

**TESIS**

**MODEL MONITORING PERSEBARAN PENYAKIT  
DEMAM BERDARAH DI INDONESIA BERDASARKAN  
ANALISIS PESAN TWITTER**

***MONITORING MODEL OF SPREADING DENGUE FEVER  
DISEASE IN INDONESIA BASED ON TWEET ANALYSIS***



PUJI WINAR CAHYO  
13/356442/PPA/04412

**PROGRAM STUDI S2 ILMU KOMPUTER  
JURUSAN ILMU KOMPUTER DAN ELEKTRONIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS GADJAH MADA  
YOGYAKARTA  
2016**

## **TESIS**

# **MODEL MONITORING PERSEBARAN PENYAKIT DEMAM BERDARAH DI INDONESIA BERDASARKAN ANALISIS PESAN TWITTER**

***MONITORING MODEL OF SPREADING DENGUE FEVER  
DISEASE IN INDONESIA BASED ON TWEET ANALYSIS***

Diajukan untuk memenuhi salah satu syarat memperoleh derajat  
Master of Science Ilmu Komputer



PUJI WINAR CAHYO

13/356442/PPA/04412

**PROGRAM STUDI S2 ILMU KOMPUTER  
JURUSAN ILMU KOMPUTER DAN ELEKTRONIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS GADJAH MADA  
YOGYAKARTA**

**2016**

**HALAMAN PENGESAHAN**

**TESIS**

**MODEL MONITORING PERSEBARAN PENYAKIT  
DEMAM BERDARAH DI INDONESIA BERDASARKAN  
ANALISIS PESAN TWITTER**

Telah dipersiapkan dan disusun oleh

PUJI WINAR CAHYO  
13/356442/PPA/04412

Telah dipertahankan di depan Tim Penguji  
pada tanggal 5 Januari 2017

Susunan Tim Penguji

Edi Winarko, M.Sc, Ph.D  
Promotor

---

Penguji

Ko-promotor

---

Penguji

---

Penguji

Tesis ini telah diterima sebagai salah satu persyaratan  
Untuk memperoleh gelar Master of Computer Science  
Tanggal 5 Januari 2017

Dr. Tri Kuntoro Priyambodo, M.Sc.  
Pengelola Program Studi S2 Ilmu Komputer

## **PERNYATAAN**

Dengan ini saya menyatakan bahwa dalam Tesis ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar Master di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 5 Januari 2017

**PUJI WINAR CAHYO**

Karya sederhana ini kupersembahkan untuk wanita itu,  
Tegar, Sabar, Cinta, Kasih, Sayang dan Do'a darimu

Terimakasih “**Ibu**”.

*(motto)*

Kesaksian Sederhana, (Emha Ainun Nadjib, 2003)

## **PRAKATA**

Alhamdulillah, segala puji dan syukur kepada Allah SWT atas segala kemudahan dan pertolongan dari-Nya, sehingga Tesis dengan judul Model *Monitoring Persebaran Penyakit Demam Berdarah di Indonesia Berdasarkan Analisis Pesan Twitter* ini telah selesai penulis susun.

Telah banyak pihak yang membantu penulis dalam proses penulisan tesis ini. Oleh karena itu, penulis tidak lupa menghaturkan banyak terimakasih kepada semua pihak atas segala bimbingan dan bantuan yang telah diberikan, semoga amal baik tersebut mendapat balasan dan limpahan karunia dari Allah SWT. Sebagai rasa hormat dan ucapan terimakasih penyusun sampaikan kepada:

1. Ibunda saya tercinta, Ibu Suratmi, tempat curhat setia penulis. Selalu sabar, mendoakan, memberi support dan motivasi kepada penulis.
2. Mbah Putri dan Mbah Kakung, Mbah Karto Rebo. Telah mendidik dan memberikan hasil jerih payah untuk cucunya, agar bisa terus sekolah.
3. Kakak perempuanku satu – satunya, Fadis Isnawati. Atas segala bantuan dan kesabaran yang telah banyak diberikan kepada penulis.
4. Bapak Edi Winarko, M.Sc, Ph.D, selaku Pembimbing Tesis, dengan sabar telah memberikan ilmu dan pengarahan kepada penulis.
5. Mas Habibi, Yulia, Obed, Ni Made Ari Pratiwi, Mas Rahman Indra, Mas Salim, Mbak Dian dan Seluruh Teman-teman Pascasarjana ILKOM 2013 yang tidak penulis sebutkan satu persatu.
6. Bapak/Ibu Dosen serta segenap Staf dan Karyawan di jurusan Ilmu Komputer FMIPA UGM.

Tesis ini tentunya tidak lepas dari segala kekurangan dan kelemahan, untuk itu segala kritik dan saran yang bersifat membangun guna kesempurnaan tesis ini sangat diharapkan. Semoga tesis ini dapat bermanfaat bagi penulis sendiri pada khususnya dan bagi para pembaca pada umumnya.

Yogyakarta 22 November 2016

Puji Winar Cahyo

## DAFTAR ISI

<b>JUDUL .....</b>	ii
<b>HALAMAN PENGESAHAN.....</b>	iii
<b>PERNYATAAN.....</b>	iv
<b>HALAMAN PERSEMBAHAN .....</b>	v
<b>HALAMAN MOTTO .....</b>	vi
<b>PRAKATA.....</b>	vii
<b>DAFTAR ISI .....</b>	viii
<b>DAFTAR TABEL .....</b>	xi
<b>DAFTAR GAMBAR.....</b>	xii
<b>INTISARI .....</b>	xiii
<b>ABSTRACT.....</b>	xiv
<b>BAB I PENDAHULUAN.....</b>	1
1.1 Latar Belakang dan Permasalahan .....	1
1.2 Rumusan Masalah .....	2
1.3 Batasan Masalah .....	2
1.4 Tujuan Penelitian .....	3
1.5 Manfaat Penelitian .....	3
1.6 Keaslian Penelitian.....	3
1.7 Metodologi Penelitian.....	4
1.8 Sistematika Penulisan .....	5
<b>BAB II TINJAUAN PUSTAKA .....</b>	6
<b>BAB III LANDASAN TEORI .....</b>	13
3.1 <i>Pesan Tweet</i> .....	13
3.2 <i>Web Scraping</i> .....	13
3.3 <i>Natural Language Processing</i> .....	14
3.4 <i>Text Mining</i> .....	15
3.5 NER Menggunakan <i>Maximum Entropy</i> .....	16
3.5.1 <i>Maximum entropy tagger</i> .....	17
3.5.2 <i>Improved iterative scaling pada maximum entropy</i> .....	18

3.6 <i>String Matching</i> .....	19
3.7 <i>Forward Chaining</i> .....	20
<b>BAB IV ANALISIS DAN RANCANGAN MODEL MONITORING .....</b>	<b>21</b>
4.1 Analisis Model <i>Monitoring</i> .....	21
4.2 <i>Preprocessing</i> .....	23
4.2.1 Pembentukan <i>feature</i> .....	24
4.2.2 Pembobotan ( <i>Improved Iterative Scaling</i> ) .....	25
4.3 Anotasi NER ( <i>Maximum Entropy</i> ).....	28
4.4 Kombinasi NER ( <i>Maximum Entropy</i> dengan <i>String Matching</i> ) .....	29
4.5 Penalaran Tingkat Resiko .....	30
4.6 Rancangan Antar Muka Model <i>Monitoring</i> .....	32
4.6.1 Antar muka peta persebaran penyakit demam berdarah .....	32
4.6.2 Antar muka data persebaran penyakit demam berdarah .....	33
4.6.3 Antar muka <i>history</i> kejadian demam berdarah .....	33
4.7 Pengujian dan Evaluasi .....	34
<b>BAB V IMPLEMENTASI.....</b>	<b>36</b>
5.1 Pembangunan Model <i>Monitoring</i> .....	36
5.2 Implementasi <i>Scraping</i> data.....	37
5.2.1 <i>Scraping</i> twitter.com.....	37
5.2.2 <i>Scraping</i> geonames.org.....	38
5.3 Implementasi <i>Preprocessing</i> .....	39
5.4 Implementasi <i>Binary Features</i> .....	41
5.5 Implementasi Pencarian Bobot Optimal .....	42
5.6 Implementasi Deteksi Entitas .....	42
5.7 Implementasi Pemetaan Lokasi Kejadian dan Penalaran Tingkat Resiko..	44
5.8 Implementasi Pengujian NER.....	45
<b>BAB VI HASIL DAN PEMBAHASAN .....</b>	<b>46</b>
6.1 Pengujian NER tanpa <i>String Matching</i> .....	46
6.2 Pengujian NER dengan <i>String Matching</i> .....	47
6.3 Hasil Model <i>Monitoring</i> .....	48
6.4 Pengujian Hasil Model <i>Monitoring</i> .....	54
<b>BAB VII KESIMPULAN DAN SARAN .....</b>	<b>56</b>

7.1 Kesimpulan .....	56
7.2 Saran .....	57
<b>DAFTAR PUSTAKA.....</b>	<b>58</b>

## DAFTAR TABEL

Tabel 2.1 Kajian pustaka beberapa penelitian yang terkait .....	10
Tabel 4.1 Contoh <i>Preprocessing</i> .....	23
Tabel 4.2 Template <i>Features</i> .....	24
Tabel 4.3 Definisi Kumpulan <i>Gazetter</i> .....	24
Tabel 4.4 Data pelatihan .....	25
Tabel 4.5 Penjabaran x Terhadap <i>Feature</i> Anotasi .....	25
Tabel 4.6 Hasil Perhitungan $g(\Delta\lambda_1)$ Sesuai Persamaan 3.10 .....	27
Tabel 4.7 Hasil Perhitungan $g'(\Delta\lambda_1)$ Sesuai Persamaan 3.11 .....	27
Tabel 4.8 Hasil Perhitungan untuk Persamaan 3.12 .....	27
Tabel 4.9 Hasil Pencarian <i>feature</i> $f_5=1$ dan label = NUM Pada <i>history</i> Pertama	27
Tabel 4.10 Binary <i>Feature</i> Data Pelatihan .....	28
Tabel 4.11 Normalisasi Faktor Peluang Kebenaran untuk $x = \text{"situbondo"}$ .....	28
Tabel 4.12 Peluang maximum entropy kata “situbondo” .....	29
Tabel 4.13 Hasil Anotasi.....	29
Tabel 4.14 Perbandingan MaxEnt dengan Kombinasi .....	30
Tabel 4.15 <i>Knowledge base</i> .....	30
Tabel 4.16 Permasalahan dan Solusi Pada Proses Penalaran .....	31
Tabel 4.17 Keterangan Pengujian .....	34
Tabel 6.1 Pengujian <i>Maximum Entropy</i> .....	46
Tabel 6.2 Pengujian <i>Maximum Entropy</i> dan <i>String Matching</i> .....	48
Tabel 6.3 Data Hasil <i>Monitoring</i> Bulan Maret 2015 dan April 2015.....	49

## **DAFTAR GAMBAR**

Gambar 4.4 Peta Persebaran Penyakit Demam Berdarah.....	32
Gambar 4.5 Antar Muka Persebaran Penyakit Demam Berdarah .....	33
Gambar 4.6 Antar Muka <i>History</i> Kejadian Demam Berdarah Pada Suatu Daerah .....	34
Gambar 5.1 Implementasi Scrapping twitter.com .....	38
Gambar 5.2 Implementasi <i>Scraping geonames.org</i> .....	39
Gambar 5.3 Implementasi <i>Preprocessing</i> .....	40
Gambar 5.4 Implementasi <i>Binary Feature</i> .....	41
Gambar 5.5 Implementasi Pencarian Bobot Optimal .....	42
Gambar 5.6 Implementasi Deteksi Entitas.....	43
Gambar 5.7 Implementasi Pemetaan Lokasi dan Penalaran Tingkat Resiko .....	44
Gambar 5.8 Implementasi Pengujian.....	45
Gambar 6.1 Peta Persebaran Penyakit Demam Berdarah pada Bulan Maret .....	51
Gambar 6.2 10 Lokasi Tertinggi Angka Kejadian Bulan Maret 2015.....	52
Gambar 6.3 <i>History</i> Angka Kejadian Lokasi Pekanbaru Bulan Maret 2015.....	53
Gambar 6.4 Pengujian Data Hasil Model <i>monitoring</i> dengan Dinas Kesehatan..	54

## **INTISARI**

### **MODEL MONITORING PERSEBARAN PENYAKIT DEMAM BERDARAH DI INDONESIA BERDASARKAN ANALISIS PESAN TWITTER**

Oleh

PUJI WINAR CAHYO

13/356442/PPA/04412

Social media merupakan salah satu media komunikasi populer yang umum digunakan oleh masyarakat pada zaman sekarang ini, salah satunya dengan menggunakan twitter masyarakat dapat melakukan *update* informasi dan juga komunikasi dua arah atau lebih. Melalui informasi yang telah disampaikan pada pesan *tweet*, maka dapat dibentuk *history* data informasi yang dapat dilakukan analisis pengamatan dengan tujuan menghasilkan sebuah kesimpulan informasi baru. Pada penelitian ini dibangun model monitoring untuk analisis persebaran penyakit demam berdarah. Model monitoring tersebut menerapkan NER (*Named Entity Recognition*) dengan metode *maximum entropy* dan pelatihan *improve iterative scaling* untuk mendapatkan entitas yang tersirat didalam informasi pesan *tweet* yang membahas mengenai demam berdarah. Hasil data dari NER tersebut kemudian akan dilakukan analisis data menggunakan metode *forward chaining* untuk menghasilkan tiga kategori tingkat resiko angka kejadian demam berdarah, diantaranya adalah resiko rendah, sedang dan tinggi. Pengujian data hasil model *monitoring* dengan dinas kesehatan terkait menunjukan bahwa untuk mencapai kesesuaian hasil *monitoring* perlunya melihat konteks kalimat pada sumber informasi pesan *tweet* yang diambil.

Kata-kata kunci : *social media, monitoring, demam berdarah, data mining, named entity recognition*

## **ABSTRACT**

### **MONITORING MODEL OF SPREADING DENGUE FEVER DISEASE IN INDONESIA BASED ON TWEET ANALYSIS**

**By**

PUJI WINAR CAHYO

13/356442/PPA/04412

*Social media is one of the popular communication media commonly used by the public in recent times, one of them by using the twitter community can update information and two-way communication or more. Through the information from tweet message, it can be formed history data to do analysis of observations with the purpose producing a new conclusion. In this study constructed a models for the analysis of monitoring the spread of dengue fever disease. Monitoring models apply NER (Named Entity Recognition) with maximum entropy method and training with improve iterative scaling to get the message implicit in a tweet that addresses information about dengue fever. Data results from NER will be analyzed using forward chaining method for producing three categories of risk levels the incidence of dengue fever. In this case, including the risk of low, medium and high. Data results from monitoring model compared with datas from departement of health, it shows that to achieve conformity results need to see the context of the sentence on resources taken tweet message. Data results from monitoring model compared with datas from department of health, it shows that to achieve conformity results need to see the context of sentence from tweet resource.*

*Key words:* social media, monitoring, dengue fever, data mining, named entity recognition

## **BAB I**

### **PENDAHULUAN**

#### **1.1 Latar Belakang dan Permasalahan**

Demam Berdarah Dengue (DBD) merupakan penyakit dengan cara persebaran melalui gigitan nyamuk aedes yang terinfeksi virus dengue. Sampai saat ini DBD banyak ditemukan di daerah tropis dan sub-tropis. Data dari seluruh dunia menunjukkan Asia menempati urutan pertama dalam jumlah penderita DBD setiap tahunnya. Sementara itu, terhitung sejak tahun 1968 hingga tahun 2009, *World Health Organization* (WHO) mencatat negara Indonesia sebagai negara dengan kasus DBD tertinggi di Asia Tenggara.

Penyakit Demam Berdarah Dengue (DBD) merupakan salah satu masalah bidang kesehatan di Indonesia. Jumlah penderita dan luas daerah persebarannya semakin bertambah seiring dengan meningkatnya mobilitas dan kepadatan penduduk. Di Indonesia, Demam Berdarah pertama kali ditemukan di kota Surabaya pada tahun 1968, dimana sebanyak 58 orang terinfeksi dan 24 orang diantaranya meninggal dunia dengan (Angka Kematian (AK) : 41,3 %). Dan sejak saat itu, penyakit ini menyebar luas keseluruh Indonesia (Buletin Jendela Epidemiologi, 2010).

Kejadian Luar Biasa (KLB) adalah timbulnya atau meningkatnya kejadian kesakitan dan atau kematian yang bermakna secara epidemiologis pada suatu daerah dalam kurun waktu tertentu. Beberapa jenis KLB yang dinilai semakin meningkat diantaranya adalah demam berdarah, SARS dan Hepatitis E. Kejadian Luar Biasa tersebut dapat mempengaruhi peningkatan kesakitan dan kematian yang besar, sehingga berdampak pada bidang pariwisata, ekonomi dan sosial. Untuk itu perlu dilakukan deteksi secara dini dan diikuti tindakan yang cepat dan tepat. Hasil deteksi tersebut digunakan untuk peningkatan kesiap-siagaan dalam menghadapi kemungkinan KLB (Permenkes, 949-2004). Meningkatnya pengguna media sosial dikalangan masyarakat modern belakangan ini sering dijadikan tolak ukur popularitas suatu permasalahan. Media sosial seperti Facebook, Linkedin,

Google+, instagram merupakan beberapa media sosial populer yang ada di kalangan pengguna internet saat ini (Milanovic, 2015).

Dikutip dari website *cnnindonesia.com*, CEO Twitter yaitu Dick Costolo mengungkapkan jumlah pengguna twitter di Indonesia sudah mencapai 50 juta anggota, sehingga memungkinkan untuk melakukan analisis terhadap *tweet* berbahasa Indonesia. Salah satunya pada situs *petajakarta.org*, situs tersebut termasuk salah satu situs Indonesia yang melibatkan twitter dalam pengembangan ide bisnis yang dijalannya, *petajakarta.org* menyediakan informasi seputar banjir di beberapa daerah jakarta Indonesia (Apasia, 2015).

Akun twitter dinas kesehatan daerah, surat kabar ataupun personal akun belakangan ini ramai melakukan pemberitaan berbahasa Indonesia mengenai kejadian demam berdarah di indonesia. Salah satu kategori isi dari *tweet* tersebut banyak membahas mengenai informasi penderita demam berdarah disuatu daerah, sehingga data tersebut dapat diolah untuk mencapai tujuan tertentu. Seperti yang telah dilakukan oleh Dr. John Brownstein, seorang CIO (*Chief Innovation Officer*) Boston Children Hospital dan juga anggota asosiasi professor di Harvard Medical School, membangun situs *healthmap.org*, dengan mengangkat permasalahan secara global, mendeteksi kejadian penyakit berbahasa Inggris kemudian dipetakan ke dalam suatu daerah dengan melibatkan data twitter sebagai salah satu sumber data rujukan (Reddy, 2015).

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah disebutkan sebelumnya, maka diperlukan model monitoring persebaran penyakit demam berdarah yang dapat memberikan informasi angka kejadian demam berdarah di negara Indonesia melalui pesan twitter.

## 1.3 Batasan Masalah

Batasan dalam penelitian ini diantaranya:

1. Penelitian ini melakukan analisis teks *tweet* berbahasa Indonesia.
2. Data yang dikumpulkan hasil dari kata kunci pencarian “demam berdarah”, “dengue fever”, “dbd”, “dengue hemorrhagic”, “dhf” atau “sakit db”.

