TUGAS AKHIR – IF184802

ANALISIS SENTIMEN BERBASIS ASPEK DENGAN PENDEKATAN SEQUENCE LABELLING TASK

ROGO JAGAD ALIT
NRP 05111540000168

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – IF184802

ANALISIS SENTIMEN BERBASIS ASPEK DENGAN PENDEKATAN SEQUENCE LABELLING TASK

ROGO JAGAD ALIT
NRP 05111540000168

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

ASPECT BASED SENTIMENT ANALYSIS USING SEQUENCE LABELLING TASK APPROACH

ROGO JAGAD ALIT
NRP 05111540000168

First Advisor
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Second Advisor
Dini Adni Navastara, S.Kom., M.Sc.

Department of Informatics
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

ANALISIS SENTIMEN BERBASIS ASPEK DENGAN PENDEKATAN *SEQUENCE LABELLING TASK*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
ROGO JAGAD ALIT
NRP: 05111540000168

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. .....
   (NIP. 19751220 200112 2 002) (Pembimbing 1)

2. Dini Adni Navastara, S.Kom., M.Sc. .....
   (NIP. 19851017 201504 2 001) (Pembimbing 2)

SURABAYA
Juli, 2019

(Halaman ini sengaja dikosongkan)

ANALISIS SENTIMEN BERBASIS ASPEK DENGAN PENDEKATAN *SEQUENCE LABELLING TASK*

Nama Mahasiswa	**: Rogo Jagad Alit**
NRP	**: 05111540000168**
Jurusan	**: Informatika, FTIK-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Dini Adni Navastara, S.Kom., M.Sc.**

ABSTRAK

Jumlah pengguna internet di seluruh dunia bertambah dengan pesat dari 2 miliar di tahun 2010 sekarang menjadi 3 miliar pengguna seluruh dunia ditahun 2017. Bertambahnya penggunaan akses internet yang mudah, membuat sosial media menjadi sangat populer di kalangan masyarakat. Masyarakat akan cenderung bersosialisasi dengan cara mengekspresikan opini. Analisis sentimen menjadi sesuatu yang penting dalam kondisi tersebut terutama untuk keperluan bisnis. Data teks ulasan yang berisi opini terhadap suatu produk atau layanan dapat menjadi dasar pembuatan keputusan oleh konsumen dan menjadi feedback yang berharga bagi pemilik bisnis untuk mengembangkan produk atau layanannya.

Pada tugas akhir ini, akan dilakukan analisis sentimen berbasis aspek. Tujuannya adalah untuk mengetahui polaritas sentimen (positif, negatif atau netral) suatu aspek dari objek yang menjadi target opini. Data pelatihan dan uji coba diambil dari dataset "Restaurant Review ABSA-2015" yang berisi dokumen ulasan terhadap makanan dan layanan dari beberapa restoran. Pada Tugas Akhir ini proses pengenalan aspect term dari objek ulasan dilakukan dengan menggunakan pendekatan sequence labelling task. Sequence labelling task adalah sebuah proses untuk memberikan label untuk satu data di dalam sebuah rangkaian data dengan memanfaatkan fitur dan label dari data lain yang ada pada

rangkaian tersebut. Pendekatan sequence labelling task cocok digunakan pada Tugas Akhir ini karena untuk mengenali aspect term dalam sebuah kalimat, metode yang digunakan harus mengetahui informasi antar kata yang ada dari rangkaian kata atau kalimat yang diberikan. Metode yang akan digunakan untuk melakukan proses sequence labelling task untuk mengenali aspect term pada kalimat ulasan pada Tugas Akhir ini adalah Conditional Random Field (CRF). Hasil polaritas sentimen dari aspek ulasan akan didapatkan melalui metode rule-based dengan menggunakan 4 kamus kata opini dan pembobotan jarak kata. Hasil uji coba terakhir untuk pengenalan aspek target opini didapatkan nilai akurasi sebesar 98,07% dan untuk hasil analisis sentimen didapatkan nilai akurasi sebesar 85,06%

Kata kunci: *Conditional Random Field*, data teks, analisis sentimen, *aspect term*, polaritas sentimen, *sequence labelling task*

ASPECT BASED SENTIMENT ANALYSIS USING SEQUENCE LABELLING TASK

Student's Name : ROGO JAGAD ALIT
Student's ID : 05111540000168
Department : Informatics, Faculty of ICT-ITS
First Advisor : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
Second Advisor : Dini Adni Navastara, S.Kom., M.Sc.

ABSTRACT

The number of internet user worldwide increased rapidly from 2 billion in 2010 to 3 billion wordlwide in 2017. Increasing use of easy internet access, make social media become very popular among the people. People tend to socialize by expressing opions through existing social media. Sentiment analysis become something important in such situations, especially for business needs. Review texts that contains opinion towards any product or services can have such influence in consumer's decision making and become valuable feedback for business owner to improve their product or services.

This Final Project aims to do aspect-based sentiment analysis. The goal is to identify the sentiment polarity of certain aspect belongs to opinion's target. Training and testing data come from "Restaurant Review ABSA-2015" dataset which contains review documents of food and service from a restaurant. Aspect term from a review sentence is extracted using sequence labelling task approach. Sequence labelling task is an approach to give appropriate label to a data item belongs to a data sequence by using other data's label and features. Using sequence labelling task approach suits the aspect term extraction process in this Final Project because the implemented method should be able to make use of relation between words and other word's feature that belong to a review sentence. Conditional Random Field (CRF) is a method belongs to sequence labelling task that will be used in this Final Project to recognize aspect term mentioned in the review sentence.

Sentiment polarity from an aspect term is detected using rule-based method using 4 sentiment lexicons and word distance weighting. The accuracy for final test accuracy for aspect term detection is 98.07% and final test accuracy for aspect sentiment polarity analysis is 85.06%.

Keywords: ***Conditional Random Field, text data, sentiment analysis, aspect term, sentiment polarity, sequence labelling task***

KATA PENGANTAR

Puji syukur saya sampaikan kepada Tuhan yang Maha Esa karena berkat rahmat-Nya saya dapat melaksanakan Tugas Akhir yang berjudul:

"**ANALISIS SENTIMEN BERBASIS ASPEK DENGAN PENDEKATAN SEQUENCE LABELLING TASK**"

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Tuhan Yesus Kristus, karena limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Informatika ITS.
2. Kedua orangtua penulis, dan anggota keluarga lainnya yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. dan Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Dr. Eng. Radityo Anggoro, S.Kom., M.Kom. selaku Ketua Program Studi S1 Departemen Informatika ITS dan seluruh dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Informatika ITS.
5. Admin-admin Laboratorium Komputasi Cerdas & Visi (KCV) yang memberikan kesempatan penulis untuk fokus mengerjakan Tugas Akhir ini dan menyediakan tempat di laboratorium tersebut.

6. William Albertus Dembo, Cynthia Dewi Tejakusuma, Jonathan Rehuel Lewerissa, Chendrasena Oemaryoga, Nurlita Dhuha Fatmawati dan Mbak Nafia Rizky Yogayana sebagai teman – teman dari Laboratorium Pemrograman 2 yang selalu memberi semangat dan dukungan moral kepada penulis.
7. Teman – teman User TA KCV dari angkatan 2015 yang telah menemani, menghibur dan memberi dukungan kepada penulis selama pengerjaan Tugas Akhir ini.
8. Seluruh mahasiswa Informatika ITS angkatan 2015 yang telah menjadi teman penulis selama menjalani masa kuliah di Informatika ITS.
9. Hendry Wiranto atas warisan template buku, presentasi dan jurnal yang sangat membantu penulis dalam menyusun tugas akhir ini.
10. Alexander Nugi Nugroho, Hizkia Steven dan Joshua Nurdjalim atas dukungan, semangat dan doa yang diberikan kepada penulis selama menjalani masa kuliah dan mengerjakan Tugas Akhir ini.
11. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih memiliki banyak kekurangan. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis kedepannya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum.

Surabaya, April 2019

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

DAFTAR KODE SUMBER

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring berkembangnya teknologi dan bertambahnya penggunaan akses internet yang mudah, membuat sosial media menjadi sangat populer di kalangan masyarakat. Masyarakat cenderung bersosialisasi dengan cara mengekspresikan opini melalui sosial media. Analisis sentimen menjadi sesuatu yang penting pada kondisi tersebut terutama untuk keperluan bisnis. Contohnya pada bisnis restoran dimana pendapat seorang konsumen terhadap makanan atau layanan yang diberikan dapat menjadi *feedback* yang berharga bagi pelaku bisnis untuk mengembangkan bisnisnya. Dengan melakukan analisis sentimen maka setiap pendapat yang diberikan akan dianalisis. Analisis sentimen yang dilakukan secara akurat dan terus-menerus akan memberikan beberapa manfaat diantaranya, dapat meningkatkan pelayanan konsumen, dapat mengembangkan kualitas produk terkait, dapat meningkatkan pendapatan penjualan, dan dapat memberikan strategi bisnis untuk masa mendatang.

Analisis sentimen adalah kombinasi dari teknik *natural language processing* (NLP) dan teknik *data mining* [1] yang berfokus untuk mengidentifikasi polaritas sentimen (positif, negatif dan netral) pada sebuah dokumen opini atau pendapat. Seiring dengan berkembangnya teknologi, banyak perusahaan, badan riset dan universitas yang terus mengembangkan proses analisis sentimen ini untuk dilakukan sampai pada level aspek dari target opini. Sehingga, pengguna dapat mengetahui polaritas sentimen untuk suatu *aspect term* yang menjadi target opini secara spesifik. Untuk dapat mengetahui target opini yang ada pada sebuah kalimat, sistem akan memanfaatkan relasi semantik dan sintaksis dari setiap kata yang ada di dalam dokumen untuk mendapatkan fitur – fitur yang dapat dimasukkan ke dalam algoritma klasifikasi yang akan menentukan apakah kata tersebut adalah *aspect term* atau bukan. Proses klasifikasi ini termasuk

dalam *sequence labelling task* karena untuk mendapatkan label untuk satu data bergantung dari fitur dan label dari data yang lain pada sekuens data tersebut [2].

Conditional Random Field (CRF) adalah sebuah metode pemodelan statistik yang sering digunakan pada bidang pengenalan pola, *machine learning* dan prediksi data terstruktur [3]. CRF termasuk dalam kelompok *sequence modelling classifier* yang melakukan prediksi label terhadap satu data di dalam rangkaian dengan memperhatikan data lainnya yang ada di dalam rangkaian tersebut. Berbeda dengan *discrete classifier* yang hanya melakukan prediksi label terhadap satu data tanpa memperhitungkan data lainnya yang ada. Di dalam tugas akhir ini penulis mengusulkan sebuah sistem yang mampu melakukan analisis sentimen berbasis aspek dengan menggunakan *Conditional Random Field* (CRF) untuk mendapatkan *aspect term* yang menjadi target opini secara spesifik dan dilanjutkan dengan analisis polaritas sentimen untuk *aspect term* yang didapatkan menggunakan metode *rule-based*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana mengimplementasikan metode CRF untuk mendeteksi *aspect term* dari objek yang menjadi target opini?
2. Bagaimana menentukan nilai polaritas sentimen sebuah *aspect term* dari target opini menggunakan metode *rule-based*?
3. Bagaimana mengevaluasi kinerja model Conditional Random Field dan sistem analisis sentimen *aspect term* yang telah diimplementasikan?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Data teks diambil dari dataset *Restaurant Review ABSA-2015*.
2. Domain permasalahan yang digunakan pada dataset adalah domain restoran
3. Bahasa yang digunakan pada dataset adalah Bahasa Inggris
4. Implementasi program menggunakan bahasa pemrograman *Python 3*.
5. Ada 3 polaritas sentimen yang akan dikenali yaitu positif, negatif dan netral.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk membuat sistem yang mampu mengenali aspek yang diulas pada target opini dan mengenali nilai polaritas sentimen terhadap aspek tersebut.

1.5 Manfaat

Tugas Akhir ini diharapkan dapat membantu menambahkan kemampuan yang ada dalam analisis sentimen pada data teks sehingga mampu melakukan analisis sentiment sampai pada level aspek dari target opini.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir yang berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri dari latar belakang diajukannya Tugas Akhir, rumusan masalah dan batasan masalah yang ditetapkan, serta manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi

pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian literatur berupa jurnal yang digunakan sebagai referensi untuk pengerjaan tugas akhir ini. Literatur yang dipelajari pada pengerjaan tugas akhir ini berasal dari jurnal ilmiah yang diambil dari berbagai sumber di internet, beserta berbagai literatur online tambahan terkait *data processing* pada *Python 3, Conditional Random Field* dan *Sentiment Analysis*.

1.6.3 Implementasi Perangkat Lunak

Pada tahap ini akan dilaksanakan implementasi metode dan algoritma yang telah direncanakan. Implementasi sistem menggunakan *Python 3* sebagai bahasa pemrograman, *Spacy* sebagai *library* untuk melakukan *teks processing, Standford Dependency Parser* untuk melakukan *dependency parsing* pada data teks, serta *library* pendukung lainnya.

1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan menggunakan data uji yang telah disediakan dataset *Restaurant Review ABSA-2015* untuk mengetahui hasil dan performa arsitektur yang telah dibangun. Evaluasi dilakukan dengan metode pengukuran akurasi, *precision,* dan *recall*.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

Bab I Pendahuluan

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

Bab II Tinjauan Pustaka

Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang *Conditional Random Field* dan *library* yang digunakan.

Bab III Perancangan Sistem

Bab ini berisi pembahasan mengenai perancangan dari *Conditional Random Field* dan metode *rule-based* yang digunakan untuk pengenalan polaritas sentimen dari aspek objek ulasan.

Bab IV Implementasi

Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi terhadap kinerja dari sistem yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini membahas teori-teori dasar yang yang menjadi dasar pembuatan Tugas Akhir.

2.1 Analisis Sentimen

Analisis sentimen adalah sebuah proses yang memanfaat *Natural Language Processing (NLP)*, analisis data teks, komputasi linguistik atau data biometric untuk mengidentifikasi, mengekstraksi, mengukur dan mempelajari aspek psikologis dan informasi subjektif dari data yang ada. Analisis sentimen banyak dimanfaatkan untuk mengolah dan mempelajari pendapat orang terhadap suatu topik. Beberapa bidang yang dapat memanfaatkan proses analisis sentimen ini meliputi bidang informasi *feedback* dalam sebuah bisnis, pelayanan konsumen, *brand monitoring* dan lain – lain [4].

2.2 Text Processing

Text processing disebut juga sebagai *text mining* adalah sebuah pemrosesan teks untuk menghasilkan informasi atau *insights* pada suata data teks. Untuk menghasilkan beberapa informasi, t*ext mining* menggunakan beberapa metode salah satunya NLP (*Natural Laguage Processing)* [5]. Terdapat beberapa tugas dalam NLP diantaranya adalah sebagai berikut

2.1.1 Tokenization

Tokenization adalah memisahkan kata, simbol, frase, dan entitas penting lainnya (yang disebut sebagai token) dari sebuah teks untuk kemudian di analisis lebih lanjut. Token dalam NLP sering dimaknai dengan "sebuah kata", walau tokenisasi juga bisa dilakukan ke kalimat, paragraf, atau entitas penting lainnya [6].

2.1.2 Dependency Parsing

Dependency parsing adalah sebuah proses dalam NLP yang bertujuan untuk mengekstraksi struktur relasi grammatikal antar kata dari sebuah kalimat berdasarkan peraturan tata bahasa dari bahasa yang digunakan [7]. Proses *dependency parsing* akan menghasilkan nama relasi grammatikal untuk setiap relasi kata yang ada pada kalimat tersebut. Beberapa contoh relasi grammatikal dalam Bahasa Inggris adalah sebagai berikut:

1. *Nominal Subject (nsubj)*
   Sebuah kata benda yang menjadi subjek dari sebuah kata kerja.

2. *Direct Object (dobj)*
   Sebuah kata benda menjadi objek dari sebuah kata kerja.

3. *Conjuction (conj)*
   2 buah kata yang dihubungkan oleh kata sambung.

4. *Adjectival Modifier (amod)*
   Sebuah frasa kata sifat yang menjelaskan frasa kata benda.

5. *Dependent (dep)*
   Relasi ini digunakan ketika proses *dependency parsing* tidak dapat menemukan relasi yang tepat untuk sepasang kata. Hal ini dapat disebabkan oleh konstruksi gramatikal yang kurang tepat ataupun keterbatasan pada pustaka *dependency parsing* yang digunakan.

2.1.3 Lemmatisasi

Lemmatisasi adalah proses menghasilkan bentuk kata dasar dari sebuah kata yang telah diberi imbuhan. Contoh imbuhan dalam Bahasa Inggris adalah pemberian akhiran *-ed*, *-ing* atau *-en*. Lemmatisasi mengembalikan bentuk kata dasar dari sebuah kata

dengan cara mencari bentuk kata dasar yang makna morfologinya paling mendekati dari kata yang saat ini diberikan [5].

2.3 Conditional Random Field (CRF)

Conditional Random Field (CRF) merupakan suatu model probabilistik *discriminative* yang digunakan untuk melakukan segmentasi dan pelabelan suatu sekuens data yang terstruktur. CRF termasuk dalam kelompok *sequence modelling classifier* yang melakukan prediksi sekuens label Y untuk sebuah sekuens data X dengan memperhitungkan fitur dan label dari keseluruhan data yang ada pada sekuens tersebut. Berbeda dengan *discrete classifier* yang hanya melakukan prediksi label terhadap satu data tanpa memperhitungkan data lainnya [8].

