Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IS184853

SLOT FILLING UNTUK LANGUAGE UNDERSTANDING DALAM SISTEM DIALOG PADA E-COMMERCE DENGAN ARSITEKTUR BIDIRECTIONAL LONG SHORT-TERM MEMORY-CONDITIONAL RANDOM FIELDS (DOMAIN: PONSEL PINTAR)

SLOT FILLING FOR LANGUAGE UNDERSTANDING IN DIALOGUE SYSTEM ON E-COMMERCE WITH BIDIRECTIONAL LONG SHORT-TERM MEMORY-CONDITIONAL RANDOM FIELDS ARCHITECTURE (DOMAIN: SMART PHONE)

AHMAD NUR SALIM
NRP 05211540000003

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

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

Institut Teknologi Sepuluh Nopember

TUGAS AKHIR - IS184853

SLOT FILLING UNTUK LANGUAGE UNDERSTANDING DALAM SISTEM DIALOG PADA E-COMMERCE DENGAN ARSITEKTUR BIDIRECTIONAL LONG SHORT-TERM MEMORY-CONDITIONAL RANDOM FIELDS (DOMAIN: PONSEL PINTAR)

AHMAD NUR SALIM
NRP 05211540000003

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

Institut
Teknologi
Sepuluh Nopember

UNDERGRADUATE THESIS - IS184853

SLOT FILLING FOR LANGUAGE UNDERSTANDING IN DIALOGUE SYSTEM ON E-COMMERCE WITH BIDIRECTIONAL LONG SHORT-TERM MEMORY-CONDITIONAL RANDOM FIELDS ARCHITECTURE (DOMAIN: SMART PHONE)

AHMAD NUR SALIM
NRP 05211540000003

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

LEMBAR PENGESAHAN

SLOT FILLING UNTUK LANGUAGE UNDERSTANDING DALAM SISTEM DIALOG PADA E-COMMERCE DENGAN ARSITEKTUR BIDIRECTIONAL LONG SHORT-TERM MEMORY-CONDITIONAL RANDOM FIELDS (DOMAIN: PONSEL PINTAR)

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

Oleh:

AHMAD NUR SALIM
NRP. 0521 15 4000 0003

Surabaya, 18 Juli 2019

KEPALA
DEPARTEMEN SISTEM INFORMASI

Mahendrawathi ER, S.T., M. Sc., Ph.D.
NIP. 19761011 200604 2 001

LEMBAR PERSETUJUAN

SLOT FILLING UNTUK LANGUAGE UNDERSTANDING DALAM SISTEM DIALOG PADA E-COMMERCE DENGAN ARSITEKTUR BIDIRECTIONAL LONG SHORT-TERM MEMORY-CONDITIONAL RANDOM FIELDS (DOMAIN: PONSEL PINTAR)

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

Oleh:

AHMAD NUR SALIM
0521 15 4000 0003

Disetujui Tim Penguji: Tanggal Ujian: 9 Juli 2019
Periode Wisuda: September 2019

Renny Pradina K., S.T., M.T., SCJP **(Pembimbing 1)**

Faizal Johan Atletiko, S.Kom., M.T. **(Penguji 1)**

Nur Aini Rakhmawati, S.Kom., M.Sc.Eng., Ph.d **(Penguji 2)**

SLOT FILLING UNTUK LANGUAGE UNDERSTANDING DALAM SISTEM DIALOG PADA E-COMMERCE DENGAN ARSITEKTUR BIDIRECTIONAL LONG SHORT-TERM MEMORY-CONDITIONAL RANDOM FIELDS (DOMAIN: PONSEL PINTAR)

Nama Mahasiswa : Ahmad Nur Salim
NRP : 05211540000003
Departemen : Sistem Informasi FTIK-ITS
Pembimbing I : Renny Pradina Kusumawardani, S.T., M.T., SCJP

ABSTRAK

Saat ini teknologi informasi telah diadopsi dalam berbagai bidang, salah satunya adalah di bidang perdagangan hingga muncullah istilah e-commerce (perdagangan elektronik). Dengan potensi pertumbuhan pelanggan yang bergitu cepat, layanan customer service dapat menjadi suatu tantangan tersendiri pada e-commerce. Pelanggan dapat dengan mudah mengakses layanan e-commerce dari mana saja dan kapan saja. Hal ini dapat menjadi masalah apabila layanan e-commerce mengandalkan customer service konvensional dimana dengan bertambahnya jumlah pelanggan, kebutuhan tenaga untuk customer service juga meningkat . Salah satu cara untuk mengatasi masalah ini adalah dengan melakukan otomasi layanan customer service dengan memanfaatkan sistem dialog (dialogue system).

Dalam sebuah dialogue system berbasis teks terdapat sebuah komponen yang disebut Language Understanding (LU). Komponen ini berfungsi untuk menganalisa teks masukan yang didapatkan agar dapat dimengerti oleh sistem. Salah satu tugas dari LU adalah untuk secara otomatis mengisi sebuah set slot untuk membentuk bingkai semantik. Tugas tersebut biasa disebut dengan istilah slot filling.

Slot filling merupakan salah satu masalah sequence tagging. Salah satu metode untuk menyelesaikannya adalah dengan menggunakan Bidirectional Long Short-Term Memory-Conditional Random Fields (Bidirectional LSTM-CRF). Bidirectional LSTM-CRF menggabungkan kemampuan Bidirectional LSTM untuk menggunakan feature dari masukan sebelum dan sesudahnya dan kemampuan untuk mendapatkan informasi pada tingkat kalimat dari CRF. Model ini memiliki performa yang baik untuk menyelesaikan masalah sequence tagging.

Hasil dari penelitian ini didapatkan kesimpulan bahwa arsitektur BiLSTM-CRF memiliki performa dari arsitektur neural network lain yang digunakan dalam pengujian dengan nilai f-measure sebesar 80.2%. Dalam masalah slot filling yang merupakan salah satu jenis sequence tagging akurasi bukan merupakan ukuran yang tepat untuk menguji performa jika terjadi ketidakseimbangan pada jumlah label yang ada.

Kata Kunci: Dialogoue System, Sequence Tagging, Slot Filling, Bidirectional Long Short-Term Memory-Conditional Random Fields.

SLOT FILLING FOR LANGUAGE UNDERSTANDING IN DIALOGUE SYSTEM ON E-COMMERCE WITH BIDIRECTIONAL LONG SHORT-TERM MEMORY-CONDITIONAL RANDOM FIELDS ARCHITECTURE (DOMAIN: SMART PHONE)

Name	**: Ahmad Nur Salim**
NRP	**: 05211540000003**
Department	**: Information System FTIK-ITS**
Supervisor	**: Renny Pradina Kusumawardani, S.T., M.T., SCJP**

ABSTRACT

Nowadays information technology has been adopted in various fields, one of which is in the field of trade until the term e-commerce (electronic commerce) appears. With the potential of fast customer growth, customer service can be a challenge for e-commerce. Customers can easily access e-commerce services from anywhere and anytime. This can be a problem if e-commerce services rely on conventional customer service where as the number of customers increases, the requirements for customer services also increase. One way to overcome this problem is to automate customer service services using dialogue system.

In a text-based dialogue system there is a component called Language Understanding (LU). This component serves to analyze the input text obtained so that it can be understood by the system. One of the tasks of LU is to automatically fill in a set of slots to form a semantic frame. The task is commonly referred to as slot filling.

Slot filling is one of the sequence tagging problems. One method to solve it is by using Bidirectional Long Short-Term Memory-Conditional Random Fields (Bidirectional LSTM-CRF). Bidirectional LSTM-CRF combines the Bidirectional LSTM

ability to use features from past and future input and the ability to get information at the sentence level from the CRF. This model has a good performance for solving sequence tagging problems.

The results of this study are the conclusion that the BiLSTM-CRF architecture has better performance compared to other neural network architectures used in testing with f-measure value of 80.2%. In the problem of slot filling which is one type of sequence tagging, accuracy is not the right measurement to test performance if there is an imbalance in the number of labels.

Keywords: Dialogoue System, Sequence Tagging, Slot Filling, Bidirectional Long Short-Term Memory-Conditional Random Fields

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 Slot Filling untuk Language Understanding dalam Sistem Dialog pada E-Commerce dengan arsitektur Bidirectional Long Short-Term Memory-Conditional Random Fields (Domain: Ponsel Pintar. 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. Allah SWT, yang selalu menemani dan membimbing penulis dalam segala aspek kehidupan.
2. Bapak Imam Suwardi dan Ibu Ari Sulistyowati selaku orang tua penulis, serta Prasetiyo Hadi Purwoko dan Ardhi Nugroho selaku saudara kandung penulis, yang tiada hentinya mendoakan dan memberikan dukungan kepada penulis.
3. Ibu Mahendrawathi ER. S.T., M.Sc., Ph.D. selaku Ketua Departemen Sistem Informasi ITS Surabaya.
4. Ibu Renny Pradina Kusumawardani, S.T., M.T., SCJP selaku dosen pembimbing yang telah mencurahkan segenap tenaga, waktu dan pikiran dalam penelitian ini, serta memberikan motivasi yang membangun.
5. Bapak Faizal Johan Atletiko, S.Kom., M.T dan Ibu Nur Aini Rakhmawati, S.Kom, M.Sc.Eng,Ph.d selaku dosen penguji yang telah memberikan kritik dan saran yang membuat kualitas penelitian ini lebih baik lagi.
6. Segenap dosen dan karyawan Departemen Sistem Informasi.
7. Aldi, Dani, Edwin, Wahyu, Azzam, Habib dan Indra selaku teman satu atap penulis selama dua tahun merantau di Surabaya.

8. Zaza yang selalu memberikan dukungan moral dan menjadi sobat kuliner penulis selama merantau di Surabaya.
9. Andira, Azzam, Supri, Yoga, Wenny, Magrid, Evia, Faris serta teman-teman penghuni laboratorium ADDI yang telah menemani pengerjaan tugas akhir ini selama di laboratorium.
10. Rekan-rekan Lannister, HMSI Kolaborasi dan HMSI Evolve yang telah memberikan kenangan dan pengalaman semasa kuliah.
11. 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, Juni 2019

Penulis,

Ahmad Nur Salim

DAFTAR ISI

Halaman ini sengaja dikosongkan.

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan.

DAFTAR TABEL

Halaman ini sengaja dikosongkan.

DAFTAR KODE

Halaman ini sengaja dikosongkan.

DAFTAR PERSAMAAN

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

Tidak dapat dipungkiri jika perkembangan teknologi informasi dewasa ini semakin cepat, terlebih dengan semakin mudahnya akses internet. Saat ini teknologi informasi telah diadopsi dalam berbagai bidang, salah satunya adalah di bidang perdagangan hingga muncullah istilah *e-commerce* (perdagangan elektronik).

Indonesia merupakan salah satu negara dengan jumlah pengguna internet terbesar di dunia. Menurut survey yang dilakukan APJII pada tahun 2017 tercatat Indonesia memiliki 143,26 juta pengguna internet aktif[1] atau setara dengan 54,68 persen dari total jumlah penduduk Indonesia. Selain untuk mengakses media sosial, banyak masyarakat Indonesia yang menggunakan internet untuk melakukan transaksi *online* dengan nilai transaksi mencapai sekitar US$8 miliar pada tahun 2017[2]. Pengguna internet di Indonesia juga terus mengalami peningkatan. Tercatat pada tahun 2017 pengguna internet di Indonesia mengalami kenaikan sebesar delapan persen dari tahun sebelumnya yang berjumlah 132.7 juta menjadi 143.26 juta[1], [3]. Seiring dengan tumbuhnya pengguna internet di Indonesia, akan semakin banyak pula masyarakat yang memanfaatkan layanan *e-commerce*. Pada tahun 2017, terdapat 30 juta pembeli *online* dari total penduduk di Indonesia yang berjumlah 260 juta[4].

Dengan potensi pertumbuhan pelanggan yang bergitu cepat, layanan *customer service* dapat menjadi suatu tantangan tersendiri pada *e-commerce*. Pelanggan dapat dengan mudah mengakses layanan *e-commerce* dari mana saja dan kapan saja.

Hal ini dapat menjadi masalah apabila layanan *e-commerce* mengandalkan *customer service* konvensional dimana dengan bertambahnya jumlah pelanggan, kebutuhan tenaga untuk *customer service* juga meningkat. Salah satu cara untuk mengatasi masalah ini adalah dengan melakukan otomasi layanan *customer service* dengan memanfaatkan sistem dialog (*dialogue system*).

Dialogue system atau *conversational agent* merupakan sistem komputer yang dirancang untuk mensimulasikan percakapan dengan manusia. *Dialogue system* dapat menggunakan teks, ucapan, gerakan, maupun metode lain sebagai metode untuk masukan dan keluaran. Dalam sebuah *dialogue system* berbasis teks terdapat sebuah komponen yang disebut *Language Understanding*[5] (LU). Komponen ini berfungsi untuk menganalisa teks masukan yang didapatkan agar dapat dimengerti oleh sistem. Salah satu tugas dari LU adalah untuk secara otomatis mengisi sebuah set slot untuk membentuk bingkai semantik[5]. Tugas tersebut biasa disebut dengan istilah *slot filling*.

Untuk melakukan *slot filling*, terdapat berbagai metode yang bisa dilakukan. *Conditional Random Fields* (CRF) pertama kali digunakan untuk mengisi *slot* oleh Raymond dan Riccardi dan menghasilkan performa yang baik. Setelah itu muncul arsitektur-arsitektur untuk mengatasi masalah slot filling seperti *Recurrent Neural Network*[6] (RNN) dan *Long Short-Term Memory*[7] (LSTM). Tidak berhenti disitu, kemudian arsitektur-arsitektur tersebut dikombinasikan menjadi arsitektur baru, salah satunya adalah *Bidirectional Long Short-Term Memory-Conditional Random Fields* (Bidirectional LSTM-CRF). *Bidirectional* LSTM-CRF menggabungkan kemampuan *Bidirectional* LSTM untuk menggunakan *feature* dari masukan sebelum dan sesudahnya[8] dan kemampuan untuk mendapatkan informasi pada tingkat kalimat dari CRF. Model ini memiliki performa yang baik untuk menyelesaikan masalah *sequence tagging*[9].

1.2 Rumusan Masalah

Berikut rumusan masalah yang akan difokuskan dan diselesaikan dalam tugas akhir ini berdasarkan pada pemaparan latar belakang di atas:

1. Bagaimana implementasi arsitektur *Bidirectional Long Short-Term Memory-Conditional Random Fields* untuk masalah *slot filling*?
2. Bagaimana performa *slot filling* dengan arsitektur *Bidirectional Long Short-Term Memory-Conditional Random Fields* pada masalah *squence tagging* apabila dibandingkan dengan arsitektur *neural network* yang lain?

1.3 Batasan Permasalahan

Berikut batasan masalah dalam pengerjaan tugas akhir ini berdasarkan pada penguraian rumusan masalah di atas:

1. Cakupan domain *dataset* yang digunakan adalah ponsel pintar.
2. Data yang digunakan pada arsitektur *Bidirectional Long Short-Term Memory-Conditional Random Fields* bersumber dari dialog yang dikumpulkan mahasiswa mata kuliah Pengolahan Bahasa Alami pada Semester Gasal 2018/2019 dengan topik jual beli ponsel pintar pada *e-commerce*.
3. Data yang digunakan menggunakan bahasa Indonesia.

1.4 Tujuan

Berikut tujuan dari tugas akhir ini berdasarkan pada pemaparan latar belakang dan rumusan masalah di atas:

1. Melakukan implementasi *slot filling* menggunakan arsitektur *Bidirectional Long Short-Term Memory-Conditional Random Fields.*
2. Melakukan uji coba untuk membandingkan performa arsitektur *Bidirectional Long Short-Term Memory-Conditional Random Fields, Bidirectional Long Short-*

Term Memory, dan *Long Short-Term Memory* pada masalah *sequence tagging*.

1.5 Manfaat

Berikut manfaat yang diharapkan akan diperoleh dari hasil pengerjaan tugas akhir ini :

1. Mengetahui dan memahami cara implementasi *slot filling* pada *dialogue system*.
2. Memahami arsitektur *Bidirectional Long Short-Term Memory-Conditional Random Fields* dan implementasinya.
3. Dapat melakukan eksperimen untuk mengetahui performa *slot filling* dengan arsitektur *Bidirectional Long Short-Term Memory-Conditional Random Field*.
4. Dapat menjadi penelitian awal yang nantinya dapat dilanjutkan pada penelitian-penelitian selanjutnya dan/atau dapat diterapkan pada rancang bangun sebuah *dialog system*.

