	**Recall (%)**	**F1-score (%)**
Ketersediaan	90,83	85,71	87,89
Harga	84,47	75	79,44
Pengiriman	86,96	95,24	90,91
Spesifikasi	71,96	60,53	65,69
Aftersales	80,77	65,63	72,41
Sapaan	96,15	100	98
Pembayaran	73,89	77,78	75,73
Tipe	47,51	87,5	61,33
Kelengkapan	53,47	56,25	54,78
Kondisi	75,71	64,29	69,05
Aksesori	37,5	25	29,17
Keaslian	62,5	83,33	71,43
Toko	41,67	50	45,24
Transaksi	37,5	33,33	34,29
Pengembalian	33,33	50	40

Berdasarkan pengujian model RNN pada Tabel 6.45, kelas *intent* Sapaan, Pengiriman, dan Ketersediaan juga merupakan tiga kelas dengan performa terbaik. Sedangkan tiga kelas *intent* dengan performa terburuk adalah kelas Aksesori, Transaksi, dan Pengembalian.

Tabel 6.46 Hasil Pengujian Performa Kelas *Intent* Model GRU

	Precision (%)	**Recall (%)**	**F1-score (%)**
Ketersediaan	95,04	97,96	96,47
Harga	92,95	88,64	90,69
Pengiriman	82	97,62	89,13
Spesifikasi	79,76	81,58	80,54
Aftersales	91,99	71,88	80,67

	Precision (%)	Recall (%)	F1-score (%)
Sapaan	100	100	100
Pembayaran	70	77,78	73,68
Tipe	79,44	93,75	85,95
Kelengkapan	93,75	87,5	90,42
Kondisi	81,25	92,86	86,67
Aksesori	70	37,5	42,22
Keaslian	100	66,67	80
Toko	75	33,33	45
Transaksi	75	50	60
Pengembalian	50	50	50

Berdasarkan pengujian model GRU pada Tabel 6.46, kelas *intent* Sapaan, Ketersediaan, dan Harga merupakan tiga kelas dengan performa terbaik. Sedangkan tiga kelas dengan performa terburuk adalah Aksesori, Toko, dan Pengembalian.

Tabel 6.47 Hasil Pengujian Performa Kelas *Intent* Model BiLSTM

	Precision (%)	Recall (%)	F1-score (%)
Ketersediaan	95	96.94	95.96
Harga	88.93	90.91	89.9
Pengiriman	78.46	95.24	86.03
Spesifikasi	63.03	71.05	66.25
Aftersales	92.26	75	82.62
Sapaan	100	100	100
Pembayaran	72.22	83.33	76.98
Tipe	70.83	62.5	66.07
Kelengkapan	92.86	81.25	86.67
Kondisi	68.75	57.14	60.61
Aksesori	66.67	50	57.14
Keaslian	87.5	83.33	82.86
Toko	66.67	33.33	41.67
Transaksi	75	33.33	45
Pengembalian	100	50	66.67

Berdasarkan pengujian model BiLSTM pada Tabel 6.47, kelas *intent* Sapaan, Ketersediaan, dan Harga merupakan tiga kelas dengan performa terbaik. Sedangkan tiga kelas *intent* dengan performa terburuk adalah Toko, Transaksi, dan Aksesori.

6.6.5.1 Pengaruh Jumlah Data

Dari keempat hasil pengujian setiap kelas *intent* di atas, diketahui bahwa Ketersediaan, Harga, dan Pengiriman termasuk kelas dengan performa terbaik. Hal ini disebabkan kelas *intent* Ketersediaan, Harga, dan Pengiriman merupakan tiga kelas dengan jumlah data terbanyak (Tabel 6.14). Sementara itu, kelas *intent* terburuk dari pengujian keempat model adalah Toko, Transaksi, Aksesori, dan Pengembalian. Hal ini disebabkan karena kelas Toko, Transaksi, Aksesori, dan Pengembalian termasuk ke dalam lima kelas dengan jumlah data paling sedikit (Tabel 6.14).

