



**UNIVERSITAS INDONESIA**

**Pengenalan Entitas Kesehatan pada Forum Online  
dengan Menggunakan Recurrent Neural Networks**

**SKRIPSI**

**WAHID NUR ROHMAN**  
**1306381856**

**FAKULTAS ILMU KOMPUTER**  
**PROGRAM STUDI ILMU KOMPUTER**  
**DEPOK**  
**JULI 2016**



**UNIVERSITAS INDONESIA**

**Pengenalan Entitas Kesehatan pada Forum Online  
dengan Menggunakan Recurrent Neural Networks**

**SKRIPSI**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar  
Sarjana Ilmu Komputer**

**WAHID NUR ROHMAN**

**1306381856**

**FAKULTAS ILMU KOMPUTER  
PROGRAM STUDI ILMU KOMPUTER  
DEPOK  
JULI 2016**

## **HALAMAN PERNYATAAN ORISINALITAS**

**Skripsi ini adalah hasil karya saya sendiri,  
dan semua sumber baik yang dikutip maupun dirujuk  
telah saya nyatakan dengan benar.**

**Nama : Wahid Nur Rohman**  
**NPM : 1306381856**  
**Tanda Tangan :**

**Tanggal : 22 Juli 2016**

## **HALAMAN PENGESAHAN**

Skripsi ini diajukan oleh :

Nama : Wahid Nur Rohman

NPM : 1306381856

Program Studi : Ilmu Komputer

Judul Skripsi : Pengenalan Entitas Kesehatan pada Forum Online  
dengan Menggunakan Recurrent Neural Networks

**Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Ilmu Komputer pada Program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Indonesia.**

## **DEWAN PENGUJI**

Pembimbing 1 : Dra. Mirna Adriani, Ph.D. ( )

Pembimbing 2 : Alfian Farizki Wicaksono S.T., M.Sc. ( )

Penguji : Dr. Indra Budi S.Kom., M.Kom ( )

Penguji : Ir. Ito Wasito M.Sc., Ph.D. ( )

Ditetapkan di : Depok

Tanggal : 27 Juni 2016

## KATA PENGANTAR

Segala puji bagi Allah, Tuhan semesta alam, semoga keselamatan dan kesejahteraan tetap terlimpahkan atas junjungan kita Nabi Muhammad SAW, sebaik-baik teladan bagi umat manusia. Segala puji dan syukur kehadiran Allah SWT, Tuhan Yang Maha Esa, yang telah memberikan pertolongan, sehingga penulis dapat menyelesaikan skripsi ini.

Penulisan skripsi ini ditujukan untuk memenuhi salah satu syarat untuk menyelesaikan pendidikan pada Program Sarjana Ilmu Komputer, Universitas Indonesia. Penulis sadar bahwa dalam perjalanan menuntut ilmu di universitas hingga dalam menyelesaikan skripsi ini, penulis tidak sendiri. Penulis ingin berterima kasih kepada pihak-pihak yang selalu peduli, mendampingi, dan mendukung penulis, yaitu:

1. Kedua Orang Tua penulis yang selalu memberikan dukungan dan do'a kepada penulis.
2. Dra. Mirna Adriani, Ph.D. dan Alfian Farizki Wicaksono S.T., M.Sc. selaku dosen pembimbing yang banyak memberikan arahan, masukan, dan bantuan dalam menyelesaikan skripsi ini.
3. Rahmad Mahendra, S.Kom., M.Sc. yang telah memberikan banyak masukan dan saran dalam menyelesaikan skripsi ini.
4. Andreas Febrian yang telah membuat *template* dokumen skripsi ini, sehingga penulis menjadi terbantu dalam menulis skripsi.
5. Erik Dominikus yang telah mempublikasikan dan mempopulerkan *template* dokumen skripsi ini, sehingga penulis menjadi tahu bahwa ada *template* tersebut.
6. Alfian Nur Fauzan yang telah memberikan *template* dokumen skripsi yang telah diperbaiki ini, sehingga saya sangat terbantu dalam melakukan penulisan
7. Putu Wira Astika Dharma, Andi Fajar Nur Ismail dan Ken Nabila Setya, sebagai rekan yang banyak memberi masukan dan berbagi ide dengan penulis.

8. Teman-teman Lab Information Retrieval yang memberi dukungan dan semangat kepada penulis untuk menyelesaikan skripsi ini.
9. Pihak-pihak lain yang tidak dapat penulis sebutkan satu-persatu yang sudah memberikan bantuan dan dukungannya kepada penulis.

Depok, Desember 2017

Wahid Nur Rohman

## HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

**Nama** : Wahid Nur Rohman  
**NPM** : 1306381856  
**Program Studi** : Ilmu Komputer  
**Fakultas** : Ilmu Komputer  
**Jenis Karya** : Skripsi

demikian demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Right)** atas karya ilmiah saya yang berjudul:

Pengenalan Entitas Kesehatan pada Forum Online dengan Menggunakan  
Recurrent Neural Networks

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia-  
/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok  
Pada tanggal : 22 Juli 2016  
Yang menyatakan

(Wahid Nur Rohman)

## ABSTRAK

Nama : Wahid Nur Rohman  
Program Studi : Ilmu Komputer  
Judul : Pengenalan Entitas Kesehatan pada Forum Online dengan Menggunakan Recurrent Neural Networks

Saat ini, seseorang dapat memanfaatkan forum kesehatan *online* untuk mencari tahu perihal penyakit tanpa perlu tatap muka. Melalui forum tersebut, seseorang hanya perlu menuliskan keluhan dan pertanyaan pada formulir yang tersedia. Banyak sekali informasi bermanfaat yang dapat diperoleh dari forum tersebut seperti keluhan, obat atau langkah penyembuhan. Penelitian kali ini mencoba untuk melakukan ekstraksi entitas *disease*, *symptom*, *treatment* dan *drug* secara otomatis. Penulis memandang permasalahan ini sebagai permasalahan *sequence labelling*. Kami mengusulkan penggunaan teknik *Deep Learning* dengan menggunakan *Recurrent Neural Networks* (RNNs), karena RNNs merupakan *state-of-the-art* untuk permasalahan *sequence labelling* seperti permasalahan pada penelitian ini. Penulis mengusulkan fitur kata itu sendiri, *stop word*, POS-Tag, frasa kata (nomina dan verba), kata sebelum dan kata sesudah. Selain itu penulis juga mengusulkan dua arsitektur RNNs, yaitu RNNs 1 layer dan RNNs 2 layer. Hasil eksperimen menunjukkan bahwa model yang diusulkan mampu memberikan hasil yang cukup baik. Berdasarkan eksperimen dengan kombinasi fitur kata itu sendiri, *stop word*, frasa kata (nomina dan verba), 1 kata sebelum dan 1 kata sesudah, arsitektur ..., mampu mencapai rata-rata *f-measure* .... (*disease*, ...). Ada yang perlu diperhatikan bahwa performa model yang kami usulkan lebih baik dari baseline.

Hal Penelitian ini akan mencari fitur-fitur yang kontributif serta arsitektur RNNs yang memberikan akurasi terbaik. Hasil akhir penelitian ini didapatkan bahwa dengan fitur kata itu sendiri, kamus kesehatan, *stop word*, frasa kata, 1 kata sebelum dan 1 kata sesudah akan memberikan hasil yang terbaik, yaitu **Hasil mana yang harus ditampilkan?**

Kata Kunci:

MER, RNNs, *disease*, *symptom*, *treatment*, *drug*



## **ABSTRACT**

Name : Wahid Nur Rohman  
Program : Computer Science  
Title : Medical Entity Recognition on the Online Health Forum using  
Recurrent Neural Networks

Keywords:

Al-Qur'an, evaluation, MFCC, SDCC, SVM, GMM

## DAFTAR ISI

<b>HALAMAN JUDUL</b>	<b>i</b>
<b>LEMBAR PERNYATAAN ORISINALITAS</b>	<b>ii</b>
<b>LEMBAR PENGESAHAN</b>	<b>iii</b>
<b>KATA PENGANTAR</b>	<b>iv</b>
<b>LEMBAR PERSETUJUAN PUBLIKASI ILMIAH</b>	<b>vi</b>
<b>ABSTRAK</b>	<b>vii</b>
<b>Daftar Isi</b>	<b>ix</b>
<b>Daftar Gambar</b>	<b>xii</b>
<b>Daftar Tabel</b>	<b>xiii</b>
<b>Daftar Kode</b>	<b>xiv</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Perumusan Masalah . . . . .	3
1.3 Tujuan dan Manfaat Penelitian . . . . .	3
1.4 Metodologi Penelitian . . . . .	3
1.5 Ruang Lingkup Penelitian . . . . .	5
1.6 Sistematika Penulisan . . . . .	5
<b>2 LANDASAN TEORI</b>	<b>7</b>
2.1 Pengenalan Entitas Kesehatan . . . . .	7
2.2 Deep Learning . . . . .	9
2.3 Recurrent Neural Networks . . . . .	10
2.3.1 Long Short Term Memory . . . . .	12
2.3.2 Penerapan RNNs untuk MER . . . . .	14
2.4 Word Embedding . . . . .	16
<b>3 METODOLOGI</b>	<b>18</b>
3.1 Gambaran Umum Pengembangan Metodologi yang Digunakan untuk . . . . .	18
3.2 Pengumpulan Data . . . . .	19
3.3 Pra-Pemrosesan . . . . .	19
3.3.1 Pembersihan data . . . . .	20

3.3.2	Tokenisasi . . . . .	20
3.3.3	Pemotongan kalimat . . . . .	21
3.4	Pelabelan . . . . .	21
3.5	Pengembangan Model . . . . .	22
3.5.1	Ekstraksi Fitur . . . . .	22
3.5.2	Pengusulan Arsitektur RNNs . . . . .	26
3.6	Eksperimen . . . . .	29
3.6.1	Evaluasi . . . . .	29
<b>4</b>	<b>IMPLEMENTASI</b>	<b>32</b>
4.1	Pengumpulan Data . . . . .	32
4.2	Pra-Pemrosesan . . . . .	32
4.2.1	Pembersihan Data . . . . .	33
4.2.2	Tokenisasi . . . . .	33
4.2.3	Pemotongan Kalimat . . . . .	34
4.3	Pelabelan . . . . .	34
4.4	Pengembangan Model . . . . .	35
4.4.1	Ekstraksi Fitur . . . . .	35
4.4.1.1	Fitur Kata Itu Sendiri . . . . .	35
4.4.1.2	Ekstraksi Fitur Part of Speech Tag . . . . .	36
4.4.1.3	Ekstraksi Fitur Stop Word . . . . .	36
4.4.1.4	Ekstraksi Fitur Kamus Kesehatan . . . . .	37
4.4.1.5	Ekstraksi Frasa Kata Benda . . . . .	37
4.4.1.6	Ekstraksi Frasa Kata Kerja . . . . .	38
4.4.1.7	Ekstraksi Fitur 1 Kata Sebelum . . . . .	38
4.4.1.8	Ekstraksi Fitur 1 Kata Sesudah . . . . .	39
4.4.2	Pengusulan Arsitektur RNNs . . . . .	41
4.4.2.1	LSTM 1 layer . . . . .	41
4.4.2.2	LSTM Layer Bertingkat . . . . .	42
4.5	Eksperimen . . . . .	43
4.6	Evaluasi . . . . .	44
<b>5</b>	<b>EKSPERIMEN</b>	<b>46</b>
5.1	Matriks Evaluasi . . . . .	46
5.2	<i>Baseline</i> Eksperimen . . . . .	46
5.3	Skenario Eksperimen . . . . .	47
5.4	Hasil Eksperimen dan Analisis . . . . .	48
5.4.1	Hasil Ekperimen Pengujian Fitur Beserta Analisis . . . . .	48
5.4.1.1	Sub-Eksperimen Menguji Fitur Kata itu Sendiri . . . . .	48
5.4.1.2	Sub-Eksperimen Menguji Fitur Terbaik Sebelum Ditambahkan Fitur Kamus Kesehatan ( <i>Disease, Symptom, Treatment</i> dan <i>Drug</i> ) . . . . .	50
5.4.1.3	Sub-Eksperimen Menguji Fitur Terbaik Sebelum Ditambahkan Fitur Stopword . . . . .	51
5.4.1.4	Sub-Eksperimen Menguji Fitur Terbaik Sebelum Ditambahkan Fitur POS-Tag . . . . .	52

5.4.1.5	Sub-Eksperimen Menguji Fitur Terbaik Sebelum Ditambahkan Fitur Frasa Kata . . . . .	54
5.4.1.6	Sub-Eksperimen Menguji Fitur Terbaik Sebelum Dikurangi Fitur POS-Tag . . . . .	55
5.4.1.7	Sub-Eksperimen Menguji Fitur Terbaik Sebelum Ditambahkan Fitur 1 Kata Sebelum . . . .	56
5.4.1.8	Sub-Eksperimen Menguji Fitur Terbaik Sebelum Ditambahkan Fitur 1 Kata Sesudah . . . .	57
5.4.2	Hasil Ekperimen Pengujian Arsitektur RNNs . . . . .	58
5.4.2.1	Sub-Eksperimen Menguji Arsitektur RNNs 1 Layer	59
5.4.2.2	Sub-Eksperimen Menguji Arsitektur RNNs 2 Layer	60
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>62</b>
6.1	Kesimpulan . . . . .	62
6.2	Saran . . . . .	62
	<b>Daftar Referensi</b>	<b>63</b>

## DAFTAR GAMBAR

2.1	Ilustrasi Sistem MER . . . . .	7
2.2	<i>Recurrent Neural Networks</i> sederhana . . . . .	11
2.3	1 buah <i>timestep</i> dalam RNNs . . . . .	12
2.4	1 buah blok memori dalam LSTM . . . . .	13
2.5	Arsitektur Word2Vec . . . . .	17
3.1	Diagram Gambaran Umum Metodologi yang Dilakukan . . . . .	19
3.2	LSTM 1 layer . . . . .	26
3.3	1 buah blok memori dalam LSTM . . . . .	26
3.4	LSTM 2 layer . . . . .	27
5.1	Histogram Metrik Evaluasi dengan Fitur Kata Itu Sendiri . . . . .	47
5.2	Histogram Metrik Evaluasi dengan Fitur Kata Itu Sendiri . . . . .	49
5.3	Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Kamus Kesehatan . . . . .	50
5.4	Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Stopword . . . . .	52
5.5	Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur POS-Tag . . . . .	53
5.6	Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Frasa Kata . . . . .	54
5.7	Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Dikurangi Fitur POS-Tag . . . . .	55
5.8	Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sebelum . . . . .	57
5.9	Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sesudah . . . . .	58
5.10	Histogram Metrik Evaluasi dengan Arsitektur RNNs 1 Layer . . . . .	59
5.11	Histogram Metrik Evaluasi dengan Arsitektur RNNs 2 Layer . . . . .	60

## DAFTAR TABEL

5.1	Tabel Hasil Eksperimen dari Penelitian Herwando (2016) . . . . .	46
5.2	Tabel Hasil Eksperimen dengan Fitur Kata Itu Sendiri . . . . .	49
5.3	Tabel Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditam- bahkan Fitur Kamus Kesehatan . . . . .	50
5.4	Tabel Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditam- bahkan Fitur Stopword . . . . .	51
5.5	Tabel Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditam- bahkan Fitur POS-Tag . . . . .	53
5.6	Rangkuman Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Frasa Kata . . . . .	54
5.7	Rangkuman Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Dikurangi Fitur POS-Tag . . . . .	55
5.8	Rangkuman Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sebelum . . . . .	56
5.9	Rangkuman Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sesudah . . . . .	58
5.10	Rangkuman Hasil Eksperimen dengan Arsitektur RNNs 1 Layer . .	59
5.11	Rangkuman Hasil Eksperimen dengan Arsitektur RNNs 2 Layer . .	60

## DAFTAR KODE

4.1	<i>Pseudocode</i> untuk melakukan pengumpulan data . . . . .	32
4.2	<i>Pseudocode</i> untuk melakukan pembersihan data . . . . .	33
4.3	<i>Pseudocode</i> untuk melakukan tokenisasi . . . . .	34
4.4	<i>Pseudocode</i> untuk melakukan pemotongan kalimat . . . . .	35
4.5	<i>Pseudocode</i> untuk melakukan ekstraksi fitur kata itu sendiri . . . . .	36
4.6	<i>Pseudocode</i> untuk melakukan ekstraksi fitur kata itu sendiri . . . . .	36
4.7	<i>Pseudocode</i> untuk melakukan ekstraksi fitur <i>stop word</i> . . . . .	37
4.8	<i>Pseudocode</i> untuk melakukan ekstraksi fitur kamus kesehatan . . . . .	38
4.9	<i>Pseudocode</i> untuk melakukan ekstraksi fitur kamus kesehatan . . . . .	39
4.10	<i>Pseudocode</i> untuk melakukan ekstraksi fitur kamus kesehatan . . . . .	40
4.11	<i>Pseudocode</i> untuk melakukan ekstraksi fitur 1 kata sebelum . . . . .	41
4.12	<i>Pseudocode</i> untuk arsitektur RNNs pertama . . . . .	42
4.13	<i>Pseudocode</i> untuk arsitektur RNNs kedua . . . . .	43
4.14	<i>Pseudocode</i> untuk memecah <i>data</i> menjadi 10 bagian . . . . .	44
4.15	<i>Pseudocode</i> untuk melakukan eksperimen . . . . .	44
4.16	<i>Pseudocode</i> untuk melakukan evaluasi . . . . .	45

# **BAB 1**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Saat ini, perkembangan teknologi semakin mempermudah berbagai kegiatan manusia yang dilakukan sehari-hari. Sebagai contoh, ketika seseorang mengalami permasalahan kesehatan dan ingin berkonsultasi kepada para dokter, ia dapat memanfaatkan forum kesehatan *online* yang dapat memungkinkan terjadinya interaksi antara pasien dan dokter tanpa perlu tatap muka. Melalui forum tersebut, seseorang hanya perlu menuliskan keluhan dan pertanyaan pada formulir yang tersedia. Kemudian, dokter yang memiliki akun di forum kesehatan *online* tersebut dapat memberikan jawaban atas pertanyaan orang tersebut.