1.6 Relevansi

Penelitian tugas akhir ini didukung pengetahuan yang diperoleh dari mata kuliah Sistem Cerdas dan Pengolahan Bahasa Alami yang merupakan mata kuliah bidang keilmuan Laboratorium Akuisisi Data dan Diseminasi Informasi.

BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka terdiri dari landasan-landasan yang akan digunakan dalam penelitian tugas akhir ini, mencakup penelitian-penelitian sebelumnya, kajian pustaka, dan metode yang digunakan selama pengerjaan.

2.1 Penelitian Sebelumnya

Terdapat beberapa penelitian yang memiliki topik yang hampir serupa dengan penelitian ini, diantaranya akan dijelaskan pada Tabel 2.1, Tabel 2.2, Tabel 2.3 dan Tabel 2.4.

Tabel 2.1 Literatur 1

Judul	End-to-End Task-Completion Neural Dialogue Systems
Nama, Tahun	Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, Asli Celikyilmaz
Gambaran umum penelitian	Penelitian ini memaparkan beberapa masalah pada *dialog system* konvensional dan kemudian menawarkan solusi *end-to-end dialog system* untuk mengatasi masalah tersebut.
Keterkaitan penelitian	Merupakan literatur untuk memahami tentang *dialog system*

Tabel 2.2 Literatur 2

Judul	Learning End-to-End Goal-Oriented Dialog
Nama, Tahun	Antoine Bordes, Y-Lan Boureau, Jason Weston
Gambaran umum penelitian	Penelitian ini befokus pada penerapan pendekatan *end-to-end* yang berbasis pada *Memory Network* pada sistem dialog yang berorientasi pada tujuan. Dari hasil penilitian ini diketahui bahwa penerapan pendekatan *end-to-end* dapat mendapatkan hasil yang menjanjikan dibandingkan dengan sistem *slot-filling* tradisional.
Keterkaitan penelitian	Merupakan literatur untuk memahami tentang *dialog system*

Tabel 2.3 Literatur 3

Judul	Bidirectional LSTM-CRF Models for Sequence Tagging
Nama, Tahun	Zhiheng Huang, Wei Xu, Kai Yu
Gambaran umum penelitian	Penelitian ini memaparkan hasil penelitian tentang variasi dari *Long Short-Term Memory* (LSTM) yaitu *Bidirectional Long Short-Term Memory-Conditional Random Field* dimana peneliti menggabungkan *Bidirectional* LSTM dan CRF untuk membuat model untuk masalah *sequence tagging*.
Keterkaitan penelitian	Merupakan literatur untuk memahami tentang *Bidirectional Long Short-Term Memory-Conditional Random Fields* dan penerapannya dalam *sequence tagging*

Tabel 2.4 Literatur 4

Judul	Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks
Nama, Tahun	Nils Reimers, Iryna Gurevych
Gambaran umum penelitian	Pada penelitian ini peneliti mengevalusi impelmentasi dari beberapa jenis *neural network* dan *hyperparameter* yang digunakan pada lima masalah *sequence tagging* yaitu POS, *Chunking*, *Entity Recognition* dan *Event Detection*. Dari percobaan yang dilakukan diketahui bahwa terdapat beberapa parameter yang memiliki dampak besar pada performa.
Keterkaitan penelitian	Merupakan literatur yang digunakan untuk memahami pengaruh dari *hyperparameter-hyperparameter* yang ada pada LSTM dan variasinya pada *sequence tagging*

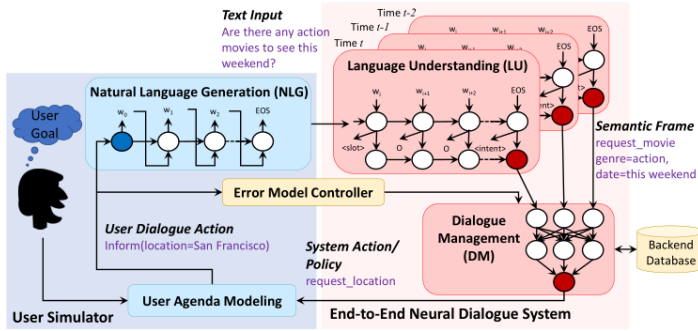
2.2 Landasan Terori

2.2.1 Dialogue System

Dialogue system atau *conversational agent* merupakan sistem komputer yang dirancang untuk mensimulasikan percakapan dengan manusia. *Dialogue system* dapat menggunakan teks, ucapan, gerakan, maupun metode lain sebagai metode untuk masukan dan keluaran. Secara umum, *dialogue system* terbagi

menjadi dua yaitu *task-oriented dialogue agents* dan *chatbot*[10].

Dalam pengembangannya, *dialogue system* terdiri dari tiga komponen utama, yaitu *Language Understanding*, *Dialogue Manager*, dan *Natural Language Generation*.



Gambar 2.1 Skema Sebuah Sistem Dialog dengan Masukan Suara[5]

Dalam pengembangannya, *dialogue system* terdiri dari tiga komponen utama, yaitu *Language Understanding*, *Dialogue Manager*, dan *Natural Language Generation*.

Pada komponen *Language Understanding*, sistem melakukan proses pengenalan tujuan dari masukan pengguna dan mengambil informasi-informasi yang ada pada masukan. Secara umum komponen ini memiliki dua tugasyaitu pengenalan *intent* dan *slot filling*.

Komponen *Dialogue Manager* memiliki dua tahapan di dalamnya yaitu *dialogue state tracking* dan *policy learning*. Pada *dialogue state tracking* dimana dilakukan penyimpanan informasi-informasi terkait *state* dari dialog yang sedang berlangsung. Kemudian pada tahapan *policy learning* sistem menentukan tindakan yang tepat berdasar dari keluaran tahap sebelumnya. Hasil keluaran dari komponen ini kemudian akan dipakai komponen *Natural Language Generation* untuk membuat respon yang tepat untuk pengguna.

2.2.2 Slot Filling

Slot filling merupakan salah satu komponen kunci dalam *Language Understanding* (LU), dimana *slot filling* biasanya dianggap sebagai masalah *sequence labelling* yaitu memberi label semantik yang sesuai pada setiap kata dari masukan yang diberikan[11] untuk kemudian mengambil informasi yang dibutuhkan. Masalah *sequence labelling* dapat diselesaikan dengan model statistika linear seperti *Hidden Markov Model*[12], *Maximum Entropy Markov Model*[13], dan *Conditional Random Fields*[14]. Selain menggunakan model statistika linear, solusi berbasis *deep neural-network* juga dapat digunakan[9][15][16]. Berikut adalah contoh kalimat dengan informasi domain, tujuan, *slot*, dan *named entity*.

Tabel 2.5 Contoh Kalimat Dengan Representasi IOB2

Sentence	*USB*	*3.0*	*ya*	*?*
Slot	B-ITEM-FITUR	I-ITEM-FITUR	O	O

Contoh di atas mengikuti representasi IOB2, dimana pada *token* awal sebuat entitas diberi tambahan “B” pada nama labelnya, sedangkan *token* dari entitas yang posisinya berada setelah *token* pertama entitas diberi tambahan “I” pada nama labelnya[17]. Label “O” digunakan pada *token* yang bukan merupakan bagian dari sebuah entitas[17].

Seperti masalah *sequence labelling* lainnya, masalah *slot filling* dapat diselesaikan dengan metode statistika linear seperti *Conditional Random Fields* (CRF)[18] atau metode berbasis *neural-network* seperti *Recurrent Neural Network* (RNN)[19][6].

2.2.3 Word Embedding

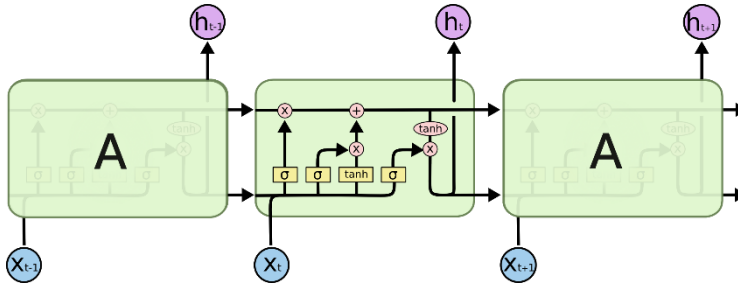
Word embedding merupakan representasi dari sebuah kata atau frasa kedalam sebuah vektor. Pada awalnya *word embedding* dikenal sebagai *word representation*[20]. *Word embedding* dapat digunakan untuk menentukan kemiripan dari kata berdasarkan suatu konteks tertentu. Selain itu, *word embedding*

diketahui dapat meningkatkan performa tugas NLP seperti *sentiment analysis*[21].

Peran *word embedding* diperlukan pada algoritma *deep learning* dan hampir semua arsitektur *deep learning* hanya menggunakan masukan berupa angka sehingga data yang berupa kata atau kalimat harus terlebih dahulu direpresentasikan sebagai vektor angka agar dapat diproses.

2.2.4 Long Short-Term Memory

Long Short-Term Memory Network (LSTM) merupakan sebuah arsitektur artifisial dari *recurrent neural network* (RNN) yang pertama kali deperkenalkan oleh Sepp Hochreiter dan Jürgen Schmidhuber pada tahun 1997[22]. LSTM mengatasi masalah yang dimiliki oleh RNN yaitu sulitnya mengatasi *long-term dependencies*[23]. Sebuah unit LSTM terdiri dari *cell* dan *gate*.



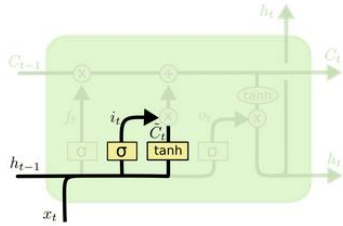
Gambar 2.2 Skema dari Sebuah Unit LSTM[24]

Gate merupakan sebuah mekanisme untuk membiarkan informasi lewat secara opsional. *Gate* terdiri dari sebuah *sigmoid neural network layer* dan sebuah operasi perkalian searah. *Sigmoid layer* ini menghasilkan nilai keluaran antara nol dan satu yang merepresentasikan seberapa banyak informasi yang bisa lewat.

LSTM memiliki kemampuan untuk menghapus atau menambah informasi ke dalam *cell* yang diatur oleh *gate* yang dimiliki unit LSTM yaitu *input gate* dan *output gate*.

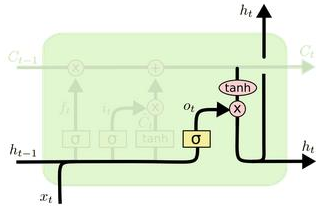
Pada *input gate* terjadi penyimpanan informasi baru dari masukan. Hal ini terdiri dari dua tahap. Pertama, *input gate*

menentukan terlebih dahulu nilai yang akan diperbarui. Selanjutnya, sebuah *tanh layer* membuat vektor dari kandidat nilai yang baru yang akan ditambahkan. Kemudian dua bagian tersebut digabungkan untuk memperbarui informasi pada *cell state*.



Gambar 2.3 Skema Input Gate[24]

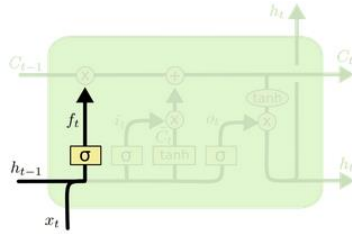
Keluaran dari sebuah unit LSTM akan dikeluarkan dari *output gate*. Keluaran didasarkan pada nilai dari *cell state* yang akan disaring terlebih dahulu. Pertama, sebuah *sigmoid layer* akan menentukan bagian mana saja yang akan dijadikan keluaran. Kemudian *cell state* akan dilewatkan pada *tanh layer* (sehingga nilainya berkisar antara -1 dan 1) dan dikalikan dengan keluaran dari *sigmoid layer* sebelumnya sehingga menghasilkan keluaran yang tepat.



Gambar 2.4 Skema Output Gate[24]

Pada perkembangannya, ditemukan kelemahan dari jaringan LSTM yaitu nilai yang disimpan dalam *cell* dapat terus menerus membesar secara linear seiring berjalannya waktu dan pada akhirnya menyebabkan jaringan tersebut rusak[25]. Untuk mengatasi kelemahan itu, sebuah *gate* baru yang disebut *forget gate* ditambahkan ke dalam struktur unit LSTM sehingga

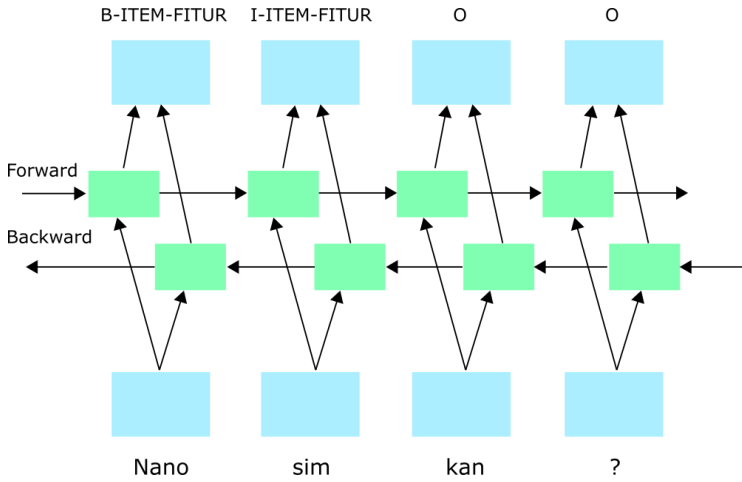
memungkinkan sebuah unit LSTM untuk belajar mengatur ulang dirinya pada waktu tertentu sehingga dapat melepaskan informasi yang tersimpan di dalamnya[25]. Skema dari *forget gate* dapat dilihat pada Gambar 6. *Forget gate* membaca nilai dari h_{t-1} dan x_t dan menghasilkan keluaran berupa angka antara 0 hingga 1 untuk setiap angka pada C_{t-1} (*cell state* pada unit LSTM sebelumnya). Nilai 1 berarti tidak ada informasi yang dilupakan sedangkan nilai 0 berarti semua informasi akan dilupakan.



Gambar 2.5 Skema Forget Gate[24]

2.2.5 Bidirectional Long Short-Term Memory

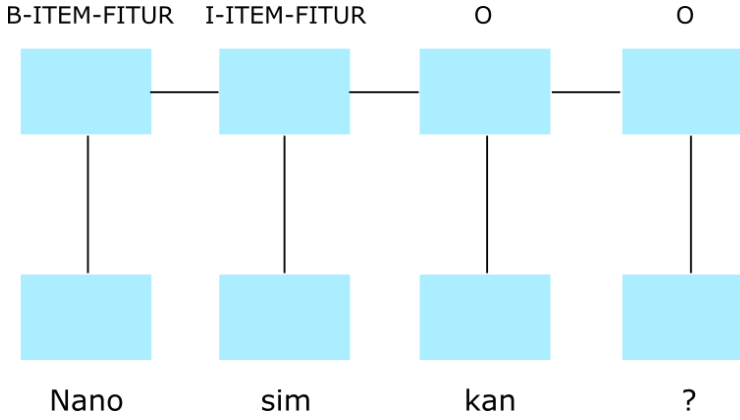
Bidirectional Long Short-Term Memory merupakan pengembangan dari *Long Short-Term Memory* (LSTM). *Bidirectional* LSTM memiliki cara kerja yang mirip dengan *Bidirectional Recurrent Neural Network* (RNN) yaitu membagi neuron menjadi dua arah, satu mengikuti arah waktu positif dan yang lain mengikuti arah waktu negatif. Dibanding dengan LSTM tradisional, *Bidirectional* LSTM dinilai memiliki performa yang lebih baik[26].