Pada CRF terdapat dua peubah acak dimana $\bar{x} = (x_1, x_2, \ldots, x_n)$ adalah sekuens data observasi dan $\bar{y} = (y_1, y_2, \ldots, y_n)$ adalah sekuens label yang akan diprediksi untuk data observasi $\bar{x}$. Probabilitas kondisional dari $\bar{y}$ terhadap $\bar{x}$ dapat dirumuskan sebagai berikut:

$$P(\bar{y}|\bar{x}) = \frac{1}{Z}\prod_{t=1}^{T} exp\left\{\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \bar{x})\right\} \qquad (2.1)$$

Dimana

- θ_k = bobot dari setiap fungsi fitur f_k
- $f_k(y_{t-1}, y_t, \bar{x})$ = fungsi fitur yang melakukan ekstraksi fitur pada komponen data posisi t berdasarkan label pada posisi t – 1, label pada posisi t dan data observasi pada $\bar{x}$ secara global.
- y_t = label untuk komponen data pada posisi t dalam sekuens data $\bar{x}$
- y_{t-1}= label untuk komponen data pada posisi t - 1 dalam sekuens data $\bar{x}$
- x_t = komponen data dalam sekuens data $\bar{x}$ pada posisi t

Z adalah konstanta normalisasi yang memastikan agar nilai yang dihasilkan adalah nilai probabilitas yang valid (berada dalam rentang 0 dan 1). Z didefiniskan pada persamaan (2.2).

$$Z = \sum_{y'} \prod_{t=1}^{T} exp\{\sum_{k=1}^{K} \theta_k f_k(y'_t, y'_{t-1}, \bar{x})\} \quad (2.2)$$

Fungsi fitur pada CRF adalah fungsi yang mengembalikan bilangan *real* dengan melakukan ekstraksi fitur untuk setiap komponen data pada sekuens data $\bar{x}$ [9]. Berikut adalah contoh fitur fungsi dalam kasus pengenalan *aspect term* menggunakan CRF:

$$f_k(y_t, y_{t-1}, \bar{x}) \begin{cases} 1, if\ y_t = B\ and\ postag(x_t) = NOUN \\ 0, otherwise \end{cases} \quad (2.3)$$

Fungsi fitur (2.3) akan mengembalikan nilai 1 apabila label kata pada posisi t adalah B dan label POS Tag dari kata pada posisi t adalah NOUN. Pada proses pelatihan akan dilakukan proses penyesuaian bobot untuk setiap fungsi fitur agar dapat memaksimalkan kemungkinan label yang sesuai pada data latih. Kemudian, pada proses klasifikasi, bobot untuk setiap fitur ini digunakan untuk memberikan kemungkinan label yang paling sesuai untuk data yang diberikan.

2.4 Beginning-Inside-Outside (BIO) Tagging

Format tagging BIO adalah format *tagging* yang umum digunakan untuk melakukan *labelling* dalam proses *chunking* pada komputasi linguistik seperti *Named Entity Recognition*. Format tagging ini pertama kali diperkenalkan oleh Ramshaw dan Marcus dalam *paper*-nya yang berjudul "Text Chunking using Transformation Based Learning".

Tag B digunakan untuk menandakan sebuah kata sebagai awal sebuah *chunk*. Tag I digunakan untuk menandakan sebuah kata sebagai bagian dari sebuah *chunk* yang terdiri dari banyak kata. Tag O digunakan untuk menandakan sebuah kata bukan bagian dari sebuah *chunk* [10].

2.5 Sentiment Lexicon

Analisis sentimen adalah sebuah proses yang bertujuan untuk mendeteksi nilai polaritas sentimen dari sebuah data teks. Proses analisis sentimen bergantung pada *sentiment lexicon* sebagai sumber data untuk menentukan nilai sentimen sebuah kata [11]. Pada Tugas Akhir ini akan digunakan 4 *sentiment lexicon* untuk melakukan penilaian sentimen untuk suatu kata.

2.5.1 Bing Liu Sentiment Lexicon

Bing Liu Sentiment Lexicon adalah kamus kata opini dalam Bahasa Inggris yang dikumpulkan dan dikembangkan oleh Bing Liu dan Minqing Hu sejak tahun 2004. *Bing Liu Sentiment Lexicon* terdiri dari sekitar 6800 kata yang dikelompokkan kedalam 2 kelompok, yaitu kata positif dan negatif [12]. Contoh kata positif dan negatif pada *Bing Liu Sentiment Lexicon* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Contoh kata positif dan negatif pada Bing Liu Sentiment Lexicon

Kata Positif	**Kata Negatif**
achievable	abnormal
adaptive	abolish
admirably	abort
adored	absence

2.5.2 General Inquirer Sentiment Lexicon

General Inquirer Sentiment Lexicon adalah kamus kata dalam Bahasa Inggris yang dikembangkan oleh Harvard

University. *General Inquirer Opinion Lexicon* terdiri dari kurang lebih 4000 kata, yang terbagi menjadi 2000 kata positif dan 2000 kata negatif [13]. Contoh kata positif dan negatif pada *General Inquirer Sentiment Lexicon* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Contoh kata negatif dan positif pada General Inquirer Sentiment Lexicon

Kata Positif	Kata Negatif
honest	horrible
honor	horrid
honorable	hostile
useful	useless

2.5.3 Multi-Perspective Question Answering (MPQA)

Multi-Perspective Question Answering adalah kamus kata dalam Bahasa Inggris yang dikembangkan oleh University of Pittsburgh. *MPQA* menyediakan 3 attribut data untuk setiap kata yang ada. Yaitu *subjectivity*, *polarity* dan label *POS Tagging* kata tersebut [14].

1) *Subjectivity* menyatakan kekuatan subjektivitas dari kata tersebut, attribut ini dapat bernilai *"weaksubj"* (subjektivitas lemah) atau *"strongsubj"* (subjektivitas kuat).

2) *Polarity* menyatakan nilai polaritas sentimen dari kata tersebut. Attribut ini dapat bernilai *negative*, *positive* atau *neutral*.

3) *POS Tagging* menyatakan kelas kata tersebut dalam Bahasa Inggris. Attribut ini dapat bernilai label *POS Tagging* apapun dalam Bahasa Inggris atau "*anypos"* apabila kata tersebut termasuk dalam beberapa kelas kata yang berbeda.

Contoh data untuk kata "*horrible*" pada *MPQA Sentiment Lexicon* dapat dilihat pada Tabel 2.3.

Tabel 2.3 Data untuk kata "*horrible*" pada MPQA Sentiment Lexicon

horrible	
subjectivity	*strongsubj*
polarity	*negative*
pos_tag	*anypos*

2.5.4 SentiWordNet

SentiWordNet adalah sumber *lexical (*kamus) untuk *opinion mining. SentiWordNet* terdiri dari beberapa *synset (Synonim Set)* yang memiliki dua nilai untuk masing-masing yaitu, positif dan negatif. Setiap nilai *term* dalam *synset* juga diurutkan berdasarkan seberapa sering digunakan dalam pengertian itu [15].

Tabel 2.4 Perubahan format pada *SentiWordNet* untuk *POS Tagging* pada Spacy

No	***POS Tagging* pada Spacy**	***POS Tagging* pada *SentiWordNet***
1	ADJ	a
2	NOUN	n
3	VERB	v
4	ADV	r

Keterangan:
a = *adjektiva*, n = *noun*, v = *verb*, r = *adverb*

Pada *SentiWordNet* terdapat empat label *POS Tagging* yang digunakan. Oleh karena itu perlu adanya perubahan format label *POS Tagging* dari Spacy ke dalam *SentiWordNet.* Perubahan format dapat dilihat pada Tabel 2.4 Perubahan format pada *SentiWordNet* untuk *POS Tagging* pada SpacyContoh data untuk

untuk kata "*horrible*" pada SentiWordNet dapat dilihat pada Tabel 2.5.

Tabel 2.5 Contoh data untuk kata "*horrible*" pada SentiWordNet

horrible	
obj_score	0.375
pos_score	0.0
neg_score	0.625

2.6 Part of Speech Tagging (POS Tagging)

POS Tagging merupakan suatu cara untuk mengkategorikan kelas kata, seperti kata benda, kata kerja, kata sifat, dan lain-lain. Dalam Bahasa Indonesia ada beberapa kelas kata yang dipakai antara lain sebagai berikut [16]:

1. Kata benda (*nomina*)
   Merupakan kata-kata yang merujuk pada bentuk suatu benda. Benda dapat bersifat konkret atau abstrak.
2. Kata kerja (*verba*)
   Merupakan jenis kata yang menyakan suatu perbuatan oleh subjek.
3. Kata sifat (*adjektiva*)
   Merupakan kelompok kata yang mampu menjelaskan atau mengubah kata benda atau kata ganti menjadi lebih spesifik. Selain itu, kata sifat mampu menerangkan kuantitas dan kualitas dari kelompok kelas kata benda atau kata ganti.
4. Kata ganti (*pronomia*)
   Merupakan kata yang digunakan untuk menggantikan suatu benda atau sesuatu yang dibendakan.
5. Kata keterangan (*adverbia*)
   Merupakan kata yang memberikan keterangan kepada kata kerja, kata sifat, dan kata bilangan, bahkan mampu memberikan keterangan pada seluruh kalimat.

6. Kata bilangan (*numeralia*)

   Merupakan jenis kata yang menyatakan jumlah, ukuran, dan urutan sesuatu yang dibendakan.

7. Kata tugas

   Merupakan kata yang memiliki arti gramatikal dan tidak memiliki arti leksikal. Dari segi bentuk umumnya, kata-kata tugas sukar mengalamai perubahan bentuk, seperti kata dengan, telah, dan tetapi.

2.7 Token Distance

Token Distance menyatakan jarak antara 2 token dalam data teks yang sudah melalui proses tokenisasi. *Token Distance* dihitung dari banyaknya token yang terdapat diantara 2 token tersebut [17].

2.8 Dependency Path Distance

Dependency path distance menyatakan jarak antara 2 token dalam data teks yang sudah melalui proses *dependency parsing*. Jarak antara 2 token ini dinyatakan dalam nilai *shortest path* antara 1 token dengan token yang lain dalam *dependency graph* pada suatu kalimat [17].

2.9 Akurasi, Precision & Recall

Ketika membangun sebuah model klasifikasi, pertanyaan yang muncul adalah bagaimana mengetahui seberapa baik model tersebut. Mengevaluasi model klasifikasi dilakukan dengan mencari tahu seberapa baik hasil prediksi dari model tersebut. *Recall* di Persamaan (2.4), *precision* di Persamaan (2.5), dan akurasi di Persamaan (2.6) adalah metode pengukuran yang biasa digunakan dalam mengevaluasi model, penjelasan variabel ada pada *confusion matrix* pada Tabel 2.6.

Tabel 2.6 *Confusion matrix*

		Kelas Prediksi	
		Benar	Salah
Kelas sebenarnya	Benar	*True Positive* (TP)	*False Negative* (FN)
	Salah	*False Positive* (FP)	*True Negative* (TN)

Keterangan :

1. *True Positive* (TP) pada teks dikenali sebagai kelas yang sesuai, pada keluaran program dikenali kelas yang sesuai.
2. *True Negative* (TN) pada teks tidak dikenali kelas yang sesuai, pada keluaran program tidak mengenali kelas tersebut.
3. *False Positive* (FP) pada teks dikenali kelas yang sesuai, pada keluaran program tidak mengenali kelas tersebut.
4. *False Negative* (FN) pada teks tidak dikenali kelas yang sesuai, pada keluaran program mengenali kelas tersebut.

$$Recall = TP / (TP + FN) \quad (2.4)$$

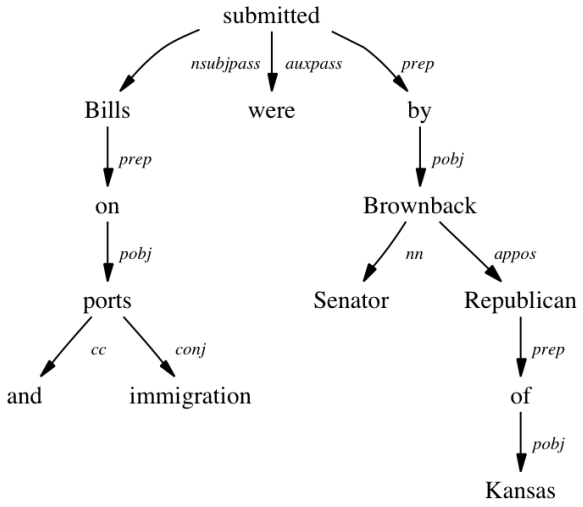
$$Precision = TP / (TP + FP) \quad (2.5)$$

$$Akurasi = (TP + TN) / (TP + FP + TN + FN) \quad (2.6)$$

2.10 Standford Dependency Parser

Standford Dependency Parser adalah sebuah *library* untuk kebutuhan *Natural Language Processing (NLP)* yang ditulis menggunakan bahasa pemrograman Java. *Library* ini dikembangkan secara khusus untuk mengolah dan merepresentasikan relasi gramatikal antar kata dalam satu kalimat dalam bahasa Inggris. Relasi gramatikal antar kata disimpan dalam bentuk *dependency graph* dimana setiap kata menjadi *node* dan nama relasi menjadi label dari *edge* pada *graph* tersebut [7]. Contoh hasil penguraian relasi gramatikal dengan *library* Standford Dependency Parser untuk kalimat "*Bills on ports and*

immigration were submitted by Senator Brownback, Republican of Kansas" dapat dilihat pada Gambar 2.1.



Gambar 2.1 Representasi penguraian relasi gramatikal menggunakan Standford Dependency Parser [7]

2.11 pycrfsuite

Pycrfsuite adalah *pustaka* antarmuka dalam untuk *library* CRFSuite yang ditulis oleh Naoaki Okazaki dalam bahasa C/C++. *Pycrfsuite* dikembangkan oleh Terry Peng dan Mikhail Koborov. *Pycrfsuite* mengimplementasikan dan menerjemahkan fitur implementasi CRF pada *library* CRFSuite ke dalam bahasa pemrograman Python [19].

2.12 Spacy

Spacy adalah *open source library* untuk kebutuhan *Natural Language Processing (NLP)* yang ditulis dalam bahasa pemrograman Python dan Cython. Spacy memberikan dukungan untuk lebih dari 49 bahasa di dunia. Spacy menyediakan fitur

tokenisasi, *POS Tagging*, *dependency parsing*, *Named Entity Recognition (NER)* dan lain – lain [20].

2.13 NetworkX

NetworkX adalah *open source library* untuk bahasa pemrograman Python. NetworkX menyediakan berbagai fungsi untuk kebutuhan pengolahan data berbentuk *graph*. NetworkX menyediakan fitur seperti struktur data *graph*, fungsi implementasi algoritma *graph*, pembuatan data *graph* acak dan lain – lain [21].

2.14 Scikit-learn

Scikit-learn adalah *open source machine learning library* untuk bahasa pemograman Python. Scikit-learn menyediakan fitur seperti *classification*, *regression*, *clustering*, termasuk juga didalamnya algoritma *support vector machines*, *random forest*, *gradient boosting*, dan lain-lain [22].

BAB III
PERANCANGAN SISTEM

Bab ini akan menjelaskan tentang analisis dan perancangan sistem untuk mencapai tujuan dari tugas akhir. Perancangan ini meliputi perancangan data dan perancangan proses. Bab ini juga akan menjelaskan tentang analisis implementasi metode secara umum pada sistem.

3.1 Perancangan Data

Data yang digunakan sebagai masukan awal dari sistem analisis sentimen berbasis aspek adalah data Restaurant Review SemEval-2015. *Restaurant Review ABSA-2015* adalah dataset yang terdiri dari 1702 data ulasan terhadap sebuah restoran. Setiap data memiliki 3 atribut, yaitu kalimat ulasan, *aspect term* dari restoran yang menjadi target opini dan polaritas sentimen untuk masing – masing *aspect term*. Contoh dataset bisa dilihat pada Gambar 3.1.

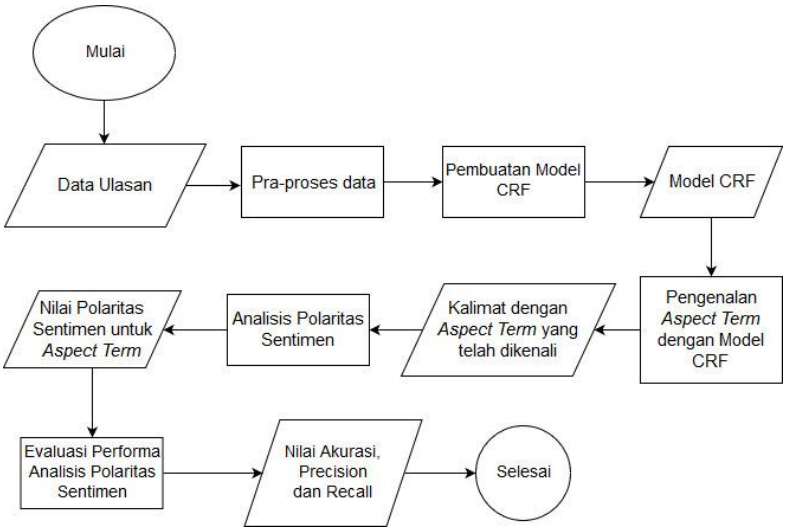

```
{
    "polarity": [
        "positive",
        "positive",
        "positive"
    ],
    "sentence": "Great food, amazing service, this place is a class act.",
    "target": [
        "food",
        "service",
        "place"
    ]
},
{
    "polarity": [
        "positive"
    ],
    "sentence": "The veal was incredible last night.",
    "target": [
        "veal"
    ]
},
```


Gambar 3.1 Contoh dataset *Restaurant Review ABSA-2015* [23]

Dataset *Restaurant Review ABSA-2015* ini terbagi menjadi 1120 data latih dan 582 data uji. Data latih diproses menggunakan *Conditional Random Field* untuk membangun model, kemudian model digunakan dalam pengenalan *aspect term* pada data uji. Kemudian kinerja model diukur dengan akurasi, *precision*, dan *recall* pengujian. Hasil pengenalan *aspect term* kemudian menjadi data masukan untuk proses analisis sentimen untuk mendapatkan nilai polaritas sentimen dari *aspect term* tersebut pada kalimat terkait.

3.2 Desain Umum Sistem

Sistem analisis sentimen berbasis aspek yang dibangun memiliki proses utama diantaranya praproses data, pembuatan model CRF, pengenalan *aspect term* dan analisis polaritas sentimen. Diagram alir keseluruhan dari sistem ditunjukan pada Gambar 3.2.



Gambar 3.2 Diagram alir keseluruhan sistem yang dibangun

Akan dilakukan praproses data terlebih dahulu sebelum data digunakan sebagai data untuk melatih dan menguji model. Pada setiap data akan dilakukan proses tokenisasi, pemberian label *POS Tag* untuk setiap kata dan *dependency parsing*. Kemudian akan dilakukan proses pemberian label BIO untuk menandakan apakah token tersebut adalah bagian dari *aspect term*.