Banyak sekali informasi bermanfaat yang dapat diperoleh dari forum kesehatan *online*. Informasi tersebut meliputi informasi keluhan yang dialami pasien, obat yang sebaiknya digunakan atau langkah penyembuhan yang dapat dilakukan. Orang lain dapat mencari obat atau langkah penyembuhan dari forum tersebut melalui pertanyaan yang sudah diajukan sebelumnya. Oleh karena itu, akan sangat baik apabila ada sebuah model atau sistem yang mampu mengekstrak secara otomatis informasi-informasi tersebut. Tantangan utama dari pengembangan model ini adalah *post* atau isi dari forum yang tidak terstruktur. Dokumen *post* tidak dibagi menjadi beberapa bagian seperti bagian keluhan, penyakit, obat dll, namun hanya menjadi satu bagian saja. Misalnya ketika seseorang menanyakan tentang keluhannya, orang tersebut hanya diberikan dua buah isian berupa judul dan isi pertanyaan. Jawaban yang diberikan oleh dokter juga sama, hanya menjadi satu bagian saja. Jawaban yang diberikan tidak terstruktur seperti memiliki bagian langkah penyembuhan, nama penyakit dan obat secara terpisah. Hal ini menyebabkan orang sulit melakukan ekstraksi informasi dari dokumen tersebut.

Dari permasalahan tersebut, terdapat sebuah solusi untuk melakukan ekstraksi informasi penyakit dalam suatu dokumen, yaitu dengan menggunakan sistem Pengenalan Entitas Kesehatan (*Medical Entity Recognition*) atau disingkat MER. Sistem MER ini dapat mengenali entitas kesehatan dalam sebuah dokumen. Apabila diberikan sebuah dokumen, sistem ini akan mengembalikan dokumen yang telah mendapatkan label pada masing-masing entitas kesehatan di dalamnya.

Penelitian dalam rangka mengembangkan sistem MER sudah banyak dilakukan



oleh beberapa peneliti. Salah satu penelitian tersebut dilakukan oleh Abacha dan Zweigenbaum (2011) dengan menggunakan dokumen medis rumah sakit berbahasa Inggris. Entitas yang mendapatkan label pada penelitian tersebut adalah *treatment*, *problem* dan *test*. Terdapat tiga pendekatan yang digunakan, yaitu pendekatan *machine learning*, pendekatan *rule based* dan pendekatan *hybrid*. Kesimpulan yang dicapai pada penelitian tersebut adalah pendekatan *hybrid* memberikan hasil terbaik, yaitu dengan *precision* 72.18%, *recall* 83.78% dan *F-Measure* 77.55%.

Pada dokumen berbahasa Indonesia, pengembangan sistem MER masih belum banyak. Ada beberapa penelitian terkait sistem MER, namun hasil yang diberikan belum memuaskan. Salah satu penelitian terkait MER dilakukan oleh Herwando (2016) yang menggunakan dokumen forum kesehatan *online* berbahasa Indonesia dari beberapa situs. Tujuan dari penelitian tersebut adalah untuk mencari kombinasi fitur yang dapat menghasilkan akurasi terbaik. Herwando (2016) menggunakan algoritma *Conditional Random Field* dengan hasil akhir yaitu *precision* 70.97%, *recall* 57.83% dan *f-measure* 63.69%. Fitur-fitur yang membuat model memiliki akurasi terbaik yaitu fitur kata itu sendiri, frasa, kamus: *symptom*, *disease*, *treatment*, *drug*, *window feature (previous word)* dan panjang kata.

Dalam penelitian ini, penulis mengusulkan model lain untuk mengembangkan sistem MER, yaitu dengan menggunakan *Recurrent Neural Network*. Sebelumnya penelitian terkait hal ini pernah dikerjakan oleh Mujiono et al. (2016), dengan jenis entitas yang digunakan adalah entitas *Drug* saja. Namun, dengan menggunakan fitur vektor kata yang menggunakan *word embedding* saja, *f-measure* yang diberikan mencapai 86.45%. Oleh karena itu, penulis mengusulkan model *Recurrent Neural Network* pada penelitian ini.

Penulis berharap bahwa penelitian ini akan memberikan banyak manfaat. Sistem MER yang dihasilkan dapat digunakan untuk membuat aplikasi lain. Misalnya dengan adanya MER pada dokumen bahasa Indonesia, dapat dibuat sistem untuk melakukan *indexing* dokumen forum sehingga pencarian dokumen kesehatan dapat dilakukan dengan lebih efisien. Selain itu, keluaran dari MER juga dapat digunakan untuk mengidentifikasi tren penyakit pada waktu tertentu dari suatu sumber, sehingga pihak terkait mampu melakukan langkah dan kebijakan yang tepat. Penulis berharap bahwa penelitian MER pada dokumen berbahasa Indonesia ini dapat dilanjutkan sehingga dapat menghasilkan model yang lebih baik dan membuat suatu aplikasi yang memanfaatkan keluaran dari penelitian ini. Masih banyak manfaat lain yang didapatkan dengan adanya sistem MER yang memiliki hasil akurat.

## 1.2 Perumusan Masalah

Berdasarkan latar belakang di atas, dalam penelitian ini penulis mengajukan rumusan masalah sebagai berikut:

1. Fitur apa saja yang membuat sistem MER memiliki performa terbaik?
2. Bagaimana pengaruh arsitektur RNNs terhadap performa sistem MER yang dikembangkan?

## 1.3 Tujuan dan Manfaat Penelitian

Penelitian ini bertujuan untuk membangun sistem yang mampu melakukan ekstraksi entitas kesehatan dari forum *online*. Sebenarnya, pada penelitian yang dilakukan oleh Herwando (2016) sudah menghasilkan sebuah sistem yang sama. Namun, fokus penelitian ini yaitu mencoba menggunakan metode yang berbeda. Metode tersebut yaitu dengan menggunakan *Recurrent Neural Network* dengan harapan mampu memberikan hasil yang lebih baik. Penelitian ini juga bertujuan untuk mendapatkan fitur-fitur yang membuat sistem memiliki performa terbaik. Selain itu, penelitian ini juga bertujuan untuk mendapatkan informasi baru terkait pembuatan sistem MER berbahasa Indonesia.

Manfaat dari penelitian ini adalah menghasilkan rancangan sistem dan metode yang dapat digunakan sebagai bahan penelitian lanjutan. Saat ini, sistem dan metode yang dihasilkan hanya mampu mengenali entitas kesehatan saja. Hal ini dapat digunakan untuk membuat sistem informasi tentang suatu jenis penyakit lengkap dengan gejala, obat dan cara penyembuhannya. Selama ini, masyarakat yang menanyakan suatu penyakit melalui forum *online* tidak membaca terlebih dahulu riwayat pertanyaan yang telah ditanyakan oleh orang lain. Oleh karena itu, diharapkan dengan sistem informasi tersebut, calon penanya hanya perlu mencari penyakit yang akan ditanyakan pada sistem informasi tersebut. Apabila tidak ada, penanya dapat mengajukan pertanyaan, kemudian pertanyaan dan jawaban yang diberikan akan terindeks oleh sistem dan menambah informasi.

Selain itu, hasil penelitian ini juga dapat digunakan untuk membangun sistem yang mengenali tren penyakit pada masyarakat, sehingga pihak terkait mampu menentukan langkah strategis yang tepat.

## 1.4 Metodologi Penelitian

Berikut merupakan metode penelitian yang penulis lakukan.

### 1. Studi Literatur

Pada tahapan ini penulis mencari literatur yang terkait dengan penelitian ini. Literatur ini digunakan sebagai bahan pembelajaran dan untuk mendukung penelitian yang penulis lakukan. Literatur yang penulis gunakan memiliki keterkaitan terhadap kasus MER , *Sequence Labelling* dan *Recurrent Neural Network*.

### 2. Pengumpulan Data

Pada tahapan ini, penulis mengumpulkan data percobaan yang diperlukan. Penulis mengumpulkan dokumen teks dari forum kesehatan *onlune* dan dari penelitian Herwando (2016). Setelah dokumen terkumpul, penulis melakukan langkah-langkah pra-pemrosesan baik pada dokumen yang penulis dapatkan dari forum maupun korpus dari penelitian Herwando (2016). Tujuan langkah tersebut yaitu untuk menghilangkan beberapa karakter yang mengganggu tahapan selanjutnya, melakukan normalisasi pada beberapa kasus token, dll. Setelah itu penulis melakukan tokenisasi dan melakukan pemecahan kalimat dengan menggunakan beberapa aturan, kemudian penulis memberikan label pada dokumen yang penulis dapat dari forum secara manual.

### 3. Pengembangan Model

Pada tahapan ini, penulis melakukan perancangan eksperimen yang akan dilakukan. Penulis mendefinisikan fitur-fitur yang akan diuji pada penelitian ini dan arsitektur RNNs yang juga akan diuji.

### 4. Eksperimen

Tahapan ini merupakan bagian inti dari penelitian. penulis melakukan langkah eksperimen dengan tujuan mendapatkan jawaban dari pertanyaan yang telah dirumuskan pada rumusan masalah. Sebelum masuk di tahap ini, penulis melakukan pemecahan data menjadi 10 bagian untuk mengimplementasikan *10-cross fold validation*. Setelah itu, data disusun sedemikian sehingga siap digunakan sebagai *resource* eksperimen.

### 5. Evaluasi dan Analisis Hasil

Pada tahapan ini penulis melakukan evaluasi dan analisis dari hasil eksperimen. Untuk mengukur akurasi dari masing-masing fitur dan arsitektur RNNs yang penulis usulkan, saya menggunakan *precision*, *recall* dan *f-measure*.

### 6. Penarikan Kesimpulan

Tahap ini merupakan tahap terakhir dari penelitian. Setelah melakukan

serangkaian eksperimen, evaluasi dan analisis, penulis memberikan kesimpulan dan informasi penting terkait penelitian ini. Selain itu penulis juga memberikan saran untuk penelitian selanjutnya.

## 1.5 Ruang Lingkup Penelitian

Pada penelitian ini terdapat beberapa batasan yang penulis tentukan, yaitu:

### 1. Entitas Kesehatan

Pengenalan entitas kesehatan pada penelitian ini berfokus pada pengenalan nama penyakit (*disease*), gejala-gejala penyakit (*symptom*), nama obat (*drug*) dan langkah pengobatan (*treatment*),

### 2. Domain Pengenalan

Pengenalan entitas kesehatan dilakukan pada bagian judul pertanyaan, isi pertanyaan/keluhan dan isi jawaban dari dokter.

## 1.6 Sistematika Penulisan

Sistematika penulisan dalam laporan penelitian ini sebagai berikut:

- Bab 1 PENDAHULUAN

Pada bab ini penulis menjelaskan mengenai motivasi dalam melakukan penelitian ini dan komponen-komponen utama penelitian seperti latar belakang, perumusan masalah, tujuan dan manfaat penelitian, metodologi penelitian, ruang lingkup penelitian dan sistematika penulisan.

- Bab 2 LANDASAN TEORI

Pada bab ini penulis melakukan studi literatur mengenai beberapa teori dan penelitian yang dilakukan oleh penulis lain.

- Bab 3 METODOLOGI

Pada bab ini penulis menjelaskan alur dari penelitian ini, yaitu pengumpulan data, pra-pemrosesan, pelabelan, pengembangan model, eksperimen dan evaluasi.

- Bab 4 IMPLEMENTASI

Pada bab ini penulis menjelaskan proses implementasi sistem dan eksperimen berdasarkan rancangan yang telah Wahid Nur Rohman tentukan pada bab sebelumnya. Selain itu penulis juga menjelaskan implementasi dari masing-masing tahapan yang dilakukan.

- **Bab 5 EKSPERIMEN**

Pada bab ini penulis menjelaskan analisis dari hasil eksperimen yang telah penulis kerjakan pada tahap sebelumnya. Hasil eksperimen penulis sajikan dalam bentuk tabel dan grafik.

- **Bab 6 KESIMPULAN DAN SARAN**

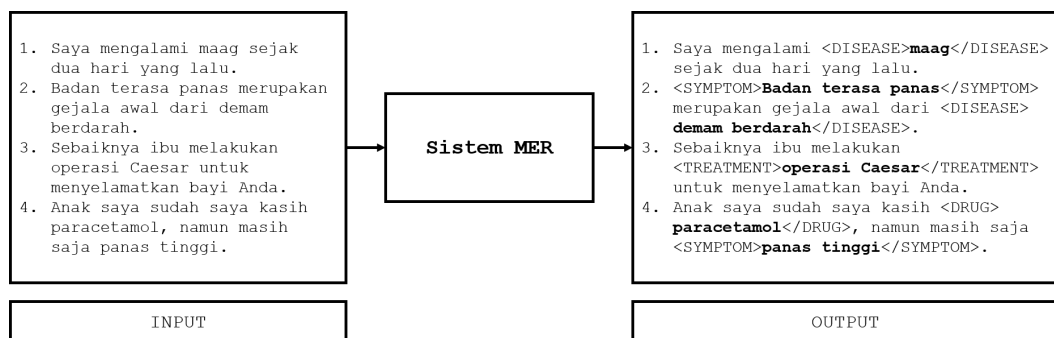
Pada bab ini penulis memberikan kesimpulan berdasarkan hasil eksperimen dan analisis yang telah dilakukan pada penelitian ini. Selain itu penulis juga memberikan saran dan masukan untuk penelitian dan pengembangan sistem mengenai MER berbahasa Indonesia selanjutnya.

## BAB 2

### LANDASAN TEORI

#### 2.1 Pengenalan Entitas Kesehatan

Pengenalan Entitas Kesehatan atau disebut juga dengan *Medical Entity Recognition* (MER) merupakan salah satu cabang dari Pengenalan Entitas Bernama (NER) dengan dokumen sumber berupa teks kesehatan. NER sendiri merupakan suatu sistem/aplikasi yang memanfaatkan teknik pada *Natural Language Processing* dan *Information Extraction* untuk mengenali entitas yang telah dikategorikan sebelumnya seperti nama, lokasi, organisasi, waktu dan sebagainya. Sedangkan pada sistem MER, entitas yang akan dikenali yaitu nama penyakit (*disease*), gejala penyakit (*symptom*), obat (*drug*) dan langkah penyembuhan (*treatment*), nama protein, DNA, RNA dll. Gambar 2.1 merupakan ilustrasi dari sebuah sistem MER.



Gambar 2.1: Ilustrasi Sistem MER

Dari ilustrasi di atas, sebuah sistem MER akan diberikan *input* berupa dokumen kesehatan, kemudian sistem diharapkan dapat memberikan *output* berupa dokumen yang sudah diberi label dengan benar. Dokumen kesehatan yang menjadi *input* dapat berupa dokumen formal seperti dokumen suatu rumah sakit atau dokumen non-formal seperti dokumen forum kesehatan *online*.

Implementasi sistem MER dapat memberikan manfaat pada beberapa bidang, seperti pada aplikasi *Question Answering* (Abacha dan Zweigenbaum, 2011) yang hasil pelabelan dari sistem MER dapat mempermudah identifikasi entitas yang ditanyakan. Selain itu, hasil pelabelan sistem MER juga dapat dimanfaatkan untuk pembuatan sistem *indexing* dokumen forum sehingga pencarian dokumen kesehatan dapat dilakukan dengan lebih efisien. Sistem MER juga dapat digunakan untuk mendukung aplikasi *entity linking* yang memungkinkan seseorang untuk

mengetahui hubungan antar entitas (Hachey et al., 2013). Misalnya dengan adanya aplikasi *entity linking*, kita dapat mengetahui obat apabila hanya diberikan *query* nama penyakit dengan *resource* dokumen-dokumen kesehatan yang telah mendapatkan pelabelan dari sistem MER. Masih banyak manfaat lain dari implementasi sistem MER ini.