6.6.5.2 Pengaruh Jumlah *Word Vocab*

Kelas *intent* Sapaan merupakan kelas dengan jumlah data terbanyak ke-6 berdasarkan Tabel 6.14 namun performanya lebih tinggi jika dibandingkan dengan kelas Ketersediaan, Harga, Pengiriman, Spesifikasi, dan *Aftersales* yang memiliki jumlah data lebih banyak. Pada model LSTM, GRU, dan BiLSTM kelas Sapaan memiliki *F1-score* 100. Sedangkan pada model GRU *F1-score* kelas Sapaan mengalami sedikit penurunan menjadi 98.

Di sisi lain, kelas *intent* Keaslian dengan jumlah data paling sedikit ke-4 memiliki performa yang cukup tinggi, mengalahkan performa kelas Aksesori dan Pembayaran pada keempat model. Pada model LSTM, RNN, GRU, BiLSTM kelas Keaslian memiliki *F1-score* secara berturut-turut sebesar 80, 71,43, 80, 82,86.

Berdasarkan pengamatan tersebut, terdapat anomali pengaruh jumlah data pada kelas Sapaan dan Keaslian. Sehingga dilakukan perhitungan jumlah *word vocab* pada seluruh data dalam setiap kelas *intent* untuk melihat apakah ada pengaruh jumlah *word vocab* terhadap performa kelas *intent*. Berikut hasil perhitungan jumlah *word vocab* pada setiap kelas *intent*.

Tabel 6.48 Jumlah *Word Vocab* Setiap Kelas *Intent*

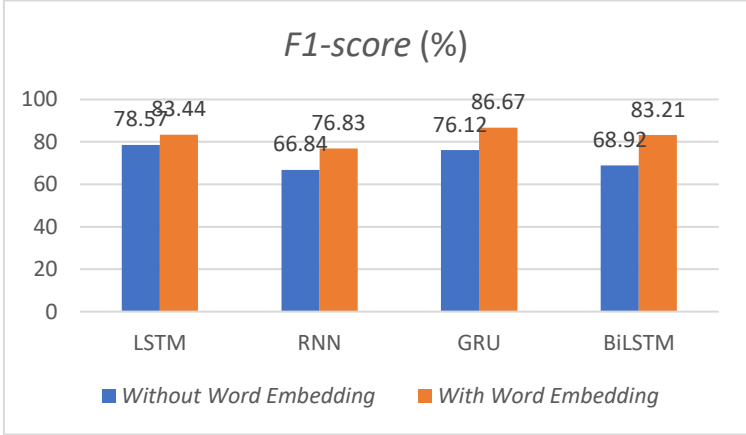
Label	Jumlah *Word Vocab*
Ketersediaan	357
Harga	299
Pengiriman	338
Spesifikasi	329
Aftersales	263
Sapaan	39
Pembayaran	213
Tipe	207
Kelengkapan	180
Kondisi	124
Aksesori	127
Keaslian	66
Toko	72
Transaksi	90
Pengembalian	99

Berdasarkan Tabel 6.48 dapat dilihat bahwa kelas *intent* Sapaan dan Keaslian merupakan kelas dengan jumlah *word vocab* paling sedikit. Sehingga dapat disimpulkan bahwa jumlah *word vocab* cukup berpengaruh terhadap performa kelas *intent.* Semakin sedikit *word vocab* suatu kelas, maka performa kelas tersebut cenderung semakin baik, begitu pula sebaliknya.

6.6.6 Pengaruh *Word Embedding*

Percobaan selanjutnya adalah menguji pengaruh *pretrained word embedding model* yang diimplementasikan untuk melakukan proses representasi kata menjadi vektor. Untuk melakukan pengujian tersebut, dilakukan pembuatan model LSTM, RNN, GRU, dan BiLSTM dengan konfigurasi parameter terbaik yang dilatih tanpa *word embedding*. Keempat model tersebut kemudian dibandingkan performanya dengan model yang dilatih menggunakan *word embedding*. Berikut

perbandingan performa model yang dilatih dengan dan tanpa *word embedding*.