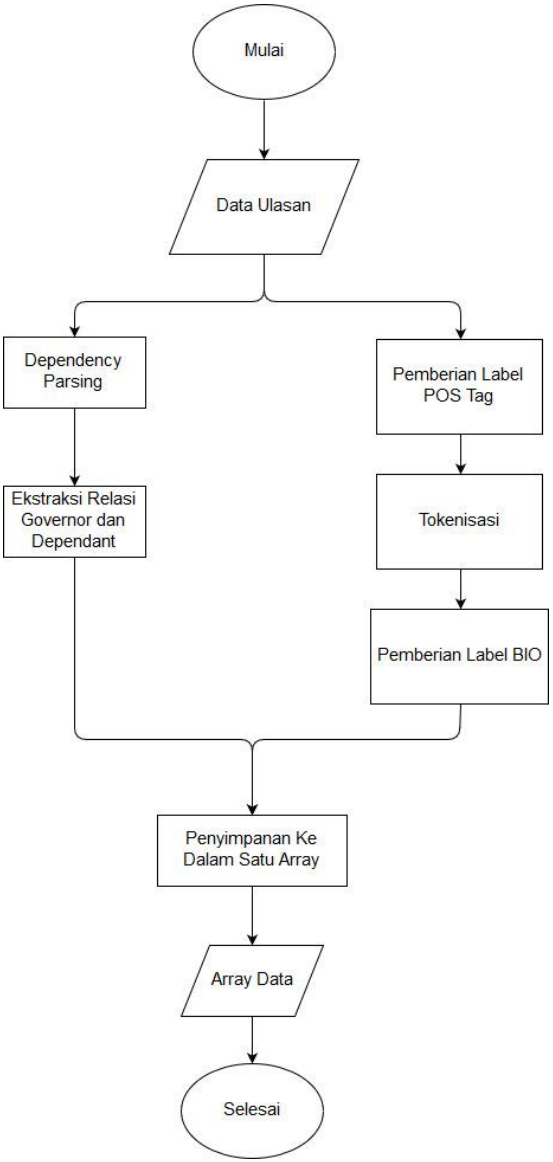
Proses pembuatan model adalah proses ekstraksi fitur yang diperlukan dan proses pelatihan model *Conditional Random Field* yang akan digunakan untuk mendeteksi *aspect term* dalam sebuah kalimat ulasan pada data uji. Data latih yang telah melalui tahap praproses data akan diekstraksi fiturnya. Selanjutnya, hasil dari proses pelatihan tersebut akan menjadi model untuk proses klasifikasi. Data uji juga akan melalui tahap praproses data dahulu sebelum diekstraksi fiturnya untuk menjadi *input* dalam proses pengujian.

Proses pengenalan *aspect term* akan menggunakan model CRF yang dihasilkan pada proses pelatihan. Hasil pengenalan *aspect term* akan dibandingkan dengan data sebenarnya, maka dapat dievaluasi nilai akurasi, *precision*, dan *recall* dari model CRF tersebut.

Proses analisis nilai polaritas sentimen bertujuan untuk mendapatkan nilai polaritas sentimen untuk setiap *aspect term* yang didapatkan pada proses sebelumnya. Hasil dari analisis sentimen akan dibandingkan dengan label sentimen sebenarnya untuk *aspect term* tersebut, maka dapat dievaluasi nilai akurasi, *precision*, dan *recall* dari proses analisis sentimen tersebut.

3.2.1 Tahap Praproses Data

Pada tugas akhir ini, praproses data yang dilakukan adalah melakukan proses *dependency parsing*, tokenisasi, pemberian label POS Tag dan pemberian label BIO. Diagram alir praproses data dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram alir pra-proses data

Sebelum digunakan pada proses selanjutnya, akan dilakukan proses *balancing* pada dataset terlebih dahulu. Tujuannya agar model yang dibentuk nantinya tidak memiliki kecenderungan terhadap suatu label tertentu. Pada tugas akhir ini, akan dilakukan proses *balancing* dengan mengurangi jumlah sampel untuk label O yang akan dilakukan dengan membuang data review yang tidak memiliki target ulasan. Setelah dilakukan proses balancing, dataset terbagi menjadi 832 data latih dan 388 data uji.

Kemudian, setiap data kalimat akan melalui proses *dependency parsing* untuk mendapatkan relasi grammatikal antar kata yang ada pada data kalimat tersebut. Dari proses *dependency parsing* yang telah dilakukan, dapat diperoleh relasi *governor* dan *dependant* untuk setiap kata yang ada pada kalimat tersebut beserta nama relasinya.

```
[(('lousy', 'JJ'), 'nsubj', ('food', 'NN')),
 (('food', 'NN'), 'det', ('The', 'DT')),
 (('lousy', 'JJ'), 'cop', ('was', 'VBD')),
 (('lousy', 'JJ'), 'conj', ('sweet', 'JJ')),
 (('sweet', 'JJ'), 'advmod', ('too', 'RB')),
 (('lousy', 'JJ'), 'cc', ('or', 'CC')),
 (('lousy', 'JJ'), 'conj', ('salty', 'JJ')),
 (('salty', 'JJ'), 'advmod', ('too', 'RB')),
 (('lousy', 'JJ'), 'cc', ('and', 'CC')),
 (('lousy', 'JJ'), 'conj', ('tiny', 'JJ')),
 (('tiny', 'JJ'), 'dep', ('portions', 'NNS')),
 (('portions', 'NNS'), 'det', ('the', 'DT'))]
```

Gambar 3.4 Contoh hasil proses *dependency parsing*

Lalu akan dilanjutkan dengan proses ekstraksi relasi *governor* dan *dependant*. Proses ini membutuhkan hasil dari proses *dependency parsing* dan tokenisasi. Proses ini bertujuan untuk mengambil nama relasi yang ada saat suatu kata ketika bersifat sebagai *governor* dan *dependant*. Nama relasi yang diambil saat suatu kata bersifat sebagai *governor* adalah *amod*, *nsubj* dan *dep*. Sedangkan nama relasi yang diambil saat suatu kata bersifat

sebagai *dependant* adalah *nsubj*, *dobj* dan *dep*. Jika pada relasi *governor* atau *dependant* untuk setiap kata tidak terdapat jenis relasi yang diperlukan, maka akan bernilai *None*. Contoh hasil proses *dependency parsing* untuk kalimat "*The food was lousy - too sweet or too salty and the portions tiny*" dapat dilihat pada Gambar 3.4.

Hasil *dependency parsing* disimpan dalam urutan: kata yang menjadi *governor*, nama relasi, kata yang menjadi *dependent*. Maka, pada Gambar 3.4 dapat dilihat bahwa kata "lousy" memiliki relasi *nsubj* dengan kata "food". Dimana kata "lousy" bertindak sebagai *governor* dan kata "food" bertindak sebagai *dependant*.

Selain melalui proses *dependency parsing* seperti yang telah dijelaskan sebelumnya, setiap data kalimat juga melalui proses pemberian label *POS Tag* untuk mendapatkan label kelas kata untuk setiap kata. Kemudian dilanjutkan dengan proses tokenisasi untuk memecah kalimat kedalam bentuk token. Setelah itu akan dilanjutkan dengan proses pemberian label BIO untuk menandakan apakah suatu kata dalam kalimat adalah bagian dari *aspect term* atau bukan. Pada proses pelatihan, label ini berfungsi sebagai acuan dalam pembuatan model CRF dan pada proses pengujian label ini berfungsi sebagai *ground truth* untuk proses evaluasi.

Proses terakhir adalah menyimpan hasil dari setiap tahap praproses data kedalam satu struktur data array. Setiap kata akan diwakili oleh struktur data *tuple* yang akan menyimpan data berikut:

1. Nilai *string* kata
2. Label *POS Tag* untuk kata tersebut
3. Relasi ketika kata tersebut bersifat sebagai *governor*
4. Relasi ketika kata tersebut bersifat sebagai *dependant*
5. Label BIO yang sesuai dengan kata tersebut.

Contoh hasil keluaran dari tahap praproses untuk kalimat "*The food was lousy - too sweet or too salty and the portions tiny*" dapat dilihat pada Gambar 3.5.

```
[('The', 'DET', None, None, 'O'),
 ('food', 'NOUN', None, 'nsubj', 'B'),
 ('was', 'VERB', None, None, 'O'),
 ('lousy', 'ADJ', 'nsubj', None, 'O'),
 (' ', 'SPACE', None, None, 'O'),
 ('too', 'ADV', None, None, 'O'),
 ('sweet', 'ADJ', None, None, 'O'),
 ('or', 'CCONJ', None, None, 'O'),
 ('too', 'ADV', None, None, 'O'),
 ('salty', 'ADJ', None, None, 'O'),
 ('and', 'CCONJ', None, None, 'O'),
 ('the', 'DET', None, None, 'O'),
 ('portions', 'NOUN', None, 'dep', 'B'),
 ('tiny', 'ADJ', 'dep', None, 'O')]
```

Gambar 3.5 Contoh hasil keluaran praproses data

3.2.2 Tahap Pembuatan Model CRF

Pembuatan model bertujuan untuk mempersiapkan fitur – fitur yang akan digunakan oleh CRF untuk melakukan pengenalan *aspect term*. Detail fitur – fitur yang akan dipersiapkan dapat dilihat pada Tabel 3.1:

Tabel 3.1 Detail fitur yang akan digunakan pada CRF

No	Nama Fitur	Keterangan
1	*word*	String dari kata saat ini
2	*word.lower*	Lowercase string dari kata saat ini
3	*postag*	Label POS Tag dari kata saat ini

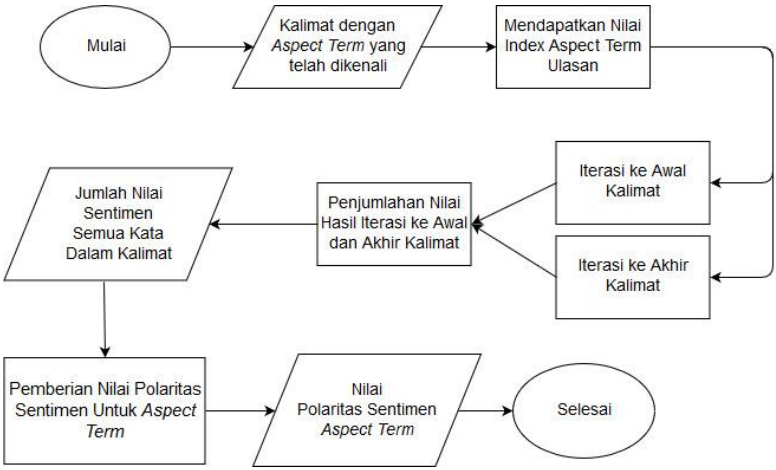
4	*word.lemmatize*	String dari kata saat ini yang sudah dilakukan proses lemmatisasi
5	*word.is_frequent_term*	Menandakan apakah kata saat ini merupakan *aspect term* yang sering muncul (*frequent aspect term)* atau tidak. *Frequent aspect term* adalah *aspect term* yang muncul sebagai target opini sebanyak lebih dari 4 kali pada data latih.
6	*word.is_stopword*	Menandakan apakah kata saat ini merupakan *stopword*.
7	*word.headword*	String *headword* dari kata saat ini
8	*word.headword_postag*	Label POS Tag dari *headword* kata saat ini
9	*word.prefix*	Prefix dari kata saat ini (3 huruf pertama)
10	*word.suffix*	Suffix dari kata saat ini (3 huruf terakhir)
11	*governor_relation*	Nama relasi yang muncul ketika kata saat ini bersifat sebagai *governor* dalam kalimat
12	*dependant_relation*	Nama relasi yang muncul ketika kata saat ini bersifat sebagai *governor* dalam kalimat

13	*-2:word.lower*	Lowercase string dari kata yang berada di jarak 2 kata sebelum kata saat ini
14	*-2:postag*	Label POS Tag dari kata yang berada di jarak 2 kata sebelum kata saat ini
15	*-1:word.lower*	Lowercase string dari kata yang berada persis sebelum kata saat ini
16	*-1:postag*	Label POS Tag dari kata yang berada persis sebelum kata saat ini
17	*+2:word.lower*	Lowercase string dari kata yang berada di jarak 2 kata setelah kata saat ini
18	*+2:postag*	Label POS Tag dari kata yang berada di jarak 2 kata setelah kata saat ini
19	*+1:word.lower*	Lowercase string dari kata yang berada persis setelah kata saat ini
20	*+1:postag*	Label POS Tag dari kata yang berada persis setelah kata saat ini
21	*BOS*	*Beginning of Sentence* Kata saat ini merupakan kata pertama dalam sebuah kalimat
22	*EOS*	*End of Sentence* Kata saat ini merupakan kata terakhir dalam sebuah kalimat

3.2.3 Tahap Pengenalan Aspect Term

Setelah melakukan praproses data dan mempersiapkan fitur – fitur yang akan digunakan, proses selanjutnya adalah membangun model CRF dengan memanfaatkan fitur – fitur yang diambil dari data latih. Kemudian, model yang terbentuk akan digunakan untuk melakukan pengenalan *aspect term* pada data uji. Lalu pada akhir tahap pengenalan *aspect term*, akan dilakukan pengujian untuk mendapatkan nilai akurasi, *precision* dan *recall*.

3.2.4 Tahap Analisis Polaritas Sentimen



Gambar 3.6 Diagram alir keseluruhan proses analisis sentimen dengan metode *rule-based*

Proses ini akan menerima *aspect term* yang telah dikenali pada data uji kemudian menentukan label sentimen yang sesuai untuk *aspect term* tersebut. Pada tugas akhir ini, proses analisis polaritas sentimen dilakukan dengan metode *rule-based*. Sehingga tidak diperlukan proses pelatihan dan data latih untuk membuat model yang nantinya akan digunakan untuk menentukan label

sentimen yang sesuai untuk *aspect term* tertentu. Diagram alir proses analisis sentimen dengan metode *rule-based* dapat dilihat pada Gambar 3.6

Untuk setiap data kalimat yang diperoleh, terlebih dahulu diambil *aspect term* yang telah dikenali oleh CRF pada kalimat tersebut beserta lokasi (*index*) nya di dalam kalimat. Kemudian, dilakukan iterasi ke kiri dan ke kanan yang dimulai dari posisi *aspect term* tersebut. Iterasi ke kiri dilakukan sampai ke awal kalimat dan iterasi ke kanan dilakukan sampai ke akhir kalimat.

Pada setiap iterasi dilakukan proses mendapatkan nilai sentimen kata dengan tahapan sebagai berikut:

1) Dilakukan pengambilan nilai sentimen kata dengan menjumlahkan nilai yang berasal dari 4 *lexicon* yang digunakan. Penjumlahan nilai dari 4 *lexicon* ini akan menghasilkan nilai yang berada pada rentang -4 dan 4 ($-4 \leq t_i \leq 4$, dimana t_i adalah jumlah nilai sentimen untuk kata ke i) [17]. Pengambilan nilai dari setiap lexicon dilakukan dengan ketentuan sebagai berikut:
    - Bing Liu Opinion Lexicon : Apabila kata yang ditemui merupakan kata positif kembalikan nilai 1, jika negatif kembalikan nilai -1
    - General Inquirer Opinion Lexicon : Apabila kata yang ditemui merupakan kata positif kembalikan nilai 1, jika negatif kembalikan nilai -1
    - MPQA Opinion Lexicon : Kembalikan nilai yang sesuai dengan ketentuan pada Tabel 3.2 pada kondisi atribut kata yang sesuai.
    - SentiWordNet : Kembalikan nilai positif apabila nilai positif lebih besar dari nilai negatif dan nilai objektif, kembalikan nilai negatif apabila nilai negatif lebih besar dari nilai positif dan nilai objektif. Kembalikan nilai 0 jika kedua kondisi tersebut tidak dipenuhi.

Tabel 3.2 Ketentuan pengambilan nilai pada MPQA Opinion Lexicon

Polarity	**Subjectivity**	**Nilai**
positive	strongsubj	1
positive	weaksubj	0.5
negative	strongsubj	-1
negative	weaksubj	-0.5

2) Untuk setiap kata yang ditemui, lakukan pemeriksaan keberadaan kata negasi dalam jarak 3 token. Jika ada, maka nilai sentimen yang didapatkan dari tahap 1 dikalikan dengan -1. Keberadaan kata negasi dalam Bahasa Inggris yang diperiksa adalah *not, no, never* dan '*nt*.

3) Lakukan pengambilan nilai jarak antara kata saat ini dengan kata yang menjadi *aspect term*. Nilai jarak ini nantinya akan menjadi bobot bagi nilai sentimen kata saat ini, karena tidak semua kata dalam kalimat dapat mempengaruhi nilai sentimen dari *aspect term* pada tingkat yang sama [17]. Nilai jarak didapatkan dari penjumlahan nilai Token Distance dan Dependency Distance dari kata saat ini ke kata yang menjadi *aspect term*.

4) Nilai sentimen akhir untuk kata saat ini didapatkan dengan melakukan pembagian antara jumlah nilai sentimen dari 4 *lexicon* yang digunakan dengan jumlah nilai jarak yang telah didapatkan sebelumnya.

Semua nilai sentimen yang didapatkan untuk setiap kata pada proses iterasi kiri dan kanan kemudian dijumlahkan dan digunakan untuk menentukan nilai polaritas sentimen untuk *aspect term* terkait dengan ketentuan sebagai berikut:

- Apabila nilai akhir sentimen lebih besar dari 0, maka polaritas sentimen positif
- Apabila nilai akhir sentimen sama dengan 0, maka polaritas sentimen netral
- Apabila nilai akhir sentimen kurang dari 0, maka polaritas sentimen negatif

Pada akhir proses anlisa polaritas sentimen dilakukan proses pengujian untuk mengevaluasi seberapa baik metode yang diimplementasikan dapat mengenali polaritas sentimen untuk setiap *aspect term* dengan mendapatkan nilai akurasi, *precision* dan *recall*.

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Bab ini menjelaskan mengenai implementasi perangkat lunak dari rancangan sistem yang telah dibahas pada Bab 3 meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran, dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

4.1 Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir ini memiliki spesifikasi perangkat keras dan perangkat lunak yang ditunjukkan oleh Tabel 4.1.