Sebelumnya Abacha dan Zweigenbaum (2011) telah melakukan penelitian terkait sistem MER pada dokumen berbahasa Inggris. Sistem MER yang dibuat bertujuan untuk melabeli entitas *treatment*, *problem* dan *test* dengan menggunakan 3 metode, yaitu (i) metode semantik dengan menggunakan *tools* MetaMap (*domain knowledge*), (ii) ekstraksi frasa berdasarkan *chunker* dan klasifikasi dengan SVM (*Support Vector Machine*) dan (iii) gabungan 2 metode sebelumnya dengan menggunakan CRF (*hybrid*). Metode *hybrid* yang dimaksud yaitu dengan menggunakan *tools* CRF sebagai *tools machine learning* yang ditambahkan fitur *domain knowledge*, yaitu fitur semantik yang diekstraksi dengan *tools* MetaMap. Hasil yang terbaik didapatkan dengan menggunakan metode *hybrid* yang menggabungkan 2 metode sebelumnya (*domain knowledge* dan *machine learning*) dan dengan *precision* 72.18%, *recall* 83.78% dan *f-measures* 77.55%.

Selain penelitian di atas, Mujiono et al. (2016) juga melakukan penelitian terkait MER pada dokumen berbahasa Indonesia. Model MER yang dikembangkan adalah untuk melabeli entitas *drug* saja. Penelitian tersebut bertujuan untuk mendapatkan representasi data yang berdasarkan karakteristik *training data*. Mujiono et al. (2016) mengusulkan tiga teknik representasi data yang berdasarkan karakteristik distribusi kata dan kemiripan kata dari hasil *training* dari model *word embedding*. Representasi data yang dimaksud adalah: (i) semua kalimat diformat sebagai *sequence* token, (ii) semua kalimat di-*generate* menjadi beberapa *sequence*, dan (iii) data direpresentasikan sebagai vektor dengan *tools* Word Embedding. Masing-masing representasi kata tersebut dievaluasi dengan masing-masing evaluator, yaitu (i) evaluasi dengan model *neural networks* standar, (ii) evaluasi dengan dua *deep network classifiers*, yaitu DBN (*Deep Belief Networks*), dan SAE (*Stacked Denoising Encoders*) serta (iii) representasi kalimat sebagai vektor *word embedding* yang dievaluasi dengan *recurrent neural networks* yaitu LSTM (*Long Short Term Memory*). Hasil yang didapatkan yaitu kalimat sebagai *sequence* yang dievaluasi dengan LSTM memberikan hasil yang terbaik, yaitu *f-measure* 86.45%.

Penelitian terkait MER pada dokumen berbahasa Indonesia sudah dilakukan sebelumnya oleh Herwando (2016). Dalam penelitiannya, Herwando (2016) menggunakan CRF (*Conditional Random Fields*) untuk proses pelabelan. Kemudian, pada pekerjaan yang Herwando (2006) lakukan, sebagian besar

digunakan untuk mencari fitur-fitur yang memang diskriminatif untuk masalah *sequence labelling* yang menghasilkan akurasi terbaik. Entitas yang akan diberi label yaitu nama penyakit (*disease*), gejala penyakit (*sympton*), obat (*drug*) dan langkah penyembuhan *treatment*. Dokumen yang menjadi input penelitian merupakan hasil *crawling* dari forum kesehatan *online* dari berbagai situs yang berisi tanya jawab. Hasil yang didapatkan yaitu *precision* 70.97%, *recall* 57.83% dan *f-measure* 63.69% dengan fitur *its own word*, frasa, kamus (*symptom*, *disease*, *treatment* dan *drug*), kata pertama sebelum dan panjang kata.

Selain itu, Suwarningsih et al. (2014) juga melakukan penelitian terkait MER pada dokumen berbahasa Indonesia dengan menggunakan SVM (*Support Vector Machine*), dengan SVM yang digunakan untuk klasifikasi per-kata. Entitas yang akan dikenali yaitu *location*, *facility*, *diagnosis*, *definition* dan *person*. Data yang digunakan sebagai korpus merupakan data dari situs *health.detik.com*, *detikhealth.com* dan *health.kompas.com/konsultasi* dengan total keseluruhan sebanyak 1000 kalimat. Akurasi yang dihasilkan yaitu 90% dengan menggunakan fitur *baseline*, *word level (morphology, POS-Tag, dll)* dan fitur dari dalam dokumen tersebut.

## 2.2 Deep Learning

*Deep Learning*, atau disebut juga *deep structured learning*, *hierarchical learning*, dan *deep machine learning* merupakan salah satu cabang dalam *machine learning* dimana model komputasinya yang terdiri dari beberapa layer mampu mempelajari dan mengekstrak representasi data/fitur secara otomatis pada abstraksi tingkat tinggi (LeCun et al., 2015). Model tersebut memberikan hasil yang sangat baik dalam penelitian di berbagai bidang seperti *speech recognition*, *object detection*, *sequence labelling* dan lain sebagainya. Struktur pembelajaran hierarki pada *deep learning* termotivasi dari bagaimana neokorteks pada otak manusia bekerja secara mendalam. Neokorteks tersebut melakukan proses pembelajaran berlayer dan secara otomatis mampu mengekstrak fitur dan melakukan abstraksi dari *resource* yang diberikan (Bengio et al., 2007).

Dalam proses pembelajarannya *deep learning* mampu menemukan struktur intrinsik dalam dataset yang besar dengan cara menerapkan *backpropagation* untuk menentukan parameter internal di dalam model. Berbeda dengan



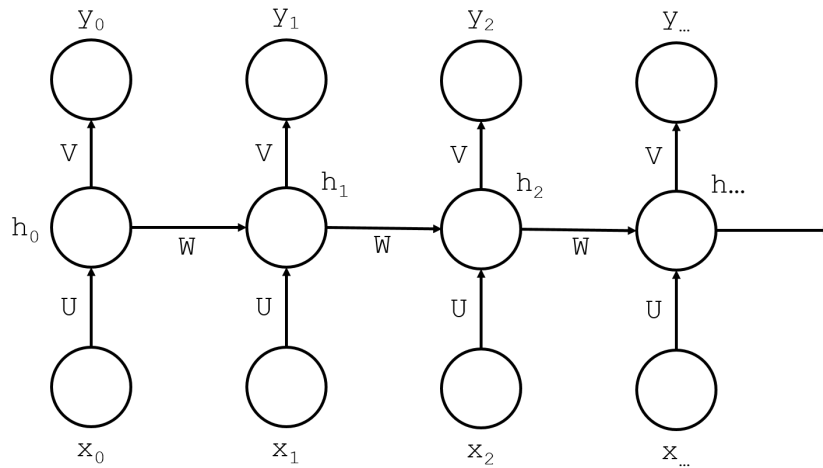
## 2.3 Recurrent Neural Networks

*Recurrent neural networks* (RNNs) merupakan salah satu arsitektur *Deep Learning* yang memiliki koneksi siklik (Graves, 2012). RNNs memiliki *neuron* yang terkoneksi dengan *neuron* lain sehingga membentuk *loop* umpan balik (Haykin et al. (2009)), tidak seperti *feedforward neural network* (FNNs) dimana aliran informasi hanya berjalan searah. RNNs memungkinkan *output* yang dihasilkan akan menjadi *input* untuk menghasilkan *output* yang lain. Hal ini menyebabkan perilaku RNNs tidak hanya bergantung pada *input* saat ini saja, namun juga bergantung pada *output* sebelumnya. Oleh karena itu, RNNs memiliki kemampuan yang sangat bagus sebagai model dalam permasalahan *sequence data* dibandingkan dengan FNNs. RNNs sendiri memiliki kemampuan yang sangat bagus dalam beberapa *task*, seperti *language model* (Mikolov et al. (2010)) dan *speech recognition* (Graves et al. (2013)).

Dibandingkan dengan FNNs, RNNs memiliki beberapa kelebihan (Mikolov et al., 2010), yaitu:

1. Pada RNNs, kata-kata sebelumnya direpresentasikan dengan *recurrent connections*, sehingga RNNs dapat menyimpan informasi kata sebelumnya dalam jumlah tak hingga. FNNs tidak bisa secara alami memodelkan hubungan kontekstual antara sebuah kata dengan kata-kata pada posisi sebelumnya dan representasi kata sebelumnya berupa konteks dari  $n - 1$  kata. Oleh karena itu, FNNs terbatas dalam penyimpanan informasi kata sebelumnya terbatas seperti pada model n-gram.
2. RNNs dapat melakukan kompresi keseluruhan riwayat kata menjadi ruang dimensi yang lebih kecil, sedangkan FNNs melakukan kompresi/proyeksi hanya dengan sebuah kata saja.

Banyak variasi RNNs yang telah diusulkan oleh beberapa peneliti, seperti *Elman networks* (Elman, 1990), *Jordan networks* (Jordan, 1986), *time delay neural networks* (Lang et al., 1990) dll. Gambar berikut merupakan contoh dari RNNs secara umum



**Gambar 2.2:** *Recurrent Neural Networks sederhana*

Dari gambar 2.2, sebuah jaringan pada RNNs memiliki 3 layer pada setiap *timestep*, yaitu *input layer*, *hidden layer* dan *output layer*. *Input layer* merupakan layer sebagai tempat masuk *resource*. Di dalam *hidden layer* tersebut terdapat beberapa unit untuk menyimpan informasi dari *timestep* sebelumnya. Sedangkan pada *output layer* merupakan layer yang memberikan *output* dari model. Pada setiap *timestep*  $t$ , RNNs di atas memiliki sebuah *input layer*  $\vec{x}(t) \in \mathbb{R}^N$ , *hidden layer*  $\vec{h}(t) \in \mathbb{R}^H$ , dan *output layer*  $\vec{y}(t) \in \mathbb{R}^M$ . Nilai  $N$ ,  $H$ , dan  $M$  merupakan panjang vektor *input*, jumlah unit di dalam *hidden layer* tersebut, dan panjang vektor *output* yang diinginkan. Terdapat tiga parameter yang akan diestimasi, yaitu  $U \in \mathbb{R}^{H \times N}$ ,  $V \in \mathbb{R}^{M \times H}$ , dan  $W \in \mathbb{R}^{H \times H}$ . Tiga parameter tersebut bersifat *shared*, yang artinya masing-masing *timestep* menggunakan dan mengestimasi tiga parameter tersebut.

Apabila tiga parameter di atas sudah diketahui,  $\vec{h}(t)$  dan  $\vec{y}(t)$  dapat dihitung dengan persamaan:

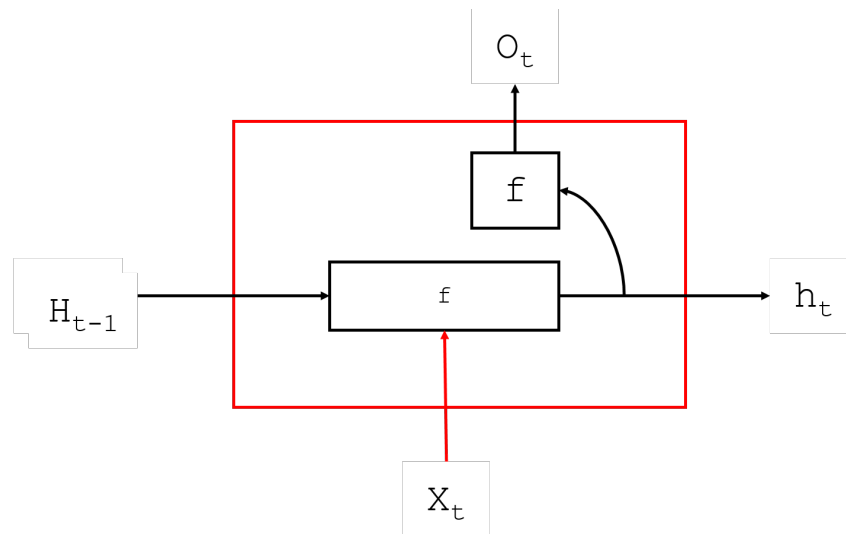
$$\vec{y}(t) = f(V \cdot \vec{h}(t)) \quad (2.1)$$

$$\vec{h}(t) = f(U \cdot \vec{x}(t) + W \cdot \vec{h}(t-1)) \quad (2.2)$$

dimana

$$\vec{h}(0) = f(W \cdot \vec{x}(0)) \quad (2.3)$$

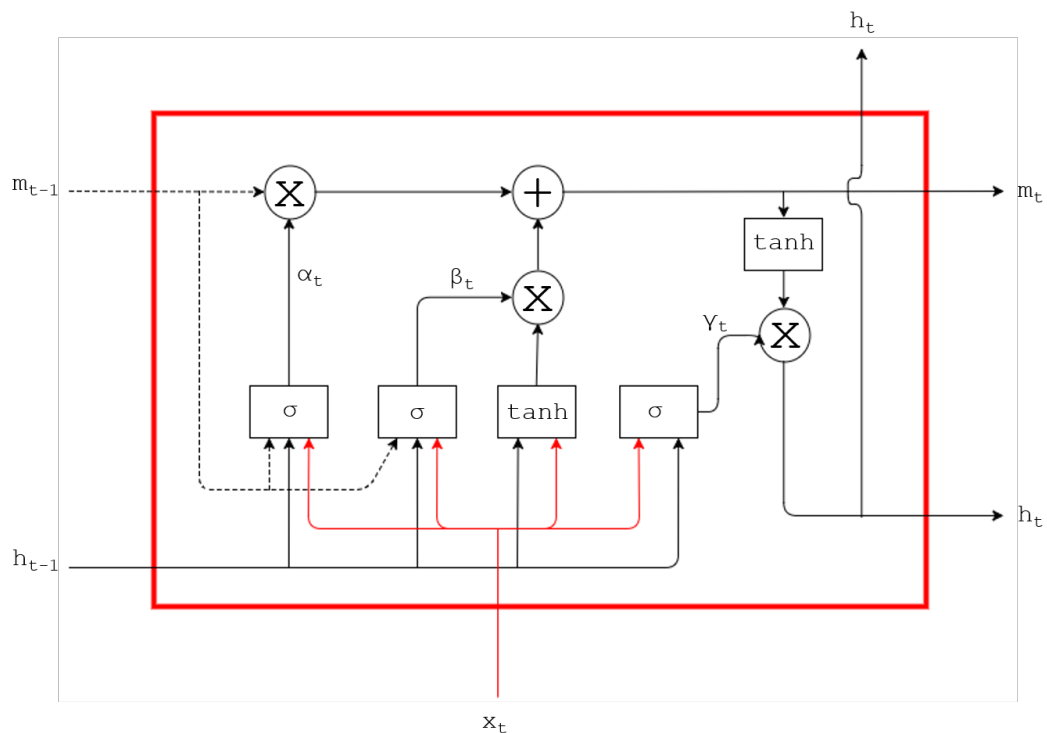
dengan  $f$  sebagai *activation function*, misalnya *tanh* atau *softmax*. Untuk lebih jelasnya, berikut merupakan gambar dari satu buah *timestep* di dalam RNNs.



**Gambar 2.3:** 1 buah *timestep* dalam RNNs

### 2.3.1 Long Short Term Memory

Pada penjelasan di atas, RNNs memiliki kelebihan mempertimbangkan konteks untuk mengolah *input* menjadi *output*. Sayangnya, *range* konteks yang dapat digunakan dalam satu blok terbatas (Graves, 2012). Efek dari keterbatasan ini yaitu informasi pada suatu blok akan hilang atau terganggu dalam perjalanan *timestep* sehingga *output* yang dihasilkan tidak sesuai harapan. Oleh karena itu RNNs tidak dapat menangani permasalahan dependensi jangka panjang. Permasalahan ini disebut dengan *vanishing gradient problem* (Hochreiter (1991); Hochreiter et al. (2001); Bengio et al. (1994)). Banyak upaya untuk mengatasi masalah ini, seperti dengan menggunakan *simulated annealing* dan *discrete error propagation* (Bengio et al., 1994), menggunakan *time delays* (Lang et al. (1990); Bakker (2001)) atau *time constant* (conference on neural information processing systems et al., 1997), dan *hierarchical sequence compression* (Schmidhuber et al., 2007). Namun sejauh ini solusi yang paling bagus yaitu dengan arsitektur *Long Short Term Memory* (LSTM) (Hochreiter dan Schmidhuber, 1997).