Gambar 6.3 Perbandingan Performa Model Dengan Dan Tanpa *Word Embedding*

Berdasarkan Gambar 6.3, *F1-score* keempat model yang dilatih dengan *word embedding* lebih baik dibanding model yang dilatih tanpa *word embedding*. Dengan menggunakan *word embedding*, *F1-score* model LSTM meningkat sebesar sebanyak 4,87%, dari 78,57% menjadi 83,44%. *F1-score* model RNN meningkat sebesar 9,99%, dari 66,84% menjadi 76,33%. *F1-score* model GRU meningkat sebesar 10,55%, dari 76,12% menjadi 86,67%. Sedangkan *F1-score* model BiLSTM mengalami peningkatan sebesar 14,29%, dari 68,92% menjadi 83,21%. Dari hasil pengujian ini terbukti bahwa penggunaan model *word embedding* dalam pelatihan model LSTM, RNN, GRU, dan BiLSTM sangat berpengaruh terhadap performa model.

Halaman ini sengaja dikosongkan.

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini dibahas mengenai kesimpulan dari semua proses yang telah dilakukan dan saran yang dapat diberikan untuk pengembangan yang lebih baik.

7.1 Kesimpulan

Kesimpulan yang didapatkan dari proses pengerjaan tugas akhir ini antara lain:

1. SGD merupakan *optimizer* paling baik untuk *training* model RNN, *Momentum* paling baik untuk *training* model BiLSTM, dan ADAM paling baik untuk *training* model GRU dan LSTM.
2. Kompleksitas arsitektur RNN memiliki pengaruh terhadap durasi *training* model. Model dengan arsitektur paling kompleks seperti BiLSTM memiliki durasi *training* paling lama. Sedangkan *non-gated* model paling sederhana yaitu *basic* RNN memiliki durasi *training* paling cepat.
3. *Class merging* dapat meningkatkan performa model LSTM dan BiLSTM, namun menurunkan performa model RNN dan GRU.
4. Jumlah data sangat berpengaruh terhadap performa kelas *intent*. Kelas *intent* dengan banyak data cenderung memiliki performa yang baik. Sebaliknya, kelas *intent* dengan sedikit data memiliki performa yang buruk.
5. Jumlah *word vocab* dalam kelas *intent* berpengaruh terhadap performa klasifikasi ujaran dengan *intent* tersebut. Semakin sedikit *word vocab* dalam suatu kelas *intent*, maka performa klasifikasi ujaran dengan *intent* tersebut akan semakin baik.
6. Pada tugas klasifikasi *intent* menggunakan *dataset* percakapan yang dimiliki, model dengan arsitektur GRU memiliki performa yang paling baik. Performa GRU dengan konfigurasi parameter terbaik menghasilkan akurasi mencapai 87,10%.

7. *Gated* RNN seperti LSTM, GRU, dan BiLSTM memiliki performa yang lebih baik dibanding *basic* RNN dalam tugas klasifikasi *intent*.
8. BiLSTM tidak selalu lebih baik dibanding dengan LSTM, terutama pada *dataset* dengan kalimat yang singkat dengan klasifikasi *intent* yang tidak membutuhkan *long-range semantic dependencies*.
9. Penggunaan *word embedding* sangat berpengaruh terhadap performa model *recurrent neural network*. Penggunaannya pada model LSTM terbukti dapat meningkatkan *F1-score* sebesar 4,87%.

7.2 Saran

Dalam pengerjaan tugas akhir, terdapat beberapa saran untuk pengembangan penelitian ke depan, yaitu:

1. Menggunakan kombinasi *optimizer*, *learning rate*, dan *num layer* yang lebih variatif, sehingga mendapatkan penemuan yang baru.
2. Memperbanyak repetisi *training* seluruh skenario model, sehingga hasil pengujian lebih akurat.
3. Menggunakan *well-balanced dataset* dengan jumlah data yang lebih banyak.
4. Menggunakan *dataset* dengan kalimat yang panjang seperti laporan keluhan pelanggan untuk melihat pengaruh panjangnya kalimat terhadap performa setiap arsitektur.
5. Melihat pengaruh *text preprocessing* seperti *stemming* terhadap akurasi model.
6. Menggunakan *contextual word embedding* seperti ELMo untuk melihat pengaruhnya terhadap performa model.

DAFTAR PUSTAKA

[1] "Penetrasi & Perilaku Pengguna Internet Indonesia," Asosiasi Penyelenggara Jasa Internet Indonesia, 2017.

[2] "Payments & E-commerce Report: High-Growth Markets," PPRO Group, 2018.

[3] K. Das, T. Tamhane, B. Vatterott, P. Wibowo, and S. Wintels, "The digital archipelago: How online commerce is driving Indonesia's economic development," p. 12.