Tabel 4.1 Spesifikasi perangkat

Perangkat	Spesifikasi
Perangkat Keras	• Prosesor: Intel® Core™ i5 5200U CPU @ 2.7GHz • Memori : 8 GB
Perangkat Lunak	• Sistem Operasi: Windows 10 Education 64-bit • Perangkat pengembang: Anaconda 3.4 • Bahasa Pemrograman: Python 3.6 • Pustaka: Spacy, *pycrfsuite*, Standford Dependency Parser, NetworkX dan Scikit-Learn

4.2 Implementasi Praproses Data

Pada subbab ini akan dijabarkan implementasi pada tahap praproses data, yaitu melakukan proses *dependency parsing*, ekstraksi relasi *governor* dan *dependent*, tokenisasi, pemberian label POS Tag dan pemberian label BIO.

4.2.1 Implementasi *Dependency Parsing*

Proses *dependency parsing* diimplementasikan pada Kode Sumber 4.1. Proses dilakukan pada teks dengan memanfaatkan fungsi dari *library* Standford Dependency Parser. Proses diawali dengan membuat pada baris ke-1 dengan membuat *instance* dari kelas *StandfordDependencyParser* dengan parameter *path* ke berkas *executables* dan model dari *library StandfordDependencyParser*. Kemudian proses Dependency Parsing dilakukan pada baris ke-4 dengan memanggil fungsi *raw_parse()* pada *instance* yang telah dibuat dengan parameter kalimat yang akan diproses.

```
1  parser = stanford.StanfordDependencyParser(
2    path_to_jar=jar_path, path_to_models_jar=model_path
3  )
4  sentences = parser.raw_parse(sentence)
5  dep = sentences.__next__()
6  parsed = list(dep.triples())
```

Kode Sumber 4.1 Fungsi *dependency parsing*

Hasil proses *Dependency Parsing* akan menghasilkan 3 nilai keluaran, yaitu kata *governor*, nama relasi dan kata *dependent*. 3 keluaran tersbut dapat diambil dengan memanggil fungsi *triples()* yang diimplementasikan pada baris ke-6.

4.2.2 Implementasi Ekstraksi Relasi *Governor* dan *Dependent*

```
1  def get_governor_relation(tokenized, results):
2      needed_relation = ["amod", "nsubj", "dep"]
3      rel = {}
4      for token in tokenized:
5          for data in results:
6              if data[0][0] == token and data[1] in needed_relation:
7                  if token in rel:
8                      if data[1] not in rel[token].split(";"):
9                          rel[token] += ";" + data[1]
10                 else:
11                     rel[token] = data[1]
12     return rel
```

Kode Sumber 4.2 Fungsi ekstraksi *governor* relation

```
1  def get_dependent_relation(tokenized, results):
2      needed_relation = ["dobj", "nsubj", "dep"]
3      rel = {}
4      for token in tokenized:
5          for data in results:
6              if data[0][0] == token and data[1] in needed_relation:
7                  if token in rel:
8                      if data[1] not in rel[token].split(";"):
9                          rel[token] += ";" + data[1]
10                 else:
11                     rel[token] = data[1]
12     return rel
```

Kode Sumber 4.3 Fungsi ekstraksi relasi *dependent*

Proses ekstraksi relasi *governor* dan *dependent* diimplementasikan pada Kode Sumber 4.2 dan Kode Sumber 4.3. Proses dilakukan dengan memanfaatkan hasil dari proses *dependency parsing* dan tokenisasi. Masing – masing fungsi ini menerima 2 parameter dengan penjelasan sebagai berikut:

1. Parameter *tokenized* adalah hasil proses tokenisasi yang akan digunakan sebagai acuan kata apa saja yang terdapat pada kalimat
2. Parameter *results* adalah hasil dari *dependency parsing* yang menyimpan relasi kata yang ada pada kalimat.

Pada baris ke-2 dilakukan deklarasi variabel *needed_relation* yang menyimpan relasi kata yang akan diambil untuk masing – masing kata *governor* dan *dependent*. Pada baris ke-4 dilakukan iterasi untuk setiap kata yang ada pada hasil tokenisasi. Pada baris 5-11 dilakukan iterasi pada hasil dari proses *dependency parsing*. Pada fungsi *get_governor_relation* hal ini dilakukan untuk mengambil relasi yang ada ketika suatu kata bertindak sebagai *governor*, pada fungsi *get_dependent_relation* hal ini dilakukan untuk mengambil relasi yang ada ketika suatu kata bertindak sebagai *dependent*. Pada baris ke-6 dilakukan pemeriksaan apakah relasi yang ada untuk kata saat ini merupakan salah satu relasi yang dibutuhkan. Apabila tidak maka fungsi akan melanjutkan iterasi. Apabila iya maka fungsi akan menyimpan relasi tersebut kedalam variabel *rel*. Baris 8-9 dilakukan apabila variabel *rel* telah menyimpan relasi untuk kata saat ini untuk menghindari duplikasi. Jika terdapat relasi lain untuk kata yang sudah disimpan didalam variabel *rel* maka relasi tersebut akan disimpan sebagai sebagai *value* dari variabel *rel* untuk *key* dari kata yang sama.

4.2.3 Implementasi Proses Tokenisasi dan Pemberian Label POS Tag

Proses tokenisasi dan pemberian label POS Tag dilakukan dengan memanfaatkan *library* Spacy yang diimplementasikan pada Kode Sumber 4.4 melalui fungsi *tokenize_and_pos_tagging*. Fungsi ini menerima parameter *sentence* yang menyimpan data kalimat yang akan diproses. Pada baris ke-2 dilakukan proses *loading* model Bahasa Inggris untuk *library* Spacy. Pada baris ke-3 dilakukan proses pengolahan kalimat yang tersimpan pada variabel *sentence* menggunakan *library* Spacy. Pada baris 7-9 dilakukan penyimpanan hasil tokenisasi dan label POS Tag yang sesuai kedalam variabel *tokens* dan *pos_labels*. Kata dan label POS Tag yang sesuai masing – masing kata disimpan pada atribut *orth_* dan *pos_* .

```
1.  def tokenize_and_pos_tagging(sentence):
2.      nlp = spacy.load("en")
3.      docs = nlp(sentence)
4.      tokens = []
5.      pos_labels = []
6.
7.      for doc in docs:
8.          tokens.append(doc.orth_)
9.          pos_labels.append(doc.pos_)
10.     return tokens, pos_labels
```

Kode Sumber 4.4 Fungsi tokenisasi dan pemberian label POS Tag

4.2.4 Implementasi Proses Pemberian Label BIO

Proses pemberian label BIO dilakukan dengan memanfaatkan hasil dari proses tokenisasi dan atribut *target* pada dataset. Proses ini bertujuan untuk memberi label apakah kata pada token merupakan bagian dari suatu *aspect term* atau bukan.

Implementasi dari proses ini dapat dilihat pada Kode Sumber 4.5. Fungsi ini menerima 3 parameter, berikut penjelasan parameter – parameter tersebut:

1. Parameter *word* adalah kata yang akan diberi label
2. Parameter *prev_label* adalah label BIO yang diberikan pada kata sebelumnya
3. Parameter *list_of_targets* adalah daftar kata yang menjadi target ulasan (*aspect term*) pada kalimat saat ini.

Pada baris 2-10 dilakukan iterasi untuk memeriksa apakah kata saat ini adalah bagian dari *aspect term* dan memberikan label yang sesuai. Pada baris 3 dilakukan pemeriksaan jika kata tersebut merupakan bagian dari *aspect term*. Pada baris ke-4 dilakukan pemeriksaan apakah kata saat ini merupakan kata pertama dalam *aspect term*, maka berikan label B. Jika kata tersebut bukan merupakan kata pertama dalam *aspect term* dan didahului oleh label B atau I, maka berikan label I. Jika bukan bagian dari *aspect term* maka berikan label O.

```
1.  def get_word_label(word, prev_label, list_of_targets):
2.      for targets in list_of_targets:
3.          if word in targets:
4.              if targets.index(word) > 0:
5.                  if prev_label == "B" or prev_label == "I":
6.                      return "I"
7.              elif targets.index(word) == 0:
8.                  return "B"
9.      else:
10.         return "O"
```

Kode Sumber 4.5 Fungsi pemberian label BIO pada kata

Pemanggilan fungsi pemberian label BIO pada kata untuk sebuah kalimat diimplementasikan pada Kode Sumber 4.6. Fungsi ini menerima 2 parameter, yaitu *data* sebagai satu data dari dataset

yang menyimpan kalimat, *aspect term* dan nilai sentimen untuk *aspect term* pada kalimat tersebut dan *tokens* sebagai kalimat yang sudah melalui proses tokenisasi.

```
1.  def entity_labelling(data, tokens):
2.      targets = get_target(data["target"])
3.
4.      prev_label = “”
5.      sentence = data["sentence"].strip("\n")
6.
7.      for token in tokens:
8.          label = get_word_label(token,prev_label, targets)
9.          prev_label = label
10.         words.append((token, label))
11.
12.      return words
```

Kode Sumber 4.6 Pemanggilan fungsi pemberian label BIO

Pada baris 2 dilakukan pengambilan *aspect term* kalimat dari dataset. Pada baris 7-10 dilakukan iterasi untuk setiap kata dari hasil tokenisasi yang akan akan diberi label BIO yang sesuai melalui pemanggilan fungsi *get_word_label*.

4.2.5 Implementasi Penyimpanan Data ke Dalam Satu Array

Untuk memudahkan penggunaan data yang dihasilkan untuk setiap kata dari tahap praproses pada tahap selanjutnya, maka hasil dari setiap proses disimpan pada sebuah *array*. Proses ini dilakukan melalui fungsi *labelling* yang diimplementasikan pada Kode Sumber 4.7. Fungsi labelling menggunakan 2 parameter, berikut adalah penjelasan parameter – parameter tersebut:

1. Parameter *data* adalah dataset yang saat ini diproses
2. Parameter *dependency_parsing_result* adalah hasil dari proses *dependency parsing* untuk kalimat pada dataset tersebut.

Pada fungsi *labelling* dilakukan pemanggilan fungsi – fungsi terkait untuk menjalankan tahap – tahap praproses dan menyimpan hasilnya. Detail pemanggilan fungsi – fungsi tersebut adalah sebagai berikut:

1. Pada baris ke-3 dilakukan pemanggilan fungsi *tokenize_and_pos_tagging()* untuk melakukan proses tokenisasi dan pemberian label POS Tag
2. Pada baris ke-4 dilakukan pemanggilan fungsi *entity_labelling()* untuk melakukan proses pemberian label BIO pada setiap kata pada kalimat
3. Pada baris ke-7 dilakukan pemanggilan fungsi *get_governor_relation()* untuk mendapatkan relasi grammatikal yang ada ketika kata bersifat sebagai *governor*.
4. Pada baris ke-9 dilakukan pemanggilan fungsi *get_dependent_relation()* untuk mendapatkan relasi grammatikal yang ada ketika kata bersifat sebagai *dependent*.
5. Pada baris ke-13 dilakukan iterasi untuk setiap data untuk kata yang ada pada variabel *entity_labelling_result*, *tokenized* dan *pos_tags* untuk disimpan ke dalam array *labelling_result.*

```
1.  def labelling(data, dependency_parsing_result):
2.      labelling_result = []
3.      tokens, pos_tags = tokenize_and_pos_tagging(data["sentence"])
4.      entity_labelling_results = entity_labelling(data)
5.
6.      governor_relation_dict = get_governor_relation(
```

```
7.          tokenized, dependency_parsing_result
8.      )
9.      dependent_relation_dict = get_dependent_relation(
10.         tokenized, dependency_parsing_result
11.     )
12.
13.     for result, token, pos_tag in zip(
14.             entity_labelling_results, tokens, pos_tags
15.         ):
16.
17.         bio_label = result[1]
18.         if token in governor_relation_dict:
19.             governor_relation = governor_relation_dict[token]
20.         else:
21.             governor_relation = None
22.
23.         if token in dependent_relation_dict:
24.             dependent_relation = dependent_relation_dict[token]
25.         else:
26.             dependent_relation = None
27.
28.         labelling_result.append(
29.             (token, pos_tag,
30.              governor_relation, dependent_relation,
31.              bio_label)
32.         )
33.     return labelling_result
```

Kode Sumber 4.7 Fungsi penyimpanan data ke dalam satu array

4.3 Implementasi Pembuatan Model CRF

Pada subbab ini akan dijelaskan proses ekstraksi fitur yang akan digunakan oleh CRF dan pelatihan model CRF.

4.3.1 Ekstraksi Fitur

Proses ini bertujuan untuk mempersiapkan fitur – fitur yang akan digunakan oleh CRF untuk melakukan pengenalan *aspect term*. Detail fitur yang akan diekstraksi dapat dilihat pada Tabel 3.1. Proses ekstraksi fitur akan dilakukan dengan melakukan iterasi untuk setiap kata yang ada pada kalimat. Untuk setiap kata yang ditemui, akan dilakukan ekstraksi untuk 3 jenis fitur. Yaitu fitur untuk kata saat ini, untuk kata yang ada di depan kata saat ini dan untuk kata yang ada di belakang kata saat ini.

Implementasi proses ekstraksi fitur untuk kata yang saat ini diperoleh dapat dilihat pada Kode Sumber 4.8.

```
features = [
        "word=" + word,
        "word.lower=" + word.lower(),
        "postag=" + postag,
        "word.lemmatize=" + lemmatized,
        "word.is_frequent_term=%s" % frequent,
        "word.is_stopword=%s" % stopword,
        "word.suffix=%s" % word[-3:],
        "word.prefix=%s" % word[0:3],
        "word.headword=" + token_headword,
        "word.headword_pos=" + headword_pos,
        "governor_relation=" + governor_relation,
        "dependent_relation=" + dependent_relation,
    ]
```

Kode Sumber 4.8 Implementasi ekstraksi fitur untuk kata pada saat ini

Pada proses ekstraksi fitur ini akan diambil fitur untuk 2 kata yang berada di depan dan 2 kata yang berada di belakang kata saat ini. Fitur yang akan diambil yaitu *string* dari kata itu sendiri dan label POS Tag dari kata tersebut. Implementasi untuk proses ekstraksi fitur kata yang berada di depan dan di belakang kata saat ini dapat dilihat pada Kode Sumber 4.9 dan Kode Sumber 4.10.

```
1.  if i - 2 > 0:
2.          word1 = doc[i - 2][0]
3.          postag1 = doc[i - 2][1]
4.          features.extend(
5.              [
6.                  "-2:word.lower=" + word1.lower(),
7.                  "-2:postag=" + postag1,
8.              ]
9.          )
10.
11. if i > 0:
12.         word1 = doc[i - 1][0]
13.         postag1 = doc[i - 1][1]
14.         features.extend(
15.             [
16.                 "-1:word.lower=" + word1.lower(),
17.                 "-1:postag=" + postag1,
18.             ]
19.         )
```

Kode Sumber 4.9 Implementasi ekstraksi fitur untuk kata yang berada di depan kata saat ini

```
1.  if i - 2 > 0:
2.          word1 = doc[i - 2][0]
3.          postag1 = doc[i - 2][1]
4.          features.extend(
5.              [
```

```
                "-2:word.lower=" + word1.lower(),
                "-2:postag=" + postag1,
            ]
        )

if i > 0:
        word1 = doc[i - 1][0]
        postag1 = doc[i - 1][1]
        features.extend(
            [
                "-1:word.lower=" + word1.lower(),
                "-1:postag=" + postag1,
            ]
        )
```

Kode Sumber 4.10 Implementasi ekstraksi fitur untuk kata yang berada di belakang kata saat ini

CRF juga akan menggunakan fitur yang menandai apakah suatu kata menjadi awal atau akhir kalimat. Apabila suatu kata menjadi awal sebuah kalimat, maka kata tersebut akan memiliki nilai "*BOS*" pada array fiturnya. Apabila suatu kata menjadi akhir sebuah kalimat, maka kata tersebut akan memiliki nilai "*EOS"* pada array fiturnya. Proses ekstraksi fitur ini dapat dilihat pada Kode Sumber 4.11.

```
if i == len(doc) - 1:
    features.append("EOS")

if i == 0:
    features.append("BOS")
```

Kode Sumber 4.11 Implementasi ekstraksi fitur BOS dan EOS

4.3.2 Pelatihan Model

Setelah melakukan ekstraksi fitur untuk setiap kata yang ada pada kalimat maka proses selanjutnya adalah membuat model CRF untuk melakukan proses klasifikasi kata dalam kalimat. Proses pelatihan akan memanfaatkan fitur – fitur kata yang telah diekstraksi dari data latih. Pada baris ke-2 dilakukan instansiasi dari kelas *Trainer*. Pada baris 4-5 dilakukan proses penyimpanan fitur untuk setiap kata beserta label yang sesuai untuk setiap kata ke dalam objek *trainer*. Pada baris ke 7 dilakukan proses pelatihan model CRF menggunakan fitur dan label dari data latih yang sudah disimpan sebelumnya. Model yang dihasilkan oleh proses pelatihan ini akan disimpan pada sebuah file yang bernama *crf.model*. Fungsi pelatihan model CRF dapat dilihat pada Kode Sumber 4.12.

```
1.  def train(X_train, y_train):
2.      trainer = pycrfsuite.Trainer()
3.
4.      for xseq, yseq in zip(X_train, y_train):
5.          trainer.append(xseq, yseq)
6.
7.      trainer.train("crf.model")
```

Kode Sumber 4.12 Fungsi pelatihan model CRF

4.4 Implementasi Tahap Pengenalan Aspect Term

Pada subbab ini akan dijelaskan proses pengenalan *aspect term* pada kalimat dan evaluasi model CRF yang dihasilkan pada proses pelatihan.

4.4.1 Pengenalan Aspect Term

Setelah dilakukan proses ekstraksi fitur dan pelatihan model *Conditional Random Field*, tahap selanjutnya adalah melakukan klasifikasi kata untuk mendeteksi *aspect term* dalam kalimat menggunakan model yang telah dilatih. Proses ini memanfaatkan fitur – fitur kata yang telah diekstraksi dari data uji.