**Gambar 2.4:** 1 buah blok memori dalam LSTM

LSTM diperkenalkan oleh Hochreiter dan Schmidhuber (1997) dan saat ini banyak digunakan dalam berbagai *task*. Gambar 2.4 merupakan ilustrasi satu buah blok memori di dalam LSTM. Pada dasarnya, arsitektur LSTM mirip dengan RNNs, namun unit *nonlinear* pada *hidden layer* di dalam RNNs diganti menjadi blok memori. Sebuah blok memori memiliki gerbang *multiplicative* yang berfungsi untuk menyimpan dan mengakses informasi dari blok sebelumnya namun dengan batasan yang jauh lebih besar dibanding RNNs, sehingga mampu menghindari *vanishing gradient problem*. Apabila *input gate* selalu tertutup, maka memori tidak akan pernah ditimpa sehingga isi memori tidak berubah.

Pada gambar 2.4, kita dapat melihat bahwa 1 blok memori pada LSTM tersebut memiliki 3 buah gerbang, yang berfungsi untuk sebagai pengatur suatu informasi apakah ditambahkan, dipertahankan atau dihapus di dalam sebuah sel. Masing-masing gerbang terdiri dari komponen *sigmoid layer* dan komponen untuk melakukan operasi penjumlahan atau perkalian untuk masing-masing *element-wise*. *Sigmoid layer* tersebut memiliki nilai antara nol sampai dengan satu, yang mendeskripsikan perilaku gerbang dalam menerima *input*. Semakin kecil nilai dari layer tersebut maka semakin kecil pula informasi masuk ke gerbang terkait dan sebaliknya.

#### 1. Forget Gate

Gerbang ini memiliki fungsi untuk menentukan informasi yang akan disimpan di dalam memori dengan formula berikut

$$\alpha_t = \tau(W_{x\alpha} + W_{h\alpha} \cdot h_{t-1} + W_{m\alpha} \cdot m_{t-1}) \quad (2.4)$$

## 2. *Input Gate*

Gerbang ini berfungsi untuk menentukan apakah informasi baru  $x(t)$  akan disimpan dalam *cell state* atau tidak.

$$\beta_t = \tau(W_{x\beta} + W_{h\beta} \cdot h_{t-1} + W_{m\beta} \cdot m_{t-1}) \quad (2.5)$$

## 3. *Output Gate*

Gerbang ini berfungsi untuk menendukan *output* dari sebuah *timestep* berdasarkan *cell state* saat ini.

$$\gamma_t = \tau(W_{x\gamma} + W_{h\gamma} \cdot h_{t-1} + W_{m\gamma} \cdot m_{t-1}) \quad (2.6)$$

Dalam setiap *timestep*  $t$ , berikut merupakan formula untuk menghitung  $m(t)$  dan  $h(t)$ :

$$m_t = \alpha_t(\times)m_{t-1} + \beta_t(\times)f(x_t, t-1) \quad (2.7)$$

$$h_t = \gamma_t(\times)\tanh(m_t) \quad (2.8)$$

dimana

$$f(x_t, t-1) = \tanh(W_{xm} \cdot x_t + W_{hm} \cdot h_{t-1}) \quad (2.9)$$

Notasi  $(\times)$  merupakan operasi perkalian untuk setiap pasang elemen, dan  $(+)$  merupakan operasi penjumlahan setiap pasang elemen.

### 2.3.2 Penerapan RNNs untuk MER

Terdapat beberapa penelitian terkait MER yang dikembangkan menggunakan RNNs, seperti *drug entity recognition* (Mujiono et al., 2016), *medical event detection on EHR* (Jagannatha dan Yu, 2016), *biomedical entity recognition* (Limsopatham dan Collier, 2016), dan *Named Entity Recognition in Swedish Health Records* (Almgren et al., 2016). Penelitian *drug entity recognition* oleh Mujiono et al. (2016) sudah dijelaskan pada subbab 2.1.

Dalam penelitiannya, Jagannatha dan Yu (2016) menggunakan LSTMs untuk memprediksi label entitasnya. Penelitian tersebut bertujuan untuk mendeteksi kejadian klinik pada *Electronic Health Records* seperti *medication*, *diagnosis*

(*Indication*), *adverse drug events (ADEs) severity*, *other SSD*, *frequency*, *drugname* dan *duration*. Sebagai pembanding, penulis tersebut juga mengimplementasikan CRF dan GRU. Ada beberapa kesulitan yang dihadapi dalam mengolah EHR tersebut, yaitu EHR lebih *noisy* dibandingkan dengan teks biasa, banyak kalimat yang tidak komplet dan penggunaan frasa. Hasil dari penelitian tersebut menunjukkan bahwa semua model RNNs (LSTMs dan GRU) memiliki akurasi yang lebih baik daripada CRF. Apabila dibandingkan dengan *baseline* yang digunakan, GRU mampu meningkatkan *recall* (0.8126), *precision* (0.7938) dan *F-score* (0.8031) sebesar 19%, 2% dan 11%.

Limsopatham dan Collier (2016) menggunakan *Bidirectional-LSTMs* untuk mengidentifikasi kalimat dengan menggunakan karakter dan kata yang diubah menjadi vektor menggunakan *word embedding*. Untuk setiap kalimatnya, peneliti tersebut mengusulkan adanya *ortographic feature* supaya modelnya dapat mempelajari fitur tersebut secara eksplisit. Evaluasi yang digunakan menggunakan tiga buah koleksi *biomedical test*, yaitu *Gene Mention task corpus*, *BioNLP 2009* dan *NCBI disease corpus*, dengan perhitungan *F1-score*. ada empat *baseline* yang digunakan sebagai pembanding, yaitu *feedforward*, *bidirectional-LSTM*, *CNN-Bidirectional LSTM* yang hanya menggunakan karakter dan *CNN-Bidirectional LSTM*. Hasil yang didapatkan mengatakan bahwa penggunaan *Bidirectional LSTM* yang dikombinasikan dengan CNN dengan diberikan *word embedding* dan *orthographic* merupakan model yang paling bagus. Penulis tersebut juga menyimpulkan bahwa penggunaan fitur *hand-crafted* tersebut mampu memberikan akurasi yang lebih tinggi.

Almgren et al. (2016) menggunakan *deep bidirectional LSTM* dalam mengembangkan NER di bidang medis. Entitas yang akan diidentifikasi adalah *disorders and findings*, *pharmaceutical drugs*, *body structure* dan *non-entity term*. Model menggunakan teks medis berbahasa Swedia sebagai *dataset*, di-*train* dengan menggunakan *end-to-end backpropagation* dan *Adam optimizer* dan *input* yang diberikan berbentuk urutan karakter. *Baseline* yang digunakan yaitu *Stockholm EPR corpus*, yang mana mendapatkan *precision* 0.67, *recall* 0.12 dan *f-measure* 0.20. Hasil yang didapatkan adalah Char-BiLSTM pada Stockholm EPR corpus memiliki *precision* yang lebih tinggi (0.67), dan *recall* yang juga lebih tinggi (0.24) dibandingkan dengan *baseline*.

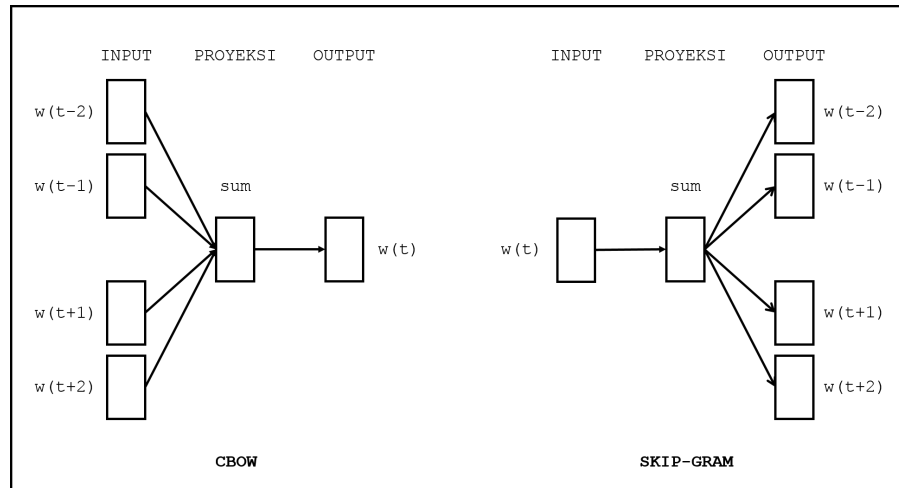
## 2.4 Word Embedding

Pada umumnya, pendekatan yang digunakan untuk merepresentasikan sebuah kata sebagai *input* model adalah dengan menggunakan *one-hot-vector*. Panjang dari sebuah vektor kata ini bergantung dari banyaknya kata unik di dalam sebuah korpus. Ada beberapa cara untuk mengubahnya menjadi *one hot vector*, seperti mengumpulkan semua kata unik kemudian mengurutkannya secara alfabetis. Vektor *one hot* tersebut bernilai 1 pada indeks kata yang bersesuaian. Misalnya kata "obat" berada di indeks ke 25 pada kumpulan kata unik, maka representasi vektornya elemen ke 21 di vektor "obat" adalah 1 sedangkan yang lainnya 0.

Dari ilustrasi singkat tersebut, representasi *one-hot-vector* memiliki kelemahan yaitu besar vektor yang tergantung jumlah kata unik di dalam korpus. Selain itu, jika terdapat sebuah kata yang muncul di korpus namun tidak muncul di *training* ataupun *testing data*, kata tersebut tidak dapat diproses. Selain itu, sangat susah untuk mencari hubungan baik sintaks maupun semantiks dari representasi kata ini, karena antar kata hanya dibedakan indeks yang berisi angka 1 saja.

Dari kelemahan di atas, terdapat sebuah representasi vektor lain dari kata yang lebih baik, yaitu dengan menggunakan *word embedding*. *Word embedding* adalah salah satu jenis dari representasi kata yang memiliki kelebihan yaitu padat, berdimensi rendah, dan memiliki nilai yang real. *Word embedding* memetakan kata dengan vektor berisi bilangan *real*, misalkan  $W(\text{"obat"}) = [0.4, -0.9, 0.1, \dots, 0.9]$ , dimana  $W$  adalah fungsi yang memetakan suatu kata menuju representasi vektor dan  $W(\text{"obat"})$  merupakan *word embedding* dari kata "obat". *Word embedding* dapat meningkatkan performa dari *tasks* dalam NLP dengan cara mengelompokkan kata-kata yang mirip, karena kata yang mirip memiliki vektor yang mirip pula. Ada beberapa metode *word embedding* yang banyak digunakan dalam beberapa *task* di NLP, seperti Glove (Pennington et al., 2014) dan Word2Vec (Mikolov et al., 2014). Pada pembahasan ini, penulis hanya menuliskan mengenai *word2vec*.

Word2Vec merupakan model linguistik yang dikembangkan oleh ? dan berdasarkan pada *neural network*. Word2Vec mempelajari *embedding* dari setiap kata untuk dipetakan ke masing-masing vektor yang berdimensi rendah dari sifat distribusinya pada korpus yang diberikan. Dari situ, Word2Vec mampu mengelompokkan kata berdasarkan kemiripannya di dalam *vector space*.



**Gambar 2.5:** Arsitektur Word2Vec

Ada dua arsitektur Word2Vec yang dikembangkan oleh Mikolov et al. (2014), yaitu arsitektur *skip-gram* dan arsitektur *continuous bag of words* (CBOw). Dari gambar 2.5, dapat dilihat bahwa arsitektur CBOw memprediksi masing-masing kata berdasarkan kata di sekelilingnya. *Input layer* dalam arsitektur ini direpresentasikan dengan *bag of words*. CBOw sendiri dapat mempelajari data dengan ukuran yang sangat besar yang tidak dapat dilakukan oleh model *neural network* yang lain. Sedangkan arsitektur *skip gram* memprediksi kata-kata di sekeliling dan konteksnya berdasarkan sebuah kata yang diberikan (gambar 2.5). *Skip-gram* mampu menangkap *co-occurrence* rata-rata dari dua buah kata di dalam *training set*.



## **BAB 3**

### **METODOLOGI**

Pada bab ini penulis akan menjelaskan metodologi penelitian yang penulis gunakan. Metodologi penelitian yang dilakukan meliputi tahap pengumpulan data, pra-pemrosesan data, pelabelan data, pengembangan model, eksperimen dan evaluasi.

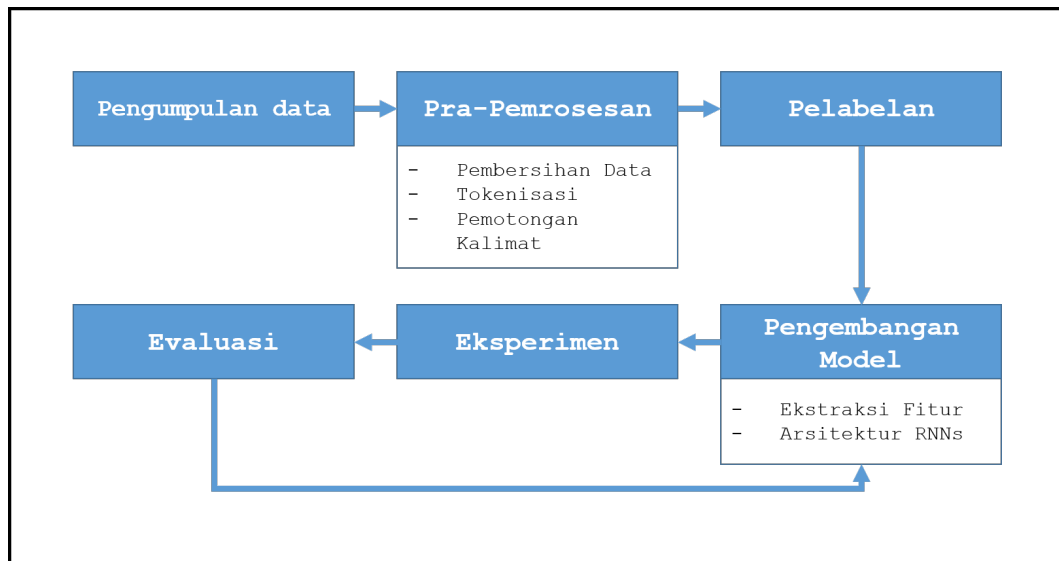
#### **3.1 Gambaran Umum Pengembangan Metodologi yang Digunakan untuk**

Penelitian ini bertujuan untuk membuat sebuah model yang mampu memberikan label entitas kesehatan pada suatu dokumen. Seperti yang telah dijelaskan pada bab sebelumnya, terdapat banyak entitas kesehatan yang dapat digunakan sebagai target pelabelan. Oleh karena itu, untuk mempermudah penelitian ini penulis menggunakan entitas-entitas yang diusulkan oleh Herwando (2016) dalam penelitiannya, nama penyakit (*disease*), gejala penyakit (*symptom*), obat (*drug*) dan langkah penyembuhan (*treatment*).

Penelitian ini menggunakan dua buah korpus, yaitu korpus dari data dokumen teks kesehatan yang digunakan Herwando (2016) dan dokumen teks hasil pengumpulan yang dilakukan oleh penulis pada situs kesehatan *online*. Setelah itu penulis melakukan pra-pemrosesan pada kedua data sebelum melakukan tahap selanjutnya. Untuk dokumen hasil pengumpulan dari forum, penulis memberi label kesehatan secara manual dengan ketentuan pelabelan pada penelitian Herwando (2016)

Setelah tahap pengusulan model, terdapat 2 eksperimen penulis lakukan, yaitu eksperimen untuk mendapatkan fitur diskriminatif yang mampu membuat model memiliki akurasi terbaik dan eksperimen untuk mendapatkan arsitektur RNNs yang membuat model menghasilkan akurasi tertinggi. Pada eksperimen pertama, penulis mencoba beberapa fitur, seperti fitur yang diusulkan oleh Herwando (2016) (fitur *its own word*, frasa, kamus (*symptom*, *disease*, *treatment* dan *drug*), kata pertama sebelum) dan fitur kata setelah. Pada eksperimen kedua, penulis mencoba dua arsitektur RNNs, yaitu RNNs yang setiap *input* digabung terlebih dahulu dengan meng-*append* semua vektor fitur. Sedangkan RNNs yang kedua yaitu RNNs yang setiap kelompok fitur menjadi *input* bagi masing-masing LSTM, baru kemudian *output* dari layer tersebut digabung.

Setelah melakukan eksperimen, penulis melakukan evaluasi dari hasil yang didapatkan dengan menghitung nilai *precision*, *recall* dan *F-measure* dari masing-masing entitas dan entitas secara keseluruhan. Untuk mendapatkan rata-rata akurasi dari setiap eksperimen, penulis melakukan *ten-fold cross validation* dengan cara membagi semua data menjadi 10 bagian, 9 bagian menjadi data *training* dan 1 bagian menjadi data *testing*. Proses tersebut diulang sebanyak sepuluh kali sehingga masing-masing bagian data menjadi data *testing*.