Gambar 2.6 Skema dari Jaringan Bidirectional LSTM

2.2.6 Conditional Random Fields

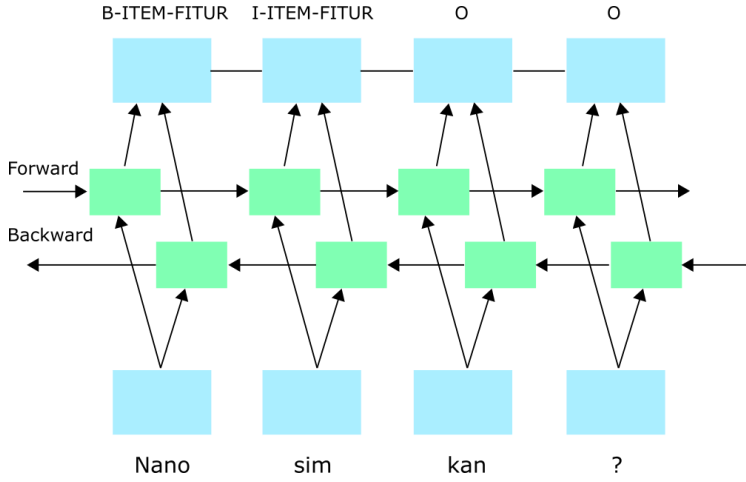
Conditional Random Fields (CRF) merupakan metode permodelan statistik yang seringkali digunakan dalam *machine learning*. CRF merupakan model diskriminatif yang biasa diaplikasikan dalam masalah sekuensial seperti POS *Tagging* dan *named entity recognition*. Metode ini meggunakan informasi kontekstual dari label masukan sebelumnya untuk memprediksi label sebuah masukan.



Gambar 2.7 Skema dari Jaringan CRF

2.2.7 Bidirectional Long Short-Term Memory-Conditional Random Fields

Bidirectional Long Short-Term Memory-Conditional Random Fields (*Bidirectional* LSTM-CRF) merupakan kombinasi dari *Bidirectional Long Short-Term Memory* dengan *Conditional Random Fields*. Selain dapat menggunakan informasi dari *feature* masukan sebelum dan sesudah, model *Bidirectional* LSTM-CRF juga dapat memanfaatkan informasi *tag* pada tingkat kalimat. *Bidirectional* LSTM-CRF sendiri memiliki ketergantungan yang lebih kecil terhadap *word embedding* dibandingkan dengan metode-metode sebelumnya[9].



Gambar 2.8 Skema Jaringan Bidirectional LSTM-CRF

2.2.8 F-Measure

F-Measure (dikenal pula dengan F_1 dan *F-score*) merupakan satuan akurasi dari sebuah percobaan. *F-Measure* menggunakan *precision* dan *recall* dalam perhitungannya. Persamaan dari *F-Measure* adalah sebagai berikut.

$$F_1 = 2 * \left(\frac{recall * precision}{recall + precision}\right)$$

Persamaan 2.2.1

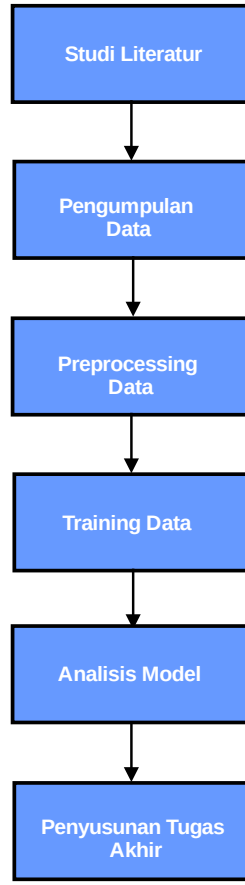
F-Measure merupakan rata-rata harmonis dari *precision* dan *recall* dimana nilainya tertingginya adalah satu dan nilai terendahnya adalah nol.

BAB III
METODOLOGI

Bab ini menjelaskan metodologi yang akan digunakan dalam penyusunan tugas akhir. Metodologi tersebut digunakan agar penyusunan terarah dan sistematis.

3.1 Tahapan Pelaksanaan Tugas Akhir

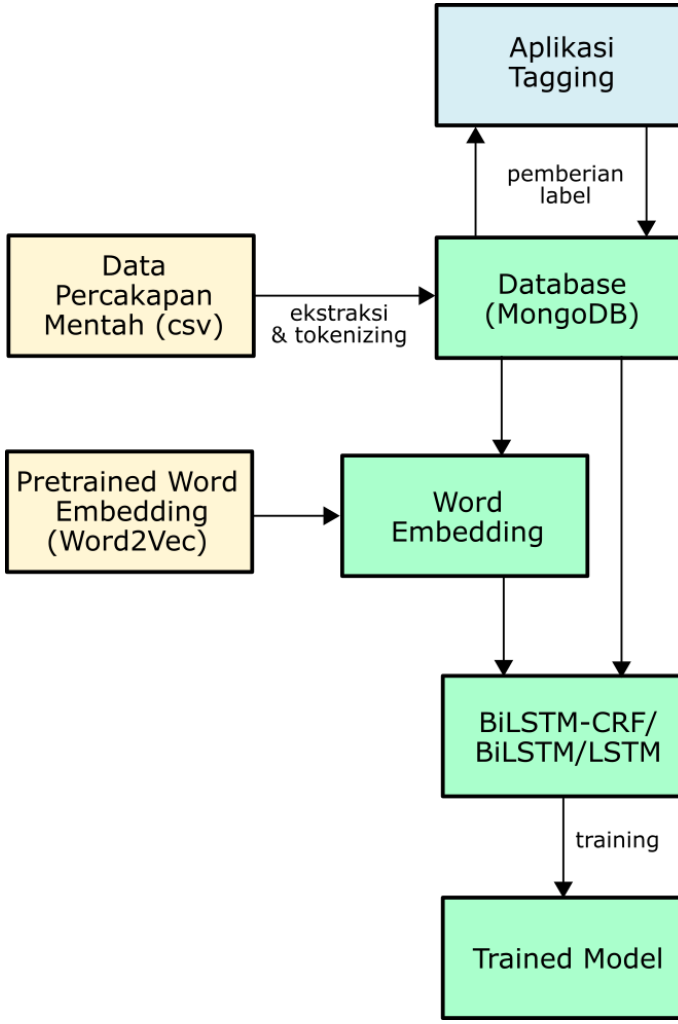
Alur pelaksanaan tugas akhir ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Metodologi Pengerjaan Tugas Akhir

3.2 Arsitektur Penelitian

Pada bagian ini akan dijelaskan garis besar arsitektur yang digunakan pada penelitian ini. Arsitektur dapat dilihat pada Gambar 3.2.



Gambar 3.2 Arsitektur Penelitian

Seperti yang ditampilkan pada Gambar 3.2 Data percakapan yang masih mentah diekstrak terlebih dahulu untuk kemudian dilakukan *tokenizing* dan disimpan pada basis data *mongodb*. Kemudian data pada basis data diberi label yang sesuai untuk kemudian digunakan pada pembuatan model.

3.3 Uraian Metodologi

3.3.1 Studi Literatur

Pada tahap ini akan dilakukan studi literatur untuk memahami konsep, metode dan teknologi yang terkait dan sesuai dengan permasalahan serta solusi yang ditawarkan. Studi literatur juga dilakukan untuk menggali informasi melalui literatur-literatur penelitian terkait dengan permasalahan dan solusi yang ditawarkan. Adapun literatur utama yang digunakan sebagai pedoman dalam penyusunan tugas akhir ini adalah *Bidirectional LSTM-CRF Models for Sequence Tagging* oleh Zhiheng Huang[9], *Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding* oleh Grégoire Mesnil[6], dan *Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks* oleh Nils Reimers[27].

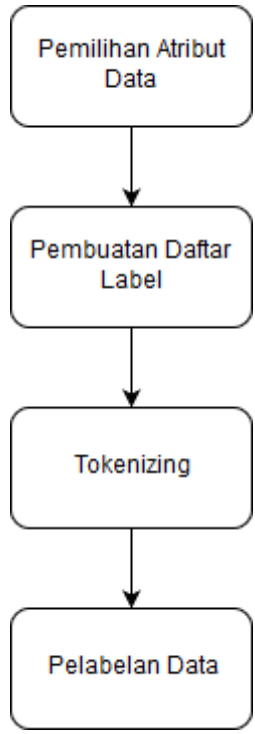
3.3.2 Pengumpulan Data

Pada tahap ini akan dilakukan pengumpulan data yang akan digunakan. Data didapatkan dari arsip mata kuliah Pengolahan Bahasa Alami pada Semester Gasal 2018/2019. Data tersebut merupakan simulasi dialog antara penjual dan pembeli ponsel pintar pada *e-commerce*. Data ini dibuat oleh setidaknya 30 mahasiswa yang familiar dengan *e-commerce*. Data yang digunakan adalah ujaran berupa pertanyaan, sedangkan ujuran yang bukan berupa pertanyaan akan dibuang.

Selain itu penelitian ini menggunakan kumpulan data representasi vektor kata berbahasa Indonesia yang berupa *pretrained word embedding*. *Pretrained word embedding* didapatkan dari arsip tugas akhir milik laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI) Departemen Sistem Informasi ITS.

3.3.3 *Preprocessing* Data

Pada tahap ini akan dilakukan langkah-langkah yang diperlukan agar data menjadi data *training* pada tahap selanjutnya. Langkah-langkah yang akan dilakukan dalam tahap ini dapat dilihat pada Gambar 3.2.



Gambar 3.3 Tahap Pre-processing Data

3.3.3.1 Pemilihan Atribut Data

Pada tahap ini akan dilakukan pemisahan antara atribut yang penting dan tidak penting dalam penelitian. Tujuan dari tahap ini adalah agar data lebih mudah dan efektif saat digunakan pada proses *training* data. Untuk memudahkan akan dibuat kode sederhana untuk mengambil atribut yang dibutuhkan dari data.

3.3.3.2 Pembuatan Daftar Label

Pada langkah ini dilakukan penentuan label *slot* apa saja yang akan digunakan berdasarkan *dataset*. Hasil dari langkah ini akan digunakan dalam pelabelan data.

3.3.3.3 *Tokenizing*

Pada langkah ini dilakukan *tokenizing* dimana kalimat-kalimat pada *dataset* dipisahkan berdasar kata. *Token-token* ini akan digunakan sebagai masukan pada model *neural network* yang dnantinya akan dibuat.

3.3.3.4 Pelabelan Data

Pada langkah ini dilakukan pelabelan pada data dialog. Hal ini bertujuan untuk menentukan bagian mana dari kalimat yang digunakan untuk mengisi *slot*. Pelabelan akan dilakukan dengan mengikuti representasi IOB2 sebagaimana yang ditunjukan pada Tabel 3.1.

Tabel 3.1 Contoh Pelabelan Kalimat

Kata dalam Kalimat	Slot
Bisakah	O
pengiriman	O
barang	O
menggunakan	O
GRAB	B-KURIR-PENGIRIMAN
?	O

Pada representasi IOB2 *token* awal sebuat entitas diberi tambahan “B” pada nama labelnya, sedangkan *token* dari entitas yang posisinya berada setelah *token* pertama entitas diberi tambahan “I” pada nama labelnya[17]. Label “O” digunakan pada *token* yang bukan merupakan bagian dari sebuah entitas[17].

3.3.4 Pengimplementasian BiLSTM-CRF

Pada tahap ini akan dilakukan implementasi model BilSTM-CRF. Untuk proses pelatihan data dialog akan dibagi menjadi data *train*, *dev* dan *test* dengan rasio 70:15:15. Pembuatan dan pelatihan model menggunakan *library* bernama *pytorch*. Selain dari implementasi sendiri, model BiLSTM-CRF dan model *neural network* lain akan dibuat juga dengan menggunakan *toolkit* NCRF++. NCRF++ merupakan *toolkit* untuk *sequence labelling* yang dibangun diatas *pytorch*[28]. Pada setiap iterasi, nilai *f-measure* model akan diuji pada *dataset dev* yang menghasilkan nilai *dev f-measure*. Untuk setiap percobaan, model dengan nilai *f-measure* tertinggi pada *dataset dev* akan disimpan.

3.3.5 Analisis Model

Pada tahap ini akan dilakukan analisis pada performa dari model-model yang dihasilkan pada tahap sebelumnya terhadap *dataset test*. Analisis performa setiap model akan dilakukan dengan menggunakan ukuran *f-measure* yang dikenal juga dengan nama *f1-score* dan *f-score*.

3.3.6 Penyusunan Tugas Akhir

Pada tahap ini akan dilakukan penyusunan dokumentasi dari proses selama pengerjaan tugas akhir yang akan menghasilkan buku tugas akhir. Buku tugas akhir yang dihasilkan diharapkan dapat dimanfaatkan sebagai referensi untuk penelitian selanjutnya.

BAB IV
PERANCANGAN

Bab ini menjelaskan tentang rancangan untuk setiap proses yang akan digunakan dalam penyusunan tugas akhir. Perancangan yang dibuat adalah perancangan pengumbpulan data, perancangan *pre-processing* data dan perancangan pembuatan model BiLSTM-CRF dan model *neural network* lain.

4.1 Perancangan Pengumpulan Data

Pada proses pembuatan model BiLSTM-CRF, data yang digunakan merupakan data dialog. Model *neural ntework* yang akan dibuat bertujuan untuk mengenali label dari kata-kata yang terdapat pada ujaran pembeli sehingga data yang diperlukan hanyalah kalimat pertanyaan yang diajukan oleh pembeli. Untuk itu perlu dilakukan ekstraksi terlebih dahulu karena pada data yang tersedia ujaran dari pembeli tercampur dengan ujaran dari penjual. Hasil dari ekstraksi ini kemudian disimpan kedalam *file csv*. Contoh ujaran yang digunakan dapat dilihat pada Gambar 4.1.

1	Tagging;message;response
2	Ketersediaan;Saya mau beli hape xiaomi redmi 5 plus masih ready stock?;Masih kak silahkan diorder
3	Ketersediaan;Untuk warna tinggal sisa apa aja ya?;Tinggal sisa hitam dan putih saja
4	Ketersediaan;Oh kalo produk yang setara dengan redmi 5 plus dan ram 4gb ada apa ya?;Untuk produk ya
5	Harga;Gan iphone x yang 256 harganya masih bisa dinego ga?;wah sudah nett kak harganya soalnya garan
6	Promo;Wah udah nett ya, kalo misalnya dibonusin aksesoris boleh ga?;Paling kita bisa kasih case aja kak
7	Ketersediaan;J2 pro blue silver ready gan?;ready gan. silahkan diorder
8	Pengiriman;bisa gosend kan gan?;bisa gan. order masuk lgsg diproses, kecuali kl sdh lwt jam 4 sore. krn a
9	Harga;Samsung galaxy note 9 masih bisa nego?;Silahkan dinego
10	Harga;10 juta boleh?;Wah masih belom dapet kak kalo segitu, palingan kita bisa ngasih kalo 11,5 juta
11	Ketersediaan;Galaxy J2 Prime Black ready? Bs gosend?;absolute black ready gan. bisa gosend. besok lgsg
12	Pengiriman;Sip.. aman kan ya by gosend?;aman gan. ada asuransinya jg kok. jd kl smp amit2 knp2, akan d
13	Ketersediaan;Gan saya mau order razer phonenya apakah masih tersedia;Mohon maaf gan untuk semua
14	Spesifikasi;Waduh udah sold out ya gan, kalo yang setara ama razer phone apa ya gan;Kalo untuk yang se
15	Ketersediaan;black 2 pcs ready?;absolute black 2unit ready gan. silahkan diorder..
16	Lainnya;Gold ada? Bs kirim lsg skr pakai gojek? Ini ram 2?;ada sis. iya ini yg ram 2gb. bisa gojek skrg. silah
17	Ketersediaan;Gan untuk poccophone kira kira kapan stok lagi ya?;Untuk restock kami belum bisa memas
18	Lainnya;Biasanya restock barang berapa lama gan?;Kami biasanya melakukan restock sekitar 1 bulan
19	Lainnya;Wah lama juga ya, kalau sudah restock bisa dikabari gan?;Silahkan masukan produk kami kedalar
20	Promo;Gan untuk produk asus zenfone max apakah ada diskon;zenfone max sedang tidak ada diskon gar
21	Lainnya;Oke gan nanti kalo saya bingung boleh ya tanya tanya;Silahkan gan saya siap membantu

Gambar 4.1 Contoh Data Dialog Mentah

Proses ekstraksi atribut dilakukan bantuan kode program sederhana. Setelah dilakukan ekstraksi data akan disimpan dalam file *csv*. Hasil dari ekstraksi adalah seperti berikut.