Proses klasifikasi kata menggunakan *pycrfsuite* dilakukan dengan membuat *instance* dari class *Tagger* yang dilakukan pada baris ke-2. Setelah itu pada baris ke-4 dilakukan proses memuat model yang telah dilatih sebelumnya dengan memanggil fungsi *open*. Pada baris ke-6 dilakukan proses klasifikasi untuk setiap kata pada kalimat data uji. Fungsi klasifikasi label kata menggunakan CRF dapat dilihat pada Kode Sumber 4.13.

```
1.  def predict(X_test):
2.      tagger = pycrfsuite.Tagger("crf.model")
3.
4.      tagger.open("crf.model")
5.
6.      result = [tagger.tag(xseq) for xseq in X_test]
7.
8.      return result
```

Kode Sumber 4.13 Fungsi klasifikasi kata menggunakan CRF

4.4.2 Evaluasi Model CRF

Library pycrfsuite tidak menyediakan fungsi untuk melakukan evaluasi performa model terhadap data uji. Sehingga perlu dilakukan cara alternatif untuk melakukan evaluasi terhadap model agar mendapatkan nilai akurasi, *precision* dan *recall*.

Pycrfsuite memberikan label klasifikasi dengan format numerik yang dimulai dari 0 sampai n-1 dimana n adalah banyak

kelas. Sehingga perlu adanya perubahan format pemberian label oleh *pycrfsuite* ke format label BIO. Hal ini diimplementasikan pada Kode Sumber 4.14 baris 2 - 9. Lalu pada baris 4 – 10, dilakukan perubahan format pemberian label kelas untuk *predictions* dan *truths*. Baris ke-11 memanfaatkan fungsi *classification_report* dari *library* Scikit-learn untuk mengevaluasi nilai prediksi (*predictions)* terhadap nilai sesungguhnya (*truths)* untuk masing – masing kelas sehingga didapatkan nilai *precision* dan *recall*. Baris ke-15 memanfaatkan fungsi *confusion_matrix* dari *library* Scikit-learn untuk menampilkan persebaran jumlah data untuk tiap kelas. Baris ke-17 memanfaatkan fungsi *accuracy_score* dari *library* Scikit-learn untuk menampilkan hasil akurasi keseluruhan dari proses klasifikasi.

```
1.  def evaluate(y_test, y_pred):
2.      labels = {"B": 0, "I": 1, "O": 2}
3.
4.      predictions = np.array(
5.          [labels[tag] for row in y_pred for tag in row]
6.      )
7.      truths = np.array(
8.          [labels[tag] for row in y_test for tag in row]
9.      )
10.
11.     print(classification_report(truths, predictions,
12.                                 target_names=["B", "I", "O"])
13.         )
14.     print(confusion_matrix(truths, predictions, labels=[0, 1,
    2]))
15.
16.     print(accuracy_score(truths, predictions))
```

Kode Sumber 4.14 Fungsi evaluasi model CRF

4.5 Implementasi Analisis Polaritas Sentimen

Untuk mengimplementasikan proses analisis polaritas sentimen, terdapat tiga tahap yang harus dilakukan. Tahap pertama adalah mendapatkan nilai sentimen dari 4 *lexicon* yang digunakan untuk kata yang diperoleh, tahap kedua adalah melakukan pembobotan untuk nilai sentimen kata yang diperoleh dan tahap ketiga adalah melakukan analisis sentimen *aspect term* bedasarkan proses yang telah dijelaskan pada subbab 3.2.4. Pengambilan nilai sentimen kata dilakukan berdasarkan *rule* yang telah dijelaskan pada subbab 3.2.4.

4.5.1 Implementasi Proses Mendapatkan Nilai Sentimen Kata

Implementasi perolehan nilai sentimen kata dari masing - masing *lexicon* yang digunakan dapat dilihat pada Kode Sumber 4.15, Kode Sumber 4.17, Kode Sumber 4.18 dan Kode Sumber 4.18.

```
1.  class BingLiuLexiconValueGetter:
2.      def get_value(self, token):
3.          if token in self.positive_lex:
4.              return 1
5.          elif token in self.negative_lex:
6.              return -1
7.          else:
8.              return 0
```

Kode Sumber 4.15 Implementasi pengambilan nilai sentimen kata dari Bing Liu Opinion Lexicon

Fungsi pengambilan nilai sentimen kata dari Bing Liu Opinion Lexicon diimplementasikan pada Kode Sumber 4.15.

Fungsi *get_value* pada kelas *BingLiuLexiconValueGetter* menerima parameter *token* yang menyimpan nilai *string* dari kata yang hendak dicari nilai sentimennya. Pada baris 3-4 apabila kata tersebut termasuk kata positif, maka fungsi mengembalikan nilai 1. Pada baris 5-6, apabila kata tersebut termasuk kata negatif, maka fungsi mengembalikan nilai -1. Pada baris 7-8, apabila kata tersebut tidak terdapat pada Bing Liu Opinion Lexicon maka kembalikan nilai 0.

```
1.  class GeneralInquirerLexiconValueGetter:
2.      def get_value(self, token):
3.          try:
4.              polarity = self.lexicon[token]
5.              if polarity == "positive":
6.                  return 1
7.              elif polarity == "negative":
8.                  return -1
9.              else:
10.                 return 0
11.         except KeyError:
12.             return 0
```

Kode Sumber 4.16 Implementasi pengambilan nilai sentimen kata dari General Inquirer Opinion Lexicon

Fungsi pengambilan nilai sentimen dari General Inquirer Opinion Lexicon diimplementasikan pada Kode Sumber 4.16 melalui kelas *GeneralInquirerLexiconValueGetter*. Fungsi *get_value* menerima parameter *token* yang menyimpan nilai *string* dari kata yang hendak dicari nilai sentimennya. Pada baris ke-4 dilakukan pengambilan nilai sentimen kata dari General Inquirer Sentiment Lexicon. Baris 5-10 dilakukan pemeriksaan dan pengembalian nilai yang sesuai dengan nilai sentimen dari kata tersebut. Apabila kata bernilai positif maka fungsi mengembalikan

nilai 1, apabila kata bernilai negatif maka fungsi mengembalikan nilai -1, Apabila kata bernilai netral maka fungsi mengembalikan nilai 0. Pada baris 11-12 dilakukan pemeriksaan, apabila kata tersebut tidak terdapat pada *lexicon* yang digunakan, maka kembalikan nilai 0.

```
1.  class MPQALexiconValueGetter:
2.      def get_value(self, word):
3.          try:
4.              subjectivity = self.lexicon[word]["subjectivity"]
5.              polarity = self.lexicon[word]["polarity"]
6.              if subjectivity == "strongsubj":
7.                  if polarity == "positive":
8.                      return 1
9.                  elif polarity == "negative":
10.                     return -1
11.                 else:
12.                     return 0
13.             elif subjectivity == "weaksubj":
14.                 if polarity == "positive":
15.                     return 0.5
16.                 elif polarity == "negative":
17.                     return -0.5
18.                 else:
19.                     return 0
20.         except KeyError:
21.             return 0
```

Kode Sumber 4.17 Implementasi pengambilan nilai sentimen kata dari MPQA Opinion Lexicon

Fungsi pengambilan nilai sentimen dari MPQA Opinion Lexicon diimplementasikan pada Kode Sumber 4.17 melalui kelas *MPQALexiconValueGetter*. Fungsi *get_value* menerima parameter *word* yang menyimpan nilai *string* dari kata yang hendak dicari nilai sentimennya. Pada baris 4-5 dilakukan pengambilan nilai *subjectivity* dan *polarity* kata dari MPQA Sentiment Lexicon. Baris 6-19 dilakukan pemeriksaan dan pengembalian nilai yang sesuai dengan nilai sentimen dari kata tersebut sesuai dengan ketentuan yang telah dijelaskan pada Tabel 3.2. Jika *subjectivity* bernilai *"strongsubj"* dan *polarity* bernilai *"positive"* maka kembalikan nilai 1, apabila *polarity* bernilai *"negative"* kembalikan nilai -1, selain itu kembalikan nilai 0. Jika *subjectivity* bernilai *"weaksubj"* dan *polarity* bernilai *"positive"* maka kembalikan nilai 0.5, apabila *polarity* bernilai *"negative"* kembalikan nilai -0.5, selain itu kembalikan nilai 0.

```
1.  class SentiWordNetValueGetter:
2.   def analyze_value(self, word, tag):
3.          synsets = wn.synsets(word, pos=tag)
4.          if not synsets:
5.              return 0
6.          swn_synset=swn.senti_synset(synsets[0].name())
7.          pos_score = swn_synset.pos_score()
8.          neg_score = swn_synset.neg_score()
9.          obj_score = swn_synset.obj_score()
10.         if pos_score > neg_score
                    and pos_score >= obj_score:
11.             return pos_score
12.         elif neg_score > pos_score
                    and neg_score >= obj_score:
13.             return neg_score * -1
```

Kode Sumber 4.18 Implementasi pengambilan nilai sentimen kata dari SentiWordNet

Fungsi pengambilan nilai sentimen dari SentiWordNet diimplementasikan pada Kode Sumber 4.18 Implementasi pengambilan nilai sentimen kata dari SentiWordNetmelalui kelas *SentiWordNetValueGetter*. Fungsi *analyze_value* menerima 2 buah parameter. Parameter *word* menyimpan nilai *string* dari kata yang hendak dicari nilai sentimennya. Parameter *tag* menyimpan label POS Tagging kata tersebut. Pada baris ke-3 dilakukan pengambilan nilai kata dari SentiWordNet. Pada baris ke-6 dilakukan pengambilan nilai *synset* pertama, yaitu nilai yang paling umum digunakan. Pada baris 7-9 dilakukan pengambilan nilai nilai positif (*pos_score*), nilai negatif (*neg_score*) dan nilai objektivitas (*obj_score*) untuk kata yang diterima. Apabila *pos_score* lebih besar dari *neg_score* dan *obj_score*, maka fungsi mengembalikan nilai *pos_score*. Apabila *neg_score* lebih besar dari *pos_score* dan *obj_score*, maka fungsi mengembalikan nilai *neg_score* dikali dengan -1 agar menjadi nilai negatif.

4 *class* ValueGetter yang telah diimplementasikan kemudian dipanggil oleh fungsi *get_lexicon_value_sum* yang diimplementasikan pada Kode Sumber 4.19. Fungsi ini akan mengembalikan hasil penjumlahan nilai sentimen kata dari 4 *lexicon* yang digunakan. Fungsi ini menerima 2 parameter, berikut adalah penjelasan dari parameter – parameter tersebut:

1. Parameter *opinion_word* adalah kata yang hendak dicari jumlah nilai sentimennya
2. Parameter *token_data* adalah *variable* yang menyimpan label POS Tag dari *opinion_word*

```
1.  def get_lexicon_value_sum(self, opinion_word, token_data):
2.      gi_lexicon_value = self.gi_lexicon_value_getter
3.                               .get_value(opinion_word.lower())
4.
5.      mpqa_value = self.mpqa_value_getter
6.                           .get_value(opinion_word.lower())
7.
```

```
8.      sentiwordnet_value = self.sentiwordnet_value_getter
9.                           .get_value(token_data)
10.
11.     bing_liu_value = self.bing_liu_value_getter
12.                          .get_value(opinion_word.lower())
13.
14.     total_sum = sentiwordnet_value
15.                 + gi_lexicon_value
16.                 + mpqa_value
17.                 + bing_liu_value
18.
19.     return total_sum
```

Kode Sumber 4.19 Fungsi pengambilan total nilai sentimen untuk kata

4.5.2 Implementasi Proses Mendapatkan Nilai Jarak

Setelah didapatkan total nilai sentimen untuk sebuah kata, maka pada tahap selanjutnya dilakukan pembobotan untuk menentukan pengaruh nilai sentimen dari kata tersebut terhadap nilai sentimen *aspect term*. Pembobotan dilakukan menggunakan nilai jarak dari kata tersebut ke *aspect term*. Nilai jarak yang digunakan adalah Token Distance dan Dependency Path Distance.

Implementasi untuk mendapatkan Token Distance dapat dilihat pada Kode Sumber 4.20. Fungsi *get_distance_value* pada *class* TokenDistanceGetter menerima 2 parameter, parameter *target_idx* sebagai nilai index dari *aspect_term* dalam kalimat dan parameter *opinion_idx* sebagai nilai index dari kata saat ini. Nilai Token Distance didapatkan melalui nilai absolut dari selisih nilai *index* kata dan nilai *index aspect term*. Pada baris 5-8 dilakukan pemeriksaan nilai yang akan dikembalikan. Apabila nilai Token Distance sama dengan 0, maka akan dikembalikan nilai 1 untuk menghindari terjadinya *error* karena pembagian dengan 0.

```
1.  class TokenDistanceGetter:
2.      def get_distance_value(self,
                               target_idx,
                               opinion_idx):
3.
4.          value = abs(opinion_idx - target_idx)
5.
6.          if value == 0:
7.              return 1
8.          else:
9.              return value
```

Kode Sumber 4.20 Implementasi untuk mendapatkan nilai Token Distance

Implementasi untuk mendapatkan Dependency Path Distance dapat dilihat pada Kode Sumber 4.21. Fungsi *get_distance_value* menerima 5 parameter, berikut adalah penjelasan dari parameter – parameter tersebut:

1. Parameter *source* adalah kata yang menjadi *aspect term* pada kalimat tersebut
2. Parameter *source_idx* adalah nilai *index aspect term* dalam kalimat
3. Parameter *target* adalah kata yang saat ini ditemui pada proses iterasi
4. Parameter *target_idx* adalah nilai *index* kata pada parameter *target* dalam kalimat
5. Parameter *sentence_idx* adalah nilai *index* kalimat tersebut pada *dataset*

Kode Sumber 4.21 memanfaatkan *library* NetworkX untuk menyimpan dan mengolah hasil *dependency parsing* dalam struktur data *graph*. Sehingga dapat memanfaatkan fungsi yang ada pada *library* NetworkX untuk melakukan operasi *graph* yang dibutuhkan tanpa harus melakukan implementasi secara manual.

Dalam hal ini, operasi yang dibutuhkan adalah mencari nilai *shortest path* dari 2 kata pada *dependency graph*.

Pada baris ke-9 dilakukan instansiasi *object* dari *class* Graph dengan parameter hasil *dependency parsing* kalimat. Pada baris 11-12 dilakukan pembuatan *query* untuk melakukan pencarian nilai *shortest path*. Format *query* yang dibuat adalah : "[kata]-[nilai index kata pada kalimat]". Pada baris ke-15 dilakukan pemanggilan fungsi *shortest_path_length* dengan 3 parameter untuk mendapatkan nilai *shortest path* dari kata pada parameter *source* ke kata pada paremeter *target*. Fungsi *shortest_path_length* menerima 3 parameter, berikut adalah penjelasan dari parameter – parameter tersebut:

1. Parameter *graph* adalah objek dari *class* Graph yang telah dibuat pada baris 9
2. Parameter *source* adalah *query string* untuk menentukan node mulai pada *graph*
3. Parameter *target* adalah *query string* untuk menentukan node tujuan pada *graph*

```
1.  class DependencyPathDistanceGetter:
2.      def get_distance_value(self,
                               source,
                               source_idx,
                               target, target_idx, sentence_idx):
3.      graph = nx.Graph(self.edges_list[sentence_idx])
4.      source += "-" + str(source_idx)
5.      target += "-" + str(target_idx)
6.
7.          try:
8.              return nx.shortest_path_length(graph,
                                               source=source,
                                               target=target)
9.          except nx.exception.NetworkXNoPath:
```

```
10.             return 1
11.         except nx.exception.NodeNotFound:
12.             return 1
```

Kode Sumber 4.21 Implementasi untuk mendapatkan nilai Dependency Path Distance

4.5.3 Implementasi Analisis Sentimen *Aspect Term*

Setelah mengimplementasikan fungsi untuk mendapatkan nilai sentimen dan jarak untuk setiap kata, maka proses selanjutnya adalah melakukan analisis sentimen *aspect term* dengan metode *rule-based*. Proses ini diimplementasikan pada fungsi *analyze* yang dapat dilihat pada Kode Sumber 4.22. Fungsi ini menerima 2 parameter, parameter *doc* adalah data kalimat yang akan dianalisis dan sudah melalui proses pengenalan *aspect term* dan parameter *doc_idx* adalah nilai index data kalimat tersebut pada *dataset*.

Pada baris 8 – 24 dilakukan iterasi untuk setiap kata yang ada pada kalimat. Untuk setiap kata yang ditemui, dilakukan pemeriksaan label BIO yang dimiliki oleh kata tersebut. Jika kata tersebut adalah bagian dari *aspect term* yang ditandai dengan label B, maka dilakukan proses analisis polaritas sentimen untuk kata tersebut. Pada baris 10 dilakukan pemanggilan fungsi *analyze_backward* untuk mendapatkan jumlah nilai sentimen semua kata yang berada di sebelah kiri *aspect term*. Pada baris 11 dilakukan pemanggilan fungsi *analyze_forward* untuk mendapatkan jumlah nilai sentimen untuk semua kata yang berada di sebelah kanan *aspect term*. Pada baris 12 dilakukan penjumlahan nilai yang dikembalikan oleh fungsi *analyze_backward* dan *analyze_forward* yang disimpan pada variabel *total_polarity_value*. Berdasarkan nilai pada variabel *total_polarity_value*, pada baris 14-19 dilakukan proses penentuan nilai polaritas yang sesuai untuk *aspect_term* yang saat ini

didapatkan. Apabila bernilai positif, maka diberikan polaritas sentimen positif. Apabila bernilai negatif, maka diberikan polaritas sentimen negatif. Apabila bernilai 0, maka diberikan polaritas sentimen netral.

```
1.  def analyze(self, doc, doc_idx):
2.      target_counter = 0
3.
4.      doc_result = dict()
5.      temp = dict()
6.      doc_result["id"] = doc_idx
7.
8.      for idx, data in enumerate(doc):
9.          if data[2] == "B":
10.             backward_value = self.analyze_backward(
                                    doc, idx, doc_idx
                                )
11.             forward_value = self.analyze_forward(
                                    doc, idx, doc_idx
                                )
12.             total_polarity_value = backward_value
                                       + forward_value
13.
14.              if total_polarity_value > 0:
15.                 polarity = "positive"
16.              elif total_polarity_value < 0:
17.                 polarity = "negative"
18.              else:
19.                 polarity = "neutral"
20.
21.              temp[
```

```
            self.current_target_words_list[target_counter]
                    ] = polarity

                target_counter += 1
            doc_result["data"] = temp
        self.result.append(doc_result)
```

Kode Sumber 4.22 Fungsi analisis sentimen *aspect term*

Setelah didapatkan nilai polaritas sentimen untuk setiap *aspect term* pada data uji, maka tahap selanjutnya adalah melakukan evaluasi performa dari implementasi analisis polaritas sentimen menggunakan metode *rule-based*. Proses evaluasi dilakukan agar memperoleh nilai akurasi, *precision* dan *recall* dari implementasi yang telah dilakukan. Kode Sumber 4.23 menunjukan implementasi evaluasi performa analisis sentimen dengan metode *rule-based*.