**Gambar 3.1:** Diagram Gambaran Umum Metodologi yang Dilakukan

## 3.2 Pengumpulan Data

Pengumpulan data dilakukan dengan tujuan untuk mendapatkan data *training* dan *testing* yang akan digunakan sebagai *resource* dalam melakukan *training* dan evaluasi model MER. Data yang dimaksud merupakan teks dari forum kesehatan *online* dari berbagai sumber. Pada penelitian ini, penulis menggunakan data penelitian Herwando (2016) dan data yang penulis dapatkan dari hasil *crawling* di forum kesehatan *online*. Data yang Herwando (2016) diambil dari beberapa situs forum kesehatan *online* dan sedangakan data yang penulis unduh bersumber dari forum kesehatan *online*.

## 3.3 Pra-Pemrosesan

Pra-pemrosesan dilakukan dengan tujuan supaya teks yang diberikan mampu dibaca oleh sistem MER. Dalam tahap ini, ada tiga pekerjaan utama yang perlu dilakukan, yaitu:

### 3.3.1 Pembersihan data

Langkah ini dilakukan dengan tujuan untuk mempermudah proses POS *tagging*. Selain itu, terdapat beberapa token yang berbeda sintaks namun memiliki jenis kata yang sama, misalnya token *email*. Model hanya perlu tahu token tersebut merupakan email, tidak peduli pemilik email tersebut. Berikut merupakan beberapa langkah yang penulis lakukan:

1. menghapus karakter yang bukan merupakan karakter ASCII,
2. mengganti token url menjadi kata "url", misalnya token tautan ([www.alodokter.com/asma/pengobatan](http://www.alodokter.com/asma/pengobatan)) diganti menjadi token "url",
3. mengganti token *email* menjadi kata "email", misalnya sebuah alamat *email* ([wahid@domain.com](mailto:wahid@domain.com)) diganti menjadi token "email",
4. mengganti karakter "\_" menjadi token "underscore",
5. mengganti karakter "&" menjadi token "dan",
6. mengganti karakter "<" dan ">" menjadi token "kurang dari" dan "lebih dari" dan
7. mengganti karakter "/" menjadi token "atau".

Pada langkah ini, penulis tidak menghapus karakter tanda baca karena karakter tersebut memiliki fungsi pada sistem POS *tagging* yang penulis gunakan.

### 3.3.2 Tokenisasi

Tokenisasi dilakukan untuk mendapatkan token yang paling tepat sebagai sebuah kata. Hal ini perlu dilakukan untuk menghindari beberapa kelompok token berbeda yang tergabung. Karakter abjad dengan karakter angka atau karakter abjad dengan karakter tanda baca dipisahkan berdasarkan kelompoknya. Misalnya token "pusing2" diubah menjadi "pusing 2". Pada tahap ini, penulis melakukan pemisahan terhadap beberapa kelompok token, yaitu:

1. <alfabet><numeric> menjadi <alfabet><spasi><numeric>
2. <numerik><alfabet> menjadi <numerik><spasi><alfabet>
3. <alfanumerik><non-alfanumerik> menjadi <alfanumerik><spasi><non-alfanumerik>

4. <alfanumerik><non-alfanumerik> menjadi <alfanumerik><spasi><non-alfanumerik>

### 3.3.3 Pemotongan kalimat

Untuk menghindari jumlah token yang timpang dalam kalimat yang berbeda dan data yang *sparse*, penulis melakukan pemotongan kata dengan langkah-langkah sebagai berikut:

1. memisahkan kalimat berdasarkan tanda baca (.!?,),
2. apabila suatu kalimat memiliki jumlah kata yang sedikit (batasan minimal jumlah kata dalam sebuah kalimat yang penulis gunakan adalah 10 kata), kalimat tersebut digabungkan dengan kalimat setelahnya.

## 3.4 Pelabelan

Pada tahap ini, penulis melakukan pelabelan pada dokumen teks yang merupakan hasil pada tahap sebelumnya dengan label *disease*, *symptom*, *drug* dan *treatment*. Berikut merupakan penjelasan dari masing-masing label:

### 1. *Disease*

Entitas *disease* yang dimaksud pada penelitian ini yaitu nama dari suatu penyakit. Penyakit merupakan keadaan abnormal yang timbul pada tubuh manusia. Contoh dari entitas *disease* yaitu:

- Skizofrenia
- Trikotilomania
- Diabetes melitus

### 2. *Symptom*

Entitas *symptom* yang dimaksud pada penelitian ini yaitu fenomena yang dialami oleh seseorang yang terkena suatu penyakit. Contoh dari entitas *symptom* yaitu:

- Napas berbunyi
- Benjolan di daerah perut
- Nyeri saat BAK

### 3. *Drug*

Entitas *drug* merupakan entitas nama obat dari suatu penyakit yang memiliki fungsi untuk mengurangi atau menyembuhkan penyakit tersebut. Contoh dari entitas *drug* yaitu:

- Paracetamol
- Diltiazem
- eritropoetin-alfa

### 4. *Treatment*

Entitas *treatment* merupakan cara atau langkah penyembuhan dari suatu penyakit. Contoh dari entitas *treatment* yaitu:

- Pemeriksaan darah rutin
- Penilaian denyut kapiler
- Terapi inhalasi

Setelah proses di atas selesai, label di dalam korpus diubah menjadi format BIO (*begin inside outside*).

## 3.5 Pengembangan Model

Pada tahap ini, penulis melakukan pengusulan dan perancangan model yang nantinya akan penulis evaluasi pada tahap eksperimen. Dalam mengembangkan model, terdapat dua pekerjaan yang penulis lakukan, yaitu:

### 3.5.1 Ekstraksi Fitur

Pada tahap ini, penulis melakukan ekstraksi fitur dari dokumen yang telah diberi label entitas. Ada beberapa fitur yang penulis usulkan dalam penelitian ini yang nantinya penulis kombinasikan supaya mendapatkan hasil terbaik. Fitur-fitur tersebut yaitu:

#### 1. Fitur 1: Kata itu sendiri

Fitur ini merupakan fitur kata dalam representasi vektor. Fitur ini merupakan fitur yang digunakan Abacha dan Zweigenbaum (2011) dalam penelitian tentang MER. Untuk mendapatkan representasi vektor dari masing-masing kata, penulis menggunakan *word embedding*. Pada penelitian mengenai MER yang dilakukan oleh Mujiono et al. (2016), hasil dari representasi data

terbaik yaitu *word embedding*. Selain itu, seperti yang dijelaskan pada Bab Tinjauan Pustaka, *word embedding* memberikan hasil yang sangat baik dalam bidang pemrosesan bahasa manusia. Oleh karena itu, penulis menggunakan *word embedding* untuk mendapatkan representasi vektor masing-masing kata. Dalam penelitian ini. Terdapat beberapa langkah yang perlu penulis lakukan dalam memanfaatkan *word embedding* ini, yaitu:

(a) Pengumpulan data *training* untuk *word embedding*

Penulis melakukan pengumpulan data teks sebagai *resource* untuk melakukan *training* model *word embedding*. Data teks yang penulis gunakan merupakan data teks dari artikel-artikel kesehatan dan data teks forum kesehatan di kaskus. Penulis menggunakan teks berjenis kesehatan supaya *domain word embedding* dengan data *training* untuk model MER sama. Selain itu, terdapat beberapa *term* kesehatan yang susah ditemukan di forum umum.

(b) *Training* untuk mendapatkan model *word embedding*

*Training* dilakukan untuk mendapatkan model yang mampu mendapatkan representasi vektor dari sebuah kata. Panjang vektor yang dihasilkan yaitu 1298 dengan besaran *windows* yaitu 5. Arsitektur yang digunakan untuk melakukan *training* ini adalah *skip-gram*.

(c) Pengubahan kata menjadi vektor dari model yang didapatkan

Pada langkah ini penulis mendapatkan suatu kata menjadi representasi vektor dengan model yang telah penulis dapatkan pada tahap *training* model *word embedding*.

2. Fitur 2: *Part of Speech Tag* (POS-Tag)

Fitur ini merupakan fitur *tag* yang dimiliki setiap kata yang diusulkan oleh Abacha dan Zweigenbaum (2011) dalam penelitiannya di bidang MER. Entitas-entitas tertentu memiliki tag yang sama, misalnya entitas obat dan penyakit pada umumnya memiliki tag "NNP" sehingga dengan digunakannya fitur ini sistem dapat mengenali jenis obat dan penyakit dengan lebih baik. Model POS-Tagger yang penulis gunakan merupakan model POS-Tag berbahasa Indonesia.

3. Fitur 3: *Stopword*

Fitur ini merupakan fitur yang berisi vektor suatu kata merupakan *stopword* atau bukan. Fitur ini penulis gunakan dalam penelitian ini untuk membantu sistem dalam menghindari kesalahan pelabelan suatu kata yang bukan entitas namun dilabeli sebagai entitas.

Ketika melakukan eksperimen, hasil yang penulis dapatkan ternyata lebih bagus apabila mempertahankan fitur ini, oleh karena itu, penulis mengusulkan untuk menggunakan fitur ini. Untuk pembahasan lebih lanjut dibahas pada Bab 5.

#### 4. Fitur 4: Kamus kesehatan

Fitur kamus kata merupakan fitur yang berisi informasi suatu kata terdapat di dalam kamus kesehatan atau tidak. Pada penelitian ini, kamus kesehatan yang dipakai merupakan kamus *disease*, kamus *symptom*, kamus *drug* dan kamus *treatment*. Dengan menggunakan fitur ini diharapkan mampu berkontribusi dalam meningkatkan akurasi karena model akan mempertimbangkan apakah suatu kata termasuk di dalam kamus atau tidak.

#### 5. Fitur 5: Frasa Kata

Pada penelitian ini, penulis mengusulkan fitur Frasa Kata karena entitas *symptom* dan *treatment* pada umumnya merupakan frasa kata kerja. Sedangkan entitas *disease* dan *drug* pada umumnya entitas yang akan dikenali pada penelitian ini merupakan frasa kata benda. Oleh karena itu, penulis berharap bahwa dengan diusulkannya fitur ini akan mampu menambah akurasi dari model yang diusulkan.

Pada penelitian ini ada dua frasa yang diuji-cobakan, yaitu:

- (a) Frasa Kata Benda (Nomina) Menurut Hs (2005), frasa kata benda sendiri merupakan kelompok kata benda yang dibentuk dengan memperluas kata benda ke sekelilingnya. Fitur frasa kata benda yang penulis gunakan dalam penelitian merupakan fitur yang berisi informasi suatu kata atau kumpulan kata merupakan frasa kata benda atau bukan. Dalam menentukan suatu kata merupakan frasa atau bukan, penulis menggunakan aturan pembentukan frasa yang digunakan pada bahasa Indonesia, yaitu:

- NP : NN
- NP : NNP
- NP : PR
- NP : PRP
- NP : NN + NN
- NP : NN + NNP
- NP : NN + PR
- NP : NN + PRP

- NP : NN + JJ
- NP : DT + NN
- NP : RB + NN
- NP : CD + NN
- NP : NND + NN

(b) Frasa Kata Kerja (Verbal)

Menurut Hs (2005), frasa verbal merupakan kelompok kata benda yang dibentuk dengan kata kerja. Fitur frasa verbal yang penulis gunakan dalam penelitian merupakan fitur yang berisi informasi suatu kata atau kumpulan kata merupakan frasa verbal atau bukan. Dalam menentukan suatu kata merupakan frasa atau bukan, penulis menggunakan aturan pembentukan frasa yang digunakan pada bahasa Indonesia, yaitu:

- VP : VB
- VP : VB + NP

6. Fitur 6: 1 Kata Sebelum

Fitur ini merupakan fitur yang berisi informasi kata sebelum kata saat ini yang direpresentasikan dalam bentuk vektor untuk masing-masing kata. Fitur ini digunakan pada penelitian penelitian Herwando (2016) yang juga berkontribusi memberikan hasil terbaik pada penelitiannya. Menurut penulis, ada beberapa entitas yang akan lebih mudah diketahui apabila diketahui kata sebelumnya. Misalnya kata "masuk angin", apabila hanya diberikan informasi kata "angin" tanpa kata "masuk", akan lebih sulit menentukan kata tersebut bagian dari suatu entitas *disease* atau bukan.

7. Fitur 7: 1 Kata Sesudah

Fitur ini merupakan fitur yang berisi informasi kata sesudah kata saat ini yang direpresentasikan dalam bentuk vektor untuk masing-masing kata. Sama seperti pada fitur 1 Kata Sebelum, ada beberapa kasus yang mana apabila suatu kata merupakan sebuah entitas, akan lebih mudah dikenali apabila melihat kata atau konteks setelahnya. Sama seperti contoh pada Fitur 1 Kata Sebelum, misal diberikan kata "masuk angin", apabila hanya diberikan informasi "masuk" tanpa "angin", akan lebih sulit mengenali apakah kata tersebut termasuk entitas *disease* atau bukan. Selain itu, fitur ini juga dapat membedakan kata berentitas dengan kata yang bukan, misalnya kata "masuk angin" dengan "masuk rumah". Apabila informasi pada saat tersebut hanya diberikan kata "masuk" saja tanpa kata setelahnya, akan lebih sulit mengenali kata tersebut termasuk kata berentitas atau bukan.

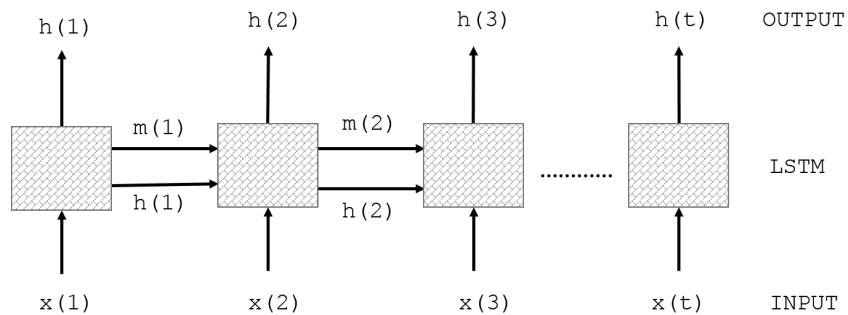


### 3.5.2 Pengusulan Arsitektur RNNs

Pada tahap ini penulis mengusulkan arsitektur RNNs yang akan digunakan pada tahap eksperimen. Ada dua arsitektur yang penulis gunakan dalam penelitian ini, yaitu

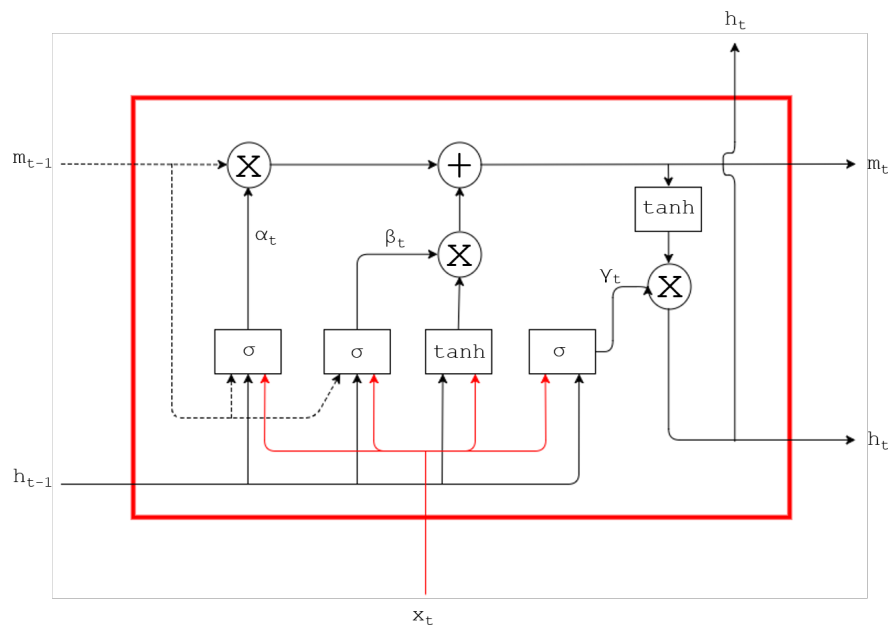
#### 1. LSTM 1 layer

Pada LSTM 1 layer, semua fitur yang menjadi input pada sebuah *timestep* digabung menjadi satu. Untuk menentukan label, penulis menggunakan *feed-forward Neural Network* pada masing-masing *timestep* di layer terakhir. Berikut merupakan ilustrasi LSTM 1 layer yang penulis gunakan dalam penelitian ini.



**Gambar 3.2:** LSTM 1 layer

Untuk masing-masing *timestep*  $t$ , berikut merupakan gambar sebuah *cell*-nya.