Sedangkan untuk *hashtag* “demamberdarah”, “denguefever”, “dbd”, “dhf” atau “sakit db”.

3. Metode kombinasi *maximum entropy* digunakan sebagai NER (*Name Entity Recognition*) mendeteksi 4 entitas entitas tersebut adalah CON untuk entitas kondisi, LOC untuk entitas lokasi, ORG untuk entitas organisasi dan NUM untuk angka insiden.
4. Metode *Forward Chaining* digunakan untuk penentuan tingkat resiko kejadian demam berdarah disuatu daerah. Tingkat resiko tersebut diantaranya resiko rendah, normal dan tinggi berdasarkan angka insiden (Buletin Jendela Epidemiologi, 2010).
5. Penentuan resiko kejadian demam berdarah berdasarkan tingkat area kabupaten.

#### **1.4 Tujuan Penelitian**

Tujuan dari penelitian ini adalah membuat model *monitoring* data demam berdarah di Indonesia, berdasar hitungan angka insiden kejadian yang ada dalam pesan *twitter*.

#### **1.5 Manfaat Penelitian**

Penelitian ini diharapkan dapat melakukan ekstraksi entitas yang dapat digunakan sebagai model *monitoring* demam berdarah di negara Indonesia.

#### **1.6 Keaslian Penelitian**

Penelitian yang dilakukan terkait Model *Monitoring* Persebaran Penyakit Demam Berdarah Di Indonesia Berdasarkan Analisis Pesan *Twitter*, dengan menggunakan *maximum entropy* sebagai NER (*Name Entity Recognition*) dan *forward chaining* untuk penentuan tingkat resiko kejadian, hingga saat ini belum pernah dilakukan.

## 1.7 Metodologi Penelitian

Metodologi penelitian dalam perancangan dan implementasi model *monitoring* ini dapat dibagi menjadi beberapa tahapan, diantaranya :

### 1. Studi Pustaka

Pustaka yang dipelajari berkaitan dengan penelitian ini menggunakan *Maximum Entropy (MaxEnt)*, *Improved Iterative Scaling (IIS)* dan *Forward Chaining* diperoleh dari berbagai sumber antara lain laporan hasil penelitian (skripsi, tesis, disertasi atau jurnal), buku, majalah ilmiah, perundangan, dokumen paten atau publikasi melalui media internet.

### 2. Pengumpulan Data

Data pesan diambil dari situs *twitter.com*, situs tersebut merupakan situs media sosial populer di indonesia. Data lokasi kota besar di indonesia diambil dari situs *http://mfdonline.bps.go.id* digunakan untuk pembentukan *gazetter* (kamus data) dengan dukungan koordinat lintang dan bujur dari *geonames.org*.

### 3. Perancangan Model

Pada tahap ini dilakukan analisis proses pembentukan model untuk dapat melakukan *monitoring* persebaran penyakit demam berdarah yang ada di negara Indonesia, mulai dari tahap pengambilan informasi dari data *text* sampai pada hasil akhir terbentuknya data kejadian demam berdarah pada suatu daerah.

### 4. Implementasi

Hasil arsitektur model *monitoring* dikembangkan menjadi perangkat lunak menggunakan bahasa pemrograman python serta beberapa *tools* bantuan dengan tujuan akhir memperoleh informasi kejadian demam berdarah suatu daerah di indonesia berdasarkan tingkat resiko kejadian.

### 5. Pengujian dan Evaluasi

Bagian dari implementasi model *monitoring*, didalamnya menggunakan deteksi entitas untuk mengambil informasi dalam sebuah *text*, hasil dari deteksi entitas tersebut diuji dengan data terlabeli manual yang diambil dari situs *twitter.com* dengan pengukuran evaluasi menggunakan *precision*, *recall* dan *F-Measure*. Untuk pengujian data kejadian demam berdarah, hasil data *monitoring* demam berdarah diujikan dengan data kejadian demam berdarah yang diambil dari dinas kesehatan setempat.

## 1.8 Sistematika Penulisan

### BAB I PENDAHULUAN

Bab ini berisi uraian mengenai latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, tinjauan pustaka, metodologi penelitian, dan sistematika penulisan.

### BAB II TINJAUAN PUSTAKA

Bab ini berisi tentang informasi hasil penelitian yang telah dilakukan sebelumnya, dimana penelitian tersebut memiliki keterkaitan dengan penelitian yang dilakukan.

### BAB III LANDASAN TEORI

Bab ini berisi landasan teori, yang merupakan pembahasan mengenai teori dasar yang berkaitan dengan konsep dasar *Named Entity Recognition (NER)*, *Maximum Entropy* dan *Improved Iterative Scaling*.

### BAB IV ANALISIS DAN RANCANGAN SISTEM

Bab ini berisi uraian mengenai analisis dan kebutuhan perancangan sistem yang diimplementasikan, analisis dan rancangan pengujian. Selain itu bab ini juga menguraikan tentang perancangan *flowchart* dan antar muka (*interface*).

### BAB V IMPLEMENTASI

Bab ini berisi potongan kode program (*sourcecode*) dan *interface* program dari implementasi model yang telah dihasilkan.

### BAB VI HASIL DAN PEMBAHASAN

Bab ini berisi hasil akhir sistem yang dibangun berdasarkan hasil pengujian sistem dan evaluasi, disertai dengan penjelasan pada keluaran proses.

### BAB VII KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran yang dihasilkan dari hasil dan pembahasan, dalam hal ini dilakukan pendefinisian beberapa kekurangan dan kelebihan model *monitoring* yang telah diterapkan.

## **BAB II**

### **TINJAUAN PUSTAKA**

Penelitian mengenai *text mining* pada dekade terakhir ini terus mengalami perkembangan yang sangat pesat, terlebih dalam ekstraksi informasi yang mengarah pada *information retrieval*. *Information retrieval* pada *text mining* bertujuan untuk mengambil informasi bermanfaat di dalam kumpulan data yang berbentuk teks, informasi yang dihasilkan dapat dilakukan pelabelan untuk pengenalan entitas terhadap sebuah teks, proses tersebut sering dikenal sebagai NER (*Named Entity Recognition*). Berikut ini dipaparkan beberapa penelitian yang berkaitan dengan penyelesaian *information retrieval* terlebih mengarah pada kasus *monitoring*.

Model *feature* dibutuhkan dalam *Named Entity Recognition*, model *feature* tersebut digunakan untuk mengelompokan label kata yang mempunyai entitas yang sama. Dalam penelitiannya, Jiang dkk (2006) menggunakan *Mutual Information*, *Information Gain* dan *AMI (Average Mutual Information)* untuk ekstraksi konteks *feature* dengan kombinasi kluster kata dan kamus kata yang diterapkan pada model *maximum entropy*. Hasil dari pengujian NER tersebut membuktikan bahwa menggunakan *Maximum Entropy* dapat mencapai akurasi yang lebih tinggi dibanding menggunakan *Hidden Markov Model*. Dilain sisi Cai dan Fan (2009) menggunakan *maximum entropy* untuk mengenali bentuk kata *frase*, dimana domain pengetahuan diintegrasikan ke dalam metode statistik untuk mendukung peningkatan klasifikasi, penelitian ini menunjukkan hasil yang cukup baik pada permasalahan klasik yang dihadapi NER, yaitu bergantung pada tersedianya anotasi *corpus* yang cukup besar. Penelitian yang dilakukan Hui dkk (2009) *Named Entity Recognition* menggunakan *maximum entropy* pada kalimat berbahasa cina, Saat melakukan ekstraksi *semantic knowledge* mengalami kendala keterbatasan korpus yang digunakan. Ekspansi *semantic* diterapkan pada pengenalan entitas, dari hasil penerapan tersebut menunjukan nilai *precision*, *recall* dan *F-Value* semakin meningkat, meskipun diterapkannya entitas nama organisasi yang cukup rumit. Dari tiga penelitian yang telah disebutkan sebelumnya, untuk mencapai

keakurasi yang lebih tinggi maka dibutuhkan peningkatan ekstraksi *feature* yang didukung dengan banyak dan *validitas* korpus yang digunakan.

Selain digunakan untuk *Named Entity Recognition*, *maximum entropy* juga sering diterapkan pada *POS Tagging (Part of Speech Tagging)*. *POS Tagging* adalah pelabelan kata menyesuaikan ilmu linguistik bahasa, seperti kata kerja, kata benda, kata sifat dan lain sebagainya. Penelitian yang dilakukan Nurwidiyantoro (2011) menggunakan *maximum entropy* dijalankan dengan *map reduce* untuk pembentukan *POS Tagging*, proses pelatihannya menggunakan IIS *Improved Iterative Scaling*. Dengan menggunakan *map reduce*, hasil pemrosesan menunjukkan waktu yang lebih efisien dibanding tanpa *map reduce*. Hasil tercepat diperlihatkan pada saat pemberian anotasi menggunakan *corpus* pelatihan berukuran 1.000.000 data, dengan 30 *process map* yang digunakan. Di sisi lain penelitian yang dilakukan oleh Putranti (2013), *maximum entropy* digunakan dalam pembentukan *POS Tagging*, *SVM (Support Vector Machine)* digunakan untuk membangun model klasifikasi *sentiment*, sedangkan *TF (Term Frequency)* *IDF (Inverse Document Frequency)* digunakan untuk proses pembobotan. Hasil penelitian tersebut menunjukkan bahwa klasifikasi diperoleh akurasi 86,81 % pada pengujian 7 *cross validation* untuk tipe kernel sigmoid, sedangkan pada kelas secara manual dengan *POS Tagger* menghasilkan akurasi 81,67%. Dari dua penelitian tersebut menunjukkan bahwa *maximum entropy* dapat diproses ke dalam *map reduce* dan menghasilkan akurasi cukup baik dalam pembentukan *POS Tag*.

Manfaat sosial media saat ini erat kaitannya dengan kehidupan sehari-hari. Sehingga memungkinkan data sosial media tersebut digunakan sebagai *monitoring* pada permasalahan tertentu, seperti penelitian yang dilakukan oleh, Ji dkk (2013) melakukan *monitoring* kekhawatiran pada kesehatan masyarakat menggunakan *Multinomial Naive Bayes* untuk klasifikasi sentiment analisis pesan *tweet*. Penelitian ini melacak persebaran epidemi penyakit kemudian berusaha memahami tingkat kepedulian penduduk sekitar terhadap epidemi penyakit yang terjadi. Dalam penelitian ini, *Multinomial Naive Bayes* menghasilkan *F-Measure* paling tinggi dan memiliki kecepatan dalam membangun model dibanding dengan Naive bayes dan *SVM*. Penelitian lain yang dilakukan Huang dan Zhang (2013), penelitian tersebut melakukan deteksi terhadap penyebaraan penyakit *influenza* di china. Dalam

penelitiannya, mereka menggunakan data dari weibo yaitu gabungan data antara *twitter* dan *facebook*. *Cosine Similarity* digunakan sebagai *filtering* data validitas *influenza*, sedangkan *Dynamic Bayesian Network* digunakan untuk deteksi persebaran penyakit. Hasil dari penelitian tersebut menggunakan simulasi dengan target 4 kota sebagai node observasi, dengan menggunakan indikator pengukuran diantaranya *Accuracy*, *MAE (Mean Absolute Error)* dan *RMSE (Root Mean Square Error)* menunjukkan model akurasi dapat mencapai lebih dari 0.7. Selain itu, penelitian yang dilakukan MacEachren dkk (2011) melakukan *monitoring* kesadaran situasi yang ditujukan untuk semua *domain*, didalamnya memanfaatkan ANNIE *named-entity extractor* digunakan untuk deteksi entitas yang ada didalam pesan *tweet*, entitas tersebut diantaranya adalah entitas lokasi, organisasi dan nama orang. Konsep penggunaan model *monitoring* ini adalah melakukan pencarian menggunakan *keyword* pada domain tertentu kemudian hasil pencarian tersebut divisualisasikan kedalam peta, lengkap dengan bentuk data *what*, *when* dan *where* situasi pada *keyword* tersebut terjadi, kemudian dilakukan beberapa pengujian, pengujian pertama menggunakan data lokasi yang berada pada kota Haiti kemudian dicocokan dengan entitas lokasi yang dihasilkan, dari pengujian ini menghasilkan nilai *accuracy* 91.4%, sedangkan untuk pengujian *multiple* lokasi yang tersirat dalam satu *tweet* menghasilkan nilai *accuracy* 69.2%. Penelitian lain mengenai *monitoring* dilakukan oleh Achrekar dkk (2011) dengan tambahan fitur prediksi persebaran penyakit flu, menggunakan metode *auto regression* digunakan untuk prediksi persebaran mengambil data dari pesan *tweet*, pesan *tweet* tersebut kemudian akan dilakukan seleksi data berdasar indikator terjadinya penyakit *influenza*. Hasil dari seleksi data tersebut dikorelasikan dengan data aktivitas ILI (*influenza-like illness*) yang diambil dari *Centers for Disease Control and Prevention (CDC)* dengan hasil koefisien korelasi *pearson* 0.9846 terbukti dapat meningkatkan akurasi dalam prediksi kasus ILI. Selain itu penelitian yang hampir sama dilakukan oleh Talvis dkk (2014) mengenai persebaran penyakit *influenza* dengan memanfaatkan geolokasi dan gejala kritis penyakit yang kemudian dilakukan analisis menggunakan *lexicon* terhadap *feature bigram*, *three gram* dan *single word* pada pesan *twitter*, hasil dari analisis tersebut kemudian dipetakan menurut geolokasi masing – masing pesan *tweet* yang diperoleh. Data hasil analisis

kemudian dilakukan perbandingan dengan data dari *Google Flu Trends*, hasil dari perbandingan tersebut menghasilkan nilai korelasi mencapai 0.79. Penelitian berikutnya dilakukan oleh Kashyap dan Nahapetian (2014) membahas tentang *monitoring* kesehatan terhadap pengguna *twitter* dengan metode yang digunakan adalah *semantic analisis* dan *sentiment analisis* yang terapkan pada *meta data*, pesan *twitter* dan *trending topic* kesehatan. Penelitian ini diterapkan pada beberapa kelompok pengguna *twitter*, diantaranya adalah ahli kebugaran, ahli diet, dokter, selebriti, pengguna *netral* hingga orang yang berkecimpung dalam bidang IT. Hasil dari penelitian ini menyebutkan bahwa ahli kebugaran, ahli diet dan selebriti menduduki posisi paling atas dalam menjalani kebiasaan hidup sehat.

Tinjauan pustaka yang telah dibahas, beberapa menguraikan permasalahan mengenai *monitoring* dalam bidang kesehatan, terutama pada kasus persebaran penyakit. Sumber data *monitoring* umumnya merujuk pada situs *social media* salah satunya situs *social media twitter*. Setelah dilakukan pengujian, hasil pengujian membuktikan nilai *F-Measure*, akurasi dan juga koefisien korelasi yang cukup bagus, sehingga membuktikan bahwa data dari *twitter* dapat digunakan sebagai salah satu sumber data penelitian terlebih dalam bidang kesehatan. Dari beberapa referensi tinjauan pustaka yang telah dikaji, maka dapat didefinisikan perbedaan penelitian yang akan dilakukan dengan peneliti sebelumnya adalah pada penyesuaian *feature maximum entropy* dan *improved iterative scaling* untuk menangani deteksi entitas teks bahasa indonesia. Selain itu perbedaan lainnya adalah pada proses kategorisasi tingkat resiko persebaran penyakit menggunakan metode *forward chaining*. Daftar simpulan tinjauan pustaka dilihat pada Tabel 2.1.

Tabel 2.1 Kajian pustaka beberapa penelitian yang terkait

No	Penelitian	Metode	Tujuan
1	Jiang dkk. (2006)	<i>Named Entity Recognition</i> , peningkatan <i>performance</i> : menggunakan kombinasi kluster dan kamus kata diterapkan pada model <i>maximum entropy</i>	Meningkatkan ekstraksi <i>feature</i> pada <i>Named Entity Recognition</i>
2	Cai dan Fan (2009)	<i>Maksimum entropy</i> menggunakan N-Gram <i>feature</i> .	Melakukan pengenalan frase nama penyakit dengan ekstraksi dari literatur kesehatan dan data rekam medis
3	Hui dkk. (2009)	<i>Maximum entropy</i> menggunakan <i>semantic expansion</i> untuk mengatasi keterbatasan <i>corpus</i>	Melakukan pengenalan entitas pada teks bahasa cina
4	MacEachren dkk. (2011)	<i>Geovisual analytics</i> dengan <i>Scenario-based design</i> digunakan untuk <i>monitoring</i> kesadaran situasi ( <i>ANNIE named-entity extractor</i> diterapkan dalam penelitian ini)	Sumber data dari pesan <i>tweet</i> melakukan pencarian situasi terhadap <i>keyword</i> yang dimasukan, kemudian dilakukan visualisasi data hasil pencarian kedalam sebuah peta
5	Achrekar dkk. (2011)	Menggunakan <i>auto regression</i> untuk prediksi persebaran penyakit flu	Melakukan <i>monitoring</i> terhadap pesan <i>tweet</i> yang mengandung indikator

			penyakit flu, indikator tersebut digunakan untuk melacak dan memprediksi persebaran <i>influenza</i> dalam suatu populasi
6	Nurwidiyantoro (2011)	<i>Maximum entropy</i> digunakan untuk <i>Tagging</i> dan <i>improve iterative scaling</i> digunakan saat proses pembobotan	Melakukan penerapan map reduce pada <i>POS Tagging</i> bahasa indonesia
7	Putranti (2013)	<i>Maksimum entropy</i> digunakan dalam <i>POS Tagging, Support Vector Mesin</i> untuk membangun model klasifikasi, dengan pembobotan TIDF	Melakukan analisis sentimen untuk object tertentu bersumber dari opini twitter bahasa indonesia
8	Ji dkk. (2013)	<i>Multinomial Naive Bayes</i> digunakan untuk klasifikasi analisisa sentimen	Melakukan <i>monitoring</i> tingkat kepedulian kesehatan masyarakat berdasarkan analisis sentimen pesan twitter
9	Zhao dkk. (2013)	Menggunakan <i>Cosine Similarity</i> sebagai <i>filtering</i> data, <i>Dynamic Bayesian Network</i> untuk deteksi persebaran penyakit flu	Melakukan deteksi persebaran penyakit <i>influenza</i> berdasarkan weibo

10	Talvis dkk. (2014)	Menggunakan analisis <i>lexicon</i> terhadap <i>feature bigram, three gram</i> dan <i>single word</i> untuk analisis pesan <i>twitter</i>	Melakukan <i>monitoring</i> persebaran penyakit <i>influenza</i> dengan memanfaatkan geolokasi dan juga analisis gejala kritis penyakit <i>influenza</i> pada pesan <i>tweet</i> .
11	Kashyap dan Nahapetian (2014)	Menggunakan semantic analisis dan sentiment analisis untuk melakukan analisis pada <i>metadata</i> dan <i>trending topic</i> kesehatan pada pesan <i>tweet</i>	Melakukan <i>monitoring</i> kesehatan pada kebiasaan sehari – hari pengguna <i>twitter</i>
12	Cahyo (2016)	Penyesuaian <i>feature</i> pada <i>maximum entropy</i> dan <i>improved iterative scaling</i> pada NER berbahasa indonesia. Kategorisasi angka insiden menggunakan <i>forward chaining</i>	Melakukan <i>monitoring</i> persebaran penyakit demam berdarah berdasarkan angka insiden penyakit demam berdarah di negara indonesia.