Pada baris 1-5 dilakukan deklarasi beberapa variable yang dibutuhkan dengan penjelasan sebagai berikut:

1. Variabel *error_counter* untuk menyimpan banyaknya *aspect_term* yang memiliki hasil analisis sentimen yang salah
2. Variabel *missing* untuk menyimpan banyaknya *aspect_term* yang tidak dikenali oleh CRF
3. Variabel *aspect_count* untuk menyimpan banyaknya *aspect_term* yang ada pada data uji
4. Variabel array *predictions* untuk menyimpan nilai polaritas sentimen hasil dari sistem yang telah diimplementasikan
5. Variabel array *true_labels* untuk menyimpan nilai polaritas sentimen yang berasal dari *dataset*

Pada baris 7-18 dilakukan iterasi pada variabel *truths* (menyimpan pasangan *aspect term* dan nilai polaritas sentimen yang benar) dan *results* (menyimpan pasangan *aspect term* dan

nilai polaritas sentimen yang berasal dari hasil CRF dan sistem analisis polaritas sentimen yang telah diimplementasikan). Untuk setiap iterasi, dilakukan penyimpanan label polaritas sentimen ke variabel *predictions* dan *true_labels*. Apabila terjadi ketidaksamaan polaritas sentimen untuk *aspect_term* pada *dataset* dan hasil dari sistem yang telah diimplementasikan, dilakukan *increment* pada variabel *error_counter* untuk menghitung banyaknya polaritas sentimen yang tidak tepat. *KeyError exception* dapat terjadi apabila ada *aspect term* pada *dataset* yang tidak dikenali oleh CRF. Banyaknya *aspect term* yang tidak dikenali oleh CRF disimpan pada variabel *missing* yang terus dilakukan *increment* apabila terjadi *KeyError exception*.

Pada baris 22-23, dilakukan perubahan format penamaan label dari penggunaan *string* ke dalam bentuk numerik sesuai dengan nilai yang ada pada variabel *labels* pada baris ke-20. Pada baris 25-27 dilakukan pencetakan nilai dari variabel *error_counter*, *missing* dan *aspect count*. Baris ke-28 memanfaatkan fungsi *classification_report* dari *library* Scikit-learn untuk mengevaluasi nilai prediksi (*predictions)* terhadap nilai sesungguhnya (*truths)* untuk masing – masing kelas sehingga didapatkan nilai *precision* dan *recall*. Baris ke-33 memanfaatkan fungsi *confusion_matrix* dari *library* Scikit-learn untuk menampilkan persebaran jumlah data untuk tiap kelas. Baris ke-34 memanfaatkan fungsi *accuracy_score* dari *library* Scikit-learn untuk menampilkan hasil akurasi keseluruhan dari proses klasifikasi.

```
1.  for truth, result in zip(truths, results):
2.      for aspect, sentiment in truth["data"].items():
3.          aspect_count += 1
4.          try:
5.              predictions.append(result["data"][aspect])
6.              true_labels.append(sentiment)
7.
8.              if result["data"][aspect] != sentiment:
```

```
                error_counter += 1
        except KeyError:
            missing += 1
            continue

labels = {"positive": 0, "negative": 1, "neutral": 2}

predictions_np = np.array(
              [labels[label] for label in predictions]
                  )
true_labels_np = np.array(
               [labels[label] for label in true_labels]
                  )

return (
    classification_report(
                    true_labels_np,
                    predictions_np,
                    target_names=["Positive"
                                  ,"Negative"
                                  ,"Neutral"]
    ),
    confusion_matrix(
                       true_labels_np,
                       predictions_np,
                       labels=[0, 1, 2]
                       )
      ), accuracy_score(true_labels_np, predictions_np)
)
```

Kode Sumber 4.23 Implementasi evaluasi performa analisis polaritas sentimen dengan metode *rule-based*

BAB V
UJI COBA DAN EVALUASI

Bab ini akan membahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba pada tugas akhir ini adalah sebuah Laptop ASUS X455LJ. Sistem operasi yang digunakan adalah Windows 10 64-bit. Perangkat yang digunakan memiliki spesifikasi perangkat keras *processor* Intel Core i5-5200U dengan kecepatan 2,2 GHz, *Random Access Memory* (RAM) sebesar 8 GB. Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan bahasa pemograman Python 3.6 dilengkapi dengan *library* antara lain *pycrfsuite*, Spacy, StanfordDependencyParser, NetworkX, Numpy dan Scikit-learn.

5.2 Dataset

Pada tugas akhir ini, data yang digunakan adalah data *Restaurant Review ABSA-2015*. Dataset *Restaurant Review ABSA-2015* sudah melakukan pembagian data untuk melakukan proses pelatihan dan pengujian. Data uji yang akan digunakan terdiri dari sebanyak 388 data. Terdapat 3 polaritas sentimen yang akan dikenali dari data uji yaitu positif, negatif dan netral. Pembagian banyak sampel untuk masing – masing polaritas sentimen dapat dilihat pada Tabel 5.1.

Tabel 5.1 Banyak sampel untuk setiap polaritas sentimen pada data uji

Polaritas Sentimen	Banyak Sampel
Positif	266
Negatif	140
Netral	9

5.3 Uji Coba Keseluruhan Sistem

Pada subbab ini akan dijelaskan tahap – tahap pemanggilan fungsi pada sistem yang telah diimplementasikan beserta dengan contoh keluaran dari setiap proses yang dijalankan. Pada Tabel 5.2 dapat dilihat contoh data yang akan digunakan pada uji coba sistem secara keseluruhan.

Tabel 5.2 Contoh data yang akan pada uji coba sistem secara keseluruhan

Id Kalimat	Kalimat
361	Here the hot dog is amazingly delicious
370	The staff was very rude and the wait for the table is ineffective.

Tahap pertama pada keseluruhan sistem adalah melakukan praproses terhadap data latih dan data uji. Tahap praproses bertujuan untuk mempersiapkan data untuk dilakukan proses ektraksi fitur. Tahap praproses meliputi proses tokenisasi, pemberian label POS Tagging, melakukan proses *dependency parsing* dan melakukan pemberian label BIO untuk menandai *aspect term* pada setiap kalimat. Hasil dari tahap praproses disimpah dengan format nilai *string* kata, label POS Tagging untuk kata tersebut, relasi *governor* kata, relasi *dependent* kata dan label BIO untuk kata tersebut. Hasil dari tahap praproses dapat dilihat pada Tabel 5.3.

Tabel 5.3 Contoh hasil dari tahap praproses

Id Kalimat	Hasil Tahap Praproses
361	[('Here', 'ADV', None, 'dep', 'O'), ('the', 'DET', None, None, 'O'), ('hot', 'ADJ', None, None, 'B'), ('dog', 'NOUN', 'dep;amod', None, 'I'), ('is', 'VERB', None, None, 'O'), ('amazingly', 'ADV', None, None, 'O'), ('delicious', 'ADJ', None, None, 'O')]
370	[('The', 'DET', None, None, 'O'), ('staff', 'NOUN', None, 'nsubj', 'B'), ('was', 'VERB', None, None, 'O'), ('very', 'ADV', None, None, 'O'), ('rude', 'ADJ', 'nsubj', None, 'O'), ('and', 'CCONJ', None, None, 'O'), ('the', 'DET', None, None, 'O'), ('wait', 'NOUN', None, 'nsubj', 'B'), ('for', 'ADP', None, None, 'O'), ('the', 'DET', None, None, 'O'), ('table', 'NOUN', None, None, 'O'), ('is', 'VERB', None, None, 'O'), ('ineffective', 'ADJ', 'nsubj', None, 'O')]

Tahap selanjutnya adalah melakukan ekstraksi fitur dan membuat model CRF. Tahap ekstraksi fitur bertujuan untuk mengambil fitur – fitur yang dibutuhkan untuk membuat model CRF. Detail fitur yang akan diambil dapat dilihat pada Tabel 3.1. Hasil dari proses ektraksi fitur untuk kata "*wait*" dari contoh kalimat dengan ID 361 dapat dilihat pada Tabel 5.4. Setelah itu dilakukan proses pelatihan model CRF menggunakan fitur yang telah diekstraksi.

Tabel 5.4 Hasil ekstraksi fitur untuk kata pertama "*wait*" dari kalimat ID 361

Kata	Fitur
wait	['word=wait', 'word.lower=wait', 'postag=NOUN', 'word.lemmatize=wait', 'word.is_frequent_term=False', 'word.is_stopword=False', 'word.headword=ineffective', 'word.headword_pos=JJ', 'word.suffix=ait', 'word.prefix=wai', 'governor_relation=', 'dependent_relation=nsubj', '-2:word.lower=and', '-2:postag=CCONJ', '-1:word.lower=the', '-1:postag=DET', '+2:word.lower=the', '+2:postag=DET', '+1:word.lower=for', '+1:postag=ADP']

Kemudian dilakukan proses pengenalan *aspect term* menggunakan model CRF yang telah dihasilkan. Contoh hasil proses pengenalan *aspect term* pada contoh data yang digunakan dapat dilihat pada Tabel 5.5 dan Tabel 5.8.

Tabel 5.5 Contoh hasil pengenalan *aspect term* untuk kalimat ID 361

#361	O	O	B	I	O	O	O
	Here	the	hot	dog	is	amazingly	delicious

Setelah didapatkan *aspect term* untuk kalimat yang digunakan, maka tahap terakhir adalah melakukan analisis sentimen untuk *aspect term* yang dikenali pada kalimat. Hasil penilaian sentimen setiap kata untuk *aspect term* "*hot dog*" pada kalimat ID 361 dapat dilihat pada Tabel 5.6. Contoh hasil dari proses analisis *aspect term* untuk kalimat ID 361 dapat dilihat pada Tabel 5.7.

Tabel 5.6 Contoh hasil penilaian sentimen setiap kata untuk *aspect term* "*hot dog*" pada kalimat ID 361

#361	Here	the	is	amazingly	delicious
	0	0	0	0,5	0,917

Tabel 5.7 Contoh hasil analisis sentimen untuk *aspect term* pada kalimat ID 361

ID Kalimat : 361	
Kalimat	Here the hot dog is amazingly delicious
***Aspect term* yang dikenali**	**Sentimen yang dikenali**
hot dog	*Positive*

Pada kalimat ID 370 terdapat 2 *aspect term*, yaitu "*staff*" dan "*wait*". Hasil dari proses penilaian sentimen tiap kata pada kalimat terhadap masing – masing *aspect term* dapat dilihat pada Tabel 5.9 dan Tabel 5.10. Contoh hasil analisis polaritas sentimen untuk masing – masing *aspect term* pada kalimat ID 370 dapat dilihat pada Tabel 5.11.

Tabel 5.8 Contoh hasil pengenalan *aspect term* untuk kalimat ID 370

#370	O	B	O	O	O	O	O
	The	staff	was	very	rude	and	the

B	O	O	O	O	O
wait	for	the	table	is	ineffective

Tabel 5.9 Contoh hasil penilaian sentimen setiap kata untuk *aspect term* "*staff*" pada kalimat ID 370

#370	The	was	very	rude	and	the
	0	0	0	-1,81	0	0

for	the	table	is	ineffective
0	0	0	0	-1,04

Tabel 5.10 Contoh hasil penilaian sentimen setiap kata untuk *aspect term* "*wait*" pada kalimat ID 370

#370	The	was	very	rude	and	the
	0	0	0	-1,21	0	0

for	the	table	is	ineffective
0	0	0	0	-1,56

Tabel 5.11 Contoh hasil analisis sentimen untuk *aspect term* pada kalimat ID 370

ID Kalimat : 370	
Kalimat	The staff was very rude and the wait for the table is ineffective.

***Aspect term* yang dikenali**	**Sentimen yang dikenali**
staff	*Negative*
wait	*Negative*

5.4 Skenario Uji Coba

Proses uji coba berguna untuk menemukan parameter yang menghasilkan performa model yang paling optimal. Parameter yang tepat akan memberikan hasil yang lebih baik pada saat proses uji coba. Hasil terbaik dari suatu skenario uji coba akan digunakan untuk skenario uji coba berikutnya. Terdapat 3 macam skenario uji coba yang akan dicoba pada proses pengenalan *aspect term* dan proses analisis sentimen *aspect term*. Skenario uji coba yang akan dilakukan yaitu:

1. Uji Coba Fitur CRF
2. Uji Coba Penggunaan *Sentiment Lexicon*
3. Uji Coba Pembobotan Nilai Sentimen Kata

5.4.1 Uji Coba Fitur CRF

Uji coba pertama adalah uji coba variasi fitur yang akan digunakan oleh CRF. Uji coba ini bertujuan untuk mengetahui penggunaan fitur yang dapat memaksimalkan performa model CRF untuk mengenali *aspect term*. Terdapat 2 jenis daftar fitur yang akan digunakan untuk uji coba pada CRF, yaitu daftar fitur A yang terdiri dari 10 fitur dan daftar fitur B yang terdiri dari 22 fitur. Daftar fitur A diperoleh dari hasil penelitian yang dilakukan oleh Z. Toh dan W. Wang [17]. Sedangkan daftar fitur B diperoleh dari hasil penelitian yang dilakukan D. K. Gupta dan K. A. Reddy [24]. Detail untuk masing – masing jenis fitur dapat dilihat pada Tabel 5.12

Tabel 5.12 Daftar fitur yang akan digunakan untuk Skenario Uji Coba Fitur CRF

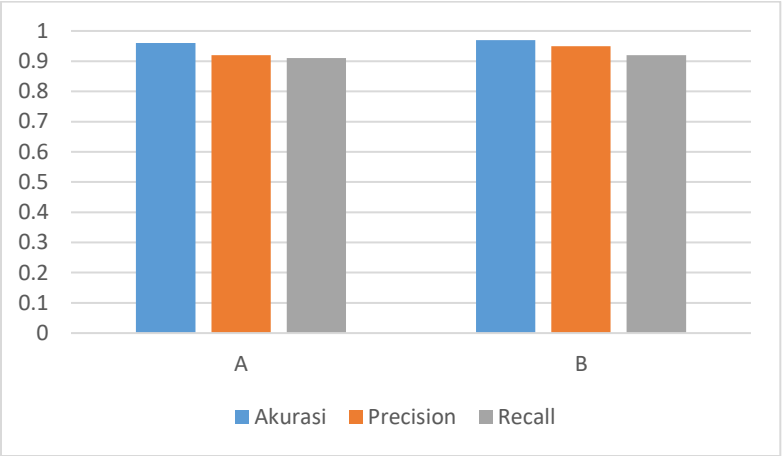
Daftar Fitur A	Daftar Fitur B
BOS	BOS
word	*word*
+1:word.lower	*postag*

-1:word.lower	*word.lower*
postag	*word.lemmatize*
word.headword	*word.is_frequent_term*
word.headword_postag	*word.is_stopword*
governor_relation	*word.headword*
dependent_relation	*word.headword_postag*
EOS	*EOS*
	word.prefix
	word.suffix
	governor_relation
	dependent_relation
	-1:word.lower
	-1:word.postag
	+1:word.lower
	+1:word.postag
	-2:word.lower
	-2:word.postag
	+2:word.lower
	+2:word.postag

Masing – masing variasi fitur akan diekstraksi dari data latih dan data uji. Fitur dari data latih akan digunakan untuk membuat model CRF dan fitur dari data uji akan digunakan pada proses pengenalan *aspect term*. Hasil penggunaan fitur yang paling baik akan dibawa ke skenario berikutnya untuk dijadikan parameter masukan.

Uji coba variasi penggunaan fitur pada CRF pembagian data masing-masing menghasilkan grafik akurasi pengujian yang dapat

dilihat pada Gambar 5.1. Perbandingan nilai akurasi, *precision* dan *recall* dapat dilihat pada Tabel 5.13.



Gambar 5.1 Grafik akurasi, *precision* dan *recall* menggunakan daftar fitur A dan B

Tabel 5.13 Perbandingan nilai akurasi, *precision* dan *recall* pada Skenario Uji Coba 1

Daftar Fitur	**Akurasi**	**Kelas**	***Precision***	***Recall***
A	96,66%	B	92%	97%
		I	85%	78%
		O	98%	99%
B	**98,07%**	**B**	**97%**	**91%**
		I	**90%**	**86%**
		O	**98%**	**99%**

Berdasarkan hasil uji coba, dapat terlihat bahwa penggunaan daftar fitur B menghasilkan performa yang lebih baik. Dengan perbedaan nilai akurasi sebesar 2%. Karena daftar fitur B menghasilkan performa yang lebih baik, maka model yang dibuat

menggunakan daftar fitur B akan digunakan pada tahap skenario uji coba berikutnya.

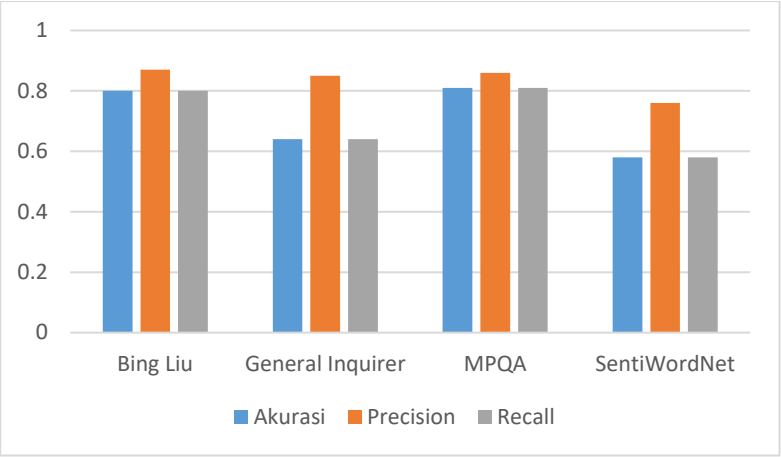
5.4.2 Uji Coba Penggunaan *Sentiment Lexicon*

Uji coba terhadap variasi penggunaan *sentiment lexicon* dilakukan untuk mengetahui kombinasi penggunaan *sentiment lexicon* yang dapat memberikan performa terbaik pada sistem yang dirancang. Pada tahap uji coba ini, pembobotan nilai kata dilakukan dengan menggunakan Token Distance dan Dependency Path Distance.