**Gambar 3.3:** 1 buah blok memori dalam LSTM

Dari gambar 3.3, sebuah *cell* membutuhkan *input*  $x(t)$  dan *output*  $h(t)$ .  $x(t)$  merupakan vektor dengan panjang  $N$ , dan  $h(t)$  merupakan vektor dengan panjang  $M$ . Seperti yang telah dijelaskan pada subbab 2.3.1, berikut merupakan formula untuk mengetahui *output* pada *timestep*  $t$ .

$$m_t = \alpha_t(\times)m_{t-1} + \beta_t(\times)f(x_t, t-1) \quad (3.1)$$

$$h_t = \gamma_t(\times)\tanh(m_t) \quad (3.2)$$

dimana

$$f(x_t, t-1) = \tanh(W_{xm} \cdot x_t + W_{hm} \cdot h_{t-1}) \quad (3.3)$$

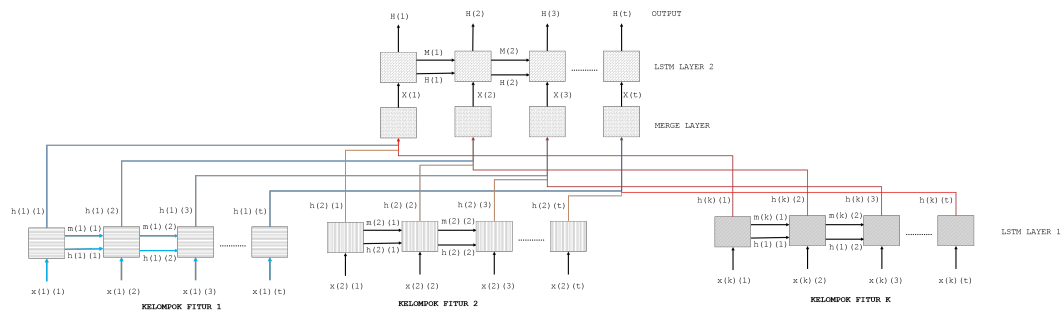
$\alpha_t$ ,  $\beta_t$  dan  $\gamma_t$  merupakan *gates*:

$$(a) \text{ Forget gates: } \alpha_t = \tau(W_{x\alpha} + W_{h\alpha} \cdot h_{t-1} + W_{m\alpha} \cdot m_{t-1})$$

$$(b) \text{ Input gates: } \beta_t = \tau(W_{x\beta} + W_{h\beta} \cdot h_{t-1} + W_{m\beta} \cdot m_{t-1})$$

$$(c) \text{ Output gates: } \gamma_t = \tau(W_{x\gamma} + W_{h\gamma} \cdot h_{t-1} + W_{m\gamma} \cdot m_{t-1})$$

2. LSTM layer bertingkat Pada LSTM 2 layer, penulis mendefinisikan 2 layer, dimana layer terbawah merupakan layer dengan jumlah LSTM sebanyak  $n$  kelompok fitur. Pertama-tama fitur dikelompokkan terlebih dahulu, kemudian dijadikan input untuk LSTM layer pertama. Setelah itu, hasil dari layer pertama tersebut akan digabung menjadi satu, dengan menggunakan layer penggabung. *Output* dari layer penggabung kemudian dimasukkan ke dalam LSTM layer kedua. Untuk menentukan label, penulis menggunakan *feed-forward Neural Network* pada masing-masing *timestep* di layer terakhir. Berikut merupakan ilustrasi dari LSTM layer bertingkat yang penulis gunakan.



**Gambar 3.4:** LSTM 2 layer

Masing-masing kelompok fitur menjadi *input* dari LSTMs yang terkait. Nantinya, masing-masing *output* akan digabung melalui *merge layer* dengan

metode *concat* dan menjadi *input* bagi LSTMs layer kedua. Di sini, penulis menotasikan  $k$  sebagai nomor kelompok fitur dan  $t$  sebagai *timestep* saat ini. Untuk masing-masing kelompok fitur, berikut merupakan formulasi *feedforward*-nya:

$$m_{k,t} = \alpha_{k,t}(\times)m_{k,t-1} + \beta_{k,t}(\times)f(x_{k,t}, k, t-1) \quad (3.4)$$

$$h_{k,t} = \gamma_{k,t}(\times)\tanh(k, m_t) \quad (3.5)$$

dimana

$$f(x_{k,t}, k, t-1) = \tanh(W_{k,xm} \cdot x_t + W_{k,hm} \cdot h_{k,t-1}) \quad (3.6)$$

$\alpha_{k,t}$ ,  $\beta_{k,t}$  dan  $\gamma_{k,t}$  merupakan *gates*:

$$(a) \text{ Forget gates: } \alpha_{k,t} = \tau(W_{k,x\alpha} + W_{k,h\alpha} \cdot h_{k,t-1} + W_{k,m\alpha} \cdot m_{k,t-1})$$

$$(b) \text{ Input gates: } \beta_{k,t} = \tau(W_{k,x\beta} + W_{k,h\beta} \cdot h_{k,t-1} + W_{k,m\beta} \cdot m_{k,t-1})$$

$$(c) \text{ Output gates: } \gamma_{k,t} = \tau(W_{k,x\gamma} + W_{k,h\gamma} \cdot h_{k,t-1} + W_{k,m\gamma} \cdot m_{k,t-1})$$

*Merge layer* berfungsi untuk menggabungkan hasil dari *feedforward* pada semua LSTMs layer pertama. Di sini, saya menotasikan  $X_t$  sebagai hasil dari *merge* di *timestep*  $t$  dan  $(\cdot)$  merupakan operasi *merging*.

$$X_t = h_{1,t}(\cdot)h_{2,t}(\cdot)h_{3,t}(\cdot)....(\cdot)h_{k,t} \quad (3.7)$$

Hasil dari *merge layer* akan digunakan sebagai *input* bagi LSTMs layer kedua. Untuk memudahkan penggunaan notasi dan membedakan dengan LSTMs pada layer pertama, penulis menggunakan huruf kapital dalam menotasikan masing-masing nilai di LSTMs layer kedua. Berikut merupakan formulasi *feed-forward*-nya.

$$M_t = \alpha_t(\times)M_{t-1} + \beta_t(\times)f(X_t, t-1) \quad (3.8)$$

$$H_t = \gamma_t(\times)\tanh(M_t) \quad (3.9)$$

dimana

$$f(X_t, t-1) = \tanh(W_{XM} \cdot X_t + W_{HM} \cdot H_{t-1}) \quad (3.10)$$

$\alpha_t$ ,  $\beta_t$  dan  $\gamma_t$  merupakan *gates*:

$$(a) \text{ Forget gates: } \alpha_t = \tau(W_{X\alpha} + W_{H\alpha} \cdot H_{t-1} + W_{M\alpha} \cdot M_{t-1})$$

$$(b) \text{ Input gates: } \beta_t = \tau(W_{X\beta} + W_{H\beta} \cdot H_{t-1} + W_{M\beta} \cdot M_{t-1})$$

$$(c) \text{ Output gates: } \gamma_t = \tau(W_{X\gamma} + W_{H\gamma} \cdot H_{t-1} + W_{M\gamma} \cdot M_{t-1})$$

## 3.6 Eksperimen

Dalam melakukan eksperimen, arsitektur *deep learning* yang penulis gunakan ada;ah *Recurrent Neural Networks*, dalam hal ini penulis menggunakan LSTM. Hal ini penulis lakukan karena pada penelitian Mujiono et al. (2016), LSTM memberikan *output* terbaik dalam MER yang dirancang. Selain itu, LSTM juga sangat baik dalam masalah *sequence labelling* seperti yang dilakukan oleh Graves et al. (2013) dan merupakan *state-of-the-art* dalam bidang ini. Masih banyak penelitian lain yang membuktikan bahwa LSTM merupakan arsitektur *deep learning* yang sangat baik dalam masalah *sequence labelling* seperti *Offline Handwriting Recognition* (Graves dan Schmidhuber, 2009), *sequence tagging* (Huang et al., 2015), *Sequence to Sequence Learning* (Sutskever et al., 2014) dan lain lain.

Eksperimen yang penulis lakukan menggunakan *10-cross fold validation*, karena keterbatasan data *training* yang penulis miliki. Sebelum melakukan eksperimen, penulis membagi data *training* menjadi 10 bagian, kemudian melakukan iterasi sebanyak 10 kali dimana pada masing-masing iterasi ke-*i*, bagian data ke-*i* menjadi data *testing* dan yang lainnya digabung menjadi data *training*.

Setelah melakukan pembagian dan pengelompokan data berdasarkan nomor iterasi, penulis membuat model dari data *training* tersebut. Setelah penulis mendapatkan model, penulis melakukan testing terhadap masing-masing model dengan data *testing* yang telah disediakan sebelumnya. Hasil dari pelabelan data *testing* ini akan penulis evaluasi di tahap selanjutnya. Setelah itu penulis kembali melakukan pembuatan model dengan fitur yang berbeda, atau dengan tambahan fitur lain. Dalam perjalanan melakukan pengujian, apabila fitur yang diuji memberikan hasil yang bagus atau menambah akurasi, penulis menggabungkan fitur ini ke percobaan selanjutnya. Namun apabila fitur pada saat ini memberikan akurasi yang lebih jelek, penulis tidak menggunakan fitur tersebut di percobaan selanjutnya.

### 3.6.1 Evaluasi

Pada tahap ini, penulis melakukan serangkaian evaluasi dari data *testing* yang telah dilabeli dengan model yang dihasilkan pada tahap eksperimen. Penulis melakukan evaluasi dengan menggunakan metode *partial evaluation* di mana sebuah token yang diprediksi entitas oleh model dihitung benar apabila terdapat fragmen yang

menyusun entitas bernama tersebut (Seki dan Mostafa, 2003). Aturan yang penulis gunakan dalam melakukan evaluasi adalah sebagai berikut:

1. Perhitungan nilai *True Positive* (TP)

Untuk masing-masing kata yang mendapat label entitas benar, nilai *TP* bertambah sejumlah kata yang diprediksi benar.

Misal:

Contoh 1

True: Bu Ani <Disease>**sakit kepala** sebelah</Disease>

Predicted: Bu Ani <Disease>**sakit kepala**</Disease> sebelah

Dari contoh di atas, nilai  $TP = 2$ , karena ada 2 kata yang mendapatkan label entitas yang benar.

Contoh 2

True : <Disease>Masuk angin</Disease> dan <Sympton>**suhu badan tinggi**</Sympton>

Predicted : <Sympton>Masuk angin</Sympton> dan <Sympton>**suhu badan tinggi**</Sympton>

Dari contoh di atas, nilai  $TP = 3$ , karena ada 3 kata yang mendapatkan label entitas yang benar

2. Perhitungan nilai *False Positive* (FP)

Untuk masing-masing kata yang mendapat label entitas namun seharusnya tidak berentitas, nilai *FP* bertambah sejumlah kata yang diprediksi salah.

Misal:

Contoh 1

True : <Disease>Sakit kepala</Disease> sudah **beberapa hari istirahat**

Predicted : <Disease>Sakit kepala</Disease> sudah <Treatment>**beberapa hari istirahat**</Treatment>

Dari contoh di atas, nilai  $FP = 3$ , karena ada 3 kata yang mendapat label entitas yang seharusnya tidak berlabel, yaitu "beberapa hari istirahat".

3. Perhitungan nilai *False Negative* (FN)

Untuk masing-masing kata yang mendapat label entitas salah, nilai *FP* bertambah sejumlah kata yang diprediksi salah.

Misal:

Contoh 1

True : Bu Ani <Disease>sakit kepala sebelah</Disease>

Predicted : Bu Ani <Disease>sakit kepala</Disease> sebelah

Dari contoh di atas, nilai  $FN = 0$ , karena tidak ada kata yang mendapat label entitas salah (kata "sebelah" tidak mendapat label).

Contoh 2

True : <Symptom>Badan terasa pegal</Symptom>, sepertinya akan <Disease>**demam**</Disease>.

Predicted : <Symptom>Badan terasa pegal</Symptom>, sepertinya akan <Symptom>**demam**</Symptom>.

Dari contoh di atas, nilai  $FN = 1$ , karena ada 1 kata yang mendapat label entitas salah, yaitu kata "demam".

Setelah mendapatkan angka  $TP, FP$  dan  $FN$ , penulis menghitung *f-measure*, *precision* dan *recall* untuk masing-masing entitas dengan menggunakan formula:

$$Precision = \frac{TP}{TP + FP} \quad (3.11)$$

$$Recall = \frac{TP}{TP + FN} \quad (3.12)$$

$$F - Measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (3.13)$$

Angka-angka hasil evaluasi ini akan menjadi pertimbangan untuk penggunaan fitur pada saat ini di eksperimen selanjutnya. Apabila akurasi dari penggunaan fitur saat ini lebih baik atau meningkat dari eksperimen sebelumnya, penulis menggunakan fitur ini pada eksperimen selanjutnya. Selain itu, penulis juga mengevaluasi arsitektur RNNs yang penulis gunakan.

## BAB 4

### IMPLEMENTASI

Bab ini akan membahas mengenai implementasi pada penelitian yang terdiri atas tahap pengumpulan data, pra-pemrosesan, pengembangan model, eksperimen dan evaluasi. Setiap fitur yang penulis usulkan pada Bab 3 juga akan dijelaskan langkah pengimplementasian pada bab ini.

#### 4.1 Pengumpulan Data

Penulis melakukan pengumpulan data dengan menggunakan ide implementasi dari Herwando (2016) yang kemudian penulis modifikasi sesuai dengan kebutuhan. Bahasa program yang penulis gunakan untuk melakukan pengumpulan data ini adalah Java, dengan menggunakan library JSoup untuk mengunduh isi forum sebuah situs. Hasil dari pengumpulan data ini penulis gabungkan dengan data penelitian milik Herwando (2016).

---

**Kode 4.1:** *Pseudocode* untuk melakukan pengumpulan data

---

```
1 Function downloadPage(link) is
   |   Input : link of an online health forum
   |   Output: content of forum
2   sql = selectFromDB(link);
3   res = execOnDB(sql);
4   if res != empty then
5   |   insertToDB(sql);
6   |   doc = JSoup.connect(link);
7   |   writeToFile(doc.getJudulKeluhan());
8   |   writeToFile(doc.getIsiKeluhan());
9   |   writeToFile(doc.getJawaban);
```

---

Hasil dari pengumpulan data ini yaitu penulis mendapatkan 2065 *post* dari forum kesehatan *online* pada situs *www.tanyadok.com*.

#### 4.2 Pra-Pemrosesan

Tahap selanjutnya yaitu tahap pra-pemrosesan. Seperti yang telah dijelaskan pada bab metodologi, penulis melakukan tiga buah pekerjaan di tahap ini, yaitu

melakukan pembersihan data, tokenisasi dan pemotongan kalimat. Berikut merupakan penjelasan dari masing-masing pekerjaan tersebut:

#### 4.2.1 Pembersihan Data

Tahap pembersihan data bertujuan untuk menghilangkan karakter yang bukan merupakan ASCII. Hal ini penulis lakukan supaya dalam tahap ekstraksi fitur POS Tagging tidak memiliki masalah karena terdapat karakter bukan ASCII. Selain itu, di dalam dokumen terdapat banyak *email* dan *url* yang unik sehingga mengakibatkan sistem akan menganggap token-token tersebut merupakan token yang unik dan berbeda. Untuk menangani hal tersebut penulis melakukan normalisasi dengan mengubah semua token *email* dan *url* menjadi kata "email" dan "url" sehingga tetap mempertahankan keberadaan kedua token tersebut. Selain itu penulis juga mengganti beberapa karakter yang bukan alfanumerik menjadi beberapa token dalam representasi kata, seperti karakter "&" menjadi "dan", "<" dan ">" menjadi token "kurang dari" dan "lebih dari". Hal ini penulis lakukan karena korpus yang penulis gunakan dalam bentuk berkas *xml* yang tidak mengizinkan adanya ketiga karakter tersebut. Kemudian penulis juga mengubah karakter "/" menjadi "atau" supaya mudah dalam ekstraksi fitur kata itu sendiri dengan menggunakan *word embedding*. Kode 4.2 merupakan *pseudocode* untuk melakukan pembersihan data yang penulis gunakan.

---

**Kode 4.2:** *Pseudocode* untuk melakukan pembersihan data

---

```

1 Function downloadPage(sentence) is
  Input : sentence before cleaning
  Output: sentence which has cleaned
2   sentence.removeByRegex(non-ASCII regex);
3   sentence.replace(email-regex, "email");
4   sentence.replace(url-regex, "url");
5   sentence.replace(&, "dan");
6   sentence.replace(<, "kurang dari");
7   sentence.replace(>, "lebih dari");
8   sentence.replace(/, "atau");
9   return sentence;
```

---

#### 4.2.2 Tokenisasi

Seperti yang dijelaskan pada 3, pada tahap tokenisasi penulis melakukan pemisahan antar kata dan antar token yang berbeda jenis, seperti token alfabet dengan



numerik, alfanumerik dengan non-alfanumerik dan menghilangkan karakter spasi yang berlebih. Dalam mengimplementasikan tahap ini, penulis menggunakan bahasa Ruby. Berikut merupakan *pseudocode* untuk melakukan tokenisasi.