## **BAB III**

### **LANDASAN TEORI**

#### **3.1 *Pesan Tweet***

Setiap pengguna *twitter* diberi kemudahan dan kecepatan untuk melakukan *posting* status (*tweet*) atau melakukan *share* status pengguna *twitter* lain (*retweet*), *tweet* tersebut mengandung teks yang dapat diolah dengan maksimal panjang karakter hanya sampai 140 karakter. Karena kecepatan dan kemudahan publikasi *twitter* tersebut, menjadikan *twitter* termasuk salah satu media komunikasi yang penting bagi orang-orang dari semua lapisan masyarakat (Kumar dkk, 2014).

Twitter memungkinkan pengguna untuk mengakses data mereka melalui API Twitter, Beberapa macam API yang disediakan *twitter* untuk permasalahan *tweet* diantaranya adalah :

1. REST API : API yang dapat digunakan untuk operasi baca dan tulis pada *twitter*, seperti posting dan pengambilan *tweet* pada akun tertentu (*user id*), lokasi tertentu (*geolocation*) atau status tertentu (*status id*)
2. Stream API : API yang dapat digunakan untuk mendapatkan data pencarian melalui kata kunci *tweet* kurun waktu saat ini (*realtime*)

#### **3.2 *Web Scraping***

*Web scraping* adalah metode pengumpulan data melalui media internet, meskipun *web scraping* bukan sesuatu hal yang baru, belakangan ini metode *web scraping* sangat populer digunakan untuk pemenuhan *data mining*. Sebelumnya metode ini dikenal kedalam beberapa istilah, diantaranya adalah *screen scraping*, *data mining*, *web harvesting* ataupun metode lain yang sejenis. Menurut teori, *web scraping* adalah cara untuk mengumpulkan data menggunakan metode yang berbeda dengan menggunakan API (*Application Programming Interface*). Cara seperti ini biasanya dimulai dengan penulisan kode program, yang dimana digunakan sebagai automatisasi *query* untuk melakukan *requests* data terhadap *server*, umumnya data tersebut dapat berbentuk *form html* dan *file* lain yang berhubungan dengan *web pages*. Data hasil *requests* tersebut dapat dilakukan ekstraksi untuk menghasilkan informasi yang akan dicari (Mitchell, 2015).

### **3.3 Natural Language Processing**

*Natural Language processing* NLP berada dalam *area human computer interaction*. Merupakan bagian dari ilmu komputer, *artificial intelligence* dan *linguistic*, mengambil fokus utama pada interaksi komputer dan bahasa manusia atau bahasa alami dengan cara mengolah informasi dari bahasa alami tersebut ke dalam informasi yang dapat dipahami oleh komputer. Informasi dapat dihasilkan dari buku, berita, bisnis, laporan pemerintahan maupun internet (Abhimanyu dkk, 2013).

Ada lima tahapan di dalam natural language *processing* menurut (Abhimanyu dkk, 2013), lima tahapan tersebut adalah :

1. Analisa *morfologi* dan *lexical*

Meliputi pembagian paragraf, kalimat hingga ke dalam bagian paling kecil yaitu kata. Kemudian dibangun perbendaharaan kata yang memuat makna sehingga satu kata dengan kata yang lain dapat diketahui perbedaannya. Contoh “kejujuran” merupakan kata jujur dapat imbuhkan awalan ‘ke’ dan akhiran ‘an’

2. Analisa Sintaksis

Melibatkan kata untuk menggambarkan struktur sebuah kalimat. Kata tersebut di transformasikan ke dalam struktur kata yang saling berelasi sehingga membentuk kalimat dengan cara penulisan yang sesuai. Contoh “*the girl the go to the school*”. Kalimat tersebut akan ditolak oleh analisa sintaksis dalam bahasa inggris karena tidak memenuhi tata penulisan yang sesuai.

3. Analisa Semantik

Dapat dikatakan arti kata secara kamus atau makna yang sesuai pada setiap konteks dipetakan ke dalam semantic model, struktur kata diciptakan oleh analisa sintaksis kemudian diberikan arti. Pemetaan secara semantik tersebut tetap mengakomodasi arti kata yang saling berelasi. Contoh : “tanpa warna ide biru”. Kalimat tersebut akan ditolak oleh analisa semantik karena sulit diketahui korelasi arti kalimatnya.

4. *Discourse Integration*

Makna dari kalimat yang bergantung kalimat yang mendahuluinya atau makna kalimat yang bergantung pada kalimat yang diikutinya. Contoh : “Ide tersebut

dapat diterapkan pada perancangan selanjutnya”. Kalimat yang mengandung kata “tersebut” kemungkinan dipengaruhi kalimat sebelumnya, kata “selanjutnya” bisa berpengaruh pada kalimat yang akan diikuti.

#### 5. Analisa Pragmatis

Merupakan analisa kalimat dengan tujuan tertentu dan bergantung pada aspek tertentu, bisa bergantung pada situasi dan tujuan utama kenapa kalimat tersebut harus dibentuk, dengan harapan kalimat yang dibentuk dapat diartikan secara benar sesuai maksud pembentukan kalimat sebelumnya.

### **3.4 *Text Mining***

*Text mining* dan *text analytics* merupakan cakupan dari data mining yang cukup luas, kemudian disatukan pada kebutuhan yang sama, yaitu untuk mengubah teks ke dalam angka sehingga teknik tersebut dapat diterapkan pada *document database* yang besar (Miner, 2012). *Text mining* mengadopsi dan mengembangkan teknik yang berasal dari bidang lain untuk menyelesaikan permasalahan. Teknik standar yang biasa digunakan dalam *text mining* diantaranya adalah *text classification*, *text clustering*, *ontology and taxonomy creation*, *document summarization* dan *latent corpus analysis* (Fienerer dkk, 2008).

Secara umum, *text mining* harus melewati beberapa tahapan proses untuk memperoleh hasil yang diinginkan. Adapun proses-proses yang harus dilakukan dalam *text mining*, diantaranya (Even dan Zohar, 2002) :

#### 1. *Text preprocessing*

Pemrosesan awal teks ini bertujuan untuk mempersiapkan teks sebelum diolah lebih lanjut.

#### 2. *Text transformation*

Proses transformasi teks melakukan perubahan kata menjadi kata dasar (*stemming*), pengurangan dimensi kata pada dokumen (*stop word removal*), *bag of word*, dan *vectorial document representation* untuk representasi dokumen.

#### 3. *Feature selection*

Pada tahapan ini dilakukan pembuangan bagian-bagian yang tidak terkait untuk mendapatkan bagian-bagian penting.

#### 4. *Pattern discovery*

Tahapan ini bertujuan untuk menemukan pola atau *knowledge* dari teks dengan menggunakan teknik-teknik data mining seperti *classification* dan *clustering*.

#### 5. *Interpretation*

Tahapan ini melakukan interpretasi kesuatu bentuk-bentuk untuk kemudian dilakukan evaluasi.

### 3.5 NER Menggunakan *Maximum Entropy*

*Named Entity Recognition* merupakan bagian dari *information extraction* bertujuan untuk melakukan ekstraksi teks kemudian melakukan klasifikasi ke dalam entitas, akan tetapi perkembangannya dapat dipakai pada identifikasi entitas gen dan protein (Christopher dkk, 2004). Berikut contoh deteksi NER pada sebuah kalimat. “13/Jumlah pasien/O demam/O berdarah/O di/O RSUD/ORG jombang/LOC meninggal/CON, sepuluh/NUM kritis/CON”. Hasil dari proses NER dapat diketahui “13” merupakan entitas Jumlah, “jombang” merupakan entitas Lokasi, “meninggal” merupakan entitas Kondisi, “sepuluh” merupakan entitas Jumlah, “kritis” merupakan entitas Kondisi dan lainnya termasuk ke dalam entitas O (*Other*).

Ambiguitas atau WSD (*Word Sense Disambiguation*) di dalam NER bisa terjadi, salah satunya pada kata yang memiliki penulisan yang sama, akan tetapi memiliki kelas entitas yang berbeda. Contoh, “soedirman sedang berada di jln soedirman”. Kata “soedirman” pertama masuk ke dalam entitas nama, kata “jln soedirman” masuk ke dalam entitas tempat atau lokasi. Karena terjadi ambiguitas tersebut maka perlu dilakukan pencegahan. Salah satunya dengan menyediakan *gazetter* atau kamus data yang memuat informasi mengenai entitas-entitas yang disediakan.

Metode yang sering dipakai dalam pembentukan model NER salah satunya adalah *maximum entropy classifier*, *maximum entropy* termasuk dalam keluarga klasifikasi eksponensial atau *log-linear*. Model ini bekerja dengan cara mengekstrak sekumpulan *feature* dari data pelatihan, kemudian mengkombinasikan *feature-feature* tersebut (dengan mengalikan setiap *feature* dengan sebuah pembobot untuk kemudian digabungkan kembali), dengan hasil akhir membentuk probabilitas. Model *maximum entropy* lebih fleksibel dalam menangani konteks

dan dapat digunakan sebagai alternatif terhadap model HMM dalam domain *Sequence Labeling*. Model ini mendefinisikan sekumpulan *features template* dan sistem mempelajari *feature-feature* mana yang berpengaruh dengan memanfaatkan *features template* pada *corpus* pelatihan.

### 3.5.1 Maximum entropy tagger

Berdasarkan (Ratnaparki, 1996) pada POS Tagger dan dideskripsikan kembali oleh (Hui dkk, 2009) untuk NER berbahasa cina, pelabelan *maximum entropy* dapat dimodelkan pada probabilitas Persamaan (3.1).

$$p(y|x) = \frac{1}{Z(x)} \exp \left( \sum_{i=1}^n \lambda_i f_i(x, y) \right) \quad (3.1)$$

dimana  $Z(x)$  adalah normalisasi faktor dari  $x$  :

$$Z(x) = \sum_{y \in Y} \exp \left( \sum_{i=1}^n \lambda_i f_i(x, y) \right) \quad (3.2)$$

dimana  $y$  adalah anotasi/label dan  $x$  adalah *context word* dan  $f_i(x, y)$  adalah *feature* ke  $i$  yang mana mempunyai bobot *feature* yaitu  $\lambda_i$ . Probabilitas dari urutan anotasi  $y_1 \dots y_n$  diberikan sebuah kata yang membentuk kalimat  $w_1 \dots w_n$  dengan peluang Persamaan (3.3).

$$p(y_1 \dots y_n | x_1 \dots x_n) \approx \prod_{i=1}^n p(y_i | x_i) \quad (3.3)$$

dimana  $x_i$  adalah *context* untuk *word ke i* ( $w_i$ ), pelabelan untuk penelitian ini menggunakan *sequential search* untuk mencari anotasi paling mungkin yang diberikan pada kata di dalam kalimat,  $f(x, y)$  akan menghasilkan *binary value* untuk diterapkan pada *context word* yang memiliki kesesuaian label, contoh dari *binary value* dapat dilihat pada Persamaan (3.4).

$$f(x, y) = \begin{cases} 1 & \text{jika } w_i = \text{jombang dan } y = \text{LOC} \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

### 3.5.2 Improved iterative scaling pada maximum entropy

Menurut (Jurafsky dan Martin, 2008), pencarian parameter pembobot yang memiliki nilai *maximum log likelihood* memiliki permasalahan yang disebut sebagai *convex optimization*. Ada beberapa algoritma yang dapat digunakan untuk menyelesaikan permasalahan ini, diantaranya adalah *quasi-Newton*, *gradient ascent*, *conjugate gradient*, dan berbagai macam algoritma *iterative scaling*. Menurut (Malouf, 2010), performa dari *Generalized Iterative Scaling* (GIS) kurang bagus. (Malouf, 2010) menyatakan bahwa *Improved Iterative Scaling* (IIS) yang dikembangkan oleh (Della Pietra dkk, 1997) dapat digunakan untuk memperbaiki konvergensi yang lambat dari GIS. Menurut (Malouf, 2010) dan (Berger dkk, 1996), metode Newton-Rhapson dapat digunakan dalam algoritma IIS untuk menghitung perubahan nilai parameter optimal.

Untuk itu algoritma *Improved Iterative Scaling* (IIS) digunakan untuk mencari parameter pembobot yang memaksimalkan nilai *log likelihood*. Menurut (Berger dkk, 1996), algoritma ini dapat digunakan ketika *feature*  $f_i(x, y)$  tidak negative  $f_i(x, y) \geq 0$  untuk setiap  $i$ ,  $x$  dan  $y$ . Algoritma ini adalah sebagai berikut (Berger dkk, 1996):

Masukan : fungsi *feature*  $f_1, f_2, f_3, \dots, f_n$  distribusi peluang empiris  $\tilde{p}(x, y)$

Keluaran : nilai parameter optimal  $\lambda_i^*$  model optimal dari  $p_{\lambda^*}$

1. Mulai dengan  $\lambda_1 = 0$  untuk semua  $i \in (1, 2, 3, \dots, n)$
2. Untuk setiap:

- a. Hitung nilai  $\Delta\lambda_i = 0$  sebagai solusi dari Persamaan (3.5)

$$\sum_{x,y} \tilde{p}(x) p(y|x) f_i(x, y) \exp(\Delta\lambda_i f^\#(x, y)) = \tilde{p}(f_i) \quad (3.5)$$

$$\text{dimana : } f^\#(x, y) \equiv \sum_{i=1}^n f_i(x, y) \quad (3.6)$$

$$\tilde{p}(f_i) = \sum_{x,y} \tilde{p}(x, y) f_i(x, y) \quad (3.7)$$

- b. Ubah nilai  $\lambda_i$  sesuai dengan  $\lambda_i = \lambda_i + \Delta\lambda_i$
3. Lanjutkan ke langkah 2 apabila  $\lambda_i$  belum konvergen

Menurut (Berger dkk, 1996) untuk Persamaan (3.5) yang tidak konstan, nilai  $\lambda_i$  dihitung secara numerik menggunakan metode Newton. Metode ini menghitung  $\alpha_*$  dari sebuah persamaan  $g(\alpha_*) = 0$  secara iteratif menggunakan Persamaan (3.8).

$$\alpha_{n+1} = \alpha_n - \frac{g(\alpha_n)}{g'(\alpha_n)} \quad (3.8)$$

Persamaan (3.5) dapat dinyatakan dalam bentuk:

$$\sum_{x,y} \tilde{p}(x) p(y|x) f_i(x, y) \exp(\Delta\lambda_i f^{\#}(x, y)) - \tilde{p}(f_i) = 0 \quad (3.9)$$

dari Persamaan (3.9), didapatkan Persamaan (3.10).

$$g(\Delta\lambda_i) = \sum_{x,y} \tilde{p}(x) p(y|x) f_i(x, y) \exp(\Delta\lambda_i f^{\#}(x, y)) - \tilde{p}(f_i) \quad (3.10)$$

turunan dari Persamaan (3.10) adalah :

$$g'(\Delta\lambda_i) = \sum_{x,y} \tilde{p}(x) p(y|x) f_i(x, y) \exp(\Delta\lambda_i f^{\#}(x, y)) f^{\#}(x, y) \quad (3.11)$$

Menggunakan Persamaan (3.8), Persamaan (3.10) dan Persamaan (3.11) dapat dinyatakan sebagai berikut :

$$\Delta\lambda_{n+1} = \Delta\lambda_n - \frac{g(\Delta\lambda_n)}{g'(\Delta\lambda_n)} \quad (3.12)$$

### 3.6 String Matching

Metode *String Matching* sering dilakukan dalam pencarian dan pemberahan terhadap karakter, kata, bahkan dapat digunakan hingga kalimat. Dalam metode *string matching* ini, dapat dibagi kedalam beberapa beberapa kriteria, diantaranya metode string matching dengan pencarian secara *exact*, selain itu metode *string matching* bisa juga digunakan dalam pemberahan kata yang dianggap kurang baku atau terjadi *error* yang dimungkinkan untuk dilakukan koreksi karakter terhadap kata yang mengalami *error* tersebut.

Proses *string matching* biasanya menggunakan cara diberikannya teks  $T_{1..n}$  dengan panjang  $n$  kemudian dicocokan dengan Pola  $P_{1..m}$  dengan panjang  $m$ . Untuk koreksi *error* dapat menggunakan model *levenshtein* atau *edit distance*. Dimana *error* karakter yang ditemukan akan mengalami penambahan, penghapusan atau substitusi karakter agar kata tersebut menjadi kata baku yang dimaksud (Navarro dkk, 2001). Untuk penelitian ini, metode yang digunakan adalah model *string*

*matching* untuk pencarian kata secara *exact* sehingga model koreksi *error* karakter tersebut tidak digunakan.

### **3.7 Forward Chaining**

Metode *Forward Chaining* adalah metode pencarian atau teknik pelacakan ke depan yang dimulai dengan informasi yang ada dan penggabungan *rule* untuk menghasilkan suatu kesimpulan atau tujuan. (Russel dan Norvig, 2003) Teknik pelacakan maju sangat baik jika bekerja dengan permasalahan yang dimulai dengan adanya informasi awal dan ingin dicapai penyelesaian akhir, karena seluruh proses akan dikerjakan secara berurutan maju. *Forward chaining* merupakan metode inferensi yang melakukan penalaran dari suatu masalah untuk menuju solusi permasalahan. Jika klausa premis sesuai dengan situasi (bernilai *TRUE*), maka proses akan menyatakan konklusi.