Uji coba penggunaan variasi *sentiment lexicon* akan dilakukan dengan tahapan sebagai berikut:

1. Dilakukan pengujian menggunakan masing – masing *sentiment lexicon* secara satu per satu
2. Berdasarkan hasil pengujian pada tahap 1, diambil 2 *sentiment lexicon* yang menghasilkan performa terbaik untuk dilakukan pengujian menggunakan 2 *sentiment lexicon* tersebut
3. Berdasarkan hasil pengujian dari tahap 1, diambil *sentiment lexicon* yang menghasilkan performa terbaik ketiga, kemudian digunakan bersama – sama dengan 2 *lexicon* yang telah diambil pada tahap ke-2 untuk dilakukan pengujian
4. Dilakukan pengujian menggunakan keempat *sentiment lexicon* tersebut secara bersama - sama

Hasil pengujian untuk tahap 1 menghasilkan grafik akurasi pengujian yang dapat dilihat pada Gambar 5.2. Perbandingan nilai akurasi, *precision* dan *recall* untuk penggunaan masing – masing *lexicon* dapat dilihat pada Tabel 5.14.



Gambar 5.2 Grafik hasil pengujian variasi penggunaan *sentiment lexicon*

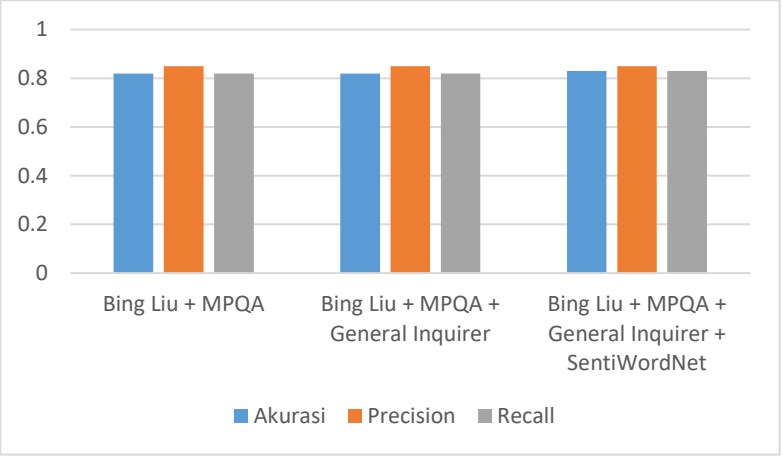
Tabel 5.14 Perbandingan nilai akurasi, precision dan recall pengujian variasi penggunaan *sentiment lexicon*

Lexicon	**Akurasi**	**Kelas**	***Precision***	***Recall***
Bing Liu	80,72%	Positif	88%	91%
		Negatif	91%	65%
		Netral	7%	30%
MPQA	**81,20%**	**Positif**	**87%**	**92%**
		Negatif	**90%**	**65%**
		Netral	**9%**	**30%**
General Inquirer	64,34%	Positif	86%	74%
		Negatif	89%	46%
		Netral	4%	50%
SentiWordNet	58,80%	Positif	81%	63%
		Negatif	72%	50%
		Netral	4%	40%

Berdasarkan hasil uji coba tahap 1, dapat diketahui bahwa penggunaan *sentiment lexicon* Bing Liu dan MPQA menghasilkan performa terbaik dengan nilai akurasi masing – masing 80,72% dan 81,20%. Diikuti oleh *lexicon* General Inquirer yang menempati peringkat 3 dengan nilai akurasi 64,34%.

Pada tahap kedua akan dilakukan pengujian dengan menggunakan *lexicon* Bing Liu dan MPQA. Kemudian akan dilanjutkan pengujian tahap ketiga, yaitu dengan menambahkan penggunaan *lexicon* General Inquirer. Terakhir akan dilakukan pengujian tahap keempat, yaitu pengujian dengan menggunakan keempat *sentiment lexicon* secara bersama – sama.

Pengujian untuk masing – masing tahap menghasilkan grafik yang dapat dilihat pada Gambar 5.3. Perbandingan nilai akurasi, *precision* dan *recall* untuk masing – masing tahap dapat dilihat pada Tabel 5.15.



Gambar 5.3 Grafik hasil pengujian kombinasi penggunaan *sentiment lexicon*

Tabel 5.15 Perbandingan nilai akurasi, precision dan recall pengujian kombinasi penggunaan *sentiment lexicon*

Lexicon	Akurasi	Kelas	Precision	Recall
Bing Liu + MPQA	82,41%	Positif	86%	92%
		Negatif	89%	67%
		Netral	12%	30%
Bing Liu + MPQA + General Inquirer	82,65%	Positif	85%	92%
		Negatif	90%	67%
		Netral	13%	30%
Bing Liu + MPQA + General Inquirer + SentiWordNet	**83,61%**	**Positif**	**86%**	**93%**
		Negatif	**87%**	**69%**
		Netral	**17%**	**30%**

Berdasarkan keseluruhan hasil skenario uji coba 2, dapat diketahui bahwa hasil performa terbaik diperoleh dengan menggunakan 4 *sentiment lexicon* secara bersamaan.

5.4.3 Uji Coba Pembobotan Nilai Sentimen Kata

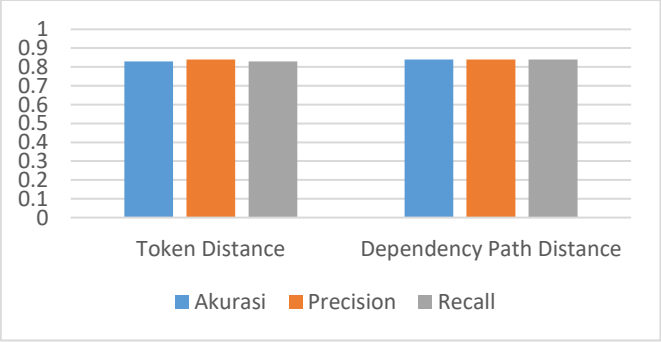
Uji coba terhadap metode pembobotan dilakukan untuk mengetahui metode pembobotan yang dapat memaksimalkan performa sistem yang telah diimplementasikan.

Skenario uji coba ini akan dilakukan dengan 2 tahap:

1. Pengujian menggunakan pembobotan Token Distance
2. Pengujian menggunakan pembobotan Dependency Path Distance

Pada tahap uji coba ini digunakan 4 *sentiment lexicon* secara bersamaan, karena berdasarkan skenario uji coba sebelumnya penggunaan 4 sentiment lexicon ini menghasilkan performa yang terbaik.

Uji coba metode pembobotan pada sistem yang telah diimplementasikan menghasilkan grafik akurasi pengujian yang dapat dilihat pada Gambar 5.4. Perbandingan nilai akurasi, *precision* dan *recall* untuk masing – masing tahap dapat dilihat pada Tabel 5.16.



Gambar 5.4 Grafik hasil pengujian dalam uji coba metode pembobotan

Tabel 5.16 Perbandingan nilai akurasi, precision dan recall pada uji coba metode pembobotan

Metode Pembobotan	**Akurasi**	**Kelas**	***Precision***	***Recall***
Token Distance	83,61%	Positif	85%	94%
		Negatif	87%	69%
		Netral	13%	20%
Dependency Path Distance	**85,06%**	**Positif**	**86%**	**93%**
		Negatif	**87%**	**71%**
		Netral	**14%**	**20%**

5.5 Hasil dan Evaluasi

Pada uji coba fitur CRF, dapat diketahui bahwa penggunaan daftar fitur B menghasilkan akurasi yang lebih tinggi dibandingkan dengan penggunaan daftar fitur A yaitu sebesar 98,07%. Hal ini terjadi karena daftar fitur B memiliki lebih banyak fitur yang menjelaskan relasi antar kata dibandingkan dengan daftar fitur A. Selain itu pada daftar fitur B terdapat fitur yang memeriksa keanggotaan suatu kata terhadap kelompok kata *stopwords* dalam Bahasa Inggris dan *frequent aspect terms* pada dataset. Sehingga model CRF dapat menggunakan informasi yang bersifat menyeluruh dan relasional pada dataset untuk menganalisis label BIO untuk suatu kata. Perbedaan bobot untuk setiap fitur pada penggunaan daftar fitur A dan B dapat dilihat pada Tabel 5.17 dan Tabel 5.18.

Nilai bobot yang terdapat Tabel 5.17 dan Tabel 5.18 menjelaskan bahwa jika nilai bobot dari suatu fitur kata terhadap suatu label target bernilai positif dan semakin tinggi maka kata dengan fitur tersebut kemungkinan akan semakin sesuai dengan label tersebut. Apabila bobot fitur bernilai negatif dan semakin kecil terhadap suatu label target, maka kata dengan fitur tersebut kemungkinan semakin tidak sesuai dengan label tersebut. Contoh,

pada Tabel 5.17, apabila suatu kata memiliki fitur “postag=DET” yang memiliki bobot 1.317998 terhadap label O, maka kemungkinan besar kata tersebut sesuai untuk memperoleh label O. Sedangkan apabila kata memiliki fitur “word=spot” yang memiliki bobot -0.667777 terhadap label B, maka kemungkinan besar kata tersebut tidak sesuai untuk memperoleh label B.

Tabel 5.17 Fitur dengan bobot tertinggi pada penggunaan daftar fitur A

Fitur	Label Target	Bobot
EOS	O	1.915851
word=service	B	1.718898
postag=DET	O	1.317998
postag=NOUN	B	1.267903
word=prices	O	1.251009
word=staff	B	1.113352
word=sushi	B	1.11253
word=atmosphere	B	1.096849
postag=CCONJ	B	-0.667777
postag=ADV	I	-0.674152
word=place	O	-0.688686
word=dishes	O	-0.693138
word=wine	O	-0.743712
word=spot	O	-0.744176
-1:word.lower=service	I	-0.744641

Pada Tabel 5.17 dapat dilihat bahwa penggunaan daftar fitur A menyebabkan model CRF lebih banyak menggunakan menggunakan informasi nilai *string* pada kata itu sendiri daripada informasi relasi antar kata untuk menentukan label yang sesuai untuk sebuah kata. Hal ini akan menjadi menyebabkan klasifikasi

label yang kurang tepat apabila ada kata yang terdapat pada data latih namun tidak terdapat pada data uji.

Tabel 5.18 Fitur dengan bobot tertinggi pada penggunaan daftar fitur B

Fitur	Label Target	Bobot
EOS	O	1.915851
postag=PROPN	B	1.168441
word.is_stopword=True	O	0.99305
-1:word.lower=the	B	0.920532
postag=NOUN	B	0.808321
postag=PRON	O	0.801654
word.is_frequent_term=False	O	0.735449
-1:word.lower=good	B	0.69584
word.is_frequent_term=False	B	-0.488053
-1:word.lower=good	I	-0.4989
dependent_relation=dobj	B	-0.514311
-1:word.lower=in	B	-0.519217
-1:postag=ADP	B	-0.52856
word.lower=seating	O	-0.537247
-2:postag=ADP	B	-0.546429

Pada Tabel 5.18 dapat terlihat pada penggunaan daftar fitur B, fitur - fitur yang bersifat menyeluruh dan menjelaskan relasi dengan kata yang lain cukup berpengaruh bagi model CRF dalam menentukan label yang sesuai untuk sebuah kata. Hal ini dibuktikan dengan fitur – fitur nilai *string* dan label POS Tagging pada kata tetangga dan keanggotaan kata pada kelompok kata *stopwords* dan *frequent aspect terms* memiliki bobot yang cukup tinggi.

Pada skenario uji coba fitur CRF, kelas kata yang mendapatkan nilai *precision* dan *recall* tertinggi adalah kelas O.

Hal ini terjadi karena pada sebuah kalimat ulasan tentu lebih banyak kata yang tidak menjadi *aspect term* daripada kata yang menjadi *aspect term*. Sehingga kelas kata O memiliki jumlah sampel lebih banyak dibandingkan kelas kata yang lain pada data latih dan data uji. Kelas kata yang mengalami salah klasifikasi terbanyak adalah pada kelas I, terlihat dari nilai *presicion* dan *recall* yang paling rendah diantara kelas kata yang lain. Hal ini disebabkan karena sedikit *aspect term* yang terdiri dari 2 kata atau lebih. Sehingga, kelas kata ini memiliki jumlah sampel yang lebih sedikit dibandingkan dengan kelas yang lain.

Pada skenario uji coba penggunaan *sentiment lexicon*, diperoleh hasil akurasi yang paling baik pada penggunaan 4 *sentiment lexicon* secara bersama – sama yaitu sebesar 83,61%. Bedasarkan data hasil pengujian pada Tabel 5.14 dan Tabel 5.15 dapat dilihat bahwa penggunaan kombinasi 2 *sentiment lexicon* atau lebih memberikan hasil akurasi yang lebih baik daripada penggunaan *sentiment lexicon* secara satu per satu secara keseluruhan. Hal ini terjadi karena ada kata yang terdapat pada satu *lexicon* tetapi tidak terdapat pada *lexicon* yang lain. Sehingga tidak semua *lexicon* dapat mengembalikan nilai sentimen untuk satu kata yang sama. Selain itu tidak semua *sentiment lexicon* menilai suatu kata bersifat positif atau negatif secara konsisten. Satu kata pada satu sentiment lexicon bisa dinilai positif namun pada sentiment lexicon yang lain bisa dinilai negatif. Sehingga dengan menggunakan lebih banyak *sentiment lexicon* secara bersama – sama dapat memberikan nilai sentimen yang lebih baik untuk sebuah kata.

Pada skenario uji coba pembobotan nilai sentimen kata, diperoleh hasil akurasi terbaik pada penggunaan metode pembobotan Dependency Path Distance yaitu sebesar 85,06%. Hal ini terjadi karena pembobotan menggunakan Token Distance sangat dipengaruhi oleh panjang kalimat. Sehingga pengaruh nilai sentimen kata – kata yang berada jauh dari *aspect term* terhadap polaritas sentimen *aspect term* tersebut akan semakin kecil walaupun secara sintaksis kata tersebut memberikan pengaruh nilai

sentimen terhadap *aspect term*. Hal ini juga terjadi ketika suatu kata berada dekat dengan *aspect term* namun secara sintaksis kata tersebut tidak memberikan pengaruh sentimen terhadap *aspect term*. Berbeda dengan pembobotan menggunakan Dependency Path Distance yang menggunakan jarak pada *dependency graph* untuk melakukan pembobotan sehingga pembobotan dilakukan bedasarkan hubungan sintaksis.

Pada proses analisis nilai polaritas sentimen untuk sebuah *aspect term* masih terdapat kesalahan. Dimana polaritas sentimen yang diberikan tidak sesuai dengan polaritas sentimen yang sebenarnya. Kesalahan ini banyak terjadi pada kalimat yang nilai polaritas sentimennya dinyatakan secara implisit. Sistem yang diimplementasikan tidak dapat mengenali polaritas sentimen dengan tepat karena sistem hanya melakukan analisis secara sintaksis.

Kesalahan analisis nilai polaritas sentimen juga banyak terjadi pada *aspect term* yang memiliki polaritas sentimen netral. Hal ini terjadi karena *rule* yang diterapkan pada sistem mengharuskan jumlah nilai sentimen semua kata yang diperoleh harus bernilai sama dengan 0 agar sebuah *aspect term* diberi label sentimen netral. Namun, karena nilai sentimen setiap kata bedasarkan 4 *sentiment lexicon* yang digunakan memiliki rentang yang sangat bervariasi, ketika nilai – nilai tersebut dijumlahkan, nilai yang dihasilkan akan sulit untuk tepat bernilai 0. Kesalahan ini juga dipengaruhi oleh komposisi kata pada kalimat dan ada atau tidaknya kata negasi pada kalimat tersebut yang dapat mempengaruhi nilai sentimen kata lainnya. Contoh kesalahan ini terjadi pada kalimat "*Our waitress wasn't mean, but not especially warm or attentive either*." dengan *aspect term* "*waitress*". Nilai

sentimen dan label POS Tagging untuk setiap kata pada kalimat tersebut ditunjukkan pada Tabel 5.19.

Tabel 5.19 Nilai sentimen setiap kata pada kalimat "*Our waitress wasn't mean, but not especially warm or attentive either*."

Our	was	nt	mean	but	not	especially
DET	VERB	ADV	ADJ	CCONJ	ADV	ADV
0	0	0	0	0	-0.625	-1

warm	or	attentive	either
ADJ	CCONJ	VERB	ADV
-3	0	3	0

Menurut *ground truth* yang terdapat pada dataset, *aspect term* "*waitress*" pada kalimat tersebut memiliki polaritas sentimen netral, tetapi sistem memberikan polaritas sentimen negatif untuk kata "*waitres*". Hal ini terjadi karena jika seluruh nilai sentimen kata pada kalimat tersebut dijumlahkan maka akan diperoleh nilai -1.625. Hal ini terjadi karena adanya kata "*not*" yang menyebabkan nilai sentimen dari kata "*especially*" dan "*warm*" dilakukan negasi yang menyebabkan jumlah nilai sentimen kata negatif pada kalimat tersebut semakin besar. Sehingga, ketika diperoleh jumlah nilai sentimen semua kata pada kalimat tersebut, sistem memberikan polaritas sentimen negatif untuk *aspect term* "*waitress*" pada kalimat tersebut.

Kalimat dengan hasil analisis polaritas sentimen netral yang benar dapat ditemukan pada contoh kalimat "*The bibimbap was average*". Sistem memberikan polaritas sentimen yang sesuai untuk *aspect term* "*bibimbap*" pada kalimat tersebut. Hal ini terjadi karena jumlah nilai sentimen semua kata pada kalimat tersebut

adalah 0. Nilai sentimen dan label POS Tagging untuk setiap kata pada kalimat tersebut dapat dilihat pada Tabel 5.20.

Tabel 5.20 Nilai sentimen setiap kata pada kalimat "*The bibimbap was average*"

The	was	average
DET	VERB	ADJ
0	0	0

Dari ketiga hasil uji coba yang telah dilakukan, pada Tabel 5.21 ditetapkan parameter optimal dari seluruh uji coba tersebut.