1	0	Vivo v7 ready stok warna apa saja gan?
2	1	Mau tanya tentang promo di counter Marvel apa masih berlaku?
3	2	Kalau 5.300.000 gimana?
4	3	oh gitu berapa tuh iphone 6nya ada bonus bonus ga?
5	4	garansi mana ini gan?
6	5	Setelah transfer, barang langsung go-s bisa?
7	6	Bedanya apa ya mas?
8	7	Poccophone hitam ready gan?
9	8	beda dari ketinganya apa mas?
10	9	redmi 6a ready warna apa aja gan?
11	10	pake gosend bisa gan?
12	11	Saya mau beli hape xiaomi redmi 5 plus masih ready stock?
13	12	Iphone 7 plusnya ready?
14	13	Kalo begitu berarti saya bisa mengajukan garansi ke semua gerai yang memiliki TAM?
15	14	Ini saya beli Redmi 5 bulan april kemaren, apakah bisa bantu claim untuk garansinya?
16	15	garansi resmi asus indonesia ada di mana ya gan?
17	16	gan beli hp disini dapet promo apaan aja ya?
18	17	gan barangnya second ya?
19	18	Yang dus global habis ya, gan?
20	19	Bisa kirim hari ini Atau masih pre-order?
21	20	Galaxy J2 Prime Black ready?
22	21	Sip.. aman kan ya by gosend?

Gambar 4.2 Contoh Hasil Ekstraksi Data Dialog

4.2 Perancangan *Preprocessing* Data

Sebelum digunakan, data harus terlebih dahulu melalui proses *pre-processing* ata pra pemrosesan terlebih dahulu. Data mentah akan diolah terlebih dahulu agar lebih mudah dan efektif untuk diolah saat pembuatan dan evaluasi model. Untuk mempermudah tahap *prepocessing* dibuat kode sederhana dengan bahasa pemrograman *python*.

4.2.1 Perancangan Pembuatan Daftar Label

Sebelum dilakukan pemberian label pada data percakapan perlu ditentukan terlebih dahulu skema pelabelan yang akan digunakan berdasarkan data yang tersedia. Skema pelabelan nantinya akan menjadi patokan dalam pemberian label pada data. Daftar label perlu dievaluasi karena mempengaruhi performa model. Apabila jumlah label terlalu sedikit, label dapat dihilangkan. Setelah skema pelabelan telah ditentukan, data dialog akan diberi label sesuai dengan skema tersebut.

4.2.2 Perancangan *Tokenizing*

Sebelum diberi label, data percakapan terlebih dahulu dipisah berdasarkan kata. Hal ini dikarenakan pelabelan pada *sequence tagging* berdasarkan kata. Selain itu model BiLSTM-CRF dan model *neural network* lain memproses kalimat masukan berupa kata per kata.

4.2.3 Perancangan Pelabelan Data

Pada tahap ini dilakukan pelabelan pada data percakapan. Proses pemberian label dibantu oleh program *tagger* yang dibuat pada penelitian sebelumnya.

4.3 Perancangan Pembuatan Model BiLSTM-CRF dan Model *Neural Network* Lain

Pada tahap ini dilakukan perancangan untuk membuat model BiLSTM-CRF dan model *neural network* lain. Data yang digunakan sebagai masukan adalah data percakapan.

4.3.1 Perancangan Dataset Model BiLSTM-CRF dan Model *Neural Network* Lain

Dataset yang digunakan dibagi menjadi *dataset train, dev* dan *test*. *Dataset-dataset* tersebut berasal dari data percakapan yang dibagi dengan rasio 70:15:15. Agar persebaran untuk setiap label rata, data terlebih dahulu dibagi berdasarkan label. Kemudian untuk setiap label dibagi menjadi tiga bagian dengan rasio 70:15:15 dan potongan-potongan tersebut disatukan dengan label yang lain sehingga didapat persebaran label yang rata. Metode ini disebut *stratified sampling*. Untuk melakukannya dibuat kode sederhana dengan bahasa pemrograman *python*.

4.3.2 Perancangan Implementasi *Word Embedding*

Untuk membuat model *neural network* dibutuhkan kamus dari vektor kata. Pada penelitian ini *word embedding* yang digunakan diperoleh dari penelitian sebelumnya. Tidak semua kata dalam *dataset* percakapan terdapat pada kamus yang dimiliki model *word embedding* yang digunakan. Untuk mengatasi hal tersebut, kata-kata yang tidak diketahui akan diberi nilai vektor secara acak.

4.3.3 Perancangan Pembuatan Model BiLSTM-CRF dan Model *Neural Network* Lain

Tahap ini bertujuan untuk menghasilkan model BiLSTM-CRF. Masukan yang digunakan adalah data percakapan yang dan *initial weight* yang didapat dari model *word* embedding. Dalam pembuatan model terdapat *hyperparameter* yang dapat diatur untuk mendapatkan hasil terbaik. Pembuatan model BiLSTM-CRF pada penelitian ini menggunakan *library pytorch* dan *toolkit* NCRF++ yang juga berbasis pada *library pytorch*. *Library* pytorch belum menyediakan implementasi dari CRF sehingga harus diimplementasikan sendiri. Dalam penggunaannya *library* ini dapat dijalankan pada GPU untuk performa *training* yang lebih baik jika dibandingkan dengan menggunakan CPU. Hal ini dapat menghemat durasi *training* secara signifikan.

Arsitektur BiLSTM-CRF merupakan gabungan dari Bidirectional LSTM dengan Conditional Random Fields, dimana model sebenarnya adalah CRF sedangkan BiLSTM bertugas untuk memberikan *feature*. Pada *library pytorch* LSTM memiliki beberapa parameter. Parameter-parameter tersebut adalah sebagai berikut.

Tabel 4.1 Parameter LSTM

Paramater	Definisi
input_size	Jumlah dari *feature* dalam suatu masukan
hidden_size	Jumlah *feature* dalam *hidden state*
num_layers	Jumlah dari *recurrent layer*
bidirectional	Jika bernilai *True* maka model akan menjadi *bidirectional*

Pada penelitian ini model-model yang dibuat akan dibandingkan dengan beberapa skenario. Skenario pertama adalah perbandingan model BiLSTM-CRF hasil implementasi dengan model BiLSTM-CRF yang dibuat dengan *toolkit* NCRF++. Selanjutnya adalah percobaan untuk mengetahui kombinasi *optimizer*, *learning rate* yang tepat pada masing-masing arsitektur.

Algoritma *optimizer* yang akan diuji adalah *Adaptive Moment Estimation* (ADAM). Selain itu akan diuji pula algoritma *Stochastic Gradient Descent* (SGD) dan kombinasinya dengan *momentum*. Sedangkan untuk nilai *learning rate* yang akan diuji coba adalah 0.0015, 0.015 dan 0.0005.

4.3.4 Perancangan Analisis dan Pengujian Model BiLSTM-CRF dan Model *Neural Network* Lain

Tahap ini dilakukan untuk menguji dan mengevaluasi model. Dalam setiap iterasi pada proses *training* model akan diuji dengan menggunakan dataset *dev* sehingga didapat nilai presisi, *recall* dan *f-measure*. Dari ketiga ukuran tersebut *f-measure* adalah ukuran utama performa model.

Ketiga ukuran tersebut memiliki fungsi masing-masing. Presisi merepresentasikan frekuensi prediksi yang benar pada setiap prediksi suatu label, *recall* merepresentasikan berapa kali model mengidentifikasi suatu label dengan tepat dari keseluruhan label tersebut, sedangkan *f-measure* merepresentasikan rata-rata harmonis dari presisi dan *recall*.

Model yang mendapatkan nilai *f-measure* yang lebih baik dari nilai sebelumnya saat diuji dengan dataset *dev* akan disimpan. Setelah seluruh iterasi pada proses *training* selesai pengujian dengan menggunakan dataset *test* akan dilakukan pada model dengan nilai *f-measure* terbaik pada *dataset dev*.

BAB V
IMPLEMENTASI

Pada bab ini, akan dijelaskan mengenai implementasi dari perancangan yang telah dilakukan sesuai dengan metode pengembangan yang dibuat.

5.1 Lingkungan Implementasi

Pengerjaan penelitian ini menggunakan dua komputer dengan spesifikasi sebagai berikut.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer 1

Processor	Intel Core i5 8400K
RAM	16 GB DDR4
GPU	Nvidia GeForce GTX 1070
OS	Ubuntu 16.04 LTS

Tabel 5.2 Spesifikasi Perangkat Keras Komputer 2

Processor	Intel Core i5 7200U
RAM	8 GB DDR4
GPU	Nvidia GeForce 930MX
OS	Windows 10 Pro

5.2 Ekstraksi Data Percakapan

Pengekstraksian pada data percakapan dilakukan dengan menggunakan program sederhana. Program yang dibuat untuk proses ini berjumlah lebih dari satu karena terdapat beberapa format dari data yang belum diekstrak sehingga membutuhkan perlakuan yang berbeda-beda.

```
1.  import os
2.  import csv
3.  import nltk
4.  import pandas as pd
5.
6.  with open("data.csv") as csvFile:
7.      headers = ['text', 'label']
8.      questionList = []
9.      questions = []
10.     reader = csv.reader(csvFile, delimiter=','
    )
11.     sentence_pos = 0
12.     pos = 0
13.     for row in reader:
14.         if pos % 2 != 0:
15.             if "?" in row[sentence_pos]:
16.                 question = row[sentence_pos].s
    trip('"')
17.                 questions.append(question)
18.         pos += 1
19.     df = pd.DataFrame(set(questions), columns=
    ['question'])
20.     df.to_csv("output.csv", header=False)
21.
22.     print("found " + str(len(set(questions)))
    + " data")
```

Kode 5.1 Potongan Kode Salah Satu Program Ekstraksi

Potongan kode pada Kode 5.1 merupakan salah satu program ekstraktor yang dibuat. Pada baris ke-10 dilakukan pembacaan dari berkas *csv* dari data percakapan yang mentah untuk kemudian ekstraksi pada perulangan di baris 13 hingga 18. Kemudian hasil dari ekstraksi disimpan pada objek *DataFrame* untuk selanjutnya disimpan kembali dalam berkas *csv*.

```
1.  import os
2.  import csv
3.  import nltk
4.  import pandas as pd
5.
6.  with open("new_data.csv") as csvFile:
7.      questionList = []
8.      questions = []
```

```
9.      reader = csv.reader(csvFile, delimiter=','
    )
10.     sentence_pos = 1
11.     for row in reader:
12.         if row[0].lower() == "pembeli":
13.             if "?" in row[sentence_pos]:
14.                 question = row[sentence_pos].s
    trip('"')
15.                 questions.append(question)
16.     df = pd.DataFrame((set(questions)), column
    s=['question'])
17.     df.to_csv("output.csv", header=False)
18.
19.     print("found " + str(len(set(questions)))
    + " data")
```

Kode 5.2 Potongan Kode Salah Satu Program Ekstraksi

Potongan kode pada Kode 5.2 juga merupakan salah satu program ekstraktor yang dibuat. Pada baris ke-9 dilakukan pembacaan dari berkas *csv* dari data percakapan yang mentah untuk kemudian ekstraksi pada perulangan di baris 11 hingga 15. Kemudian hasil dari ekstraksi disimpan pada objek *DataFrame* untuk selanjutnya disimpan kembali dalam berkas *csv*.

Setelah diekstrak data akan dimasukkan kedalam *file csv* kemudian akan disimpan dalam *database mongodb* untuk dilakukan pemberian label.

```
1.  def fromCSV(self, collection):
2.      with open("data/" + self.filename) as csvF
    ile:
3.          headers = ['text', 'label']
4.          questionList = []
5.          questions = []
6.          reader = csv.reader(csvFile, delimiter
    =',')
7.          sentence_pos = 1
8.
9.          for row in reader:
10.             if "?" in row[sentence_pos]:
11.                 question = row[sentence_pos].s
    trip('"')
```

```
12.                 questions.append([row[sentence
    _pos]])
13.                 word_tokenized = nltk.word_tok
    enize(question)
14.
15.                 tags_init = ["O" for i in rang
    e(len(word_tokenized))]
16.
17.                 questionList.append([word_toke
    nized, tags_init])
18.
19.         df = pd.DataFrame(questionList, column
    s=headers)
20.         with open(os.path.splitext("data/" + s
    elf.filename)[0] + "-
    out.csv", "w", newline='') as output:
21.             wr = csv.writer(output)
22.             wr.writerows(questions)
23.
24.         json_df = json.loads(df.T.to_json()).v
    alues()
25.
26.         db = DbInstance()
27.         db.insert_data(collection, json_df)
```

Kode 5.3 Potongan Kode Untuk Mengunggah Data Ke Database

Potongan kode pada Kode 5.3 berfungsi untuk mengunggah berkas *csv* hasil ekstraksi data percakapan ke *database mongodb*. Pada baris enam dilakukan pembacaan berkas *csv* untuk mengambil data hasil ekstraksi. Kemudian pada bari 13 dilakukan *tokenization* untuk membagi kalimat berdasarkan kata/*token*. Pada baris 15 dilakukan inisiasi label untuk setiap *token* menjadi "O", label ini nantinya akan dirubah pada seiring dengan dilakukannya pelabelan. Selanjutnya data percakapan yang sudah diolah disimpan kedalam berkas *json* pada baris 22 untuk kemudian diunggah ke *database mongodb* pada baris ke 27.

5.3 Pelabelan Data Percakapan

Pada proses pelabelan data percakapan sebelum melakukan pemberian label harus ditentukan dahulu label-label yang akan

digunakan. Penentuan label dilakukan dengan cara membaca seluruh data terlebih dahulu kemudian menentukan label apa saja yang tepat untuk digunakan. Pertama-tama semua entitas yang berhubungan dengan jual beli ponsel pintar dicatat terlebih dahulu kemudian dilakukan pemberian label.

Tabel 5.3 Label Awal

No.	Label
1	JUMLAH-ORDER
2	ITEM-PERLENGKAPAN
3	MODEL-BANDING
4	ITEM-BONUS
5	TUJUAN-PENGIRIMAN
6	WAKTU-PENGIRIMAN
7	ITEM-SPEC
8	ITEM-FITUR
9	ITEM-SUBMODEL
10	KURIR-PENGIRIMAN
11	MERK
12	WARNA
13	MODEL
14	O
15	TRANSAKSI-NOMOR
16	ASAL-PENGIRIMAN
17	ITEM-SPEC-BANDING
18	PROMO-NAMA
19	LOKASI

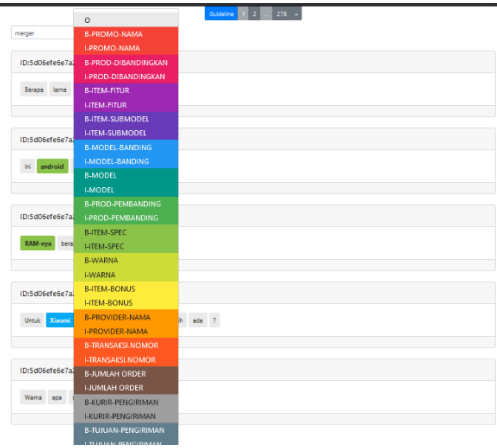
Setelah semua data memiliki label, semua label dihitung jumlahnya. Selanjutnya label dengan jumlah kurang sepuluh dibuang. Hal ini bertujuan agar model memiliki cukup sampel

sehingga dapat mengenali label yang ada. Berikut label akhir yang digunakan.

Tabel 5.4 Label yang Digunakan

Label	**Deskripsi**
JUMLAH-ORDER	Kata yang merepresentasikan jumlah pembelian
ITEM-PERLENGKAPAN	Kata yang merepresentasikan perlengkapan dari ponsel pintar
MODEL-BANDING	Kata yang merepresentasikan model ponsel pintar yang menjadi pembanding
ITEM-BONUS	Kata yang merepresentasikan bonus yang didapatkan dari pembelian ponsel pintar
TUJUAN-PENGIRIMAN	Kata yang merepresentasikan tujuan pengiriman dari pembelian
WAKTU-PENGIRIMAN	Kata yang merepresentasikan lamanya pengiriman ke tujuan pengiriman
ITEM-SPEC	Kata yang merepresentasikan spesifikasi umum dari ponsel pintar
ITEM-FITUR	Kata yang merepresentasikan fitur-fitur yang didukung ponsel pintar
ITEM-SUBMODEL	Kata yang merepresentasikan varisi dari model ponsel pintar
KURIR-PENGIRIMAN	Kata yang merepresentasikan nama kurir pengiriman
MERK	Kata yang merepresentasikan merk dari ponsel pintar
WARNA	Kata yang merepresentasikan warna dari ponsel pintar
MODEL	Kata yang merepresentasikan model dari ponsel pintar
O	Kata yang tidak termasuk kedalam label yang lain

Proses pelabelan dilakukan secara manual dibantu dengan program *tagger* yang dibuat dalam penelitian sebelumnya agar hasil pelabelan dapat tersimpan kedalam *database* secara *real-time*.