Tahapan *Forward Chaining* :

1. Permasalahan direpresentasikan ke dalam satu atau beberapa kondisi.
2. Setiap kondisi bergantung dengan *rule* dan *knowledge base*, setiap *rule* akan dibentuk dengan menggunakan logika kondisional *IF-THEN* untuk mengolah informasi dan kondisi sehingga dihasilkan sebuah konklusi.
3. Setiap *rule* dapat menghasilkan kondisi baru dari kesimpulan yang telah diproses sebelumnya, biasanya terletak pada bagian *THEN*. Kondisi baru ini dapat ditambahkan ke kondisi lain untuk dilakukan proses pelacakan berikutnya.
4. Setiap kondisi yang ditambahkan ke sistem akan diproses. Jika ditemui suatu kondisi baru dari konklusi yang diminta, sistem akan kembali ke langkah 2 dan mencari *rule-rule* yang sesuai pada *knowledge base*. Sebaliknya, apabila tidak ditemukan konklusi yang menjadikan kondisi baru, maka sesi pelacakan akan berakhir

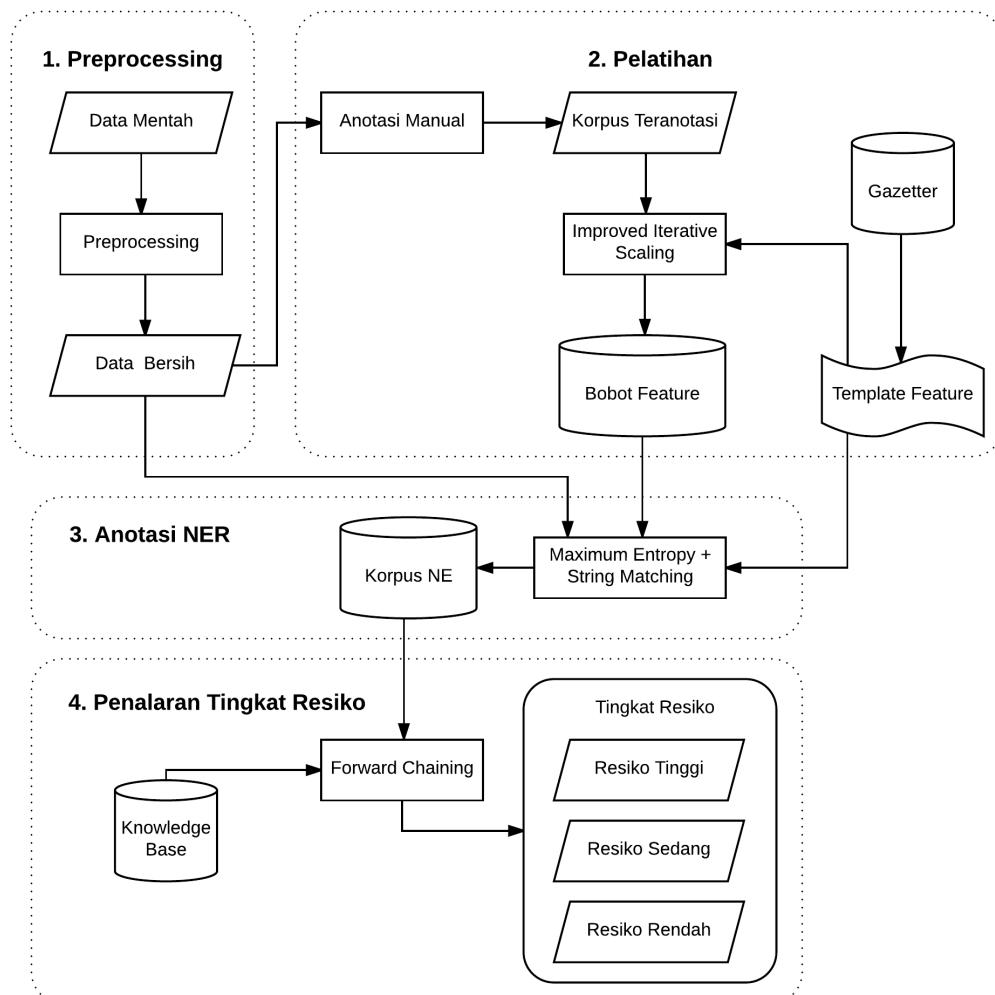
## BAB IV

### ANALISIS DAN RANCANGAN MODEL MONITORING

Bab ini menguraikan tentang deskripsi umum dari penelitian yang telah dilakukan, analisis terhadap metode yang dilakukan, perancangan sistem serta perancangan pengujian sistem.

#### 4.1 Analisis Model *Monitoring*

Pada tahap ini dilakukan analisis data yang telah diperoleh dari tahap sebelumnya, sehingga menghasilkan model arsitektur penelitian yang akan diterapkan, model arsitektur tersebut dapat dilihat pada Gambar 4.1.



Gambar 4.1 Arsitektur Model *Monitoring*

Dari Gambar 4.1 dapat diketahui 4 proses utama pembentukan *model monitoring*, tahap pertama *preprocessing*, tahap kedua pelatihan, tahap ketiga anotasi NER dan tahap ke empat adalah penalaran tingkat resiko. Tahap pertama yaitu *preprocessing*, data *twitter* hasil dari *scraping* akan dibersihkan sesuai pada bagian 4.2 sehingga data yang akan diproses kedalam pelatihan maupun anotasi NER adalah data yang sudah siap untuk diproses kedalam pelatihan atau siap untuk dilakukan pendeksi entitas.

Tahap kedua yaitu pelatihan, didalam pelatihan ini mengakomodasi metode *improved iterative scaling*, dengan terlebih dahulu menyediakan korpus anotasi hasil dari anotasi manual. Metode *improve iterative scaling* ini memanfaatkan *template template feature* yang nantinya digunakan sebagai *input* pencarian pembobot terhadap *feature set* yang telah dihasilkan, hasil dari *feature set* ini akan digunakan untuk deteksi NER menggunakan *maximum entropy*.

Tahap ketiga yaitu anotasi NER, pada tahap ini menggunakan kombinasi *maximum entropy* dan *string matching*, memanfaatkan *feature set lengkap dengan pembobot yang telah dihasilkan pada saat training*, *maximum entropy* melakukan penghitungan probabilitas klasifikasi untuk menemukan entitas yang sesuai terhadap kata yang diproses. Hasil deteksi entitas dari *maximum entropy* kemudian diproses kembali kedalam *string matching*, guna validasi nama lokasi yang umumnya seringkali mengalami WSD (*Word Sense Disambiguation*) pada saat deteksi entitas, Proses anotasi NER dapat dilihat pada bagian 4.4.

Tahap keempat adalah penalaran tingkat resiko, data entitas yang sudah diproses pada tahap anotasi NER akan dilakukan penalaran terhadap basis aturan yang telah ditentukan, dengan tujuan mengelompokan daerah terjangkit demam berdarah kedalam beberapa kelas resiko demam berdarah berdasarkan angka insiden demam berdarah disetiap daerah. Proses penalaran tingkat resiko dapat dilihat pada bagian 4.5.

## 4.2 Preprocessing

*Preprocessing* merupakan proses awal untuk memenuhi syarat ketersediaan data yang sesuai, dalam proses ini data pesan *twitter* dilakukan penyesuaian dengan beberapa tahapan, tahapan tersebut diantaranya :

1. *Case Folding* : Mengubah huruf yang ada pada token menjadi huruf kecil
2. Mengubah kata terbilang menjadi angka, contoh : tiga → 3  
sebagai contoh dapat dilihat pada Tabel 4.1, tahap pemrosesan data pesan *tweet* sehingga menghasilkan data yang sesuai:

Tabel 4.1 Contoh *Preprocessing*

Data Awal	Hasil <i>Preprocessing</i>
tiga Warga Depok Meninggal Akibat DBD <a href="http://bit.ly/1PsLQia">http://bit.ly/1PsLQia</a>	3 warga depok tinggal akibat dbd
bantul waspada 4 orang terkena demam berdarah @pemkabbantul	bantul waspada 4 orang kena demam berdarah

3. *Tokenization* yaitu proses pembagian teks ke dalam token, yaitu teks yang berbentuk paragraf akan dibagi kedalam beberapa tokenisasi kalimat, kemudian dari beberapa token kalimat akan dibagi kedalam bentuk tokenisasi kata
4. *Stopword removal* yaitu proses untuk menghilangkan kata yang tidak relevan pada hasil *parsing*, dalam tahap ini menggunakan 526 *stopword removal* dari (Tala, 2003), Misalnya : 'yang', 'tapi', 'maka', dsb. Selain *stopword removal* dari tala dilakukan penghilangan kata yang masih mengandung *syntax*, diantaranya pada URL dan *Mention*, seperti contoh :
  - a. URL (*Unified Resource Location*) : '<http://goo.gl/fb/69fgxW>'
  - b. *Mention* : '@nyamukdbd'
5. *Stemming* yaitu proses mengubah kata berimbuhan menjadi kata dasar, (dalam tahap ini menggunakan sastrawi *stemming*). Misalnya : Mencapai → capai,

Pada tahap pelatihan hasil dari tahap *preprocessing* tersebut dilakukan penyeleksian data secara *unique* (*filtering* data, agar tidak ada data yang sama) sehingga proses pelatihan dapat dilakukan tanpa mengulang proses data pelatihan yang sama.

#### 4.2.1 Pembentukan *feature*

Sebelum masuk kedalam proses pembobotan menggunakan *improved iterative scaling* maka diperlukan *feature – feature* yang sesuai agar dapat menghasilkan bobot dan juga entitas yang dimaksud. *Feature – feature* tersebut dikombinasikan dengan *gazetter* (kamus data) sebagai pendukung akurasi, untuk itu pada penelitian ini didefinisikan *template feature* yang mengakomodasi *feature* dari nlp.stanford.edu (Klein and Manning, 2003) kemudian dilakukan penyesuaian *template feature* untuk dapat diterapkan kedalam empat deteksi entitas pada pesan *tweet* yang akan digunakan pada model *monitoring*. *Template feature* tersebut dapat dilihat pada Tabel 4.2 sedangkan *gazetter* pada Tabel 4.3

Tabel 4.2 Template *Features*

No	Feature
1	$w_i$ in GAZ ORG
2	$w_i - 1 = \text{"di"}$
3	$w_i - 1$ in GAZ ORG
4	$w_i$ in GAZ LOC
5	$w_i$ is number AND ( $w_i + 1$ in GAZ KND OR $w_i + 1$ in GAZ SP)
6	$w_i$ in GAZ CON
7	$w_i - 1$ in GAZ SP

Tabel 4.3 Definisi Kumpulan *Gazetter*

Kode	Gazetter	Contoh
GAZ SP	<i>Gazetter</i> Satuan Penderita	pasien, kasus, anak, orang, jiwa, penderita
GAZ LOC	<i>Gazetter</i> Kota	sleman, bantul, pekanbaru, banjarmasin
GAZ CON	<i>Gazetter</i> Kondisi	derita, mati, kritis, sakit, kena, rawat
GAZ ORG	<i>Gazetter</i> Organisasi	rs, kabupaten, pemkot, kecamatan, kelurahan

#### 4.2.2 Pembobotan (*Improved Iterative Scaling*)

Pada tahap sebelumnya telah di definisikan *template features* yang dapat digunakan dalam proses pelatihan. Contoh data pelatihan dapat ditunjukkan pada Tabel 4.4, dengan menggunakan contoh data pelatihan tersebut maka dapat dilakukan proses pencarian *binary feature* yang memanfaatkan Persamaan (3.4). Sehingga dapat dihasilkan *binary feature* yang dimiliki oleh  $\vec{x}$  (vector kata) terhadap  $y$  (anotasi entitas). Maka dapat dihasilkan penjabaran *feature set* yang dimiliki oleh  $x$  terhadap  $y$ , penjabaran *binary feature* dapat dilihat pada Tabel 4.5

Tabel 4.4 Data pelatihan

<b>Kata</b>	4	warga	meninggal	riau	waspada	demam	berdarah
<b>Tag</b>	NUM		CON	LOC			
<b>Posisi</b>	1	2	3	4	5	6	7

Tabel 4.5 Penjabaran  $\vec{x}$  Terhadap *Feature Anotasi*

<b>No</b>	$\vec{x}$	$y$								
		ORG			LOC			NUM	CON	
		f1	f2	f3	f2	f3	f4	f5	f6	f7
1	4	0	0	0	0	0	0	1	0	0
2	warga	0	0	0	0	0	0	0	0	1
3	meninggal	0	0	0	0	1	0	0	1	1
4	riau	0	0	0	0	0	1	0	0	0
5	waspada	0	0	0	0	0	0	0	0	0
6	demam	0	0	0	0	0	0	0	0	0
7	berdarah	0	0	0	0	0	0	0	0	0

Dari Tabel 4.5 maka dapat dilakukan penghitungan bobot berdasarkan *word x* dan *feature anotasi y* yang dihasilkan. Proses pembobotan mengakomodasi algoritma *improved iterative scaling*, algoritma tersebut dapat dilihat pada (Bab 3.5.2). Berikut ini adalah contoh pembentukan *feature set* dengan melakukan *training* label NUM dan Nilai *feature 5* ( $f5 = 1$ ) yaitu pada kata “4” yang telah

terdeteksi sebagai NUM (angka kejadian). Langkah perhitungan dimulai dari *history* pertama mencari nilai  $g(\Delta\lambda_1)$  ditunjukan pada Tabel 4.6,  $\tilde{p}(x)$  merupakan prediksi kemunculan *word*  $x$  terhadap jumlah *word* pada data *training history* pertama  $\frac{1}{7} = 0.143$ , sedangkan  $p(y|x)$  merupakan probabilitas peluang *tag* terhadap *word*  $x$  pada data *training history* pertama  $\frac{1}{3} = 0.333$ , Sedangkan  $f_1(x, y)$  merupakan nilai *history feature* pertama dari *feature* ke-5 yang diambil dari Tabel 4.5. Selain itu  $\Delta\lambda_1$  merupakan nilai kenaikan bobot dari  $\lambda_1$ , untuk iterasi pertama  $\Delta\lambda_1 = 0$ ,  $f^*(x, y)$  merupakan jumlah banyaknya *feature* pada *word*  $x$  yang mempunyai nilai sejenis (pada *word* “4” *value feature* = 1 hanya ada satu).  $\tilde{p}(f_1)$  dihasilkan dari persamaan 3.7,  $\tilde{p}(x, y)$  dihasilkan dari  $\frac{1}{N(word)} \times$  berapa kali  $(x, y)$  terjadi dalam *history* ke - i dapat dihitung  $\tilde{p}(f_1) = \frac{1}{7} \times 1 = 0.143$ . Sedangkan Tabel 4.7 merupakan hasil perhitungan dari  $g'(\Delta\lambda_1)$ . Dari perhitungan  $g(\Delta\lambda_1)$  dan  $g'(\Delta\lambda_1)$  dapat dihasilkan nilai  $\Delta\lambda_{1+1} = 0 - \frac{-0.095}{0.048} = 1.979$  ditunjukan pada Tabel 4.8, sehingga nilai  $\Delta\lambda_1 = 1.979 - 0 = 1.979$ . Pencarian pembobot dilakukan seperti itu seterusnya sampai batas *convergence* terpenuhi. Apabila proses IIS ini telah mencapai batas *convergence*, maka tahap IIS (2b) menjadi  $1.979 - 0 = 1.979$  nilai tersebut dapat digunakan sebagai *update*  $\lambda$  (pembobot) pada label NUM dengan nilai *feature* 5 ( $f_5 = 1$ ) dapat dilihat pada Tabel 4.9. *Feature-feature* yang dihasilkan akan membentuk *feature set* yang nantinya digunakan untuk proses pencarian entitas menggunakan *maximum entropy*. Pencarian *feature set* menggunakan *improve iterative scaling* dilakukan seperti itu seterusnya sampai *convergence* batas *cutoff*  $\min\_All$  (*minimum rata-rata delta log likelihood*) yang telah ditentukan. Didalam *nltk* nilai *log likelihood* dihasilkan dari log rata-rata probabilitas entitas keseluruhan saat iterasi pelatihan ke  $i$  sesuai Persamaan 4.1. dimana  $y$  adalah label dan  $x$  adalah *feature set* pada *word*  $x$ .

$$ll_i = \log \left( \text{mean} \left( \sum_x p(y|x) \right) \right) \quad (4.1)$$

Tabel 4.6 Hasil Perhitungan  $g(\Delta\lambda_1)$  Sesuai Persamaan 3.10

$\vec{x}$	$\tilde{p}(x)$	$p(y x)$	$f_1(x,y)$	$\Delta\lambda_1$	$f^{\#}(x,y)$	$\exp(\Delta\lambda_1 f^{\#}(x,y))$	$\sum_{x,y} \tilde{p}(x) p(y x) f_1(x,y) \exp(\Delta\lambda_1 f^{\#}(x,y)) - \tilde{p}(f_1)$
4	0.143	0.333	1	0	1	1	-0.095

Tabel 4.7 Hasil Perhitungan  $g'(\Delta\lambda_1)$  Sesuai Persamaan 3.11

$\vec{x}$	$\tilde{p}(x)$	$p(y x)$	$f_1(x,y)$	$\Delta\lambda_1$	$f^{\#}(x,y)$	$\exp(\Delta\lambda_1 f^{\#}(x,y))$	$\sum_{x,y} \tilde{p}(x) p(y x) f_1(x,y) \exp(\Delta\lambda_1 f^{\#}(x,y)) f^{\#}(x,y)$
4	0.143	0.333	1	0	1	1	0.048

Tabel 4.8 Hasil Perhitungan untuk Persamaan 3.12

$\Delta\lambda_1$	$g(\Delta\lambda_1)$	$g'(\Delta\lambda_1)$	$\Delta\lambda_{1+1}$
0	-0.095	0.048	1.979

Tabel 4.9 Hasil Pencarian feature f5=1 dan label=NUM Pada History Pertama

History	Feature	Feature Value	Entity	$\lambda$
1	F5	1	NUM	1.979

### 4.3 Anotasi NER (*Maximum Entropy*)

Proses IIS (*improved iterative scaling*) telah menghasilkan *feature set* yang mempunyai bobot setiap *feature*-nya, maka dapat dilakukan anotasi kata ke dalam entitas yang dikenali. Menggunakan data pelatihan yang belum teranotasi, maka dibutuhkan Persamaan (3.4) untuk menghasilkan *binary feature*, untuk diproses ke dalam model *maximum entropy*, hasil *binary feature* ditunjukkan pada Tabel 4.10

Tabel 4.10 Binary Feature Data Pelatihan

No	$\vec{x}$	y								
		ORG			LOC			NUM	CON	
		f1	f2	f3	f2	f3	f4	f5	f6	f7
1	3	0	0	0	0	0	0	1	0	0
2	warga	0	0	0	0	0	0	0	0	0
3	situbondo	0	0	1	0	1	1	0	0	1
4	meninggal	0	0	0	0	0	0	0	1	0
5	karena	0	0	0	0	0	0	0	0	0
6	dbd	0	0	0	0	0	0	0	0	0

Dari kasus Tabel 4.10 dapat diambil contoh anotasi pada pelatihan kata “situbondo” dengan urutan tokenisasi pada posisi 3, maka dapat dihitung nilai normalisasi “situbondo” menggunakan Persamaan (3.2) dengan  $x$  sebagai *context* kata “situbondo” dan  $y$  adalah anotasi, perhitungan anotasi tersebut dapat dihasilkan pada Tabel 4.11.

Tabel 4.11 Normalisasi Faktor Peluang Kebenaran untuk  $x = \text{“situbondo”}$

y	i	$\lambda_i$	$f_i(x, y)$	$\lambda_i f_i(x, y)$	$\exp\left(\sum_{i=1}^{N_F} \lambda_i f_i(x, y)\right)$	$\sum_{y \in Y} \exp\left(\sum_{i=1}^{N_F} \lambda_i f_i(x, y)\right)$
ORG	1	0.610	0	0	0.753	4.604
	2	0.032	0	0		
	3	-0.284	1	-0.284		
LOC	2	0.426	0	0	1.736	4.604
	3	0.408	1	0.408		
	4	0.144	1	0.144		
NUM	5	0.764	0	0	1	
CON	6	0.509	0	0	1.114	4.604
	7	0.138	1	0.14		

Dari hasil normalisasi kata “situbondo” tersebut, dapat dilakukan pencarian  $p(y|x)$  menggunakan Persamaan (3.1) ditunjukan pada Tabel 4.12.

Tabel 4.12 Peluang maximum entropy kata “situbondo”

$p(ORG situbondo)$	$p(LOC situbondo)$	$p(NUM situbondo)$	$p(CON situbondo)$
0.164	0.377	0.217	0.242

Peluang *maximum entropy* kata “situbondo” menghasilkan nilai paling tinggi pada anotasi LOC dengan nilai peluang 0.377 , maka NER melakukan pelabelan kata “situbondo” ke dalam entitas lokasi (LOC), dari hasil keseluruhan pengenalan entitas dapat dihasilkan entitas terpilih seperti pada Tabel 4.13.

Tabel 4.13 Hasil Anotasi

Kata	3	warga	situbondo	meninggal	karena	dbd
Tag	NUM	O	LOC	CON	O	O
Posisi	1	2	3	4	5	6

#### 4.4 Kombinasi NER (*Maximum Entropy* dengan *String Matching*)

Dikarenakan anotasi NER *maximum entropy* (*MaxEnt*) pada penelitian ini hanya bisa melakukan deteksi entitas satu kata tunggal, bukan berdasarkan struktur penulisan secara frase (satu kesatuan kata yang tidak boleh dipisah). Maka perlunya penggabungan kata (*grouping*), kata yang dimaksud adalah memang benar – benar terbukti menjadi entitas kata dengan penulisan secara *frase*, entitas tersebut banyak ditemukan pada entitas lokasi.