[4] D. Jurafsky and J. H. Martin, "Speech and Language Processing." 2000.

[5] S. Ravuri and A. Stolcke, "A comparative study of recurrent neural network models for lexical domain classification," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, 2016, pp. 6075–6079.

[6] S. Ravuri and A. Stoicke, "A comparative study of neural network models for lexical intent classification," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Scottsdale, AZ, USA, 2015, pp. 368–374.

[7] J. Y. Lee and F. Dernoncourt, "Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks," *arXiv:1603.03827 [cs, stat]*, Mar. 2016.

[8] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," p. 21, 1959.

[9] S. J. Russell, P. Norvig, and E. Davis, *Artificial intelligence: a modern approach*, 3rd ed. Upper Saddle River: Prentice Hall, 2010.

[10] L. Claesson and B. Hansson, "Deep Learning Methods and Applications," Chalmers University of Technology, Gothenburg, Sweden, 2017.

[11] E. D. Liddy, "Natural Language Processing," p. 15.

[12] L. Daubigney, M. Geist, S. Chandramohan, and O. Pietquin, "A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 8, pp. 891–902, Dec. 2012.

[13] J. Gao, M. Galley, and L. Li, “Neural Approaches to Conversational AI,” *arXiv:1809.08267 [cs]*, Sep. 2018.
[14] Z. Yan, N. Duan, P. Chen, M. Zhou, J. Zhou, and Z. Li, “Building Task-Oriented Dialogue Systems for Online Shopping,” p. 8.
[15] C. Olah, “Understanding LSTM Networks -- colah’s blog.” [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 25-Feb-2019].
[16] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, “Recurrent Neural Network Based Language Model,” p. 4.
[17] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” 1997.
[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *arXiv:1301.3781 [cs]*, Jan. 2013.
[19] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative Study of CNN and RNN for Natural Language Processing,” *arXiv:1702.01923 [cs]*, Feb. 2017.

BIODATA PENULIS

Penulis lahir di Situbondo pada tanggal 16 Februari 1997. Merupakan anak pertama dari 4 bersaudara. Penulis telah menempuh beberapa pendidikan formal yaitu; SDIF Al-Fikri, SMP Negeri 2 Depok, dan SMA Negeri 1 Depok.

Pada tahun 2015 pasca kelulusan SMA, penulis melanjutkan pendidikan di Jurusan Sistem Informasi Fakultas Teknologi Informasi dan Komunikasi – Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 05211540000022. Selama menjadi mahasiswa, penulis mengikuti berbagai kegiatan kemahasiswaan seperti beberapa kepanitiaan serta pernah menjabat sebagai Staf Departemen Riset dan Teknologi HMSI FTIK ITS dan pada tahun ketiga menjabat sebagai Ketua Departemen Riset dan Teknologi HMSI FTIK ITS. Di bidang akademik, penulis aktif mengikuti lomba dan telah meraih beberapa prestasi seperti Juara Favorit Pengembangan Aplikasi pada perlombaan MAGE 2017, Finalis pada perlombaan Hackathon 3.0 Firetech Neotelemetri Universitas Andalas, Finalis pada perlombaan Hackfest IFEST 2018 Universitas Atma Jaya Yogyakarta, dan Finalis pada perlombaan Appcelerate ITS 2017 & 2018.

Pada tahun keempat, karena penulis memiliki ketertarikan di bidang pengolahan data dan NLP, penulis mengambil bidang minat Akuisisi Data dan Diseminasi Informasi (ADDI). Penulis dapat dihubungi melalui *email* di muhammadazzam1602@gmail.com.

Halaman ini sengaja dikosongkan.