Gambar 5.1 Pelabelan pada Data Percakapan dengan *Tagger*

5.4 Pembuatan Model BiLSTM-CRF

5.4.1 Pembersihan Data Duplikat

Tahap ini bertujuan untuk menghapus data ganda pada data dialog.

```
1.  def remove_duplicate(target):
2.      db = DbInstance()
3.      datas, _, _ = db.get_data(target)
4.      headers = ['text', 'label']
5.      tokens = []
6.      sentences = []
7.      for data in datas:
8.          if data.tokens not in tokens:
9.              tokens.append(data.tokens)
10.             sentences.append([data.tokens,data
    .tags])
11.
12.     duplicates_count = len(datas) - len(senten
    ces)
```

```
13.     print("found " + str(duplicates_count) + "
    duplicate(s) in " + target)
14.
15.     collection = db.get_collection(target)
16.     collection.remove({})
17.
18.     df = pd.DataFrame(sentences, columns=heade
    rs)
19.     json_df = json.loads(df.T.to_json()).value
    s()
20.     db.insert_data(target, json_df)
```

Kode 5.4 Potongan Kode Untuk menghapus Data Duplikat

Pembersihan duplikat dilakukan dengan fungsi *remove_duplicate()*. Data yang belum diproses terlebih dahulu diambil dari basis data dengan bantuan *helper class DBInstance* yang diinisiasi pada baris dua. Setelah data diambil dari *database* dengan kode pada baris tiga, dilakukan proses pembersihan data duplikat dilakukan dengan melakukan iterasi pada data tersebut pada baris tujuh hingga baris sepuluh. Hasilnya kemudian disimpan dalam variabel *sentences*. Kemudian basis data yang lama dikosongkan dengan fungsi *remove()* pada baris 16 untuk kemudian diisi kembali dengan data yang terlah bersih dengan fungsi *insert_data()* pada baris 20.

```
1.  from DataInstance import DataInstance
2.  from pymongo import MongoClient
3.
4.  class DbInstance():
5.      def __init__(self):
6.          self.db = self.get_connection()
7.
8.      def get_connection(self):
9.          username = "YOUR USERNAME HERE"
10.         password = "YOUR PASSWORD HERE"
11.         host = "YOUR HOST HERE"
12.         database = "DATABASE NAME"
13.
14.         return MongoClient(f"mongodb+srv://{us
    ername}:{password}@{host}/{database}?retryWrit
    es=true")
```

```
15.
16.     def get_collection(self, collection):
17.         db = self.db['skripsi']
18.
19.         return db[collection]
20.
21.     def get_data(self, collection):
22.         cursor = self.get_collection(collectio
    n).find({})
23.
24.         datas = []
25.         labels = []
26.         words  = []
27.
28.         for i in cursor:
29.             token = i['text']
30.             label = i['label']
31.             data = DataInstance(token, label)

32.             datas.append(data)
33.
34.             labels.extend(label)
35.             words.extend(token)
36.
37.         return datas, labels, words
38.
39.     def get_data_json(self, collection):
40.         cursor = self.get_collection(collectio
    n).find({})
41.
42.         datas = []
43.
44.         for i in cursor:
45.             token = i['text']
46.             label = i['label']
47.             data = [token, label]
48.             datas.append(data)
49.
50.         return datas
51.
52.     def insert_data(self, collection, data):
53.         coll = self.get_collection(collection)

54.         coll.insert(data)
```

Kode 5.5 Potongan Kode Helper Class Database

Potongan Kode 5.5 merupakan kode dari kelas *helper DBInstance* yang dibuat untuk memudahkan operasi yang berhubungan dengan basis data. Fungsi *get_connection()* berguna untuk menginisiasi objek untuk berkomunikasi dengan *database* berdasarkan identitas yang dibutuhkan seperti *username*, *password*, alamat *host* dari basis data dan nama *database* yang dituju. Fungsi *get_data()* berguna untuk mengambil data dari database dari *collection* yang dituju. Data yang dikembalikan berupa objek *DataInstance* yang berisikan *array* dari *token* kalimat dan label untuk masing-masing *token*.

5.4.2 Pembuatan Dataset Model BiLSTM-CRF

Tahap ini bertujuan untuk membuat dataset yang akan digunakan dalam pembuatan model BiLSTM-CRF. Data perlu diolah agar dapat menghasilkan model yang baik. Dataset terbagi menjadi tiga yaitu *train*, *dev* dan *test* degan rasio 70:15:15. Pada saat ekstraksi data dan pelabelan data, *database* data percakapan terbagi-bagi berdasarkan sumbernya sehingga perlu digabungkan terlebih dahulu.

```
1.  def merge_data(index):
2.      headers = ['text', 'label']
3.      db = DbInstance()
4.      target = "k" + str(index)
5.      origin = db.get_data_json(target)
6.
7.      df = pd.DataFrame(origin, columns=headers)

8.      json_df = json.loads(df.T.to_json()).value
    s()
9.
10.     db.insert_data("merger", json_df)
```

Kode 5.6 Potongan Kode untuk Menggabungkan Data

Kode 5.6 merupakan kode untuk menggabungkan *collection* setiap data kedalam *collection* “merger”. Pertama-tama dibuat *instance* dari *helper class database* pada baris tiga kemudian data pada *collection* yang dituju diambil dengan fungsi

get_data_json() pada baris lima dan dijadikan objek *dataframe*. Objek *dataframe* tersebut kemudian diubah menjadi *json* pada baris delapan untuk kemudian dimasukkan kedalam *collection* “merger”.

```
1.  def distribute_tag(datas):
2.      tag_list = ["JUMLAH-ORDER", "ITEM-
    PERLENGKAPAN", "MODEL-BANDING", "ITEM-
    BONUS", "TUJUAN-PENGIRIMAN", "WAKTU-
    PENGIRIMAN", "ITEM-SPEC","ITEM-FITUR", "ITEM-
    SUBMODEL", "KURIR-
    PENGIRIMAN", "MERK", "WARNA", "MODEL", "O"]
3.      tag_dict = {"JUMLAH-ORDER":[], "ITEM-
    PERLENGKAPAN":[], "MODEL-BANDING":[], "ITEM-
    BONUS":[], "TUJUAN-PENGIRIMAN":[], "WAKTU-
    PENGIRIMAN":[], "ITEM-SPEC":[],"ITEM-
    FITUR":[], "ITEM-SUBMODEL":[], "KURIR-
    PENGIRIMAN":[], "MERK":[], "WARNA":[], "MODEL"
    :[], "O":[]}
4.      train = []
5.      dev = []
6.      test = []
7.      print(len(datas))
8.      counter = 0
9.      for _tag in tag_list:
10.         padding = 0
11.         for i in range(len(datas)):
12.             data = datas[i - padding]
13.             for tag in data.tags:
14.                 x = tag[tag.find("-")+1:]
15.                 if x == _tag:
16.                     tag_dict[x].append(data)
17.                     del datas[i - padding]
18.                     padding += 1
19.                     break
20.     print(len(datas))
21.     counter = 0;
22.     for tag in tag_list:
23.         counter += len(tag_dict[tag])
24.         tag_train, tag_dev, tag_test = np.spli
    t(tag_dict[tag], [int(len(tag_dict[tag]) * 0.7
    0), int(len(tag_dict[tag]) * 0.85)])
25.
26.         for x in tag_train:
27.             train.append(x)
```

```
28.         for x in tag_dev:
29.             dev.append(x)
30.         for x in tag_test:
31.             test.append(x)
32.         print("total " + tag + " " + str(len(t
    ag_dict[tag])))
33.         print("train " + tag + ": " + str(len(
    tag_train)))
34.         print("dev " + tag + ": " + str(len(ta
    g_dev)))
35.         print("test " + tag + ": " + str(len(t
    ag_test)))
36.     print("total sentences: " + str(counter))

37.     return train, dev, test
```

Kode 5.7 Potongan Kode Pendistribusian Label

Kode diatas berfungsi untuk mengatur agar semua label terdistribusi ke setiap dataset dengan rasio yang sama. Pada baris dua terdapat *array* yang berisi label-label yang digunakan yang telah diurutkan berdasarkan jumlah dari paling sedikit ke label dengan jumlah paling banyak. Kemudian pada baris sembilan dilakukan perulangan untuk menghilangkan tambahan dari skema IOB2 pada label dari setiap *token* dan setiap data percakapan dimasukkan ke objek *dictionary* sesuai dengan label masing-masing. Setelah itu pada fungsi perulangan di baris 22 dilakukan untuk membagi setiap label ke tiga *dataset* berdasarkan dari objek *dictionary* yang sebelumnya dibuat. Rasio yang digunakan sesuai dengan rasio *dataset* yaitu 70:15:15. Hal ini penting agar tidak ada label yang tidak dimiliki oleh satu atau lebih *dataset* atau label yang terkonsentrasi pada *dataset* tertentu sehingga mempengaruhi kemampuan model untuk mengenali label tersebut. Pembagian dilakukan dengan fungsi *split()*.

```
1.  def write_file(file_out, data):
2.      for t in data:
3.          for i in range(len(t.tokens)):
4.              token = t.tokens[i]
5.              tag = t.tags[i]
```

```
6.                 file_out.writelines(token + " " +
    tag)
7.                 file_out.write("\n")
8.             file_out.write("\n")
9.         file_out.close()
```

Kode 5.8 Potongan Kode Untuk Menyimpan Hasil Distribusi Label

Kode yang terdapat pada Kode 5.8 berfungsi untuk menulis hasil dari proses pembuatan dataset kedalam *file*. Pada baris tiga *token* dan label dari *token* tersebut diambil untuk dituliskan dalam satu baris dan untuk *token* selanjutnya akan ditulis pada baris berikutnya. *File* yang dihasilkan fungsi ini kemudian akan digunakan sebagai masukan dalam pembuatan model.

```
1.  def build_data():
2.      np.random.seed(42)
3.
4.      db = DbInstance()
5.      datas, _, _ = db.get_data('merger')
6.      np.random.shuffle(datas)
7.      train, dev, test = distribute_tag(datas)
8.
9.      print("Train data count: " + str(len(train
    )))
10.     print("Dev data count: " + str(len(dev)))
11.     print("Test data count: " + str(len(test))
    )
12.
13.     f_train = open("train.bio", "w")
14.     f_dev = open("dev.bio", "w")
15.     f_test = open("test.bio", "w")
16.
17.     write_file(f_train, train)
18.     write_file(f_dev, dev)
19.     write_file(f_test, test)
```

Kode 5.9 Potongan Kode Untuk Menjalankan Pembagian dan Penyimpanan

Proses pembagian dataset dan penulisan *file* keluaran dijalankan dalam fungsi *build_data()*. Data terlebih dahulu diambil dari *database* pada baris lima dan diacak pada baris

enam untuk kemudian dibagi dengan fungsi *distribute_tag()* pada baris tujuh. Hasil dari pembagian data disimpan dalam variabel *f_train, f_dev* dan *f_test* kemudian masing-masing ditulis kedalam *file* keluaran dengan fungsi *write_file()* pada baris 17, 18 dan 19.

5.4.3 Pembuatan *Training* dan *Testing* Model BiLSTM-CRF

Pembuatan model BiLSTM-CRF menggunakan *library* bernama *pytorch*. Langkah pertama adalah membuat implementasi dari masing-masing model terlebih dahulu.

```
1.  class BiLSTM_CRF(nn.Module):
2.      def __init__(self, embedding_dim, hidden_d
    im, vocab_size, tag_to_ix, pretrained):
3.          super(BiLSTM_CRF, self).__init__()
4.          self.embedding_dim = embedding_dim
5.          self.hidden_dim = hidden_dim
6.          self.vocab_size = vocab_size
7.          self.tag_to_ix = tag_to_ix
8.          self.tagset_size = len(tag_to_ix)
9.
10.         self.START_TAG = "<START>"
11.         self.STOP_TAG = "<STOP>"
12.
13.         weights = torch.FloatTensor(pretrained
    )
14.         weights = torch.FloatTensor(pretrained
    )
15.         self.word_embeds = nn.Embedding.from_p
    retrained(weights)
16.         self.lstm = nn.LSTM(embedding_dim, hid
    den_dim // 2,
17.                             num_layers=1, bidi
    rectional=True)
18.
19.         self.hidden2tag = nn.Linear(hidden_dim
    , self.tagset_size)
20.
21.         self.transitions = nn.Parameter(
22.             torch.randn(self.tagset_size, self
    .tagset_size)).to(device)
23.
24.         self.transitions.data[tag_to_ix[self.S
    TART_TAG], :] = -10000
```

```
25.         self.transitions.data[:, tag_to_ix[sel
    f.STOP_TAG]] = -10000
26.
27.         self.hidden = self.init_hidden()
28. . . .
```

Kode 5.10 Potongan Kode Implementasi BiLSTM-CRF

Potongan kode pada Kode 5.10 merupakan *class* implementasi dari model BiLSTM-CRF. *Class* ini membutuhkan beberapa parameter seperti dimensi vektor kata, jumlah *feature* dalam *hidden state*, jumlah *vocabulary* kata, pemetaan nilai label, dan model *word embedding*. Pada implementasi BiLSTM-CRF ini yang bertindak sebagai model sesungguhnya adalah CRF dengan BiLSTM menyediakan *feature* yang menjadi masukan untuk CRF. Model *word embedding* yang sebelumnya dibuat digunakan untuk membuat objek *embedding* dengan bobot awal dari setiap kata yang diambil pada baris 13. Parameter *word embedding* pada model BiLSTM-CRF dibuat dengan fungsi *nn.Embedding.from_pretrained()* pada baris 15 dengan masukan bobot yang sebelumnya telah diambil.

```
1.  . . .
2.  def _get_lstm_features(self, sentence):
3.      self.hidden = self.init_hidden()
4.      embeds = self.word_embeds(sentence).view(l
    en(sentence), 1, -1)
5.      lstm_out, self.hidden = self.lstm(embeds,
    self.hidden)
6.      lstm_out = lstm_out.view(len(sentence), se
    lf.hidden_dim)
7.      lstm_feats = self.hidden2tag(lstm_out)
8.      return lstm_feats
9.
10. def neg_log_likelihood(self, sentence, tags):

11.     feats = self._get_lstm_features(sentence)

12.     forward_score = self._forward(feats)
13.     gold_score = self._score_sentence(feats, t
    ags)
14.     return forward_score - gold_score
15.
```

```
16. def forward(self, sentence):
17.     lstm_feats = self._get_lstm_features(sente
    nce)
18.     score, tag_seq = self._viterbi_decode(lstm
    _feats)
19.     return score, tag_seq
```

Kode 5.11 Potongan Kode Implementasi BiLSTM-CRF

Kode 5.11 merupakan potongan kode dari kelas implementasi model BiLSTM-CRF. Untuk setiap iterasi model akan melakukan *feed forward* dengan fungsi *forward()*. Pada BiLSTM-CRF nilai *feed forward* yang dihasilkan pada fungsi *_get_lstm_features()* di baris 17 tidak langsung dikeluarkan akan tetapi dijadikan masukan dalam model CRF sebagai *emission score* untuk kemudian dihitung dengan algoritma *viterbi* pada baris 18 yang menghasilkan keluaran berupa *sequence* terbaik yang dihasilkan dan nilai dari *sequence* tersebut.

```
1.  def _viterbi_decode(self, feats):
2.          backpointers = []
3.          init_vvars = torch.full((1, self.tagse
    t_size), -10000.).to(device)
4.          init_vvars[0][self.tag_to_ix[self.STAR
    T_TAG]] = 0
5.          forward_var = init_vvars
6.          for feat in feats:
7.              bptrs_t = []
8.              viterbivars_t = []
9.              for next_tag in range(self.tagset_
    size):
10.                 next_tag_var = forward_var + s
    elf.transitions[next_tag]
11.                 best_tag_id = argmax(next_tag_
    var)
12.                 bptrs_t.append(best_tag_id)
13.                 viterbivars_t.append(next_tag_
    var[0][best_tag_id].view(1))
14.             forward_var = (torch.cat(viterbiva
    rs_t) + feat).view(1, -1)
15.             backpointers.append(bptrs_t)
16.         terminal_var = forward_var + self.tran
    sitions[self.tag_to_ix[self.STOP_TAG]]
```

```
17.         best_tag_id = argmax(terminal_var)
18.         path_score = terminal_var[0][best_tag_
    id]
19.         best_path = [best_tag_id]
20.         for bptrs_t in reversed(backpointers):

21.             best_tag_id = bptrs_t[best_tag_id]

22.             best_path.append(best_tag_id)
23.         best_path.reverse()
24.         return path_score, best_path
```

Kode 5.12 Potongan Kode Implementasi Algoritma Viterbi

Potongan kode pada Kode 5.12 merupakan implementasi dari algoritma *viterbi*. Berdasarkan dari *feature* yang menjadi parameter, dilakukan fungsi perulangan untuk menghitung nilai kemungkinan dari setiap label untuk menjadi label yang tepat. Setelah itu diambil label dengan nilai terbaik untuk ditambahkan ke *backpointers* pada baris 15. Setelah itu dilakukan penyusunan urutan label berdasarkan *backpointers* yang terlebih dahulu dibalik urutannya pada baris 20. Kemudian *sequence* yang dihasilkan dibalik agar menjadi seperti semula pada baris 23.