Entitas lokasi yang sudah ditemukan (hasil dari deteksi *maximum entropy* sebelumnya) kemudian di proses kedalam *string matching* untuk memastikan apakah entitas tersebut benar – benar lokasi atau kata lain yang mempunyai penulisan yang sama, akan tetapi mempunyai pemaknaan arti yang berbeda. Pencocokan dalam *string matching* tersebut menggunakan data nama – nama daerah yang ada di negara indonesia, data tersebut diambil dari <http://mfdonline.bps.go.id>. Berikut ini adalah contoh perbandingan antara anotasi menggunakan *maximum entropy* dengan anotasi kombinasi *maximum entropy* dan *string matching* kata entitas dapat dilihat pada Tabel 4.14.

Tabel 4.14 Perbandingan MaxEnt dengan Kombinasi

Contoh	Metode	Pesan <i>Tweet</i>
1	<i>MaxEnt</i>	2/NUM warga jakarta/LOC selatan/LOC terkena/CON demam berdarah
	<i>MaxEnt + String Matching</i>	2/NUM warga jakarta/LOC selatan/LOC terkena/CON demam berdarah
2	<i>MaxEnt</i>	air/LOC jakarta/LOC sangat disukai nyamuk demam berdarah
	<i>MaxEnt + String Matching</i>	air jakarta/LOC sangat disukai nyamuk demam berdarah

Dapat dilihat pada Tabel 4.10 pada contoh 1 deteksi LOC ada 2 entitas, deteksi, dalam kombinasi *Maxent + String Matching*, dua lokasi tersebut akan dideteksi 1 lokasi “jakarta selatan” struktur penulisan secara *frase*. Sedangkan pada contoh 2 terdapat kesalahan deteksi LOC, pada contoh 2 tersebut lokasi sebenarnya hanya ada satu entitas lokasi dengan kata “jakarta”, maka dapat dihilangkan deteksi entitas lokasi pada kata “air”.

#### 4.5 Penalaran Tingkat Resiko

Menggunakan *knowledge base* yang diambil dari (Pusat Data dan Surveilans Epidemiologi Kementerian Kesehatan RI, 2010) ditunjukan pada Tabel 4.15 maka dapat dilakukan proses penalaran untuk mengelompokan tingkat resiko kejadian demam berdarah di suatu daerah menggunakan *forward chaining*.

Tabel 4.15 *Knowledge base*

Rule No	Premis 1	Premis 2	Konklusi
1	LOC=TRUE	NUM<20	Resiko Rendah
2	LOC=TRUE	20<=NUM<=55	Resiko Sedang
3	LOC=TRUE	55<NUM	Resiko Tinggi

Dari hasil proses anotasi pada Tabel 4.13 dilanjutkan menuju proses penalaran menurut *knowledge base* yang telah didefinisikan pada Tabel 4.15 dengan keterangan Premis 1 LOC=TRUE adalah entitas lokasi harus terpenuhi (tidak boleh

kosong), sedangkan pada Premis 2 entitas NUM atau angka insiden juga harus terpenuhi (tidak boleh kosong) agar dapat dilakukan logika perbandingan nilai batasan yang nantinya berpengaruh pada hasil kesimpulan. Dapat dilakukan contoh dari data yang diambil dari Tahap 4.3, Anotasi NER menghasilkan notasi LOC = “situbondo” dan NUM = 3 maka dapat dilakukan proses penalaran tingkat resiko, sehingga menghasilkan lokasi “situbondo” berada pada tingkat “Resiko Rendah”.’

Sebelum dilakukannya penalaran tingkat resiko, perlu dilakukan seleksi data yang benar - benar mengandung lokasi dan angka kejadian, apabila benar telah ditemukan data pesan *tweet* yang mengandung entitas LOC dan NUM, maka proses penalaran akan dilakukan. Beberapa permasalahan dihadapi dalam penyediaan data entitas LOC dan NUM untuk proses penalaran, permasalahan tersebut perlu solusi agar pesan *tweet* dapat diproses kedalam langkah penalaran menggunakan *forward chaining*. Permasalahan dan solusi dapat dijelaskan sebagai pada Tabel 4.16 :

Standarisasi pesan *tweet* :

\*satu pesan *tweet* terdiri dari satu entitas LOC dan satu entitas NUM.

Tabel 4.16 Permasalahan dan Solusi Pada Proses Penalaran

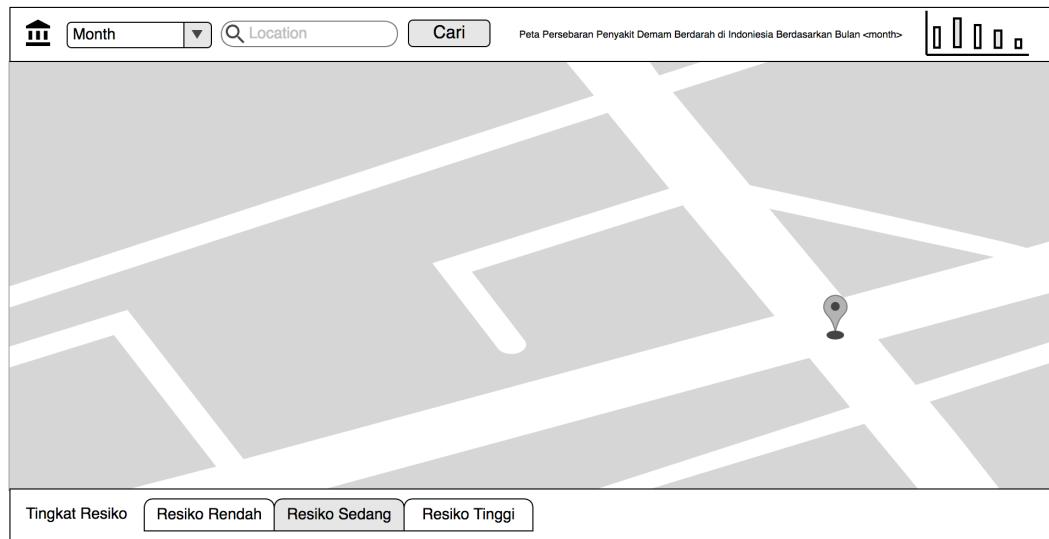
Permasalahan	Solusi
Satu pesan <i>tweet</i> bisa terdiri dari satu entitas LOC dan dua atau lebih entitas NUM	Perlunya dilakukan deteksi nilai maksimum NUM, apabila didapatkan nilai maksimum NUM maka ambil nilai yang paling besar tersebut.
Satu pesan <i>tweet</i> bisa terdiri dari dua atau lebih entitas LOC dan satu entitas NUM	Setiap lokasi akan diberikan nilai entitas NUM yang sama.
Satu pesan <i>tweet</i> bisa terdiri dari dua atau lebih entitas LOC dan dua atau lebih entitas NUM.	Perlunya dilakukan deteksi kedekatan kata entitas LOC dan NUM. Apabila kedekatan kata sudah didapat maka dicocokan berdasarkan kedekatan entitas LOC dan NUM masing - masing

## 4.6 Rancangan Antar Muka Model *Monitoring*

Tampilan antar muka model *monitoring* dirancang dengan menggunakan konsep *wizard*, agar pengguna dapat dengan mudah memahami hasil proses model *monitoring* yang telah dibuat. Rancangan antar muka tersebut dibagi kedalam 3 rancangan antar muka diantaranya antar muka peta persebaran penyakit demam berdarah, antar muka data persebaran demam berdarah dan antar muka *history* kejadian demam berdarah. Ketiga rancangan antar muka tersebut dapat dijelaskan sebagai berikut.

### 4.6.1 Antar muka peta persebaran penyakit demam berdarah

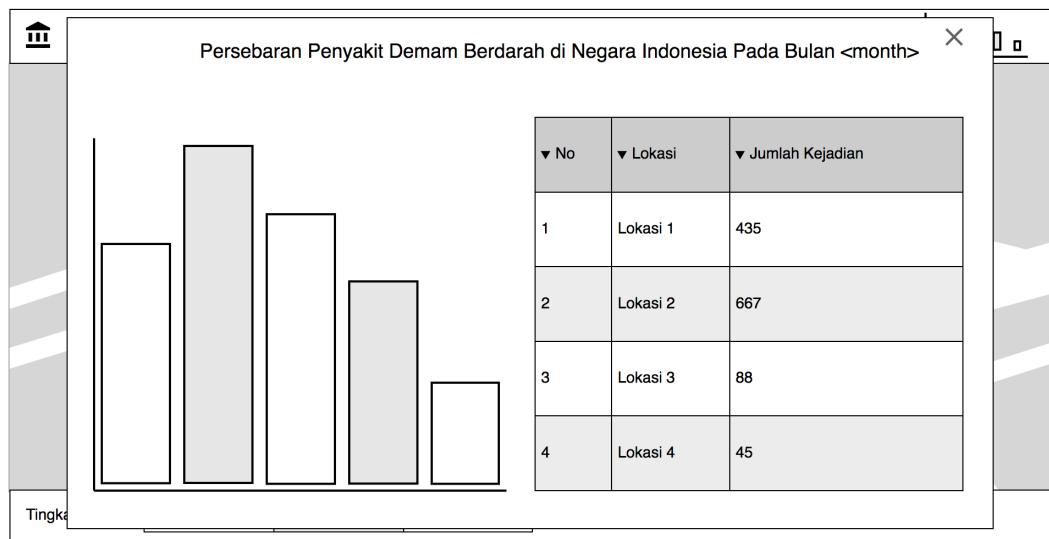
Kejadian demam berdarah selama satu bulan dapat dilihat melalui peta persebaran penyakit demam berdarah yang ditunjukan dengan adanya titik lokasi kejadian pada suatu daerah. Data peta tersebut hasil dari *parsing tweet* yang mengandung entitas LOC dan NUM kemudian dicari NUM maksimal setiap lokasi pada bulan tersebut, kemudian dipetakan menurut koordinat bujur dan lintang yang diambil dari geonames.org. Rancangan antar muka peta persebaran demam berdarah dapat dilihat pada Gambar 4.4



Gambar 4.4 Peta Persebaran Penyakit Demam Berdarah

#### 4.6.2 Antar muka data persebaran penyakit demam berdarah

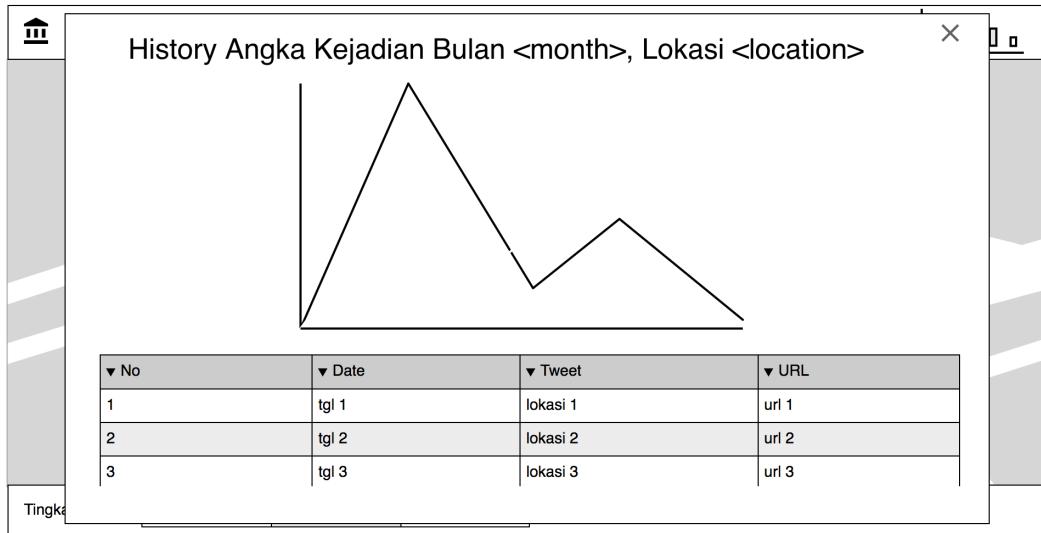
Antar muka data persebaran penyakit demam berdarah selama satu bulan direpresentasikan menggunakan grafik batang, sehingga dapat diketahui dengan mudah perbandingan jumlah kejadian demam berdarah disuatu lokasi dengan lokasi lainnya. Dikarenakan data grafik batang tersebut diurutkan berdasarkan data kejadian paling tinggi menuju kejadian paling rendah disuatu daerah, rancangan antar muka grafik dapat dilihat pada Gambar 4.5.



Gambar 4.5 Antar Muka Persebaran Penyakit Demam Berdarah

#### 4.6.3 Antar muka *history* kejadian demam berdarah

*History* kejadian demam berdarah disajikan dalam bentuk grafik garis pada periode bulanan, sehingga memudahkan pengguna untuk melakukan *monitoring* perkembangan kejadian demam berdarah disuatu daerah. Selain grafik garis, data tersebut juga disajikan dalam bentuk tabel, yang didalamnya berisi data *tweet* sumber berita yang sudah terdeteksi entitas – entitas yang diketahui. Data *tweet* tersebut merupakan sumber data acuan terbentuknya *monitoring* kejadian demam berdarah. Rancangan antar muka *history* demam berdarah dapat dilihat pada Gambar 4.6



Gambar 4.6 Antar Muka *History* Kejadian Demam Berdarah Pada Suatu Daerah

#### 4.7 Pengujian dan Evaluasi

Dalam *Information Retrieval* terdapat beberapa tolak ukur dalam menentukan kinerja dari sebuah sistem diantaranya adalah akurasi, *F-Measure*, *precision*, dan *recall*. Nilai-nilai tersebut dapat digunakan untuk penentuan tolak ukur model yang paling bagus, salah satunya pada saat melakukan pencarian nilai optimal dari pada waktu training.

Akurasi tidak digunakan dalam pengujian ini, dikarenakan mayoritas token bukan entitas. Pengenalan entitas dapat diukur menggunakan *precision*, *recall*, dan *harmonic mean* dari *precision* dan *recall* yakni *F-measure* (Dermawan, 2016). Untuk melakukan penghitungan *precision*, *recall* dan *F-Measure* maka harus diketahui *True Positif (TP)*, *False Positif (FP)*, *False Negative (FN)* dan *True Negative (TN)* seperti yang disajikan pada Tabel 4.17 Keterangan pengujian.

Tabel 4.17 Keterangan Pengujian

	Benar	Tidak Benar
Terpilih	TP	FP
Tidak Terpilih	FN	TN

*Precision (P)* pada pengenalan entitas adalah persentase model *monitoring* dapat melakukan pelabelan benar dari entitas yang dikenali. Nilai *precision* dapat dihitung dengan Persamaan (4.1).

$$precision = \frac{TP}{TP + FP} \quad (4.1)$$

*Recall* (R) pada pengenalan entitas adalah persentase seberapa banyak entitas dapat dikenali oleh model *monitoring*. Nilai *recall* dapat dihitung dengan Persamaan (4.2).

$$recall = \frac{TP}{TP + FN} \quad (4.2)$$

*F-measure* pada pengenalan entitas adalah penghitungan evaluasi temu kembali kalimat yang mempunyai entitas benar dengan kalimat yang memiliki entitas hasil dari pelabelan model *monitoring*. Nilai *F-measure* dapat dihitung dengan Persamaan (4.3):

$$F_1 = 2 \left( \frac{P \cdot R}{R + P} \right) \quad (4.3)$$

## **BAB V**

## **IMPLEMENTASI**

### **5.1 Pembangunan Model *Monitoring***

*Core Model monitoring* ini dibangun menggunakan Python 2.7.10 dengan sistem operasi OS X El Capitan 10.11.3. Dikembangkan kedalam model sistem berbasis web agar pengguna dapat dengan mudah memahami dan melakukan operasi *monitoring*. Keseluruhan model *monitoring* ini membutuhkan beberapa basis data, modul atau *python-library* dari luar diantaranya :

1. NLTK 3.2.1  
Merupakan paket *library* yang cukup lengkap digunakan untuk *text mining*.  
Beberapa *library* digunakan untuk *preprocessing* dan pembentukan NER
2. MongoDB 3.2.8  
Digunakan sebagai basis data penyimpanan dengan bentuk *array* teks.
3. PyMongo 3.3.0  
Digunakan sebagai konektor antara python dengan MongoDB, sehingga data dapat diproses dari atau menuju MongoDB.
4. Sastrawi 1.0.1  
Digunakan sebagai *stemmer* kata berbahasa indonesia.
5. Scrapy 1.1.1  
Digunakan untuk mengambil data *tweet* secara *scraping*.
6. Numpy 1.11.1  
Digunakan untuk merepresentasikan dan mengoperasikan *array* matrik dalam jumlah besar.
7. Flask 0.11.1  
Digunakan sebagai *framework* pengembangan aplikasi berbasis web.
8. Jinja 2.8  
Digunakan sebagai antar muka Flask kedalam tampilan *web browser*.
9. GoogleMap 2.4.4  
Digunakan untuk memetakan persebaran penyakit demam berdarah
10. Pickleshare 1.0.5  
Digunakan untuk penyimpanan hasil *training improved iterative scaling*

## 5.2 Implementasi *Scraping* data

*Scraping* dilakukan untuk pemenuhan data *monitoring* demam berdarah, sebagian besar data yang diperlukan diambil menggunakan *scrap* halaman situs *website*, data tersebut adalah pesan *tweet* yang diambil dari *twitter.com* dan data kota besar di negara indonesia yang diambil dari *geonames.org*. *Scraping* kedua *website* tersebut membutuhkan *library scrapy*, Implementasi *scraping* halaman situs tersebut dapat dilihat pada implementasi berikut.