LAMPIRAN A

Contoh Data Berlabel Untuk Klasifikasi *Intent*

Ujaran	Label
mas hp ready?	ketersediaan
hijau lumutnya ada?	ketersediaan
Mas redmi 5a masih ada kah?	ketersediaan
warna hitam masih ada gan ?	ketersediaan
ready warna apa saja mas?	ketersediaan
yang blue rede gan?	ketersediaan
kira kira armor edition adanya kapan gan?	ketersediaan
stok hp redmi 5a warna merah ada brapa gan?	ketersediaan
restock lagi kapan?	ketersediaan
tipe kevlar sudah ready gan?	ketersediaan
Oh iya harganya sesuai dengan iklan itu yaa?	harga
kalau beli 10 ada diskon?	harga
harga hpnya bener segitu gan?	harga
iphone x harganya belum turun ya gan?	harga
harga bisa nego?	harga
harga sudah pas?	harga
sudah termasuk ongkir?	harga
gadapet diskon kah kalo beli 10 unit?	harga
sama ongkir jadi berapa?	harga
gan mau grosiran kena brp nih?	harga
dikirim hari ini bisa?	pengiriman
ongkir brapa mas?	pengiriman
pake gojek bisa kan?	pengiriman
bisa datang sebelum sore gan?	pengiriman
dikirim dari mana?	pengiriman
packing menggunakan apa ?	pengiriman
pengiriman menggunakan apa saja?	pengiriman
kalau mau cepet kirim pake apa ya?	pengiriman

Ujaran	Label
ok saya order sekarang besok sampai kan?	pengiriman
kirim hari ini bisa?	pengiriman
mas hpnya bisa 4g?	spesifikasi
silent camera?	spesifikasi
semua fitur aman kan?	spesifikasi
ini s9+ single sim atau dual sim?	spesifikasi
midnight black kan ya warnanya?	spesifikasi
s9+nya ini yang snapdragon kan gan?	spesifikasi
128gb?	spesifikasi
ini dual camera atau ga?	spesifikasi
kelebihannya apa ya dual camera?	spesifikasi
purplenya gaada gan?	spesifikasi
garansinya TAM kan mas?	*aftersales*
garansi resmi TAM ya?	*aftersales*
garansi brapa tahun bro?	*aftersales*
kalau jatuh apa bisa digaransikan?	*aftersales*
garansi distributor?	*aftersales*
garansi distributor resmi 1 tahun ada ?	*aftersales*
ok gan ini garansi apa ya gan?	*aftersales*
garansi SEIN bukan?	*aftersales*
ga bisa klaim garansi di indo dong ya?	*aftersales*
SEIN?	*aftersales*
Hallo	sapaan
Haloo permisi gan	sapaan
Selamat malam min	sapaan
Pagi Bang	sapaan
Siang Bang	sapaan
Sore Bang	sapaan
Malem Bang	sapaan
Assalamualaikum gan	sapaan
Halo, gan	sapaan

Ujaran	Label
Pagi, gan	sapaan
bisa cod mas?	pembayaran
bisa dicicil gan?	pembayaran
di transfer kemana bos?	pembayaran
bisa dicicil gak barangnya?	pembayaran
bisa cod ga gan?	pembayaran
bisa tuker tambah s7?	pembayaran
oke sip saya tf kemana nih kk?	pembayaran
mandiri ga ada?	pembayaran
Bisa COD?	pembayaran
sip transfer ke mn nih?	pembayaran
armor edition gan?	tipe
Ini s9+ atau s9 biasa?	tipe
s9+ kan ini?	tipe
misi kak ini seri yg baru?	tipe
ini yg limited edition itu?	tipe
o gitu yg pro ada?	tipe
Itu iphone 7 plus apa yang 7 biasa gan	tipe
ini yg Global version gan?	tipe
ini mi8 SE apa bukan gan?	tipe
oya mas ini bukan yang se kan yah	tipe
include box?	kelengkapan
udah free tumi atau softcasenya ga?	kelengkapan
ini include charger?	kelengkapan
Isi box apa aja?	kelengkapan
Fullset gan?	kelengkapan
include box ?	kelengkapan
isi box apa saja ?	kelengkapan
include box ?	kelengkapan
box lengkap kan?	kelengkapan
Kak iphone 8 plus second 258GB kelengkapannya apa aja?	kelengkapan
hp lolos qc gan ?	kondisi
kondisi gimana?	kondisi
segel gan?	kondisi