Hasil perhitungan dari algoritma *viterbi* digunakan untuk menghitung *loss* dan *gradient* kemudian dilakukan pembaruan pada parameter model.

```
1.  def train(net, _optimizer, _lr):
2.  . . .
3.          training_data, training_tokens, traini
    ng_tags = read_bio("train.bio")
4.          dev_data, dev_tokens, dev_tags = read_
    bio("dev.bio")
5.          test_data, test_tokens, test_tags = re
    ad_bio("test.bio")
6.  . . .
7.          pretrained, emb_dim = build_pretrain_e
    mbedding("data/embed/embedding.bin", words)
8.  . . .
```

Kode 5.13 Potongan Kode Untuk Memuat Data dan Word Embedding

Kode 5.13 merupakan potongan kode dari fungsi *train()* yang digunakan untuk menjalankan proses pelatihan. Pertama-tama yang dilakukan adalah memuat dataset yang dilakukan pada baris tiga, empat dan lima dengan fungsi *read_bio()* dan memuat model *word embedding* pada baris tujuh.

```
1.  for epoch in progressbar.progressbar(range(num
    _epoch)):
2.              for data in training_data:
3.                  model.zero_grad()
4.                  sentence_in = prepare_sequence
    (data.tokens, word_to_ix).to(device)
5.                  targets = prepare_sequence(dat
    a.tags, tag_to_ix).to(device)
6.                  tag_scores, _ = model(sentence
    _in)
7.                  if net == "bilstmcrf":
8.                      loss = model.neg_log_likel
    ihood(sentence_in, targets)
9.                  else:
10.                     loss = loss_function(tag_s
    cores, targets.view(-1))
11.                 loss.backward()
12.                 optimizer.step()
13.             print("epoch " + str(epoch))
14.             _, _, train_accuracy = getAccuracy
    (model, training_data, word_to_ix, tag_to_ix,
    crf, net_dir=net_dir, data="train")
15.             acc_train.append(train_accuracy)
16.             loss_train.append(loss.cpu().numpy
    )
17.             print("Training accuracy is " + st
    r(train_accuracy))
18.             print("Training loss: " + str(loss
    .data))
19.             mp, mr, mf, dev_accuracy = getAccu
    racy(model, dev_data, word_to_ix, tag_to_ix, c
    rf, data="dev", net_dir=net_dir, to_tag=ix_to_
    tag)
20.             acc_dev.append(dev_accuracy)
21.             p_dev.append(mp)
22.             r_dev.append(mr)
23.             f_dev.append(mf)
24.             is_best = False
```

```
            print("Dev accuracy is " + str(dev
_accuracy))
            print("precision:" + str(mp) + " r
ecal:" + str(mr) + " f:" + str(mf))
            if mf > best_dev:
                print("Exceed latest fmeasure
by " + str(mf - best_dev))
                best_model = epoch
                torch.save(model.state_dict(),
 net_dir + "/" + str(best_model) + "-
best.pt")
                best_dev = mf
                is_best = True
            list_best_dev.append(is_best)
            print("")
        end = time.time()

        df = pd.DataFrame({"acc": acc_train,"l
oss": loss_train})
        df.to_csv(net_dir + "/training_perform
ance.csv", header=True)
        df = pd.DataFrame({"acc": acc_dev,"pre
cision": p_dev,"recall": r_dev,"f-
measure": f_dev, "best":list_best_dev})
        df.to_csv(net_dir + "/dev_performance.
csv", header=True)

        print("The time it takes to execute "
+ str(num_epoch) + " epochs is " + str(end - s
tart) + " seconds.")
        str_duration = "duration: " + str(end
- start) + "s"
        writefile(net_dir,"duration.txt", str_
duration)
        print("Test the model ...")
        num_tokens = 0
        num_correct = 0
        num_tokens_exclude_o = 0
        num_correct_exclude_o = 0
        preds = []
        with torch.no_grad():
            if net == "bilstmcrf":
                model = BiLSTM_CRF(EMBEDDING_D
IM, HIDDEN_DIM, len(word_to_ix), tag_to_ix, pr
etrained).to(device)
```

```
54.             elif net == "bilstm":
55.                 model = BiLSTM(EMBEDDING_DIM,
    HIDDEN_DIM, len(word_to_ix), len(labels), pret
    rained).to(device)
56.             elif net == "lstm":
57.                 model = LSTM(EMBEDDING_DIM, HI
    DDEN_DIM, len(word_to_ix), len(labels), pretra
    ined).to(device)
58.             model.load_state_dict(torch.load(n
    et_dir + "/" + str(best_model) + "-
    best.pt"))
59.             mp, mr, mf, test_accuracy = getAcc
    uracy(model, test_data, word_to_ix, tag_to_ix,
     crf, data="test", net_dir=net_dir, to_tag=ix_
    to_tag)
60.             df = pd.DataFrame({"acc": [test_ac
    curacy],"precision": [mp],"recall": [mr],"f-
    measure": [mf]})
61.             df.to_csv(net_dir + "/test_perform
    ance.csv", header=True)
62.             print("Test accuracy is " + str(te
    st_accuracy))
63.             print("precision:" + str(mp) + " r
    ecal:" + str(mr) + " f:" + str(mf))
```

Kode 5.14 Potongan Kode Untuk Pelatihan Model

Kode 5.14 merupakan potongan kode dari fungsi *train()*. Proses *training* pada implementasi BiLSTM-CRF dilakukan dengan iterasi sebanyak 50. Pada baris dua dilakukan iterasi untuk setiap kalimat pada *dataset train*. Kalimat tersebut dipetakan terlebih dahulu menjadi angka sesuai dengan fungsi *prepare_sequence()* pada baris empat. Kalimat yang sudah dipetakan kemudian dijadikan masukan pada model untuk menjalankan *feed forward* pada baris enam. Hasil dari *feed forward* pada baris enam dijadikan masukan pada fungsi *loss* pada baris delapan yang kemudian digunakan untuk memperbarui parameter dari model dengan fungsi s*tep()* pada objek *optimizer* di baris 12.

Pada setiap iterasi, model diuji pada *dataset dev* untuk mengetahui nilai *f-measure* dari model. Nilai *f-measure* didapatkan melalui fungsi *getAccuracy()* pada baris 19. Nilai *f-*

measure ini disimpan pada variabel *f_dev* dan setiap kali model mendapat nilai *f-measure* lebih tinggi dari model sebelumnya, model pada iterasi tersebut akan disimpan pada variabel *best_dev*. Metode ini disebut *early stopping* dimana metode ini bertujuan untuk menghindari *overfitting* dalam proses *training*. Setelah seluruh iterasi selesai model dengan nilai *f-measure* tertinggi pada *dataset dev* diuji pada *dataset test*. Model terbaik terlebih dahulu dimuat dengan fungsi *load_state_dict()* pada baris 58 untuk kemudian diuji dan dihitung nilai *f-measure* dari model tersebut dengan fungsi *getAccuracy()* pada baris 59. Dari pengujian tersebut dihasilkan nilai dari akurasi, presisi, *recall* dan *f-measure* yang kemudian disimpan pada objek *DataFrame* untuk kemudian disimpan dalam berkas *csv* dengan fungsi *to_csv()* pada baris 61.

Halaman ini sengaja dikosongkan.

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 Ekstraksi Data Percakapan

Data yang akan digunakan dari *dataset* dialog hanya pada bagian pertanyaan dari pembeli sehingga perlu diproses terlebih dahulu. Dari proses ekstraksi data yang didapatkan adalah sebanyak 2.454 ujuran pembeli. Seluruh data diubah menjadi format seragam untuk kemudian disimpan dalam *file* csv. Beberapa hasil dari ekstraksi dapat dilihat pada Tabel 6.1.

Tabel 6.1 Hasil Ekstraksi Data Percakapan

No	Kalimat
1	Kirim Go-Send bisa ya?
2	Gan, iphone XS nya ready nggak?
3	Kira-kira kapan ready nya?
4	Vivo v7 ready stok warna apa saja gan?
5	pake gosend bisa gan?

6.2 Hasil Pelabelan Data Percakapan

Proses pelabelan dilakukan dengan program yang dibuat pada penelitian sebelumnya. Pelabelan dilakukan pada setiap *token* atau kata pada kalimat-kalimat yang ada pada dataset. Contoh dari kalimat yang telah diberi label dapat dilihat pada Tabel 6.2.

Tabel 6.2 Contoh Kalimat yang Telah Diberi Label

Token	Nano	sim	kan	?
Label	B-ITEM-FITUR	I-ITEM-FITUR	O	O

6.3 Hasil Pembuatan Model BiLSTM-CRF dan Model *Neural Network* Lain

6.3.1 Hasil Pembuatan *Dataset* untuk Model BiLSTM-CRF dan Model *Neural Network* Lain

Setelah data percakapan diekstrak dan diberi label, proses selanjutnya adalah membagi seluruh data menjadi tiga *dataset* yang akan digunakan untuk proses *training* pada model yang dibuat. Seluruh data dibagi dengan rasio 70:15:15 dengan cara terlebih dahulu dibagi pada tingkat label. Rincian dari distribusi label pada setiap dataset dapat dilihat pada Tabel 6.3.

Tabel 6.3 Distribusi Label pada Setiap *Dataset*

Label	***Dataset***		
	train	**dev**	**test**
JUMLAH-ORDER	7	1	2
ITEM-PERLENGKAPAN	9	2	3
MODEL-BANDING	13	3	3
ITEM-BONUS	18	4	5
TUJUAN-PENGIRIMAN	33	7	8
WAKTU-PENGIRIMAN	20	4	5
ITEM-SPEC	57	12	13
ITEM-FITUR	47	10	11
ITEM-SUBMODEL	51	11	11

KURIR-PENGIRIMAN	52	11	12
MERK	115	25	25
WARNA	115	25	25
MODEL	92	20	20
O	1082	232	233

Berdasarkan Tabel 6.3 dapat dilihat bahwa terjadi ketidak seimbangan data pada setiap *dataset* dimana label O berjumlah jauh lebih banyak dibanding label lain. Hal ini terjadi karena kalimat-kalimat pada *dataset* memiliki lebih sedikit *token* yang merupakan bagian dari entitas dibanding *token* yang bukan merupakan bagian dari entitas seperti yang bisa dilihat pada Tabel 6.4.

Tabel 6.4 Contoh Kalimat pada Dataset

Token	pake	gosend	bisa	gan	?
Label	O	B-KURIR-PENGIRIMAN	O	O	O

Selain itu ada juga kalimat yang tidak memiliki entitas sama sekali seperti contoh yang dapat dilihat pada Tabel 6.5.

Tabel 6.5 Contoh Kalimat yang Tidak Memiliki Entitas

Token	Apakah	ada	toko	fisiknya	?
Label	O	O	O	O	O

6.3.2 Hasil Pembuatan dan Pengujian Implementasi Model BiLSTM-CRF

Sebelum menguji hasil implementasi BiLSTM-CRF, terlebih dahulu dilakukan konfigurasi model. Pengujian juga akan dilakukan pada model yang dihasilkan NCRF++ dengan

konfigurasi yang sama. Rincian terkait konfigurasi yang digunakan dapat dilihat pada Tabel 6.4.

Tabel 6.6 Parameter Pengujian Performa Implementasi BiLSTM-CRF

Parameter	Nilai
Epoch	50
Batch	1
Word Embedding	Word2Vec Skip-gram
Hidden size	300
Learning rate	0,00015

Pengujian dilakukan pada komputer 1, berikut adalah hasil dari pengujian performa model.

Tabel 6.7 Hasil Pengujian Implementasi BiLSTM-CRF

Model	Akurasi	Presisi	Recall	F-measure
BiLSTM-CRF Implementasi	0.9640	0.7814	0.6970	0.7368
BiLSTM-CRF NCRF++	0.9640	0.7805	0.6919	0.7335

Dari Tabel 6.5 dapat diketahui bahwa performa dari model implementasi BiLSTM-CRF tidak jauh berbeda dengan model yang dibuat dengan *toolkit* NCRF++. Hasil implementasi BiLSTM-CRF bahkan sedikit lebih unggul dibandingkan dengan model yang dihasilkan NCRF++. Meskipun begitu, model implementasi BiLSTM-CRF memiliki waktu pelatihan yang jauh lebih lama dibandingkan model yang dihasilkan NCRF++ sebagaimana tertera pada Tabel 6.5.

Tabel 6.8 Durasi Proses Pelatihan Implementasi BiLSTM-CRF

Model	Durasi (detik)
BiLSTM-CRF Implementasi	21.410,78
BiLSTM-CRF NCRF++	940.94

Dari Tabel 6.6 dapat diketahui bahwa durasi pelatihan model implementasi BiLSTM-CRF lebih dari 20 kali lipat dibandingkan durasi pelatihan model BiLSTM-CRF yang dibuat NCRF++. Hal ini wajar terjadi karena algoritma pada NCRF++ sudah sangat teroptimalisasi sehingga jauh lebih efisien. Dikarenakan waktu pelatihan yang jauh berbeda akhirnya diputuskan model yang digunakan untuk analisa perbandingan performa arsitektur BiLSTM-CRF dengan arsitektur *neural network* yang lain dibuat dengan menggunakan NCRF++.

6.3.3 Hasil Analisis dan Pengujian Performa Model BiLSTM-CRF dan Model *Neural Network* Lain

Pada tahap ini dilakukan percobaan untuk mengetahui performa dari BiLSTM-CRF dan model *neural network* lain. Model-model tersebut dibuat dengan menggunakan *toolkit* NCRF++ pada komputer 2 dengan jumlah iterasi sebanyak 100 dengan dua kali percobaan. Percobaan dilakukan dengan memvariasikan nilai *learning rate* dan algoritma *optimizer* menghasilkan sembilan model untuk setiap arsitektur. Sehingga untuk tiga arsitektur dihasilkan 27 model.

Tabel 6.9 Konfigurasi Pengujian Model BiLSTM-CRF dan Model Neural Network Lain

Parameter	Nilai
Epoch	100
Mini batch	30
Word Embedding	Word2Vec Skip-gram
Hidden size	200
Layer number	2
Learning rate	0.015, 0.0015, 0.0005
Optimizer	Adam, SGD, SGD + *momentum*
Dropout	0.5
Learning rate decay	0.001 (hanya pada *optimizer* yang menggunakan SGD)
L2-regularization	1^{-8}

Pelatihan model dilakukan pada arsitektur BiLSTM-CRF, BiLSTM dan LSTM dengan menggunakan konfigurasi pada Tabel 6.7. Durasi proses pelatihan pada masing-masing arsitektur dapat dilihat pada Tabel 6.8 sedangkan untuk hasil

pengujian performa dapat dilihat pada Tabel 6.9, Tabel 6.10 dan Tabel 6.11.