### 5.2.1 *Scraping* *twitter.com*

Data utama persebaran demam berdarah diambil dari situs *twitter.com* dengan mengambil pesan *tweet* kemudian disimpan kedalam basis data. Implementasi *scraping* halaman *twitter.com* ditunjukan pada Gambar 5.1 *Scraping tweet* dilakukan dengan langkah dan aturan sebagai berikut ;

1. *Scraping* dilakukan dengan dimulai dari halaman pencarian dengan kata kunci pencarian “demam berdarah”, “dengue fever”, “dbd”, “dengue hemorrhagic”, “dhf” atau “sakit db”. Sedangkan untuk *hashtag* “demaberdarah”, “denguefever”, “dbd”, “dhf” atau “sakit db”. Sedangkan tanggal pencarian di setting berdasarkan tanggal mulai pencarian dan tanggal akhir pencarian selama satu bulan.
2. Untuk mengatasi *asynchronous* maka menggunakan url dari class “w-button-more” baris 23, kemudian lakukan *scraping* data *asynchronous* yang telah diambil dari server.
3. Data hasil *scraping* tersebut kemudian disimpan kedalam MongoDB dengan nama *collection* sesuai nama bulan *scraping*.

```

1  from scrapy.spiders import Spider
2  from scrapy.http.request import Request
3  from scrapy.selector import Selector
4  from tweetmobile.items import TweetmobileItem
5  from w3lib.html import remove_tags
6  import re
7
8  class TwitterSpider(Spider):
9      index = 0
10     start = '2015-02-28'
11     end = '2015-04-01'
12     name = "twitter_mobile"
13     allowed_domains = ["mobile.twitter.com"]
14
15     start_urls = [
16         "https://mobile.twitter.com/search?f=tweets&vertical=default&q=%22demam%20berdarah%22"
17     ]
18
19     def parse(self, response):
20         s = Selector(response)
21         next_link = s.xpath('//div[@class="w-button-more"]/a/@href').extract()
22         if len(next_link):
23             yield Request("https://mobile.twitter.com"+next_link[0], callback=self.parse)
24         itemselector = Selector(response).xpath('//*[@id="main_content"]/div/div[3]/table')
25         for sel in itemselector:
26             self.index += 1
27             item = TweetmobileItem()
28             item['index'] = self.index
29             item['username'] = ''.join(
30                 map(unicode.strip, sel.xpath('tr[1]/td[2]/a/div/text()').extract()))
31             tweet = remove_tags(''.join(
32                 map(unicode.strip, sel.xpath('tr[2]/td/div').extract())))
33                 .replace('&', '&').replace(' ', '').replace('\n', '').replace('\n\n', '\n')
34             item['text_tweet'] = u''+tweet
35             item['original_tweet'] = ''.join(sel.xpath('tr[2]/td/div/div').extract())
36             item['time_tweet'] = ''.join(
37                 map(unicode.strip, sel.xpath('tr[1]/td[3]/a/text()').extract()))
38             item['url'] = ''.join(
39                 map(unicode.strip, sel.xpath('tr[2]/td/div/@data-id').extract()))
40             item['data_id'] = ''.join(
41                 map(unicode.strip, sel.xpath('tr[3]/td/span[1]/a/@href').extract()))
42             yield item

```

Gambar 5.1 Implementasi Scraping twitter.com

### 5.2.2 Scraping geonames.org

Data lokasi kota besar yang diambil dari *geonames.org* lengkap dengan koordinat lintang dan bujur yang nantinya digunakan untuk proses pemetaan kedalam peta google. Implementasi dari *scraping* dapat ditunjukan pada Gambar 5.2. *Scraping* kota besar dilakukan dengan aturan dan langkah sebagai berikut.

1. *Scraping* dimulai dari halaman menu pencarian lokasi kota yang ada di negara indonesia,
2. Data diambil menggunakan proses *scraping* dari halaman yang tampil pada halaman situs *geonames*, berulang sampai halaman terakhir dapat dilihat pada baris 18.

3. Data hasil *scraping* tersebut disimpan kedalam database *array* teks menggunakan MongoDB

```

1  from scrapy.spiders import Spider
2  from scrapy.http.request import Request
3  from scrapy.selector import Selector
4  from scrapy.loader import ItemLoader
5  from kotabesar.items import KotabesarItem
6
7  class GeonamesSpider(Spider):
8      name = "geonames"
9      allowed_domains = ["geonames.org"]
10     start_urls = [
11         "http://www.geonames.org/advanced-search.html?q=&country=ID&featureClass=P&continentCode=AS"
12     ]
13
14     def parse(self, response):
15         s = Selector(response)
16         next_link = s.xpath('//div[@id="search"]/a[last()]/@href').extract()
17         if len(next_link):
18             yield Request("http://www.geonames.org/" + next_link[0], self.parse)
19         itemselector = Selector(response).xpath('//div[@id="search"]/table/tr')
20         for sel in itemselector:
21             item = KotabesarItem()
22             item['urutan'] = ''.join(sel.xpath('td[1]/small/text()').extract())
23             item['kota'] = ''.join(sel.xpath('td[2]/a/text()').extract())
24             item['alias'] = ''.join(sel.xpath('td[2]/small/text()').extract())
25             item['bagian'] = ''.join(sel.xpath('td[3]/text()').extract()).encode('utf8')
26             strBagian = str(item['bagian'])
27             if strBagian:
28                 item['bagian'] = strBagian.replace(',', '')
29                 item['feature'] = ''.join(map(unicode.strip, sel.xpath('td[4]/text()').extract()))
30                 item['populasi'] = ''.join(sel.xpath('td[4]/small/text()').extract())
31                 strPopulasi = str(item['populasi'])
32                 if strPopulasi:
33                     item['populasi'] = (strPopulasi.replace('population ','')).replace(',','')
34                 item['latitude'] = ''.join(sel.xpath('td[5]/text()').extract())
35                 item['longitude'] = ''.join(sel.xpath('td[6]/text()').extract())
36             yield item

```

Gambar 5.2 Implementasi *Scraping geonames.org*

### 5.3 Implementasi *Preprocessing*

*Preprocessing* yang dilakukan seperti tahap yang telah dibahas pada Bab 4.2 dengan urutan *case folding*, konversi dari string ke angka, tokenisasi, stopword removal dan juga *stemming*. Implementasi *preprocessing* tersebut dapat dilihat pada Gambar 5.3 dengan fungsi utama *process* baris 45 -51, Dari fungsi *process* tersebut data kalimat *twitter* akan dibersihkan sehingga menghasilkan data yang siap untuk diproses pada tahap pelatihan dan juga anotasi. Pada baris 24 fungsi *stopword\_removal* dilengkapi dengan *tag\_removal*, didalam *tag removal* tersebut kalimat *tweet* yang masih mengandung *emoticon*, symbol dan juga url dibersihkan.

Kendala dalam pembersihan url adalah pada filterisasi kata, karena url tidak hanya berakhiran .com atau .id masih banyak lagi *top level domain* yang sering digunakan, untuk itu pada pembersihan url memerlukan daftar *top level domain*, daftar top-level-domain tersebut diambil dari <https://publicsuffix.org/list>.

Pada proses *stemming* ini menggunakan *sastrawi stemming*, *sastrawi stemming* yang digunakan memiliki kendala dalam membentuk kata dasar, terlebih pada kata yang memiliki artian lokasi, hasil stemming membuktikan banyak nama kota yang seharusnya tidak dijadikan kata dasar akan tetapi oleh *sastrawi stemming* nama kota tersebut dijadikan kata dasar, sehingga perlu filterisasi kata untuk dimasukan kedalam proses stemming, filterisasi kata tersebut membutuhkan data kota diseluruh Indonesia, data kota tersebut diambil dari <http://mfdonline.bps.go.id>. Selain kata lokasi, filterisasi juga dilakukan untuk kamus kata organisasi dan kata sambung, seperti contoh : “kecamatan”, “kelurahan”, “di”, “dan”.

```

9   class Preprocessing :
10     client = MongoClient()
11     db = client.indo_db
12
13     factory = StemmerFactory()
14     stemmer = factory.create_stemmer()
15
16     tag = r.RemoveTag()
17     stopword = s.StopwordRemoval()
18     func = func.Func()
19
20   def string_to_number(self, sentence):
21     return self.func.terbilang_to_number(sentence)
22
23   def stopword_removal(self, sentence, condition):
24     sentence_clean_tag = self.tag.tag_removal(sentence.encode("utf8"), condition) #clean tag
25     return self.stopword.stopword_removal(sentence_clean_tag, condition)
26
27   def stemming(self, paragraph):
28     sentence = sent_tokenize(paragraph)
29     train = []
30     for index, data in enumerate(sentence):
31       tokenize = word_tokenize(data)
32       div_sentence = []
33       for word in tokenize:
34         check_kota = (self.db.location.find({"$text": {"$search": word.lower()}}).count())>=1
35         if not check_kota and word not in g.gaz_o and word not in g.gaz_org:
36           #apabila kata bukan kota, organisasi dan kata sambung maka dibuat kata dasar
37           sent_stem = self.stemmer.stem(word.encode("utf8"))
38           word = sent_stem
39           div_sentence.append(word)
40       train.append(" ".join(div_sentence))
41
42     return train
43
44   def process(self, sentence, condition):
45     a = sentence.lower()
46     b = self.string_to_number(a)
47     #bedakan proses training dan anotasi menggunakan condition "ner"/"train"
48     c = self.stopword_removal(b, condition)
49     d = self.stemming(c)
50     return d

```

Gambar 5.3 Implementasi *Preprocessing*

Hasil dari *preprocessing* tersebut kemudian akan dilanjutkan menuju proses *binary feature* yang nantinya digunakan untuk pelatihan pencarian bobot menggunakan *improved iterative scaling* atau deteksi entitas menggunakan *maximum entropy*. Implementasi pencarian *binary feature* dapat dilihat pada bagian 5.4 Implementasi *Binary Features*.

## 5.4 Implementasi *Binary Features*

*Binary feature* yang diterapkan pada penelitian ini sesuai dengan *template feature* yang telah dibuat pada Bab 4.2.1, dengan 7 *feature* yang saling berbeda. *Feature – feature* tersebut diterapkan pada setiap kata pada kalimat *tweet* yang akan dilatih ataupun kata yang akan dilakukan pencarian entitas. Tahap *binary feature* dapat dilihat pada Gambar 5.4.

```

36     def binary_feature(self, sentence, type_feature):
37         self.sentence = sentence
38         #define unclean temporary array data train and label
39         train = []
40         #jika training iis maka lakukan pencarian binary feature dengan label
41         #contoh : (dict(f1=0, f2=0, f3=0, f4=0, f5=1, f6=0, f7=1}, "NUM")
42         if type_feature == "train_iis":
43             for index, data in enumerate(sentence):
44                 label = []
45                 token = word_tokenize(data)
46                 for index, data in enumerate(token):
47                     if "/" in data :
48                         #add label to array
49                         label.append(self.lbl.search(token[index]).group(1))
50                         #add word to array
51                         token[index] = self.w.search(token[index]).group(1)
52                     else:
53                         label.append("0")
54                 for index, data in enumerate(token):
55                     #feature processing panggil class feature
56                     featuretrain = f.Feature()
57                     result = featuretrain.template_feature(token, label, index)
58                     #result = template_feature(token, label, index)
59                     train.append(result)
60             else:
61                 #jika anotasi ner maka hanya melakukan pencarian binary feature, tidak dengan label
62                 #contoh : (dict(f1=0, f2=0, f3=0, f4=0, f5=0, f6=0, f7=1})
63                 token = word_tokenize(sentence)
64                 label = []
65                 for index, data in enumerate(token):
66                     #feature processing panggil class feature
67                     featuretrain = f.Feature()
68                     result = featuretrain.template_feature(token, label, index)
69                     #result = template_feature(token, label, index)
70                     train.append(result)
71             # filter array empty/none karena Other atau entitas 0 tidak diproses
72             train_set = filter(None, train)
73         return train_set

```

Gambar 5.4 Implementasi *Binary Feature*

Dari Gambar 5.4 dapat diketahui process *binary feature* terbagi menjadi 2 bagian proses yang berbeda, proses pertama adalah proses *binary feature* untuk pelatihan dan proses kedua adalah proses *binary feature* untuk pencarian entitas. Pada proses pelatihan hanya melakukan proses pembentukan *binary feature* pada kata yang terlabeli *manual*, proses tersebut membentuk *dictionary* berupa *binary feature* dengan label yang nantinya digunakan pada proses selanjutnya yaitu *improve iterative scaling*. Sedangkan pada proses pencarian entitas melakukan proses pembentukan *binary feature* pada semua kata dengan membentuk *dictionary* berupa *binary feature* tanpa label, *dictionary* tersebut nantinya digunakan untuk proses deteksi entitas menggunakan *maximum entropy*.

## 5.5 Implementasi Pencarian Bobot Optimal

Hasil *binary feature* pada saat pelatihan telah membentuk *dictionary* berupa *binary feature* dengan label, hasil tersebut kemudian diteruskan menuju proses pencarian bobot optimal menggunakan *improve iterative scaling* memanfaatkan *library nltk*. Didalam *nltk* proses iterasi menggunakan *improved iterative scaling* dapat dibatasi menggunakan *cutoff*, *cutoff* yang dipakai pada penelitian ini adalah *cutoff min\_lldelta* yaitu *cutoff* yang menjaga agar *delta rata - rata log likelihood* terus berada diatas batas minimal yang telah ditentukan. Apabila *delta rata - rata log likelihood* mengalami perubahan dan berada dibawah batas minimal, maka proses iterasi akan berhenti, sehingga menghasilkan *feature set* dengan bobot masing - masing *feature*. Implementasi pencarian pembobot menggunakan *improved iterative scaling* dapat dilihat pada Gambar 5.5

```

75 ▼  def training_weight_iis(self, paragraph):
76      train = []
77 ▼      for index, data in enumerate(paragraph):
78          sentence = sent_tokenize(data)
79          # PEMECAHAN PARAGRAF KEDALAM KALIMAT
80          for index, data in enumerate(sentence):
81              sent_lower = data.lower()
82              tokenize = word_tokenize(sent_lower)
83              div_sentence = []
84              for data in tokenize:
85                  if "/" not in data:
86                      # ubah menjadi kata dasar
87                      sent_stem = self.stemmer.stem(data)
88                      data = sent_stem
89                  elif "/con" in data:
90                      # ubah menjadi kata dasar kemudian dicocokan kedalam gazeter kondisi
91                      sent_stem = self.stemmer.stem(self.w.search(data).group(1))
92                      data = sent_stem + "/CON"
93                  elif "/" in data:
94                      word = self.w.search(data).group(1)
95                      label = self.lbl.search(data).group(1)
96                      data = word + "/" + label.upper()
97                      div_sentence.append(data)
98              train.append(" ".join(div_sentence))
99
100             #MELAKUKAN TRAINING DENGAN SENTENCE YANG SUDAH DIUBAH KEDALAM KATA DASAR
101             me_classifier = MaxentClassifier.train(self.binary_feature.train, "train_iis", algorithm='iis', trace=100, max_iter=2000, min_lldelta=1)
102             # me_classifier = MaxentClassifier.train(self.binary_feature.train, "train_iis", algorithm='iis', trace=100, max_iter=2000, min_ll=0.1)
103             return me_classifier

```

Gambar 5.5 Implementasi Pencarian Bobot Optimal

Selain itu dari gambar 5.5 dapat dijelaskan maksimal iterasi dibatasi sampai 2000 iterasi, apabila iterasi melebihi 2000 maka proses iterasi akan dihentikan. Selain itu menggunakan *min\_lldelta* = 1, sehingga apabila *delta rata - rata log likelihood* berada dibawah 1 maka iterasi akan berhenti.

## 5.6 Implementasi Deteksi Entitas

Implementasi deteksi entitas menggunakan dua model yang berbeda, model pertama deteksi entitas menggunakan *maximum entropy*, model kedua deteksi entitas menggunakan kombinasi antara *maximum entropy* dengan *string matching*. Metode *string matching* diterapkan untuk mengatasi keterbatasan *maximum*

*entropy* pada penanganan WSD (*Word Sense Disambiguation*) dan juga deteksi pada susunan kata secara *frase*, salah satunya pada saat deteksi entitas lokasi. Sebagai contoh apabila ada kata nama lokasi yang terdiri dari dua kata atau lebih, kata tersebut apabila benar merupakan satu kesatuan nama sebuah lokasi maka kata tersebut digabung menjadi satu kesatuan secara *frase*. Untuk lebih jelasnya dapat dilihat pada Gambar 5.6.

```

70 # -----
71 # NER
72 #
73 for date_day in range(1,32):
74     day = ""
75     day_str = str(date_day)
76     if len(day_str) == 1:
77         day = "0"+day_str
78     else:
79         day = day_str
80
81 cursor_get_data_preprocessor = dbmodel.get_data_preprocessor(month_data_preprocessor,day)
82 for document in cursor_get_data_preprocessor:
83     data = document["data"]
84
85     for sentences in data :
86         sentence = sentences["text_tweet"]
87         if sentence :
88             print sentences["id"]
89             ner = classify.training_ner(sentence.encode("utf8"), classifier)
90
91             if combination_match_string == True:
92                 sentence_ne = ner["text_tweet"]
93                 entity_position = ner["entity_position"]
94                 if "LOC" in ner["entity"]:
95                     # ====== clearance not location ======
96                     entity_location = ner["entity"]["LOC"]
97
98                     temp_location = dbmodel.is_candidate_loc(db_location, location_collection, entity_position, entity_location)
99                     loc_clear = dbmodel.is_real_loc(db_location, location_collection, temp_location)
100                    sentence_ne = dbmodel.ner_replace_loc(ner["text_tweet"], loc_clear)
101                    # ====== end clearance not location ======
102
103                     if "CON" in ner["entity"] :
104                         if "sakit" in ner["entity"]["CON"]:
105                             sentence_ne = dbmodel.validation_CON(sentence_ne, "sakit")
106
107                     sentence_ne = ner["text_tweet"]
108
109                     ner["id"] = sentences["id"]
110                     ner["url"] = sentences["url"]
111                     ner["username"] = sentences["username"]
112                     ner["text_tweet"] = sentence_ne
113                     ner["time"] = sentences["time"]
114                     # if ner and ("/NUM" in sentence_ne and "/LOC" in sentence_ne):
115                     # apabila array ner tidak kosong dan berisikan entitas NUM dan LOC maka jalankan statement berikut
116                     cursor_insert_data = dbmodel.insert_ner_to_db(month_data_ner, day, ner)
117                     print cursor_insert_data

```

Gambar 5.6 Implementasi Deteksi Entitas

Dari Gambar 5.6 dapat dilihat proses deteksi entitas dijalankan berdasarkan periode bulanan, ditunjukan pada baris 73, baris tersebut menjelaskan iterasi pengambilan tanggal pada bulan yang sudah ditentukan dimulai dari iterasi tanggal 1 sampai batas iterasi maksimal 32, apabila tanggal hanya sampai 28 ataupun 31 proses iterasi akan otomatis berhenti.

Sedangkan proses kombinasi *maximum entropy* dan *string matching* dapat dilihat pada baris 91 sampai 104, baris tersebut menjelaskan apabila statement *combination\_match\_string* = *True* maka proses kombinasi akan dilakukan, data NER yang dihasilkan *maximum entropy* pada baris 89 akan dilakukan *filter* kembali

berdasarkan data entitas lokasi dan data entitas kondisi untuk mencapai hasil data yang lebih akurat. Data hasil akhir deteksi entitas tersebut kemudian disimpan kedalam database MongoDB, ditunjukkan pada baris 115.

## 5.7 Implementasi Pemetaan Lokasi Kejadian dan Penalaran Tingkat Resiko

Pemetaan persebaran lokasi kejadian didasarkan pada koordinat lintang dan bujur yang diambil dari *geonames.org*, data koordinat tersebut semula menggunakan format DMS kemudian akan dilakukan *konversi* kedalam format *decimal*. Setelah didapatkan koordinat lintang dan bujur, kemudian data persebaran akan dipetakan dan dilakukan pengelompokan lokasi sesuai tingkat resiko berdasarkan *knowledge base* yang telah dibahas pada Bab 4.5, lebih jelasnya dapat dilihat pada Gambar 5.7.

```

91     for data in datas:
92         location = data["_id"]
93         label_graph.append(location)
94         data_graph.append(data["NUM"])
95         coordinate = list(dbmodel.get_lat_long_location("bigcities", "cities", location))
96         data_locations = {}
97         for loc in coordinate:
98             temp_long = loc["longitude"].encode('ascii', 'ignore').replace("'", "").replace("o", " ").split(" ")
99             s = decimal.Decimal(int(temp_long[3])/60)
100            m = decimal.Decimal((int(temp_long[2])+s)/60)
101            d = decimal.Decimal(int(temp_long[1])+m)
102            if temp_long[0] == 'S' or temp_long[0] == 'W':
103                longitude = 0 - d
104            else:
105                longitude = d
106
107            temp_lat = loc["latitude"].encode('ascii', 'ignore').replace("'", "").replace("o", " ").split(" ")
108            s = decimal.Decimal(int(temp_lat[3])/60)
109            m = decimal.Decimal((int(temp_lat[2])+s)/60)
110            d = decimal.Decimal(int(temp_lat[1])+m)
111            if temp_lat[0] == 'S' or temp_lat[0] == 'W':
112                latitude = 0 - d
113            else:
114                latitude = d
115
116            if data["NUM"] > 55 :
117                btn_style = "btn-danger"
118                data_locations["icon"] = "http://maps.google.com/mapfiles/ms/icons/red-dot.png"
119            elif data["NUM"] > 19 :
120                btn_style = "btn-warning"
121                data_locations["icon"] = "http://maps.google.com/mapfiles/ms/icons/yellow-dot.png"
122            else:
123                btn_style = "btn-success"
124                data_locations["icon"] = "http://maps.google.com/mapfiles/ms/icons/green-dot.png"
125
126            data_locations["lat"] = float(latitude)
127            data_locations["lng"] = float(longitude)

```

Gambar 5.7 Implementasi Pemetaan Lokasi dan Penalaran Tingkat Resiko

Pada Gambar 5.7 dapat dilihat jika data memiliki entitas lokasi maka akan dilakukan proses pengambilan koordinat bujur dan lintang, seperti yang ditunjukkan pada baris 95. Data koordinat yang telah diperoleh kemudian dilakukan konversi kedalam format decimal, seperti yang ditunjukkan pada baris 98 - 114. Selain itu data *tweet* juga akan dilakukan kategorisasi berdasarkan tingkat resiko sesuai *knowledge base*, ditunjukkan pada baris 116 – 124.