Tabel 5.21 Parameter optimal yang ditetapkan

Keterangan	**Parameter Optimal**
Penggunaan daftar fitur	Daftar fitur B
Penggunaan *sentiment lexicon*	Kombinasi 4 *sentiment lexicon*
Pembobotan nilai sentimen	Dependency Path Distance

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

6.1 Kesimpulan

Dalam pengerjaan Tugas Akhir ini setelah melalui tahap perancangan aplikasi, implementasi metode, serta uji coba, diperoleh kesimpulan sebagai berikut:

1. Penggunaan daftar fitur B menghasilkan akurasi yang lebih baik dibanding penggunaan daftar fitur A. Maka dipilihlah daftar fitur B yang menghasilkan akurasi sebesar 98,07%.
2. Penggunaan 4 *sentiment lexicon* secara bersama - sama menghasilkan akurasi yang lebih baik dibandingkan dengan penggunaan *sentiment lexicon* secara satu per satu dan kombinasi penggunaan 2 dan 3 *sentiment lexicon* yaitu sebesar 83,61%.
3. Penggunaan metode pembobotan Dependency Path Distance menghasilkan akurasi yang lebih baik dibandingkan dengan pembobotan menggunakan Token Distance maupun menggunakan kombinasi dari kedua metode tersebut yaitu sebesar 85,06%.
4. Sistem analisis sentimen berbasis aspek telah berhasil diimplementasikan dengan akurasi tertinggi untuk proses pengenalan *aspect term* sebesar 98,07% dan untuk proses analisis sentimen sebesar 85,06% yang didapatkan dari uji coba menggunakan daftar fitur B, kombinasi dari 4 *sentiment lexicon* dan metode pembobotan dengan Dependency Path Distance.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem analisis sentimen berbasis aspek, yaitu:

1. Melakukan eksplorasi lebih banyak fitur yang dapat digunakan oleh model CRF untuk mengenali *aspect term* dalam sebuah kalimat ulasan
2. Pengembangan sistem yang mampu melakukan analisis untuk sentimen yang bersifat implisit
3. Pengembangan sistem yang mampu melakukan analisis untuk kelas sentimen netral dengan lebih baik
4. Melakukan eksplorasi metode untuk melakukan *balancing* banyak sampel antar kelas yang dapat mempertahankan struktur kalimat pada data teks

DAFTAR PUSTAKA

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," vol. 2, no. 1, 2008.

[2] Z. Toh and W. Wang, "DLIREC: Aspect Term Extraction and Term Polarity Classification System," *Proc. 8th Int. Work. Semant. Eval. (SemEval 2014)*, no. SemEval, pp. 235–240, 2014.

[3] C. Sutton and A. McCallum, "An Introduction to Conditional Random Fields," Nov. 2010.

[4] "Sentiment Analysis." [Online]. Available: https://en.wikipedia.org/wiki/Sentiment_analysis. [Diakses: 18-Jun-2019].

[5] J. Nabi, "Machine Learning — Text Processing," 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958. [Diakses: 17-Jun-2019].

[6] "Tokenization." [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html. [Diakses: 18-Jun-2019].

[7] "The Stanford Natural Language Processing Group." [Online]. Available: https://nlp.stanford.edu/software/stanford-dependencies.shtml. [Diakses: 27-Apr-2019].

[8] C. Sutton and A. Mccallum, "1 An Introduction to Conditional Random Fields for Relational Learning," 2002.

[9] "A NLP Guide to Text Classification using Conditional Random Fields." [Online]. Available: https://www.analyticsvidhya.com/blog/2018/08/nlp-guide-conditional-random-fields-text-classification/. [Diakses: 20-Mar-2019].

[10] "BIO Tagging," 2554. [Online]. Available: https://en.wikipedia.org/wiki/Inside–outside–beginning_(tagging). [Diakses: 06-May-2019].

[11] "Sentiment Analysis Tools Overview, Part 1. Positive and Negative Words Databases." [Online]. Available: https://medium.com/@datamonsters/sentiment-analysis-tools-overview-part-1-positive-and-negative-words-databases-ae35431a470c. [Diakses: 18-Jun-2019].
[12] "Bing Liu Opinion Lexicon." [Online]. Available: https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon. [Diakses: 27-Apr-2019].
[13] "Inquirer Home Page." [Online]. Available: http://www.wjh.harvard.edu/~inquirer/. [Diakses: 27-Apr-2019].
[14] "Subjectivity Lexicon | MPQA." [Online]. Available: http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/. [Diakses: 27-Apr-2019].
[15] "aesuli/SentiWordNet: The SentiWordNet sentiment lexicon." [Online]. Available: https://github.com/aesuli/sentiwordnet. [Diakses: 17-Jun-2019].
[16] B. Wahyono, "Macam Macam Kelas Kata." [Online]. Available: https://erlangga.co.id/materi-belajar/sma/8792-macam-macam-kelas-kata.html. [Diakses: 27-Apr-2019].
[17] J. Wagner *et al.*, "DCU: Aspect-based Polarity Classification for SemEval Task 4," no. SemEval, pp. 223–229, 2014.
[18] "About Python™ | Python.org." [Online]. Available: https://www.python.org/about/. [Diakses: 27-Apr-2019].
[19] "Pycrfsuite Documentation." [Online]. Available: https://pypi.org/project/python-crfsuite/.
[20] M. Honnibal, *Introducing spaCy*. explosion.ai, 2015.
[21] "NetworkX — NetworkX." [Online]. Available: https://networkx.github.io/. [Diakses: 17-Jun-2019].
[22] "scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation." [Online]. Available: https://scikit-learn.org/stable/index.html. [Diakses: 27-Apr-2019].
[23] "AspectBasedSentimentAnalysis/dataset at master ·

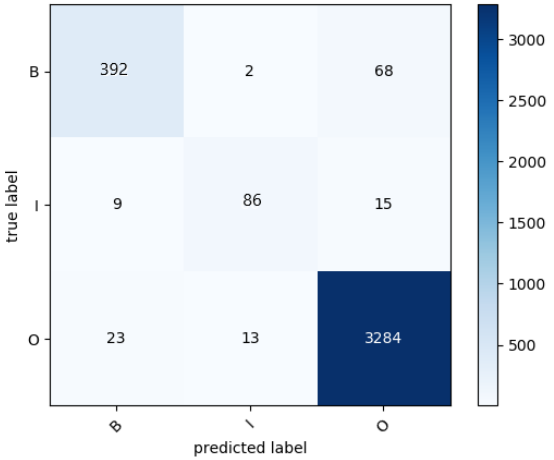
yardstick17/AspectBasedSentimentAnalysis." [Online]. Available: https://github.com/yardstick17/AspectBasedSentimentAnalysis/tree/master/dataset. [Diakses: 17-Jun-2019].

[24] C. Biemann, S. Handschuh, A. Freitas, F. Meziane, and E. Métais, "Natural language processing and information systems: 20th international conference on applications of natural language to information systems, NLDB 2015 Passau, Germany, june 17-19, 2015 proceedings," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9103, pp. 220–233, 2015.

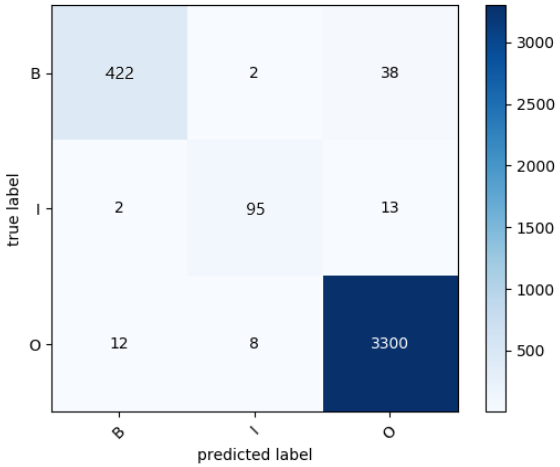
(Halaman ini sengaja dikosongkan)

LAMPIRAN

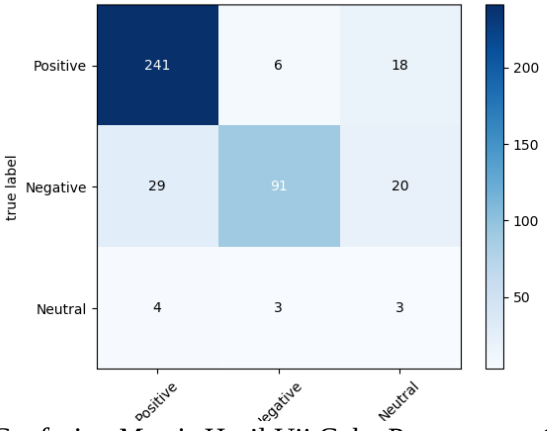
1. *Confusion Matrix* Hasil Uji Coba Daftar Fitur A Pada CRF



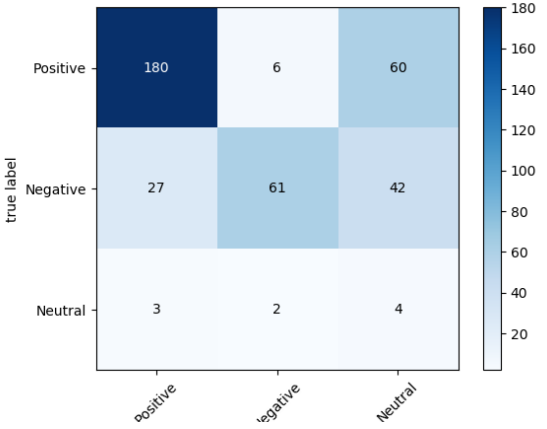
2. *Confusion Matrix* Hasil Uji Coba Daftar Fitur B Pada CRF



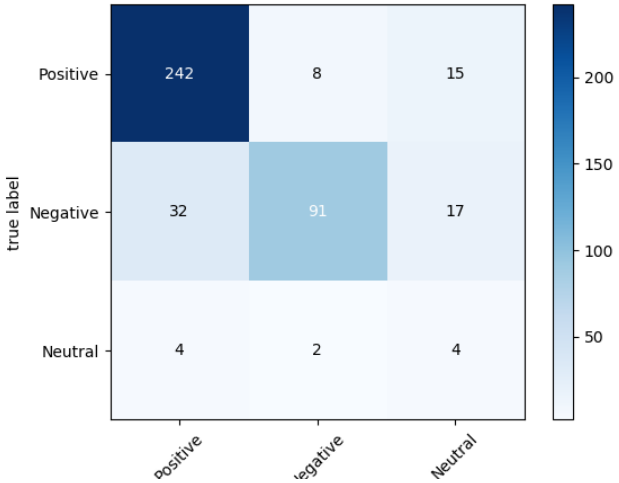
3. *Confusion Matrix* Hasil Uji Coba Penggunaan *Bing Liu Sentiment Lexicon*



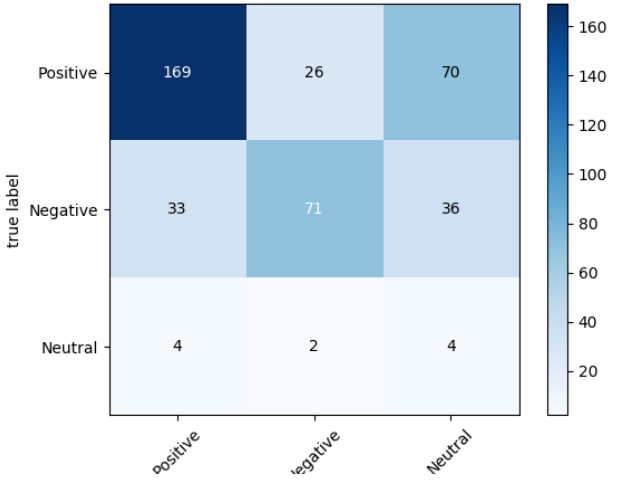
4. *Confusion Matrix* Hasil Uji Coba Penggunaan *General Inquirer Sentiment Lexicon*



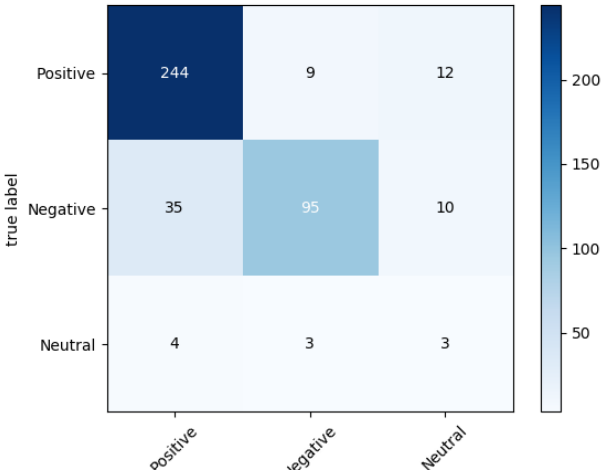
5. *Confusion Matrix* Hasil Uji Coba Penggunaan *MPQA Sentiment Lexicon*



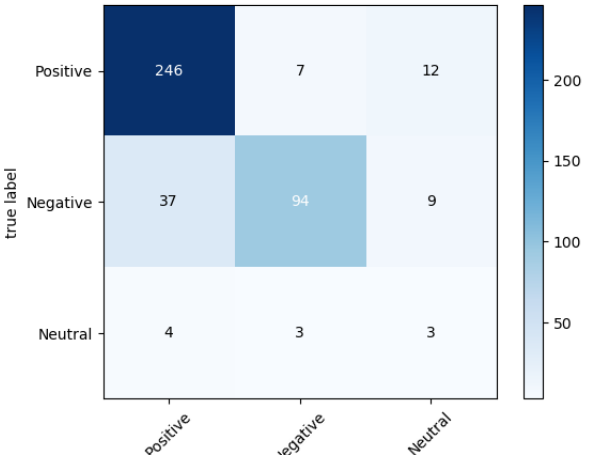
6. *Confusion Matrix* Hasil Uji Coba Penggunaan *SentiWordNet Sentiment Lexicon*



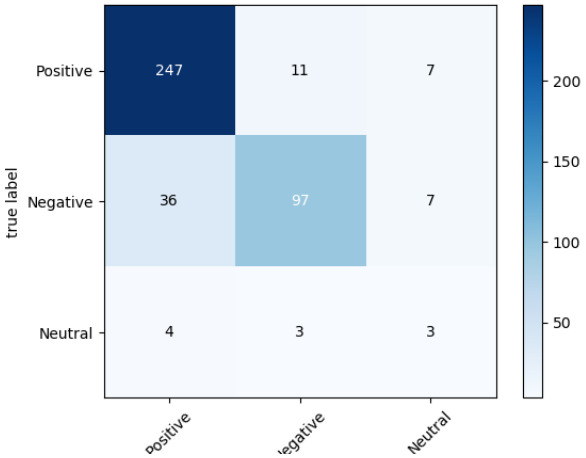
7. *Confusion Matrix* Hasil Uji Coba Kombinasi *MPQA* dan *Bing Liu Sentiment Lexicon*



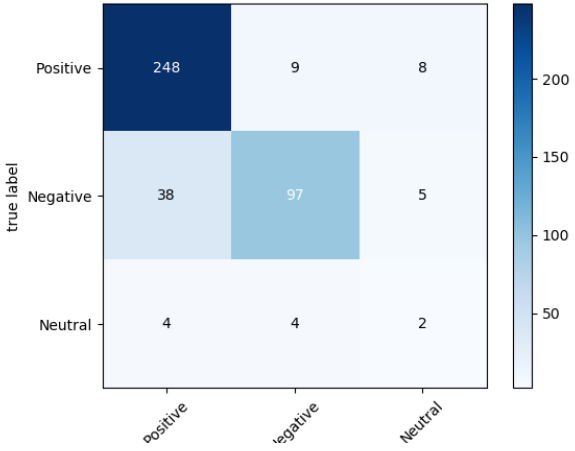
8. *Confusion Matrix* Hasil Uji Coba Kombinasi *MPQA, Bing Liu dan General Inquirer Sentiment Lexicon*



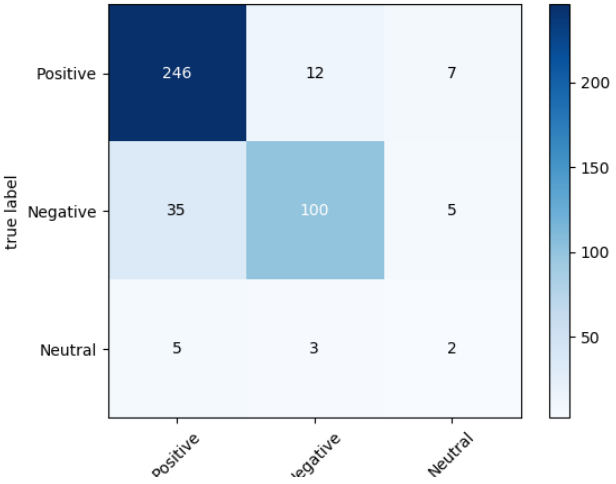
9. *Confusion Matrix* Hasil Uji Coba Kombinasi *4 Sentiment Lexicon*



10. *Confusion Matrix* Hasil Uji Coba Pembobotan Menggunakan *Token Distance*



11. *Confusion Matrix* Hasil Uji Coba Pembobotan Menggunakan *Dependency Path Distance*



BIODATA PENULIS

Rogo Jagad Alit, lahir di Kediri pada tanggal 17 Agustus 1997. Penulis menempuh pendidikan mulai dari TK Mardi Yuana Depok (2002-2003), SD Mardi Yuana Depok (2003-2009), SMP Mardi Yuana Depok (2009-2012), SMA Regina Pacis Bogor (2012-2015), dan sekarang sedang menjalani pendidikan S1 Informatika di ITS. Penulis aktif dalam organisasi dan kepanitiaan Himpunan Mahasiswa Teknik Computer (HMTC), Schematics dan ITS Expo 2017. Diantaranya adalah menjadi staff ahli Departemen Kaderisasi dan Pemetaan HMTC ITS 2017-2018, staff Departemen *National Programming Contest* Schematics ITS 2016, staff ahli Departemen *National Programming Contest* Schematics ITS 2017 dan staff ahli Divisi Web Apps ITS Expo 2017. Penulis juga merupakan salah satu penerima BukaBeasiswa dari Bukalapak pada tahun 2018. Komunikasi dengan penulis dapat melalui telepon: +6289658591489 dan *email*: **rogojagadalit@gmail.com**.