Tabel 6.10 Rata-Rata Durasi Proses *Training* Setiap Model

Optimizer	Model	Waktu (detik)		
		lr = 0.0015	**lr = 0.015**	**lr = 0.0005**
SGD + Momentum	BiLSTM-CRF	842.8585	842.2702	831.6492
	BiLSTM	339.2935	344.5631	344.3682
	LSTM	328.1195	320.5289	318.6326
SGD	BiLSTM-CRF	853.8252	834.7578	857.4749
	BiLSTM	338.0882	325.5675	333.3821
	LSTM	317.5291	325.4289	316.0327
Adam	BiLSTM-CRF	893.1665	842.2214	868.5276
	BiLSTM	362.6541	354.9540	362.2520
	LSTM	341.8115	346.4785	348.7213

Tabel 6.11 Hasil Pengujian Dengan Learning Rate 0.0015

lr = 0.0015									
Optimizer	**Model**	**accuracy**		**precision**		**recall**		**f-score**	
		rata-rata	**maksimal**	**rata-rata**	**maksimal**	**rata-rata**	**maksimal**	**rata-rata**	**maksimal**
SGD + Momentum	**BiLSTM-CRF**	0.9679	0.9690	0.7742	0.7892	0.7865	0.7892	0.7802	0.7892
	BiLSTM	0.9678	0.9702	0.7637	0.7659	0.7946	0.8108	0.7787	0.7853
	LSTM	0.9606	0.9610	0.7426	0.7433	0.7486	0.7514	0.7456	0.7473
SGD	**BiLSTM-CRF**	0.9599	0.9629	0.6863	0.7092	0.7432	0.7514	0.7135	0.7297
	BiLSTM	0.9629	0.9632	0.6925	0.6931	0.7487	0.7568	0.7195	0.7235
	LSTM	0.9581	0.9595	0.6950	0.7041	0.7270	0.7459	0.7106	0.7244
Adam	**BiLSTM-CRF**	0.9594	0.9599	0.6883	0.6990	0.7568	0.7730	0.7207	0.7222
	BiLSTM	0.9642	0.9662	0.7379	0.7457	0.7216	0.7459	0.7293	0.7380
	LSTM	0.9558	0.9573	0.6984	0.7102	0.6811	0.6865	0.6895	0.6925

Tabel 6.12 Hasil Pengujian Dengan Learning Rate 0.015

lr = 0.015									
Optimizer	**Model**	**accuracy**		**precision**		**recall**		**f-score**	
		rata-rata	**maksimal**	**rata-rata**	**maksimal**	**rata-rata**	**maksimal**	**rata-rata**	**maksimal**
SGD + Momentum	**BiLSTM-CRF**	0.9524	0.9570	0.6934	0.7342	0.6486	0.6703	0.6689	0.6764
	BiLSTM	0.9537	0.9581	0.6778	0.7127	0.6649	0.6973	0.6713	0.7049
	LSTM	0.9511	0.9515	0.6762	0.6897	0.6324	0.6486	0.6536	0.6685
SGD	**BiLSTM-CRF**	0.9675	0.9691	0.7830	0.8043	0.7973	0.8000	0.7900	0.8022
	BiLSTM	0.9671	0.9680	0.7700	0.7833	0.7676	0.7730	0.7687	0.7726
	LSTM	0.9515	0.9614	0.6986	0.7540	0.6784	0.7622	0.6880	0.7581
Adam	**BiLSTM-CRF**	0.9472	0.9478	0.6523	0.6707	0.6108	0.6270	0.6304	0.6304
	BiLSTM	0.9432	0.9452	0.6077	0.6369	0.6270	0.6378	0.6165	0.6264
	LSTM	0.9435	0.9456	0.6753	0.7073	0.6108	0.6270	0.6414	0.6648

Tabel 6.13 Hasil Pengujian Dengan Learning Rate 0.0005

lr = 0.0005									
Optimizer	**Model**	**accuracy**		**precision**		**recall**		**f-score**	
		rata-rata	**maksimal**	**rata-rata**	**maksimal**	**rata-rata**	**maksimal**	**rata-rata**	**maksimal**
SGD + Momentum	**BiLSTM-CRF**	0.9658	0.9662	0.7584	0.7742	0.7946	0.8108	0.7757	0.7763
	BiLSTM	0.9647	0.9665	0.7598	0.7720	0.8108	0.8162	0.7844	0.7884
	LSTM	0.9608	0.9632	0.7368	0.7435	0.7568	0.7676	0.7466	0.7553
SGD	**BiLSTM-CRF**	0.9529	0.9537	0.6181	0.6277	0.6297	0.6378	0.6238	0.6327
	BiLSTM	0.9542	0.9548	0.6030	0.6080	0.6405	0.6541	0.6212	0.6302
	LSTM	0.9456	0.9474	0.6110	0.6243	0.5649	0.5838	0.5870	0.6034
Adam	**BiLSTM-CRF**	0.9647	0.9669	0.7488	0.7542	0.7486	0.7676	0.7485	0.7553
	BiLSTM	0.9645	0.9665	0.7392	0.7568	0.7568	0.7568	0.7478	0.7568
	LSTM	0.9570	0.9588	0.6841	0.6931	0.7378	0.7568	0.7099	0.7235

Dapat dilihat dari tabel 6.8 bahwa secara umum model dengan durasi pelatihan paling cepat adalah LSTM. Hal ini disebabkan oleh kompleksitas arsitektur dimana LSTM merupakan arsitektur paling sederhana dibanding BiLSTM dan BiLSTM-CRF. BiLSTM merupakan LSTM yang dua arah sedang BiLSTM-CRF merupakan kombinasi antara BiLSTM dan CRF dimana BiLSTM bertindak sebagai penyedia *feature* untuk CRF.

Parameter juga ikut mempengaruhi durasi pelatihan. Secara umum model dengan *optimizer* Adam memiliki waktu pelatihan yang lebih lama dibanding model yang menggunakan *optimizer* lain.

Dari hasil pengujian pada Tabel 6.9, Tabel 6.10 dan Tabel 6.11, analisis dilakukan dengan membandingkan hasil pengujian pada setiap konfigurasi. Ukuran yang digunakan adalah *f-measure*. Hasil dari analisis tersebut adalah konfigurasi terbaik bagi setiap arsitektur sebegaimana tercantum dalam Tabel 6.12.

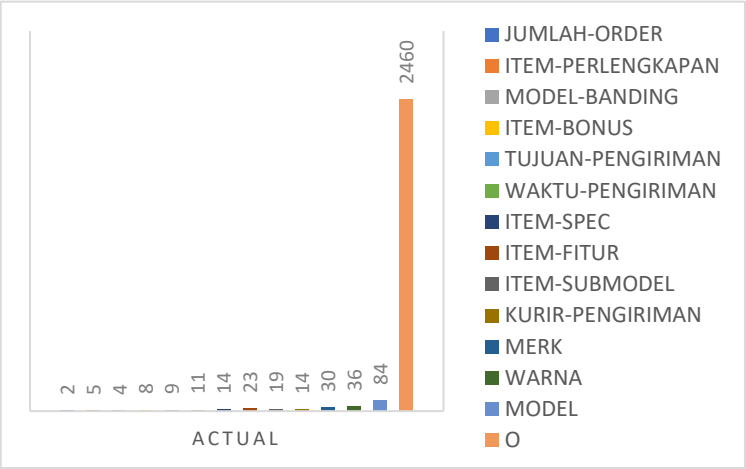
Tabel 6.14 Konfigurasi Terbaik Setiap Arsitektur

Arsitektur	**Optimizer**	**Learning Rate**	**F-Measure**
BiLSTM-CRF	SGD	0.015	0.802
BiLSTM	SGD + Momentum	0.0005	0.788
LSTM	SGD	0.015	0.758

Pada Tabel 6.12 dapat dilihat bahwa diantara hasil terbaik dari setiap arsitektur BiLSTM-CRF memiliki performa terbaik dengan nilai *f-measure* sebesar 0.802 atau 80.2%.

Dari hasil pengujian dapat dilihat jika performa model dengan *optimizer* SGD + momentum dan Adam cenderung menurun saat menggunakan *learning rate* yang lebih tinggi.

Selanjutnya dilakukan analisis performa untuk setiap label. Analisis ini bertujuan untuk mengetahui bagaimana performa model dalam mengenali setiap label. Ukuran performa yang digunakan adalah *f-measure*. Pada penelitian ini akurasi kurang tepat apabila dijadikan ukuran performa karena pada *dataset* yang digunakan terjada ketidakseimbangan label sebagaimana bisa dilihat pada Gambar 6.1 dimana jumlah label O jauh melampaui jumlah label-label yang lain.



Gambar 6.1 Visualisasi Jumlah Sebenarnya Dari Setiap Label

Tabel 6.15 Hasil Pengujian Performa Label Model BiLSTM-CRF

Label	presisi	recall	f-measure
JUMLAH-ORDER	1.00	1.00	1.00
ITEM-PERLENGKAPAN	1.00	0.20	0.33
MODEL-BANDING	1.00	1.00	1.00
ITEM-BONUS	1.00	1.00	1.00
TUJUAN-PENGIRIMAN	0.64	0.78	0.70
WAKTU-PENGIRIMAN	0.75	0.55	0.63
ITEM-SPEC	0.47	0.57	0.52
ITEM-FITUR	0.79	0.48	0.59

ITEM-SUBMODEL	0.68	0.89	0.77
KURIR-PENGIRIMAN	0.81	0.93	0.87
MERK	0.92	0.73	0.81
WARNA	0.86	1.00	0.92
MODEL	0.91	0.93	0.92
O	0.98	0.98	0.98

Berdasarkan hasil pengujian model BiLSTM-CRF pada tabel 6.13 kelas JUMLAH-ORDER, MODEL-BANDING dan ITEM-BONUS merupakan label dengan performa terbaik. Sedangkan kelas ITEM-PERLENGKAPAN merupakan label dengan performa paling buruk.

Tabel 6.16 Hasil Pengujian Performa Label Model BiLSTM

Label	**presisi**	**recall**	**f-measure**
JUMLAH-ORDER	1.00	1.00	1.00
ITEM-PERLENGKAPAN	1.00	0.20	0.33
MODEL-BANDING	1.00	1.00	1.00
ITEM-BONUS	0.70	0.88	0.78
TUJUAN-PENGIRIMAN	0.89	0.89	0.89
WAKTU-PENGIRIMAN	0.47	0.64	0.54
ITEM-SPEC	0.56	0.71	0.63
ITEM-FITUR	0.63	0.52	0.57
ITEM-SUBMODEL	0.58	0.79	0.67
KURIR-PENGIRIMAN	0.81	0.93	0.87
MERK	0.92	0.77	0.84
WARNA	0.88	1.00	0.94
MODEL	0.91	0.94	0.92
O	0.99	0.98	0.98

Berdasarkan hasil pengujian model BiLSTM pada tabel 6.14 kelas JUMLAH-ORDER dan MODEL-BANDING merupakan label dengan performa terbaik. Sedangkan kelas ITEM-

PERLENGKAPAN merupakan label dengan performa paling buruk.

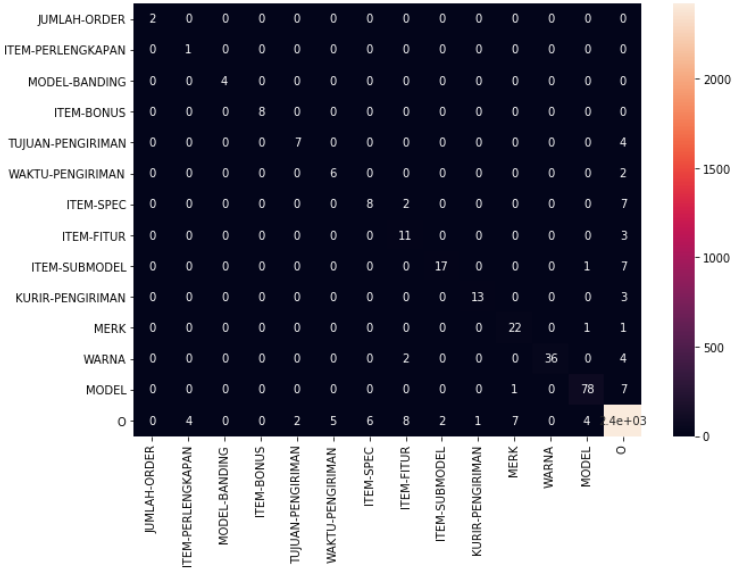
Tabel 6.17 Hasil Pengujian Performa Label Model LSTM

Label	presisi	recall	f-measure
JUMLAH-ORDER	1.00	1.00	1.00
ITEM-PERLENGKAPAN	1.00	0.20	0.33
MODEL-BANDING	0.75	0.75	0.75
ITEM-BONUS	1.00	0.75	0.86
TUJUAN-PENGIRIMAN	0.75	0.67	0.71
WAKTU-PENGIRIMAN	0.50	0.45	0.48
ITEM-SPEC	0.43	0.43	0.43
ITEM-FITUR	0.80	0.52	0.63
ITEM-SUBMODEL	0.56	0.53	0.54
KURIR-PENGIRIMAN	0.81	0.93	0.87
MERK	0.84	0.90	0.87
WARNA	0.83	0.97	0.90
MODEL	0.81	0.90	0.85
O	0.98	0.98	0.98

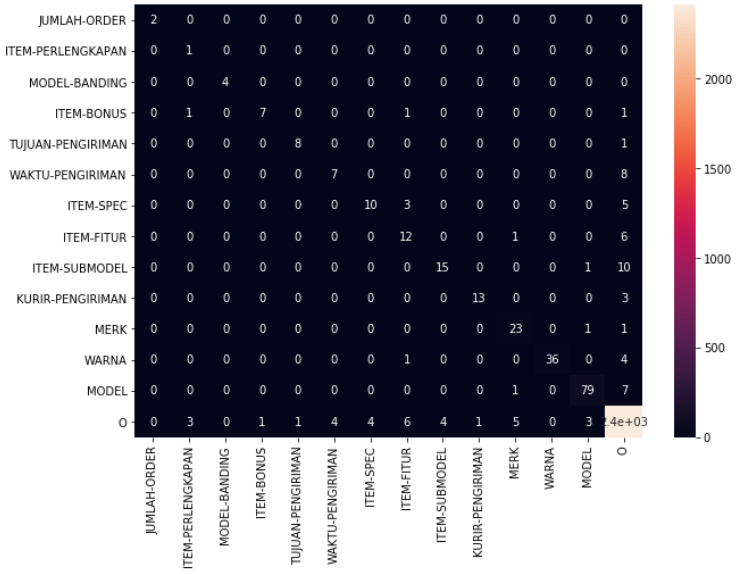
Berdasarkan hasil pengujian model LSTM pada tabel 6.15 kelas JUMLAH-ORDER, MODEL-BANDING merupakan label dengan performa terbaik. Sedangkan kelas ITEM-PERLENGKAPAN merupakan label dengan performa paling buruk.

Label ITEM-PERLENGKAPAN menjadi label dengan performa paling buruk pada ketiga model yang diuji. Ketika dilihat dari *confussion matrix* dari semua model, ternyata model sering kali salah menebak label ITEM-PERLENGKAPAN dengan label O. Hal ini menunjukkan bahwa model belum bisa mengenali label ITEM-PERLENGKAPAN dengan baik

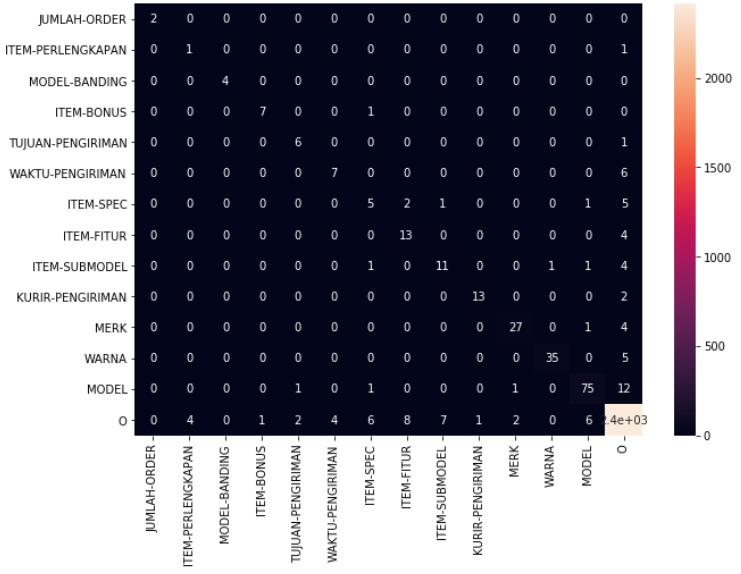
sehingga masih condong ke label O yang jumlahnya paling banyak dalam *dataset*. *Confussion matrix* dari setiap arsitektur dapat dilihat pada Gambar 6.2, Gambar 6.3 dan Gambar 6.4.