## 5.8 Implementasi Pengujian NER

Pengujian menggunakan nilai *precision*, *recall* dan *f-measure*, nilai tersebut nantinya akan digunakan sebagai tolak ukur evaluasi dari model yang telah diterapkan. Pengujian yang dilakukan menggunakan data yang telah terlabeli manual kemudian dicocokan dengan hasil pelabelan NER yang telah diproses. Proses pengujian tersebut dilakukan berdasarkan periode bulanan, seperti yang dapat ditunjukkan oleh Gambar 5.8 pada baris 27.

```

27     for date_day in range(1,32):
28         day = ""
29         day_str = str(date_day)
30         if len(day_str) == 1:
31             day = "0"+day_str
32         else:
33             day = day_str
34
35         cursor = dbmodel.get_data_all(db_source,day);
36         for doc in cursor :
37             sentences+=1
38             cur = dbmodel.get_data_one_from_id(db_testing, day, doc["id"])
39             for doc2 in cur:
40                 data_label = doc["text_tweet"]
41                 data_test = doc2["text_tweet"]
42                 token_label = word_tokenize(data_label)
43                 token_test = word_tokenize(data_test)
44                 if(len(token_label) == len(token_test)):
45                     for index, label in enumerate(token_label):
46                         arr_lab = token_label[index].split("/")
47                         arr_test = token_test[index].split("/")
48                         if arr_lab[0] == arr_test[0]:
49                             if ("/" in token_label[index]):
50                                 labels_training+=1
51                                 token_label[index] = arr_lab[0].lower() + "/" + arr_lab[1]
52                                 if (token_label[index] == token_test[index]):
53                                     # terpilih benar
54                                     labels_test+=1
55                                     tp+=1
56                                 elif ("/" in token_label[index]) and ("/" not in token_test[index]):
57                                     # tidak terpilih padahal benar
58                                     fn+=1
59                                 elif ("/" in token_test[index]) and (arr_test[1] != arr_lab[1]):
60                                     # terpilih tetapi tidak benar
61                                     labels_test+=1
62                                     fp+=1
63
64                     else:
65                         # tidak terpilih dan tidak benar (TIDAK DIPAKAI)
66                         tn+=1
67                 else:
68                     if (( "/" in token_test[index]) and ("/" not in token_label[index])):
69                         # terpilih tetapi tidak benar
70                         labels_test+=1
71                         fp+=1
72
73             p = Decimal(tp/Decimal((tp+fp)))
74             r = Decimal(tp/Decimal((tp+fn)))
75             fm = 2*((p*r)/(r+p))
76
77             result = {}
78             result["SENTENCES"] = sentences
79             result["LABEL_TRAINING"] = labels_training
80             result["LABEL_TEST"] = labels_test
81             result["TP"] = tp
82             result["FP"] = fp
83             result["FN"] = fn
84             result["PRECISION"] = p.normalize()
85             result["RECALL"] = r.normalize()
86             result["F-MEASURE"] = fm.normalize()
87
88         return result

```

Gambar 5.8 Implementasi Pengujian.

## BAB VI

### HASIL DAN PEMBAHASAN

Pada bagian ini dijelaskan mengenai hasil dan pengujian model *monitoring* yang telah dibuat, hasil dan pembahasan tersebut dibagi menjadi beberapa bagian yang berbeda. Bagian pertama mengenai pengujian NER (deteksi entitas), bagian kedua NER dikombinasikan dengan *string matching*, dilanjutkan pada bagian ketiga pembahasan mengenai hasil model *monitoring* dan keempat pengujian data hasil model *monitoring* dengan data persebaran penyakit demam berdarah yang diambil dari dinas kesehatan setempat.

#### 6.1 Pengujian NER tanpa *String Matching*

Pengujian deteksi entitas pada kali ini menggunakan data pada bulan april 2015, dengan jumlah data uji sebanyak 3198 data *tweet*, didalamnya berisikan 859 kata terlabeli manual. Proses pengujian dibagi menjadi 2 bagian, bagian pertama pengujian MaxEnt pada persamaan yang diaplikasikan oleh (Hui, 2009) dan bagian ketiga adalah pengujian menggunakan MaxEnt menggunakan *library nltk*, hasil dari pengujian tersebut dapat dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian *Maximum Entropy*

min_All	MaxEnt (Hui, 2009)			MaxEnt (NLTK)		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
0.02	0.21	0.98	0.35	0.21	0.98	0.35
0.05	0.21	0.98	0.35	0.22	0.98	0.36
0.1	0.21	0.98	0.35	0.22	0.98	0.36
1	0.21	0.98	0.35	0.21	0.98	0.35

Pada kedua pengujian tersebut hasil deteksi entitas menggunakan MaxEnt menunjukan nilai yang kurang baik meskipun dilakukan pengujian menggunakan *cutoff min\_All* yang bervariasi, nilai *precision* paling tinggi mencapai 0.22, nilai *recall* paling tinggi mencapai 0.98, sehingga nilai *F-Measure* paling tinggi

mencapai 0.36. Nilai tersebut dihasilkan pada pengujian MaxEnt (NLTK) dengan *cutoff*  $\text{min\_All} = 0.05$  dan  $\text{min\_All} = 0.1$

Nilai dihasilkan dari *precision* sangat rendah dikarenakan dari banyaknya entitas yang terdeteksi kemudian dicocokan dengan entitas yang diujikan, hanya sedikit entitas yang dapat terbukti kebenarannya, sebagian besar mengalami kesalahan pendekripsi. Berbeda dengan *recall* yang mencapai nilai tinggi, dikarenakan dari banyaknya entitas yang diujikan hampir keseluruhan entitas yang diujikan tersebut dapat terdeteksi dengan benar oleh deteksi entitas yang telah dibangun. Sedangkan usaha temu kembali antara *precision* dan *recall* menghasilkan nilai yang kurang bagus, untuk itu perlu dilakukannya evaluasi terhadap model deteksi entitas telah dibangun, yaitu dibahas pada bagian 6.2

## 6.2 Pengujian NER dengan *String Matching*

Setelah dilakukan pengujian pada bagian 6.1 maka dilakukan beberapa analisa pengujian, hasil pengujian tersebut menunjukkan nilai yang kurang bagus, hal tersebut disebabkan oleh beberapa kendala dalam deteksi entitas yang berhasil diketahui yaitu :

1. Terkendala pada deteksi entitas kata yang bermakna ganda WSD (*Word Sense Disambiguation*), kata bermakna ganda tersebut salah satunya dialami pada deteksi entitas lokasi.

Contoh : genangan air/LOC, kata air dideteksi sebagai entitas lokasi karena ada nama lokasi yang mengandung kata air yaitu air sialang hilir.

2. Terkendala dalam melakukan pendekripsi kata yang mempunyai susunan secara *frase* (satu kesatuan kata).

Contoh : “Bandar Lampung” (entitas lokasi).

Dari kendala yang ditimbulkan tersebut, maka perlu dilakukan evaluasi guna meningkatkan nilai *precision* dan *recall*, sehingga dapat meningkatkan nilai *F-Measure*. Evaluasi yang dilakukan adalah menerapkan kombinasi string matching terhadap hasil deteksi entitas *maximum entropy*. *String matching* tersebut diterapkan pada proses filterisasi entitas lokasi dan juga kondisi, apabila entitas

lokasi atau kondisi tersebut bukan merupakan entitas lokasi ataupun kondisi yang valid maka label yang ada pada kata tersebut akan dihilangkan, sehingga kata tersebut menjadi kata yang tidak memiliki label. Sedangkan hasil dari pengujian kombinasi antara *maximum entropy* dengan *string matching* dapat ditunjukkan pada Tabel 6.2.

Tabel 6.2 Pengujian *Maximum Entropy* dan *String Matching*

MaxEnt (Hui, 2009) + String Matching				MaxEnt (NLTK) + String Matching		
min_All	Precision	Recall	F-Measure	Precision	Recall	F-Measure
0.02	0.91	0.96	0.93	0.92	0.96	0.94
0.05	0.91	0.96	0.93	0.85	0.97	0.90
0.1	0.92	0.95	0.93	0.85	0.97	0.90
1	0.92	0.95	0.93	0.95	0.94	0.94

Dapat dilihat pada Tabel 6.2, pengujian dari kombinasi *maximum entropy* dan *string matching* menunjukkan nilai *precision* dan *recall* yang meningkat. Sehingga *F-Measure* paling tinggi dapat mencapai nilai 0.943953 pada *cutoff min\_All* = 0.02 dan *min\_All* = 1. Dari hasil pengujian tersebut model deteksi entitas menggunakan *maximum entropy* dan *string matching* dapat digunakan sebagai deteksi entitas pada proses model *monitoring* yang akan diterapkan.

### 6.3 Hasil Model *Monitoring*

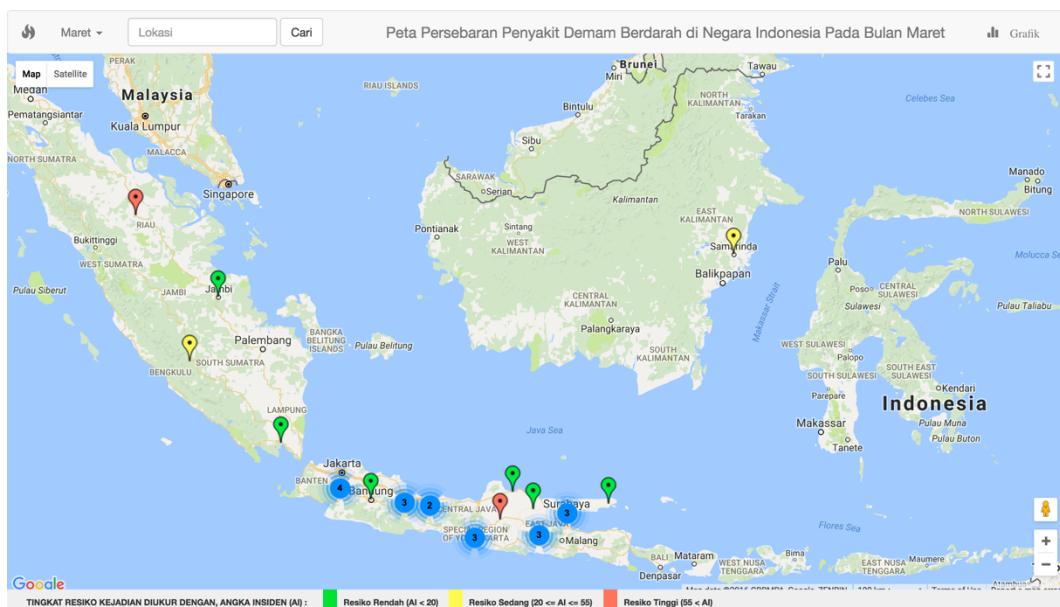
Hasil dari deteksi entitas pada bulan maret 2015 dan april 2015 digunakan sebagai data acuan untuk model *monitoring*, Sebelum data tersebut diproses maka harus diseleksi terlebih dahulu menggunakan *forward chaining*, dengan memastikan bahwa yang akan diproses hanyalah data yang mempunyai entitas lokasi (LOC) dan entitas angka kejadian (NUM). Hasil model *monitoring* keseluruhan dapat dilihat pada Tabel 6.3

Tabel 6.3 Data Hasil *Monitoring* Bulan Maret 2015 dan April 2015

No.	Lokasi	Angka Kejadian		
		Maret 2015	April 2015	Peningkatan Maret 2015 - April 2015
1	bandar lampung	7	-	-
2	bandung	1	-	-
3	banjar	1	-	-
4	banjarmasin	494	-	-
5	bantul	332	-	-
6	baru	3	-	-
7	baubau	13	-	-
8	bekasi	25	4	-
9	blitar	28	-	-
10	bogor	242	-	-
11	bojonegoro	2	4	2
12	brebes	7	-	-
13	cilacap	394	-	-
14	cilegon	3	-	-
15	cirebon	4	21	17
16	denpasar	209	-	-
17	garut	7	-	-
18	jakarta barat	111	-	-
19	jakarta selatan	343	-	-
20	jakarta timur	228	-	-
21	jambi	3	-	-
22	kediri	383	-	-
23	kendal	6	-	-
24	klaten	172	20	-

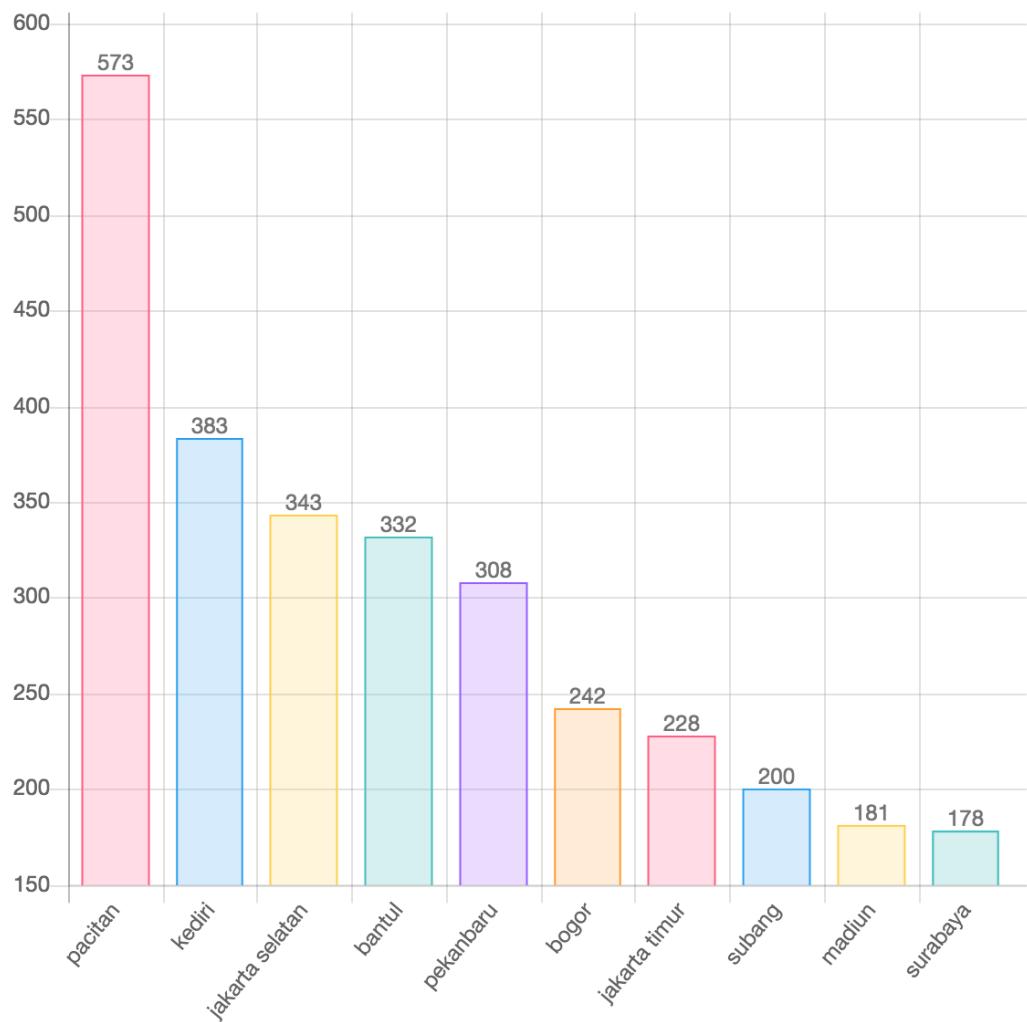
25	kuningan	9	-	-
26	lebak	82	95	13
27	lubuklinggau	24	-	-
28	madiun	181	-	-
29	malang	5	15	10
30	pacitan	573	-	-
31	pangandaran	2	-	-
32	pasaman	6	-	-
33	pasuruan	2	-	-
34	pekanbaru	308	355	47
35	pelalawan	16	-	-
36	probolinggo	3	-	-
37	purbalingga	1	-	-
38	purwakarta	2	-	-
39	rembang	10	-	-
40	samarinda	22	-	-
41	sampang	2	-	-
42	sarolangun	50	-	-
43	sleman	161	-	-
44	sragen	140	-	-
45	subang	200	-	-
46	sukabumi	4	301	297
47	sumedang	4	-	-
48	sumenep	8	-	-
49	surabaya	178	318	140
50	tanggamus	37	-	-
51	wonogiri	4	-	-

Tabel 6.3 merupakan data hasil *monitoring* yang dikelompokan kedalam daerah lokasi tingkat 2 yaitu area kabupaten, dengan cara mendeteksi angka kejadian diambil nilai kejadian paling tinggi disetiap lokasi terjadinya demam berdarah selama satu bulan. Dari data yang telah dihasilkan selama dua bulan tersebut, dapat diketahui beberapa daerah mengalami peningkatan angka kejadian demam berdarah. Sedangkan sebagian besar daerah pada bulan april 2015 tidak memiliki data kejadian demam berdarah, hal tersebut bisa dikarenakan lokasi terkait memang tidak mengalami peningkatan jumlah angka kejadian demam berdarah atau data pesan *tweet* yang membahas lokasi tersebut memang tidak tersedia. Selain itu, hasil dari *forward chaining* yang telah dibuat dapat memetakan hasil data *monitoring* kedalam peta google menurut tingkat resiko kejadian demam berdarah disetiap daerah, tingkat resiko rendah di representasikan menggunakan *point* kuning, resiko sedang direpresentasikan kedalam *point* warna hijau, sedangkan representasi tingkat resiko tinggi kedalam *point* warna merah. Pengelompokan data sesuai tingkat resiko kejadian pada peta persebaran bertujuan untuk mengetahui kondisi persebaran penyakit demam berdarah dan juga melakukan analisis terhadap arah persebaran penyakit demam berdarah pada periode waktu satu bulan, untuk lebih jelasnya dapat dilihat pada Gambar 6.1.