Ujaran	Label
refurbish itu barang apa gan?	kondisi
masih segel kah?	kondisi
minus apa aja?	kondisi
beneran mulus nih?	kondisi
masih segel kan?	kondisi
normal kan?	kondisi
beneran ga ada lecet2?	kondisi
SAMSUNG S9 di toko anda apakah menyediakan screen guard juga?	aksesori
untuk case ada gan? Sekalian anti goresnya juga	aksesori
Bisa sekalian beli tamper glassnya?	aksesori
jual temper atau casenya sekalian gan?	aksesori
bisa sekalian dipasangi temper gak gan?	aksesori
Bang kalau cuma pengan beli casingnya aja bisa ga ?	aksesori
Itu chargerkaki 2 atau 3?	aksesori
oh iya maaf gan, seri berapa ya temper nya ?	aksesori
kak ini model chargernya apa ya ?	aksesori
min hp nya ini nanti kalo mau beli aksesorisnya banyak nggak ?	aksesori
Ori semua?	keaslian
tp yg dijual sama kk asli gak?	keaslian
Ori/scd?	keaslian
Ka ini barangnya asli atau engga??	keaslian
untuk samsung S8 juga ori?	keaslian
iphon se ori ga?	keaslian
ori?	keaslian
asli?	keaslian
ini brg ori bukan refurbish?	keaslian
Ini barangnya asli atau KW ya?	keaslian

Ujaran	Label
wah toko dimana gan?	toko
buka sampai jam brapa	toko
minta alamat tokonya deh gan	toko
Ready kk region mana	toko
daerah mana?	toko
tokoknya di mana?	toko
Bogornya mana gan?	toko
Malang mana?	toko
Toko dimana gan ?	toko
Ini benar Marvel Phone WTC ?	toko
Baik mas, bagaimana sistem pre-ordernya ?	transaksi
kalau mau order gmn caranya?	transaksi
Gan saya baru bisa bayar setelah jam 12 siang apakah masih bisa dilakukan transaksinya?	transaksi
Kak kmarin saya order galaxy s9, saya mau cancel pembelian bisa tidak kak?	transaksi
Caranya pre-order gimana kak?	transaksi
Mau tanya kenapa pemesanan saya untuk pembelian HP Huawei gagal terus ya gan?	transaksi
mba mau tanya itu kok saya engga bisa order ya? udah 3x coba gagal terus	transaksi
Halo gan, saya mau pre-order nokia 6.1 plus nya masih bisa?	transaksi
Siang gan, saya mau pre-order Pocophone masih bisa?	transaksi
min, apakah pesanan saya masuk? padahal sudah saya bayar tapi kok tulisannya telah dicancel?	transaksi
Baik mas, bagaimana sistem pre-ordernya ?	transaksi
kalau mau order gmn caranya?	transaksi

Ujaran	Label
Gan saya baru bisa bayar setelah jam 12 siang apakah masih bisa dilakukan transaksinya?	transaksi
Kak kmarin saya order galaxy s9, saya mau cancel pembelian bisa tidak kak?	transaksi
Caranya pre-order gimana kak?	transaksi
Mau tanya kenapa pemesanan saya untuk pembelian HP Huawei gagal terus ya gan?	transaksi
mba mau tanya itu kok saya engga bisa order ya? udah 3x coba gagal terus	transaksi
Halo gan, saya mau pre-order nokia 6.1 plus nya masih bisa?	transaksi
Siang gan, saya mau pre-order Pocophone masih bisa?	transaksi
min, apakah pesanan saya masuk? padahal sudah saya bayar tapi kok tulisannya telah dicancel?	transaksi
Gan untuk pembelian hape redmi 4x ada cacat dibagian layarnya, apakah bisa diretur?	pengembalian
Kira kira berapa lama proses returnya ya?	pengembalian
Gan untuk kriteria hape yang bisa dilakukan retur apa saja ya?	pengembalian
Untuk pengembalian barang maksimal dilakukan berapa hari setelah diterima ya gan	pengembalian
Gan ini kok packagingnya rusak begini, apakah hapenya bisa dilakukan pengembalian jika bermasalah	pengembalian
Untuk jasa ekspedisi untuk pengembalian barang apakah ada persyaratan khusus?	pengembalian

Ujaran	Label
Apakah saya bisa request barang pengembalian yang dikirimkan menggunakan ekspedisi ekspres seperti JNE YES	pengembalian
Bang semisal ga cocok nih boleh tuker atau ga?	pengembalian
min nanti kalo barangnya nggak sesuai kondisi bisa dibalikin nggak ?	pengembalian
kalau semisal rusak apakah diganti dengan hp yg baru Atau gimana	pengembalian