Gambar 6.2 Confussion Matrix Model BiLSTM-CRF

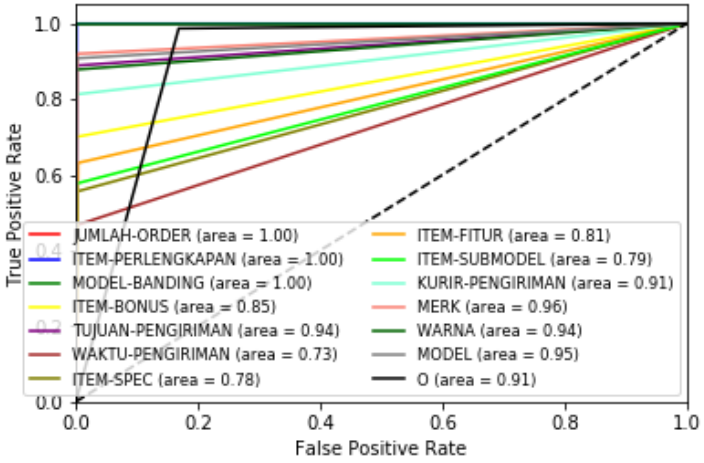


Gambar 6.3 Confussion Matrix Model BiLSTM

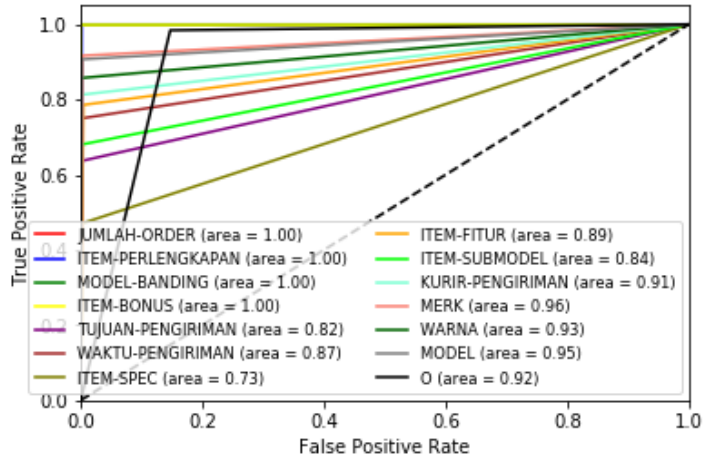


Gambar 6.4 Confussion Matrix Model LSTM

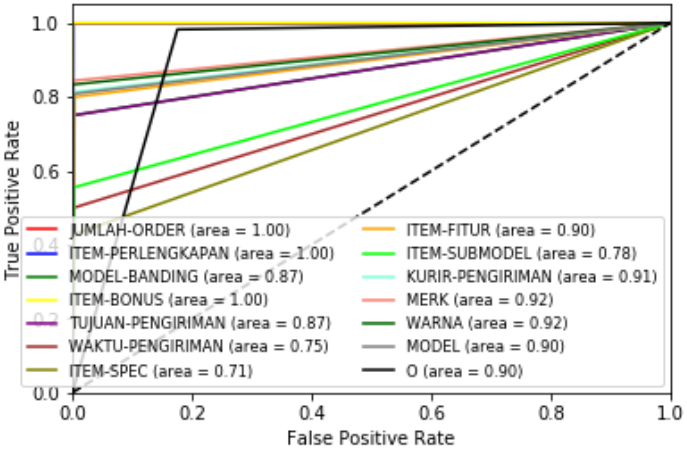
Untuk melihat performa model-model yang dihasilkan lebih jauh, dibuat juga kurva ROC untuk setiap model untuk selanjutnya dihitung nilai AUC dari setiap model yang dapat dilihat pada Gambar 6.5, Gambar 6.6 dan Gambar 6.7.



Gambar 6.5 Kurva ROC dari Model BiLSTM



Gambar 6.6 Kurva ROC dari Model LSTM



Gambar 6.7 Kurva ROC dari Model BiLSTM-CRF

Dari nilai AUC yang dihasilkan dari setiap model, dapat dilihat bahwa secara umum semua model dapat mengenali setiap label dengan baik karena hampir semua label memiliki nilai AUC diatas 0.90. Nilai AUC paling rendah yang dihasilkan adalah 0.71 untuk label ITEM-SPEC pada model BiLSTM-CRF. Label ITEM-SPEC sendiri juga merupakan label dengan performa paling rendah jika dilihat dari nilai AUC untuk setiap model dimana model LSTM mendapat nilai AUC 0.73 dan model BiLSTM mendapatkan nilai AUC 0.78 pada label ini.

Halaman ini sengaja dikosongkan.

BAB VII
KESIMPULAN DAN SARAN

Bab kesimpulan dan saran membahas mengenai kesimpulan proses penelitian yang telah dilakukan dan saran yang diusulkan baik untuk perusahaan maupun untuk penelitian serupa di masa mendatang.

7.1 Kesimpulan

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

1. Penelitian ini menunjukkan bahwa *tuning* parameter sangat berpengaruh pada performa model. Pemilihan nilai parameter yang tepat dapat meningkatkan performa model secara signifikan.
2. *Library pytorch* belum menyediakan implementasi untuk CRF sehingga jika ingin menggunakannya harus mengimplementasikan sendiri.
3. Pada proses pembuatan *dataset* perlu memperhatikan distribusi dari semua label. Pembuatan *dataset* harus diatur sedemikian rupa sehingga semua label terdistribusi merata pada setiap *dataset*.
4. Model BiLSTM-CRF memiliki performa terbaik dibanding model *neural network* lain akan tetapi tidak pada semua konfigurasi parameter. Diperlukan pengaturan nilai parameter yang tepat agar model ini dapat menyaingi model lain.
5. Nilai akurasi bukan merupakan ukuran yang ideal untuk mengukur performa model *sequence tagging*. Pada penelitian ini hal itu disebabkan akurasi sangat dipengaruhi label O yang memiliki jumlah jauh lebih banyak dibanding label lain.

7.2 Saran

Dalam pengerjaan tugas akhir, terdapat beberapa saran yang diharapkan dapat bermanfaat pengembangan penelitian kedepan, yaitu:

1. Menggunakan jumlah data yang lebih banyak.
2. Menambah variasi *hyperparameter* yang diterapkan pada model dan menguji bagaimana pengaruhnya terhadap performa.
3. Melihat pengaruh penggunaan *epoch* yang berbeda selama proses training untuk mengetahui *epoch* yang optimal untuk pelatihan model.
4. Menguji pengaruh dari penggunaan *character embedding*.
5. Membandingkan performa dengan model *neural network* yang lain, seperti GRU.

DAFTAR PUSTAKA

[1] Asosiasi Penyelenggara Jasa Internet Indonesia, "Potret Zaman Now Pengguna & Perilaku Pengguna Internet Indonesia," *Buletin APJII Edisi 23*, p. 1, Apr-2018.

[2] D. Hutabarat, "Indonesia Kompetitif di Era Digital," 2018. [Online]. Available: https://kominfo.go.id/content/detail/14206/indonesia-kompetitif-di-era-digital/0/sorotan_media. [Accessed: 18-Feb-2019].

[3] Asosiasi Penyelenggara Jasa Internet Indonesia, "Saatnya Menjadi Pokok Perhatian Pemerintah dan Industri," *Buletin APJII Edisi 05*, p. 1, Nov-2016.

[4] K. Das, T. Tamhane, B. Vatterott, P. Wibowo, and S. Wintels, "The Digital Archipelago: How Online Commerce is Driving Indonesias Economic Development," 2018.

[5] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz, "End-to-end task-completion neural dialogue systems," *arXiv Prepr. arXiv1703.01008*, 2017.

[6] G. Mesnil *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE Trans. Audio, Speech Lang. Process.*, 2015.

[7] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *2014 IEEE Workshop on Spoken Language Technology, SLT 2014 - Proceedings*, 2014.

[8] A. Graves, A. R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2013.

[9] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-

CRF Models for Sequence Tagging," 2015.

[10] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed. Prentice Hall, 2008.

[11] Y. Gong *et al.*, "Deep Cascade Multi-task Learning for Slot Filling in Online Shopping Assistant," 2018.

[12] J. Kupiec, "Robust part-of-speech tagging using a hidden Markov model," *Comput. Speech Lang.*, 1992.

[13] A. McCallum, D. Freitag, and F. Pereira, "Maximum Entropy Markov Models for Information Extraction and Segmentation," in *International Conference on Machine Learning (ICML)*, 2000.

[14] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[15] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural Architectures for Named Entity Recognition," 2016.

[16] X. Ma and E. Hovy, "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF," pp. 1064–1074, 2016.

[17] E. F. T. K. Sang and J. Veenstra, "Representing Text Chunks," pp. 173–179, 1999.

[18] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for Spoken Language Understanding," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2007.

[19] K. Yao, G. Zweig, M. Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *Proceedings of the Annual Conference of the International Speech Communication*

Association, INTERSPEECH, 2013.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, 1986.

[21] A. Y. N. and C. P. Richard Socher, Alex Perelygin, Jean Y.Wu, Jason Chuang, Christopher D. Manning, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," *PLoS One*, 2013.

[22] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, 1997.

[23] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Trans. Neural Networks*, 1994.

[24] C. Ola, "Understanding LSTM Networks -- colah's blog, http://colah.github.io/posts/2015-08-Understanding-LSTMs/," *2015-08-27*, 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 10-Apr-2019].

[25] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, 2000.

[26] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," in *Neural Networks*, 2005.

[27] N. Reimers and I. Gurevych, "Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks," 2017.

[28] J. Yang and Y. Zhang, "NCRF++: An Open-source Neural Sequence Labeling Toolkit," 2018.

Halaman ini sengaja dikosongkan.

BIODATA PENULIS

Penulis lahir di Kediri pada tanggal 3 Maret 1997. Merupakan anak kedua dari tiga bersaudara. Penulis pernah menempuh pendidikan formal di SD Negeri Mojo, SMP Negeri 4 Kediri dan SMA Negeri 1 Kediri.

Setelah lulus dari SMA penulis melanjutkan pendidikan di Departemen Sistem Informasi Fakultas Teknologi Informasi dan Komunikasi – Institut Teknologi Sepuluh Nopember. Penulis terdaftar sebagai mahasiswa dengan NRP 05211540000003. Selama menjadi mahasiswa, penulis mengikuti berbagai kegiatan kemahasiswan seperti kepanitiaan dan himpunan mahasiswa departemen. Di bidang akademik penulis aktif menjadi asisten praktikum pada beberapa mata kuliah. Selain itu penulis sempat meraih prestasi sebagai juara 3 pada perlombaan Gemastik 2018 pada kategori pengembangan perangkat lunak.

Pada tahun keempat perkuliahan, penulis memiliki ketertarikan pada bidang pengolahan data dan *natural language processing* sehingga penulis mengambil bidang minal Akuisisi Data dan Diseminasi Informasi (ADDI). Penulis dapat dihubungi melalui email di *nurskediri@gmail.com.*

Halaman ini sengaja dikosongkan.

LAMPIRAN A

Hasil Percobaan Pertama Pengujian Model *Neural Network*

lr = 0.0015						
	Model	**accuracy**	**precision**	**recall**	**f-score**	**learning time (s)**
SGD + Momentum	BiLSTM-CRF	0.969	0.7892	0.7892	0.7892	851.4821305
	BiLSTM	0.9654	0.7659	0.7784	0.7721	338.6656957
	LSTM	0.9602	0.7419	0.7459	0.7439	320.3800142
SGD	BiLSTM-CRF	0.9569	0.6634	0.7351	0.6974	868.9962573
	BiLSTM	0.9625	0.6931	0.7568	0.7235	345.1080122
	LSTM	0.9595	0.7041	0.7459	0.7244	316.734659
Adam	BiLSTM-CRF	0.9599	0.6777	0.773	0.7222	937.1808655
	BiLSTM	0.9662	0.7302	0.7459	0.738	370.5961728
	LSTM	0.9573	0.6865	0.6865	0.6865	345.551909

lr = 0.015						
	Model	**accuracy**	**precision**	**recall**	**f-score**	**learning time (s)**
SGD + Momentum	BiLSTM-CRF	0.948	0.653	0.670	0.661	868.8613253
	BiLSTM	0.949	0.643	0.632	0.638	355.4042032
	LSTM	0.951	0.690	0.649	0.669	324.1307533
SGD	BiLSTM-CRF	0.969	0.804	0.800	0.802	1015.964345
	BiLSTM	0.966	0.757	0.773	0.765	325.8572109
	LSTM	0.942	0.643	0.595	0.618	339.000
Adam	BiLSTM-CRF	0.948	0.671	0.595	0.630	836.942786
	BiLSTM	0.945	0.637	0.616	0.626	356.3237889
	LSTM	0.942	0.643	0.595	0.618	345

lr = 0.0005						
	Model	**accuracy**	**precision**	**recall**	**f-score**	**learning time (s)**
SGD + Momentum	**BiLSTM-CRF**	0.965	0.743	0.811	0.775	845.3963578
	BiLSTM	0.967	0.772	0.805	0.788	345.1485586
	LSTM	0.963	0.743	0.768	0.755	319.3441398
SGD	**BiLSTM-CRF**	0.954	0.628	0.638	0.633	877.7269034
	BiLSTM	0.955	0.608	0.654	0.630	335.1171484
	LSTM	0.947	0.624	0.584	0.603	315
Adam	**BiLSTM-CRF**	0.967	0.754	0.730	0.742	886.4302976
	BiLSTM	0.967	0.757	0.757	0.757	362.3509407
	LSTM	0.9551	0.6931	0.7568	0.7235	341

Hasil Percobaan Kedua Pengujian Model *Neural Network*

lr = 0.0015						
	Model	**accuracy**	**precision**	**recall**	**f-score**	**learning time (s)**
SGD + Momentum	**BiLSTM-CRF**	0.9669	0.7592	0.7838	0.7713	834.2349
	BiLSTM	0.9702	0.7614	0.8108	0.7853	339.9213
	LSTM	0.9610	0.7433	0.7514	0.7473	335.8589
SGD	**BiLSTM-CRF**	0.9629	0.7092	0.7514	0.7297	838.6541
	BiLSTM	0.9632	0.6919	0.7405	0.7154	331.0685
	LSTM	0.9566	0.6859	0.7081	0.6968	318.3235
Adam	**BiLSTM-CRF**	0.9588	0.6990	0.7405	0.7192	849.1521
	BiLSTM	0.9621	0.7457	0.6973	0.7207	354.7121
	LSTM	0.9544	0.7102	0.6757	0.6925	338.0711

lr = 0.015						
	Model	**accuracy**	**precision**	**recall**	**f-score**	**learning time (s)**
SGD + Momentum	**BiLSTM-CRF**	0.9570	0.7342	0.6270	0.6764	815.6791
	BiLSTM	0.9581	0.7127	0.6973	0.7049	333.7219
	LSTM	0.9507	0.6628	0.6162	0.6387	316.9270
SGD	**BiLSTM-CRF**	0.9658	0.7617	0.7946	0.7778	826.2268
	BiLSTM	0.9680	0.7833	0.7622	0.7726	325.2779
	LSTM	0.9614	0.7540	0.7622	0.7581	311.8578
Adam	**BiLSTM-CRF**	0.9467	0.6339	0.6270	0.6304	847.5000
	BiLSTM	0.9412	0.5784	0.6378	0.6067	353.5842
	LSTM	0.9456	0.7073	0.6270	0.6648	347.9569

lr = 0.0005						
	Model	**accuracy**	**precision**	**recall**	**f-score**	**learning time (s)**
SGD + Momentum	**BiLSTM-CRF**	0.9662	0.7742	0.7784	0.7763	817.9020
	BiLSTM	0.9629	0.7475	0.8162	0.7804	343.5879
	LSTM	0.9584	0.7302	0.7459	0.7380	317.9211
SGD	**BiLSTM-CRF**	0.9522	0.6085	0.6216	0.6150	837.2228
	BiLSTM	0.9537	0.5979	0.6270	0.6121	331.6470
	LSTM	0.9437	0.5976	0.5459	0.5706	317.0654
Adam	**BiLSTM-CRF**	0.9625	0.7435	0.7676	0.7553	850.6249
	BiLSTM	0.9625	0.7216	0.7568	0.7388	362.1532
	LSTM	0.9588	0.6751	0.7189	0.6963	356.4425