---

**Kode 4.3:** *Pseudocode* untuk melakukan tokenisasi

---

```

1 Function tokenization(sentence) is
    Input : sentence before tokenization
    Output: sentence which has tokenized

2 sentence.replaceByRegex([alfabet][numerik], [alfabet] [numerik]);
3 sentence.replaceByRegex([numerik][alfabet], [numerik] [alfabet]);
4 sentence.replaceByRegex([alfanumerik][non-alfanumerik], [alfanumerik]
    [non-alfanumerik]);
5 sentence.replaceByRegex([non-alfanumerik][alfanumerik],
    [non-alfanumerik] [alfanumerik]);
6 sentence.replaceByRegex([\\s]+, " ");
7 return sentence;

```

---

#### 4.2.3 Pemotongan Kalimat

Implementasi yang penulis lakukan tahap ini bertujuan untuk mendapatkan sebuah *instance* sebagai *input* dari program RNNs di tahap eksperimen. Pemotongan dilakukan pada masing-masing *post*. Pada pemotongan kalimat ini, penulis menerapkan aturan berbeda yang telah dijelaskan pada bab 3 karena jumlah kata pada sebuah kalimat yang dipisahkan dengan tanda baca ".", "!" dan "?" sangat jauh berbeda. Dengan implementasi pemotongan kalimat ini, penulis berupaya untuk menghindari kasus kalimat yang *sparse*, yaitu adanya kalimat yang memiliki jumlah token sangat renggang. Kode 4.4 merupakan *pseudocode* untuk melakukan pemotongan kalimat.

### 4.3 Pelabelan

Pada tahap ini penulis melakukan pelabelan pada data baru yang telah diunduh. Sebelumnya, Herwando (2016) telah melabeli 200 buah *post* dan pada penelitian ini penulis melakukan pelabelan terhadap 109 buah *post* yang penulis pilih dari hasil pengumpulan data. Penulis melakukan pemilihan berdasarkan banyaknya kalimat dalam sebuah *post*. Untuk aturan pelabelan, penulis mengikuti aturan pelabelan yang dilakukan oleh Herwando (2016) dalam penelitiannya. Pelabelan ini dilakukan selama 2 minggu.

---

**Kode 4.4:** *Pseudocode* untuk melakukan pemotongan kalimat
 

---

```

1 Function sentenceSplitting(post, limit) is
  Input : post, minimal limit number of word in a sentence
  Output: array of sentence
2   arrSentence = post.splitByRegex([?!,]);
3   temp = [];
4   arrResult = [];
5   foreach sentence in arrSentence do
6     if len(temp) > limit then
7       arrResult.append(temp);
8       temp = [];
9     else
10      temp += sentence
11  return arrResult;

```

---

## 4.4 Pengembangan Model

### 4.4.1 Ekstraksi Fitur

Ekstraksi fitur dilakukan dengan menggunakan program yang diimplementasikan dalam bahasa Python. Keluaran dari ekstraksi fitur ini adalah vektor kata untuk masing-masing kalimat yang disimpan dalam format JSON. Masing-masing kalimat dalam sebuah *post* disimpan dalam sebuah *array* yang kemudian keseluruhan *post* disimpan dalam *hash* dengan indeks yang telah didefinisikan pada saat tahap pengumpulan data.

#### 4.4.1.1 Fitur Kata Itu Sendiri

Dalam melakukan ekstraksi fitur kata itu sendiri, penulis menggunakan *library* *gensim* (Řehůřek dan Sojka, 2010) yang disediakan secara gratis. *Gensim* mengimplementasikan *word embedding* melalui *library* bernama *word2vec*. Sebelum melakukan ekstraksi fitur, penulis melakukan *training* model *word embedding* dengan data yang penulis unduh dari berbagai artikel kesehatan di beberapa situs. Setelah model didapatkan, penulis melakukan ekstraksi dari masing-masing kata pada korpus. Kode 4.5 merupakan *pseudocode* untuk melakukan ekstraksi fitur kata itu sendiri.

---

**Kode 4.5:** *Pseudocode* untuk melakukan ekstraksi fitur kata itu sendiri
 

---

```

1 Function wordToVector(model, arrWord) is
    Input : model word embedding, array of word in a sentence
    Output: array of word vector
2   arrVector = [];
3   foreach word in arrWord do
4     arrVector.append(model.getVector(word))
5   return arrVector;
  
```

---

#### 4.4.1.2 Ekstraksi Fitur Part of Speech Tag

Dalam mengimplementasikan ekstraksi fitur POS Tag, penulis menggunakan *tools* Stanford POS Tagger dan model POS *tagger* yang dikembangkan oleh Dinakaramani et al. (2014). Pertama-tama penulis melakukan pemberian tag pada setiap kalimat di dalam korpus, kemudian mengubah hasil tag tersebut menjadi bentuk *one-hot-vector*. Kode 4.6 merupakan *pseudocode* dalam melakukan ekstraksi fitur POS Tag untuk sebuah kalimat yang penulis lakukan.

---

**Kode 4.6:** *Pseudocode* untuk melakukan ekstraksi fitur kata itu sendiri
 

---

```

1 Function posTagExtract(model, sentence) is
    Input : tagger of POS Tag, sentence
    Output: array of one hot vector
2   sentenceTagged = model.tagSentence(sentence);
3   tagOnly = getTagOnly(sentenceTagged);
4   posTagFeature = [];
5   foreach tag in tagOnly do
6     posTagFeature.append(tag.convertToOneHotVector())
7   return posTagFeature;
  
```

---

#### 4.4.1.3 Ekstraksi Fitur Stop Word

Ekstraksi fitur *stop word* penulis lakukan dengan menggunakan kamus *stop word* yang digunakan oleh Taufik (2015) dalam melakukan pengenalan entitas bernama. Setiap kata yang merupakan *stop word* memiliki nilai fitur [ 0.0, 1.0 ] dan kata yang bukan merupakan *stop word* memiliki nilai fitur [ 1.0, 0.0 ]. Penulis menggunakan bahasa pemrograman Python dalam mengimplementasikan ekstraksi fitur ini. *Pseudocode* untuk melakukan ekstraksi fitur ini dapat dilihat pada kode 4.7.

---

**Kode 4.7:** *Pseudocode* untuk melakukan ekstraksi fitur *stop word*


---

```

1 Function stopWordExtract(dictionary, sentence) is
  Input : dictionary of stop word,sentence
  Output: array of one hot vector
2   stopWordFeature = [];
3   foreach word in tagOnly do
4     posTagFeature.append(dictionary.isExist(word))
5   return stopWordFeature;

```

---

#### 4.4.1.4 Ekstraksi Fitur Kamus Kesehatan

Pada dasarnya implementasi ekstraksi fitur kamus kesehatan mirip dengan implementasi ekstraksi fitur *stop word*. Perbedaannya yaitu pada penggunaan *resource*, yang mana ekstraksi fitur *stop words* penulis lakukan dengan menggunakan kamus *stop word*, sedangkan pada fitur ini penulis menggunakan kamus kesehatan. Kamus kesehatan yang saya gunakan sama dengan kamus pada penelitian Herwando (2016), yang mana terdapat 4 kamus, yaitu kamus *disease*, *symptom*, *treatment* dan *drug*. Setiap kata yang terdaftar di dalam kamus kesehatan memiliki nilai fitur [ 0.0, 1.0 ] dan kata yang bukan merupakan *stop word* memiliki nilai fitur [ 1.0, 0.0 ]. Penulis menggunakan bahasa pemrograman Python dalam mengimplementasikan ekstraksi fitur ini. *Pseudocode* untuk melakukan ekstraksi fitur ini dapat dilihat pada kode 4.8.

#### 4.4.1.5 Ekstraksi Frasa Kata Benda

Dalam mengimplementasikan ekstraksi fitur kata benda, penulis menggunakan *library* NLTK yang mengimplementasikan *chunking*, yang merupakan proses segmentasi dan pelabelan pada *multi-token sequences*. Untuk mengimplementasikannya, penulis menggunakan informasi POS-Tag yang didapatkan pada implementasi fitur POS Tag, kemudian menentukan *rule* pada proses *chunking* ini. *Rule* yang penulis gunakan sudah dijelaskan pada Bab 3. Keluaran dari ekstraksi fitur ini yaitu *array of one hot vector* dari masing-masing kata dalam 1 kalimat, dimana apabila suatu kata merupakan bagian dari frasa kata benda akan bernilai [0.0, 1.0], sedangkan yang bukan akan bernilai [1.0, 0.0]. Berikut merupakan *pseudocode* dari implementasi ekstraksi fitur frasa kata benda. *Pseudocode* untuk melakukan ekstraksi fitur ini dapat dilihat pada kode 4.9.

---

**Kode 4.8:** *Pseudocode* untuk melakukan ekstraksi fitur kamus kesehatan
 

---

```

1 Function dictExtract(dictionary, sentence) is
    Input : dictionary of stop word,sentence
    Output: array of one hot vector
2     dictFeature = [];
3     foreach word in sentence do
4         dictFeature.append(dictionary.isExist(word))
5     end
6     return dictFeature;
7 end

8 Function dictExtractAll(sentence) is
    Input : dictionary of stop word,sentence
    Output: array of one hot vector
9     dictExtract(symptomDict, sentence);
10    dictExtract(diseaseDict, sentence);
11    dictExtract(treatmentDict, sentence);
12    dictExtract(drugDict, sentence);
13 end
  
```

---

#### 4.4.1.6 Ekstraksi Frasa Kata Kerja

Sama seperti pada pengimplementasian ekstraksi fitur kata benda, penulis menggunakan *library* NLTK yang mengimplementasikan *chunking*, yang merupakan proses segmentasi dan pelabelan pada *multi-token sequences*. Untuk mengimplementasikannya, penulis menggunakan informasi POS-Tag yang didapatkan pada implementasi fitur POS Tag, kemudian menentukan *rule* pada proses *chunking* ini. *Rule* yang penulis gunakan sudah dijelaskan pada Bab 3. Keluaran dari ekstraksi fitur ini yaitu *array of one hot vector* dari masing-masing kata dalam 1 kalimat, dimana apabila suatu kata merupakan bagian dari frasa kata kerja akan bernilai [0.0, 1.0], sedangkan yang bukan akan bernilai [1.0, 0.0]. Berikut merupakan *pseudocode* dari implementasi ekstraksi fitur frasa kata benda. *Pseudocode* untuk melakukan ekstraksi fitur ini dapat dilihat pada kode 4.10.

#### 4.4.1.7 Ekstraksi Fitur 1 Kata Sebelum

Ekstraksi fitur ini penulis lakukan dengan menggunakan hasil dari ekstraksi fitur kata itu sendiri, yaitu mengambil vektor kata dengan indeks saat ini dikurangi satu. Untuk awal kalimat, penulis memberikan vektor  $\vec{0}$  dimana setiap elemen di dalam *array* merupakan bilangan nol. 4.11 merupakan implementasi dari ekstraksi fitur 1 kata sebelum.

---

**Kode 4.9:** *Pseudocode* untuk melakukan ekstraksi fitur kamus kesehatan
 

---

```

1 Function npExtract(chunker, sentence, label) is
    Input : chunker for a sentence, sentence, label of chunking
    Output: array of one hot vector

2     chunkedSentence = chunker.chunk(sentence);
3     chunkFeature = [];
4     foreach token in chunkedSentence do
5         if token.isLabel(label) then
6             | chunkFeature.append([0.0, 1.0]);
7         end
8         else
9             | chunkFeature.append([1.0, 0.0]);
10        end
11    end
12    return chunkFeature;
13 end

14 Function main() is
15     corpus = readFile("corpus");
16     rule = ruleOfNounPhrase;
17     chunker = nltk.RegexpParser(rule);
18     corpusChunked = [];
19     foreach sentence in corpus do
20         | corpusChunked.append(npExtract(chunker, sentece));
21     end
22     writeFile(corpusChunked)
23 end
  
```

---

#### 4.4.1.8 Ekstraksi Fitur 1 Kata Sesudah

Ekstraksi fitur 1 kata sesudah yang penulis lakukan ini mirip dengan ekstraksi fitur 1 kata sebelum, perbedaannya pada indeks yang diambil dalam pada saat ekstraksi. Untuk masing-masing kata, penulis mengambil vektor kata dengan indeks 1 kata setelahnya. Untuk vektor kata di akhir kalimat, penulis memberikan vektor  $\vec{0}$  dimana setiap elemen di dalam *array* merupakan bilangan nol. ?? merupakan implementasi dari ekstraksi fitur 1 kata sesudah.

```

\SetKwInOut { Input } { Input }
\SetKwInOut { Output } { Output }

\SetAlgoLined
\SetKwProg{Fn}{Function}{ is}{end}
  
```

---

**Kode 4.10:** *Pseudocode* untuk melakukan ekstraksi fitur kamus kesehatan
 

---

```

1 Function vpExtract(chunker, sentence, label) is
    Input : chunker for a sentence, sentence, label of chunking
    Output: array of one hot vector

2     chunkedSentence = chunker.chunk(sentence);
3     chunkFeature = [];
4     foreach token in chunkedSentence do
5         if token.isLabel(label) then
6             | chunkFeature.append([0.0, 1.0]);
7         end
8         else
9             | chunkFeature.append([1.0, 0.0]);
10        end
11    end
12    return chunkFeature;
13 end

14 Function main() is
15     corpus = readFile("corpus");
16     rule = ruleOfVerbPhrase;
17     chunker = nltk.RegexpParser(rule);
18     corpusChunked = [];
19     foreach sentence in corpus do
20         | corpusChunked.append(npExtract(chunker, sentece));
21     end
22     writeToFile(corpusChunked)
23 end
  
```

---

```

\Fn{oneWordBeforeExtract(sentenceVector)}{
    \Input{array of vector in a sentence}
    \Output{array of vector}
    \BlankLine

    oneWordAfterFeature = []\;
    \For{i in 0...sentenceVector.length - 1}{
        oneWordAfterFeature.append(sentenceVector[i+1])\;
    }
    oneWordAfterFeature.append(zeroth)\;
    \BlankLine

    \Return oneWordAfterFeature;

}
  
```

---

**Kode 4.11:** *Pseudocode* untuk melakukan ekstraksi fitur 1 kata sebelum
 

---

```

1 Function oneWordBeforeExtract(sentenceVector) is
    Input : array of vector in a sentence
    Output: array of vector
2   oneWordBeforeFeature = [];
3   oneWordBeforeFeature.append(zeroth);
4   for i in 1...sentenceVector.length do
5     | oneWordBeforeFeature.append(sentenceVector[i-1]);
6   end
7   return oneWordBeforeFeature;
8 end

```

---

```

\BlankLine
\caption{\textit{Pseudocode} untuk melakukan ekstraksi fitur 1
        kata sesudah}
\label{code:ekstraksi1wordaft}

```

#### 4.4.2 Pengusulan Arsitektur RNNs

Sesuai dengan yang telah dijelaskan pada Bab 3, penulis mengusulkan dua arsitektur RNNs yang akan digunakan pada tahap eksperimen. Pada bagian ini penulis akan menjelaskan implementasi dari masing-masing arsitektur tersebut. Dalam melakukan implementasi RNNs, penulis menggunakan *library* Keras Chollet (2015) dalam bahasa Python. Keras sendiri dapat berjalan di atas dua *library deep learning* lain, yaitu Theano dan Tensorflow, namun dalam penelitian ini penulis menggunakan Theano. Penulis menggunakan *Sequential model* yang merupakan layer *linear stack* dalam mengembangkan model dan jenis RNNs yang penulis gunakan dalam penelitian ini adalah LSTM.

##### 4.4.2.1 LSTM 1 layer

LSTM 1 layer yang dimaksud adalah model yang digunakan memiliki satu layer LSTM saja dan semua fitur yang menjadi input program digabung terlebih dahulu menjadi satu buah *array*. Seperti yang telah dijelaskan pada Bab 3, susunan layer yang penulis gunakan terdiri dari *Masking Layer*, LSTM Layer, dan *Time Distributed Layer* yang masing-masing *timestep* berisi *Dense Layer*. Untuk *Masking Layer*, dimensi yang menjadi parameter tergantung dari *array* yang menjadi masukan, untuk LSTM Layer, dimensi masukan sama dengan dimensi



*Masking Layer* dan dimensi keluaran untuk masing-masing *timesteps* adalah panjang input dalam satu *timestep* dibagi 2. Untuk masing-masing Dense Layer, dimensi masukan yang diminta sama dengan dimensi keluaran pada LSTM Layer dan dimensi keluaran sesuai dengan jumlah kelas yang telah didefinisikan.