Gambar 6.1 Peta Persebaran Penyakit Demam Berdarah pada Bulan Maret

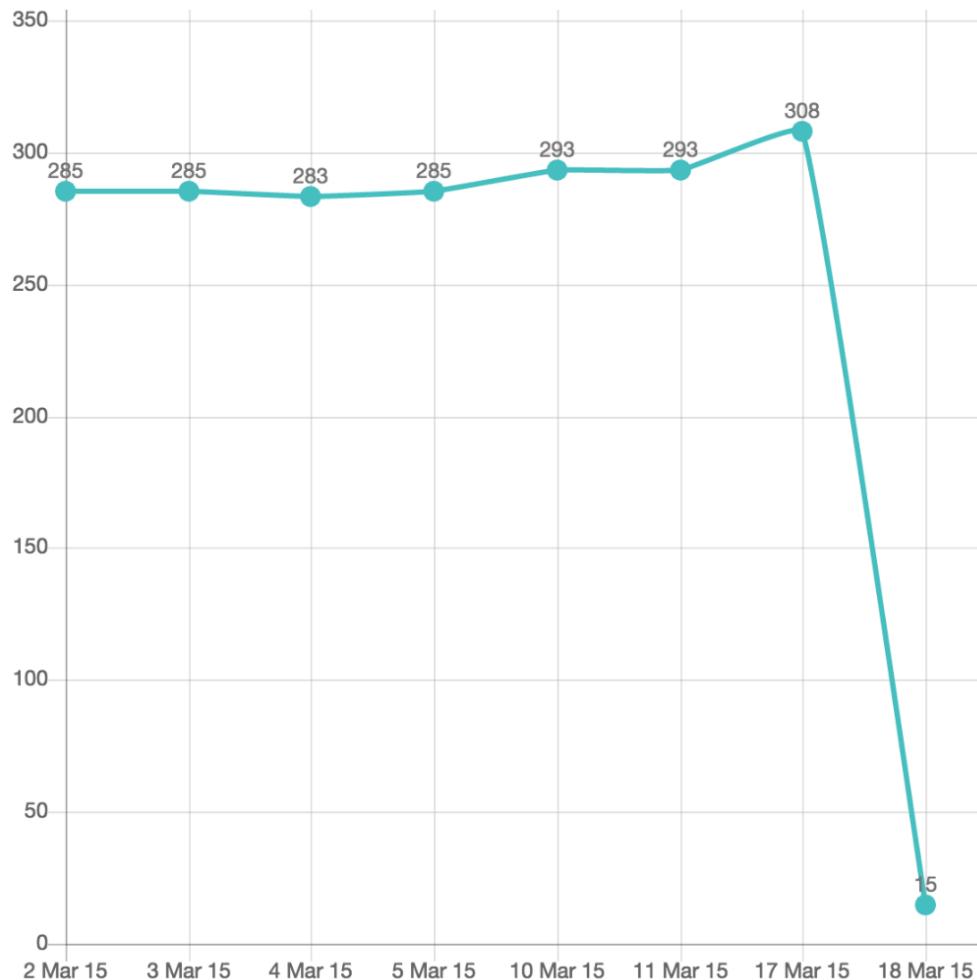
Selain representasi data kedalam peta persebaran penyakit demam berdarah, model *monitoring* juga merepresentasikan data 10 lokasi tertinggi angka kejadian demam berdarah disetiap bulan kedalam grafik batang, dengan contoh representasi data 10 lokasi kejadian bulan maret dapat dilihat pada Gambar 6.2.



Gambar 6.2 10 Lokasi Tertinggi Angka Kejadian Bulan Maret 2015

Dari Gambar 6.2 dapat diketahui 10 lokasi yang memiliki tingkat resiko angka kejadian paling tinggi pada bulan maret, data grafik batang tersebut dapat digunakan oleh pengguna model *monitoring* sebagai data awal dilakukan analisis pembuktian kejadian demam berdarah dilokasi terkait, yang nantinya bertujuan untuk melakukan langkah kedepan terhadap lokasi terjangkit. Selain itu *history* kejadian demam berdarah di setiap lokasi kejadian dapat dilihat kedalam grafik

garis *history* kejadian, merujuk pada lokasi pekanbaru yang masuk kedalam 10 besar lokasi dengan tingkat kejadian paling tinggi, maka dapat dilihat *history* kejadian pada lokasi Pekanbaru tersebut pada Gambar 6.3



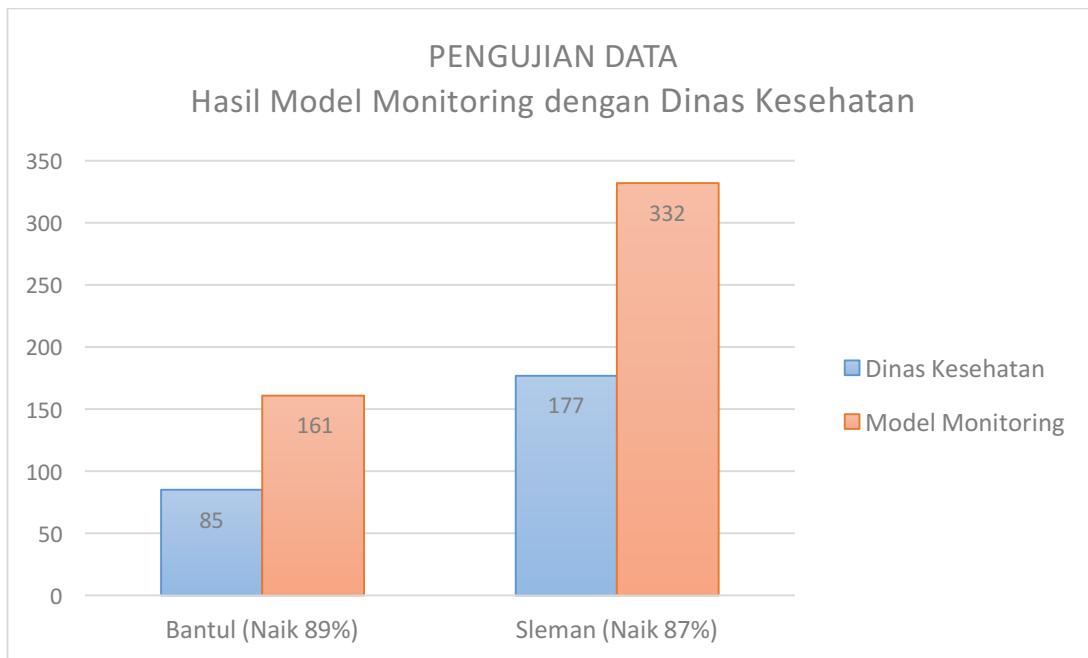
Gambar 6.3 *History* Angka Kejadian Lokasi Pekanbaru Bulan Maret 2015

Gambar 6.3 menunjukkan *history* angka kejadian lokasi Pekanbaru yang dihasilkan dari model *monitoring*, dari grafik yang ditunjukan dapat diketahui kondisi kejadian demam berdarah pada lokasi tersebut berada pada tingkat resiko tinggi dimulai dari tanggal 2 maret 2015 hingga 5 maret 2015, kemudian mengalami peningkatan sampai pada tanggal 17 maret 2015, sedangkan tanggal 18 maret 2015 berada pada angka 15, hal tersebut bisa dimaksud mengalami peningkatan 15 angka kejadian dari tanggal sebelumnya atau data 15 tersebut

termasuk dalam hitungan angka kejadian 308, untuk itu perlu dilihat lagi sumber pesan *tweet* yang diambil dan perlu dilakukan analisis ulang terhadap data kejadian sebenarnya terhadap dinas kesehatan setempat.

#### 6.4 Pengujian Hasil Model *Monitoring*

Dikarenakan data kejadian sebenarnya pada bulan maret 2015 hanya tersedia dari Dinas Kesehatan Kabupaten Sleman dan Dinas Kesehatan Kabupaten Bantul, maka pengujian data hasil model *monitoring* hanya berfokus pada dua kabupaten tersebut. Hasil pengujian dapat dilihat pada Gambar 6.4



Gambar 6.4 Pengujian Data Hasil Model *monitoring* dengan Dinas Kesehatan

Dari Gambar 6.4 dapat diketahui data hasil model *monitoring* lokasi Sleman dan Bantul berada diatas angka kejadian sebenarnya (data dari dinas terkait). Dilihat dari persentase perbandingan data pada kedua pengujian tersebut, model *monitoring* menunjukkan persentasi data kenaikan pada lokasi Bantul sebesar 89% dan lokasi Sleman sebesar 87% dibanding data kejadian sebenarnya, sehingga dapat diartikan hasil pencapaian model *monitoring* tersebut hampir dua kali lipat dari data kejadian sebenarnya.

Setelah dilakukan analisis terhadap sumber data pesan *tweet* pada kedua pengujian tersebut. Lokasi kejadian Sleman pada bulan maret 2015 menunjukan 161 angka insiden yang dihasilkan dari model *monitoring*, sedangkan data dari dinas kesehatan hanya 84 orang penderita 1 meninggal dunia, setelah dilihat pada sumber data *tweet* secara langsung ternyata data tersebut merujuk angka insiden selama 2 bulan. “tribun jogja 161/NUM kasus dbd terjadi di sleman/LOC dalam 2 bulan”.

Sedangkan hasil pengujian data lokasi Bantul pada bulan maret 2015, angka insiden yang dihasilkan mencapai 332, sementara itu data dari dinas kesehatan kabupaten bantul hanya mencapai 177 penderita dan 0 kejadian meninggal dunia. Setelah dilihat dari sumber data *tweet* secara langsung ternyata merujuk angka insiden dari januari hingga maret, “bernas jogja kab/ORG bantul/LOC 2015 332/NUM warga terkena dbd”.

## BAB VII

### KESIMPULAN DAN SARAN

#### 7.1 Kesimpulan

Setelah dilakukan implementasi dan pembahasan, maka disimpulkan hasil yang dapat diambil dari penelitian ini adalah:

1. Sesuai hasil pengujian NER yang telah dilakukan, nilai *min\_all* (*min delta log likelihood*) tidak memiliki pengaruh pada tingkat keberhasilan NER dalam melakukan deteksi entitas. Variasi *min\_all* 0.02, 0.05, 0.1 dan 1, menghasilkan nilai *precision* dan *recall* yang tidak begitu jauh, sehingga peningkatan nilai *F-Measure* menjadi kurang stabil terutama pada penggunaan maximum entropy nltk dengan dan tanpa kombinasi *string matching*.
2. Pengujian NER membuktikan bahwa kombinasi *maximum entropy* (nltk) dan *string matching* dapat menghasilkan deteksi entitas yang lebih baik. Hal ini, dilihat dari peningkatkan nilai *precision* dan *recall* pada setiap variasi pengujian NER *maximum entropy* tanpa kombinasi dibanding dengan menggunakan kombinasi *string matching*, sehingga memicu peningkatan nilai *F-Measure* yang semula memiliki nilai rata – rata dibawah 0.50 (*maximum entropy* tanpa kombinasi) menjadi nilai rata – rata diatas 0.90 (setelah dilakukan kombinasi).
3. Persentase pengujian data hasil model *monitoring* lokasi Sleman menunjukan kenaikan 89% dan lokasi bantul 87% dari angka kejadian sebenarnya. Persentase tersebut menunjukan data hasil *monitoring* hampir mencapai dua kali lipat melebihi data sebenarnya, oleh sebab itu data hasil model *monitoring* tidak dapat secara langsung dijadikan sebagai penentu kejadian persebaran penyakit demam berdarah yang sebenarnya. Perlunya memperhatikan struktur dan maksud kalimat *tweet* yang menghasilkan data kejadian tersebut.

## 7.2 Saran

Penelitian ini masih jauh dari sempurna dan masih banyak hal yang perlu dikembangkan. Beberapa saran untuk pengembangan penelitian selanjutnya yaitu:

1. Pembenahan atau penambahan *feature* yang mendukung untuk diterapkannya NER *maximum entropy* pada kata berbahasa Indonesia.
2. Pengembangan model *monitoring* menuju detail jumlah angka kejadian meninggal dan menderita secara lebih spesifik.
3. Dikembangkan kedalam *forecasting* kejadian demam berdarah di negara Indonesia.
4. Dikembangkan kedalam analisis sentimen kejadian demam berdarah di negara Indonesia.

## DAFTAR PUSTAKA

- Abhimanyu, C., Abhinav, P., dan Chandresh, S., 2013, Natural Language Processing, *International Journal of Technology Enhancements and Emerging Engineering Research*, vol 1, issue 4 131issn 2347-4289, India.
- Achrekar, H., Gandhe, A., Lazarus, R., Yu, S.H., dan Liu, B., 2011, Predicting Flu Trends using Twitter Data, *The First International Workshop on Cyber-Physical Networking Systems, IEEE 978-1-4577-0248-8/11*, University of Massachusetts Lowell
- Berger, A.L., Stephen, A., dan Vincent, J. Della Petra., 1996, A Maximum Entropy Approach to Natural Language Processing, *IBM T.J Watson Research Center*, Yorktown Heights, NY 10598.
- CAI,X., dan FAN, X., 2009, A Maximum Entropy Method to Recognize Disease Named Phrase, *International Conference on Complex Medical Engineering (ICCME)*, 1-4.
- Curran, J.R., and Clark, S., 2003, Investigating GIS and Smoothing for Maximum Entropy Taggers, *In Proceedings of the 11th Meeting of the European Chapter of the ACL*, Budapest, Hungary.
- Dermawan, R., 2016, Klasifikasi Tweet dan Pengenalan Entitas Bernama Pada Tweet Bencana Dengan Support Vector Machine, Skripsi, Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada Yogyakarta.
- Escolano, F., Suano, P., dan Bonev, B., 2009, *Information Theory in Computer Vision and Pattern Recognition*, Springer, London-New York.
- Even, Y., dan Zohar, 2002, Introduction to Text Mining, Automated Learning Group National Center For Supercomputing Applications, University of Illionis.
- Fienerer, I., Hornik, K., dan Meyer, D., 2008, Text Mining Infrastrucure in R, *Jurnal of Statistical Software*, 25(5), 1-54.
- Gungor, T., Indurkhy N., dan Damerau F.J, 2010, Part-sudo date --set "10 Apr 2016 17:38:00"of-Speech Tagging, *Handbook of Natural Language Processing*, edisi 2, CRC Press, Florida.sudo date --set "10 Apr 2016 17:38:00"
- Apasia, H.N., 2015, Jumlah Pengguna Twitter di Indonesia Akhirnya Terungkap, <http://www.cnnindonesia.com/teknologi/20150326141025-185-42076/jumlah-pengguna-twitter-di-indonesia-akhirnya-terungkap/>, diakses : 18-03-2016 10:51 WIB.

- Hui, N., Hua, Y., Ya-zhou, T., Hao, W., 2009, A Method Of Chinese Named Entity Recognition Based on Maximum Entropy Model, *International Conference on Mechatronics and Automation (ICMA)*, 2472 - 2477.
- Jiang, W., Guan, Y., Wang, X.L., 2006, Improving Feature Extraction in Named Entity Recognition Based on Maximum Entropy Model, *International Conference on Machine Learning and Cybernetics*, 2630 – 2635.
- Ji, X., Chun, S.A., Geller, J., 2013, Monitoring Public Health Concerns Using Twitter Sentiment Classifications, *IEEE International Conference on Healthcare Informatics (ICHI)*, 335 – 344.
- Jurafsky, D., dan Martin, J.H., 2008, Speech and Language Processing: An Introduction to Natural Language Processing, *Computational Linguistics, and Speech Recognition*, edisi 2, Prentice Hall, New Jersey.
- Kashyap, R., dan Nahapetian, A., 2014, Tweet Analysis for User Health Monitoring, *ICST 978-1-63190-014-3*, CA, USA
- Klein, D., and Manning, C., 2003, *Maxent Models, Conditional Estimation and Optimization, HLT-NAACL2003 and ACL2003 Tutorial*, hal 14-16. Standford University
- Kumar, S., Morstatter, F., dan Liu, H., 2014, Introduction, in: Twitter Data Analytics, *SpringerBriefs in Computer Science*, hal. 1-3. New York.
- Manning, C., Shipra, D., Jenny, F., Malvina, Nissim., dan Beatrice, A., 2004, Exploring the Boundaries: Gene and Protein Identification in Biomedical Text. *Proceedings of the BioCreative Workshop*, Granada.
- MacEachren, A.,M., Jaiswal, A., Robinson, A.C., Pezanowski, S., Savelyev, A., Mitra, P., Zhang, X., dan Blanford, J., 2011, SensePlace2: GeoTwitter Analytics Support for Situational Awareness, *IEEE Symposium on Visual Analytics Science and Technology October 23 - 28*, Providence, RI, USA
- Randy, M., 2015, *The World's 21 Most Important Social Media Sites and Apps in 2015*, <http://www.socialmediatoday.com/social-networks/2015-04-13/worlds-21-most-important-social-media-sites-and-apps-2015>, diakses : 17-03-2016 20:53 WIB.
- Miner, G., Delen, D., Elder, J., Fast, A., Hill, T., dan Nisbet, R., 2012, *Practical Text Mining and Statistical Analysis for Non-Structured Text Data Applications*, Elsevier Inc
- Mitchell, R., 2015, *Web Scraping with Python*, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., United State of America

- Navarro, G., Yater, R.B., Sutinen, E dan Tarhio, J., 2001, Indexing Methods for Approximate String Matching, *Buletin IEEE Computer Society Technical Committee on Data Engineering*, Finland.
- Nurwidayantoro, A., 2011, Paralelisasi Maximum Entropy Part Of Speech Tagging Untuk Bahasa Indonesia Dengan Map reduce, *Tesis*, Program Studi S2 Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada Yogyakarta.
- Peraturan Menteri Kesehatan Republik Indonesia Nomor 949/MenKes/SK/VIII/2004, 2004, *Pedoman Penyelenggaraan Sistem Kewaspadaan Dini Kejadian Luar Biasa (KLB)*, Kementerian Kesehatan.
- Pusat Data dan Surveilans Epidemiologi, 2010, *Jendela Epidemiologi*, Kementerian Kesehatan RI, buletin : volume kedua.
- Putranti, N.D., dan Winarko, E., 2014, Analisis Sentimen Twitter untuk Teks Berbahasa Indonesia dengan Maximum Entropy dan Support Vector Machine, *IJCCS-Indonesian Journal of Computing and Cybernetics Systems*, 91-100.
- Ratnaparkhi, A., 1996, A Maximum Entropy Part of-Speech Tagger, *In Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.
- Reddy Elaine, 2015, Using Twitter data to study the world's health, <https://blog.twitter.com/2015/twitter-data-public-health>, diakses : 18-03-2016 13:06 WIB
- Russel, S, J., dan Norvig, P., 2003, Artificial Intelligence A Modern Approach Second Edition, Pearson Education, Inc, Upper Saddle River, New Jersey 07458.
- Tala, F. Z, 2003, A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. M.S. thesis. M.Sc. Thesis. Master of Logic Project. Institute for Logic, Language and Computation. Universiteti van Amsterdam The Netherlands.
- Talvis, K., Chorianopoulos, K., dan Kermanidis, K.L., 2014, Real-time monitoring of flu epidemics through linguistic and statistical analysis of Twitter, *IEEE 978-1-4799-6814-5/14*, Ionian University, Corfu, Greece
- Zhao, H., Zhang, J., Huang, J., 2013, Detecting Flu Transmission by Social Sensor in China, *IEEE International Conference on and IEEE Cyber, Physical and Social Computing Green Computing and Communications (GreenCom), IEEE and Internet of Things (iThings/CPSCom)*, 1242 - 1247.