Masukan yang diminta yaitu *array* yang masing-masing elemennya merupakan *array* dari vektor fitur dan sudah digabung menjadi satu. Keluaran yang diminta merupakan hasil dari pelabelan otomatis dari program ini. Kode 4.12 merupakan kode untuk mengimplementasikan model ini.

---

**Kode 4.12:** *Pseudocode* untuk arsitektur RNNs pertama

---

```

1 Function lstm1(arrTraining, arrTesting) is
   Input : training data, testing data
   Output: predicted label
2   shape = arrTraning.shape();
3   model = Sequential();
4   model.add(Masking(input_shape:shape));
5   model.add(LSTM(output = shape/2));
6   model.add(TimeDistributed(Dense(output = 9)));
7   model.input(arrTraining);
8   prediction = model.predict(arrTesting);
9   return prediction;
```

---

#### 4.4.2.2 LSTM Layer Bertingkat

LSTM layer bertingkat yang dimaksud yaitu terdapat dua tingkat, tingkat pertama untuk menerima *input* yang setiap kelompok fitur menjadi input bagi LSTM masing-masing. Misalnya terdapat 3 kelompok fitur, masing-masing kelompok tadi akan menjadi input bagi layer LSTM masing-masing. Tingkat kedua sebagai penggabung hasil dari tingkat pertama.

Layer pada tingkat pertama terdiri dari *Masking Layer* dan sebuah Layer LSTM. Untuk dimensi *input* dan *output Masking Layer* secara otomatis mengikuti dimensi dari data masukan. Dimensi *output* dari Layer LSTM yaitu dimensi awal dibagi 2. Pada layer tingkat kedua, layer tersebut terdiri dari *Merge Layer*, *Time Distributed* dengan masing-masing *timestep* merupakan *Dense Layer* dan sebuah Layer LSTM. Keluaran dari *Merge Layer* sesuai dengan total dimensi *output* dari masing-masing LSTM di tingkat 1. Dimensi keluaran dari masing-masing *Dense Layer* yaitu 32 dan sesuai jumlah kelas. Masukan yang diminta yaitu *array* yang masing-masing elemennya merupakan *array* dari vektor fitur dan sudah digabung menjadi satu.

Keluaran yang diminta merupakan hasil dari pelabelan otomatis dari program ini. Kode 4.13 merupakan kode untuk mengimplementasikan model ini.

---

**Kode 4.13:** *Pseudocode* untuk arsitektur RNNs kedua

---

```

1 Function lstm2(groupOfArrTraining, groupOfArrTraining) is
    Input : grop of training data, group of testing data
    Output: predicted label

2     modelArr = [];
3     foreach groupFeature in groupOfArrTraining do
4         shape = arrTraning.shape();
5         model = Sequential();
6         model.add(Masking(input_shape:shape));
7         model.add(LSTM(output = shape/2));
8         modelArr.append(model);

9     mainModel = Sequential();
10    mainModel.add(Merge(mode='concat', modelArr));
11    mainModel.add(TimeDistributed(Dense(output = 32)));
12    mainModel.add(LSTM(output = 32));
13    mainModel.add(TimeDistributed(Dense(output = 9)));

14    mainModel.input(groupOfArrTraining);
15    prediction = mainModel.predict(groupOfArrTraining);
16    return prediction;

```

---

## 4.5 Eksperimen

Pada tahap ini penulis melakukan eksperimen model yang dikembangkan pada tahap sebelumnya. Sebelum masuk ke tahap eksperimen, penulis melakukan beberapa tahap pra-eksperimen seperti melakukan pemecahan data sebagai implementasi *cross-fold validation*. Penulis memecah data menjadi 10 bagian dan disimpan dalam sebuah *array* untuk masing-masing fitur. Berikut merupakan *pseudocode* untuk melakukan pemecahan data

Setelah masing-masing fitur dipecah menjadi 10 bagian, penulis melakukan penggabungan antar fitur sebagai *input* untuk melakukan eksperimen. Seperti yang dijelaskan pada tahap sebelumnya, penulis menggunakan dua arsitektur RNNs. Hasil dari eksperimen tersebut ditulis dalam sebuah file dengan format JSON yang nantinya akan menjadi *input* pada tahap evaluasi. Berikut merupakan implementasi eksperimen dengan masing-masing arsitektur tersebut.

---

**Kode 4.14:** *Pseudocode* untuk memecah *data* menjadi 10 bagian
 

---

```

1 Function splitting(featureArr) is
   Input : array of feature
   Output: splitted array of feature
2   lenSplit = len(featureArr)/10;
3   arrSplitted = [];
4   for i=0; i<10;i++ do
5     start = i * lenSplit;
6     end = (i+1) * lenSplit;
7     arrSplitted.append[start:end];
8   return arrSplitted;
```

---



---

**Kode 4.15:** *Pseudocode* untuk melakukan eksperimen
 

---

```

1 arrAllFeature = [];
2 foreach feature in arrSplitted do
3   arrAllFeature.join(feature);
4 for i=0; i<10;i++ do
5   training = arrSplitted[0:i] + arrSplitted[i+1:10] testing = arrSplitted[i];
6   result1 = lstm1(training, testing);
7   result2 = lstm2(training, testing);
8   writeToJSON(result1);
9   writeToJSON(result2);
```

---

## 4.6 Evaluasi

Dalam melakukan implementasi pada tahap evaluasi, penulis menghitung nilai *precision*, *recall* dan *F-Measures* untuk mengukur tingkat keakuratan model yang dikembangkan pada tahap sebelumnya. Penulis menggunakan aturan yang telah dijelaskan pada Bab 3. Berikut merupakan implementasi kode untuk melakukan evaluasi.

---

**Kode 4.16:** *Pseudocode* untuk melakukan evaluasi
 

---

```

1 resultTag = load(resultRNN);
2 originalTag = load(originalTag);

3 TP = newHash();
4 FP = newHash();
5 FN = newHash();
6 for i = 0; i < len(resultTag); i++ do
7   sentenceResult = resultTag[i];
8   sentenceOriginal = originalTag[i];
9   for j = 0; j < len(sentenceOriginal); j++ do
10    wordResult = sentenceResult[j];
11    wordOri = sentenceOriginal[j];
12    if wordOri != O then
13      if wordResult != O then
14        if wordOri == wordResult then
15          TP[wordOri] += 1;
16        else
17          FN[wordOri] += 1;
18      else
19        FN[wordOri] += 1;
20    else
21      if wordResult != O then
22        FP[wordOri] += 1;

23 prec = newHash();
24 rec = newHash();
25 fMeas = newHash();
26 foreach label in TP do
27   prec[label] = TP[label] / (TP[label] + FP[label]);
28   rec[label] = TP[label] / (TP[label] + FN[label]);
29   fMeas[label] = 2 * (prec[label] * rec[label]) / (prec[label] + rec[label]);

30 foreach label in prec do
31   print "Precision", label, prec[label];
32   print "Recall", label, rec[label];
33   print "F-Measure", label, fmeas[label];

```

---

## BAB 5

### EKSPERIMEN

Pada bab ini penulis akan menjelaskan mengenai skenario, hasil dan analisis dari eksperimen yang telah dilakukan.

#### 5.1 Matriks Evaluasi

Pada eksperimen ini, untuk mendapatkan nilai akurasi dari masing-masing eksperimen penulis menggunakan *precision*, *recall* dan *f-measure*. Penulis menggunakan *10-cross fold validation* dalam menjalankan eksperimen. Terkait dengan penjelasan mengenai cara penghitungan dan evaluasi sudah dijelaskan pada Bab 3.

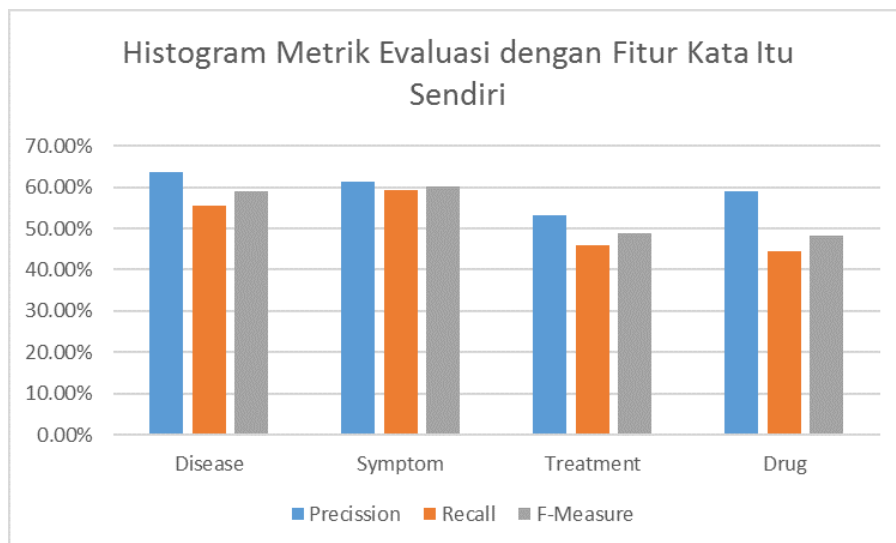
#### 5.2 Baseline Eksperimen

Pada penelitian ini, penulis mencoba melakukan implementasi ulang penelitian yang dilakukan oleh Herwando (2016). Data yang digunakan adalah data yang penulis gunakan dalam penelitian ini. Implementasi dan eksperimen ini bertujuan sebagai *baseline* eksperimen dan penelitian yang penulis lakukan. Selain itu, juga untuk mengetahui secara singkat fitur yang diskriminatif dalam melakukan *sequence labelling* pada MER ini.

Berikut merupakan hasil implementasi ulang penelitian yang dilakukan oleh Herwando (2016).

**Tabel 5.1:** Tabel Hasil Eksperimen dari Penelitian Herwando (2016)

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	63.68%	55.45%	59.13%
<i>Symptom</i>	61.43%	59.21%	60.18%
<i>Treatment</i>	53.10%	45.97%	48.82%
<i>Drug</i>	58.99%	44.46%	48.23%



**Gambar 5.1:** Histogram Metrik Evaluasi dengan Fitur Kata Itu Sendiri

### 5.3 Skenario Eksperimen

Pada penelitian ini, penulis melakukan 2 buah skenario utama, yaitu skenario untuk menguji fitur yang memiliki kontribusi untuk meningkatkan akurasi dari setiap eksperimen dan skenario untuk menguji arsitektur RNNs yang penulis usulkan. Berikut merupakan skenario yang penulis rancang dalam penelitian ini:

#### 1. Skenario untuk menguji fitur

Skenario ini bertujuan untuk mendapatkan kombinasi fitur terbaik sehingga memberikan akurasi terbaik. Penulis mencoba masing-masing fitur dengan menggunakan model arsitektur LSTM biasa. Apabila penggunaan fitur memberikan hasil yang lebih dari pada hasil eksperimen sebelumnya, fitur tersebut akan dipertahankan untuk eksperimen yang selanjutnya. Skenario ini memiliki 9 sub-skenario, yaitu:

- (a) Sub-skenario menguji fitur kata itu sendiri
- (b) Sub-skenario menguji fitur terbaik sebelumnya ditambahkan fitur kamus
- (c) Sub-skenario menguji fitur terbaik sebelumnya ditambahkan fitur stop word
- (d) Sub-skenario menguji fitur terbaik sebelumnya ditambahkan fitur POS-Tag
- (e) Sub-skenario menguji fitur terbaik sebelumnya ditambahkan fitur Frasa kata

- (f) Sub-skenario menguji fitur terbaik sebelumnya dikurangi fitur POS-Tag
- (g) Sub-skenario menguji fitur terbaik sebelumnya ditambahkan fitur 1 kata sebelum
- (h) Sub-skenario menguji fitur terbaik sebelumnya ditambahkan fitur 1 kata sesudah

## 2. Skenario untuk menguji arsitektur RNNs

Skenario ini bertujuan untuk melihat pengaruh arsitektur RNNs pada penelitian ini. Penulis mencoba kedua arsitektur RNNs yang telah diusulkan sebelumnya dengan menggunakan kombinasi fitur terbaik dari eksperimen di skenario pengujian fitur di atas. Pada skenario ini, terdapat 2 sub-skenario, yaitu:

- (a) Sub-skenario untuk menguji arsitektur 1 layer
- (b) Sub-skenario untuk menguji arsitektur 2 layer

## 5.4 Hasil Eksperimen dan Analisis

Pada bagian ini akan dilaporkan hasil dari eksperimen yang telah penulis rancang sesuai dengan skenario sebelumnya beserta analisisnya.

### 5.4.1 Hasil Eksperimen Pengujian Fitur Beserta Analisis

Hasil eksperimen ini adalah laporan dari pengujian kombinasi fitur kata itu sendiri, kamus, *stop word*, POS-Tag, frasa kata (frasa kata benda dan kata kerja), 1 kata sebelum, dan 1 kata sesudah.

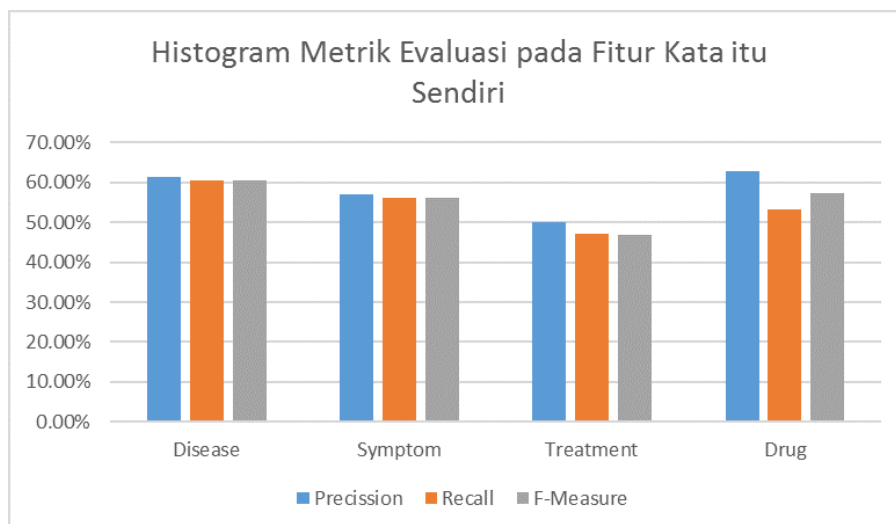
#### 5.4.1.1 Sub-Eksperimen Menguji Fitur Kata itu Sendiri

Merujuk pada penelitian Mujiono et al. (2016), penelitian tersebut bertujuan untuk mendapatkan *non-handcrafted feature*, yaitu fitur kata itu sendiri dengan menggunakan *tools Word Embedding*. Oleh karena itu, penulis menguji fitur ini untuk mengetahui pengaruhnya pada program MER di penelitian ini. Tabel 5.2 menampilkan hasil pelabelan otomatis dengan menggunakan fitur kata itu sendiri yang direpresentasikan dengan menggunakan vektor *word embedding*.

**Tabel 5.2:** Tabel Hasil Eksperimen dengan Fitur Kata Itu Sendiri

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	61.38%	60.42%	60.37%
<i>Symptom</i>	57.05%	56.13%	56.19%
<i>Treatment</i>	49.92%	47.17%	46.96%
<i>Drug</i>	62.86%	53.32%	57.28%

Pada tabel tersebut, dengan menggunakan fitur kata itu sendiri terlihat bahwa entitas *Disease* memiliki *recall* dan *f-measure* tertinggi, yaitu 60.42%, dan 60.37%. Sedangkan entitas *drug* memiliki *precision* tertinggi, yaitu 62.86%. Grafik 5.2 menunjukkan perbandingan *precision*, *recall* dan *f-measure* untuk masing-masing entitas.

**Gambar 5.2:** Histogram Metrik Evaluasi dengan Fitur Kata Itu Sendiri

Pada eksperimen ini, ada beberapa nilai *precision*, *recall* dan *f-measure* yang dicapai masih lebih kecil apabila dibandingkan dengan hasil yang dicapai Herwando (2016). Menurut penulis hal ini terjadi karena pada eksperimen ini hanya menggunakan fitur kata itu sendiri tanpa melibatkan informasi lain seperti pada penelitian yang dilakukan oleh Herwando (2016). Oleh karena itu perlu adanya informasi lain, misalnya seperti apakah suatu kata terdapat dalam sebuah kamus kesehatan, informasi POS-Tag atau informasi yang lain. Oleh karena itu, penulis mencoba menggunakan tambahan fitur lain untuk meningkatkan akurasi pada penelitian ini, yaitu pada sub-eksperimen 5.4.1.2.



#### 5.4.1.2 Sub-Eksperimen Menguji Fitur Terbaik Sebelumnya Ditambahkan Fitur Kamus Kesehatan (*Disease, Symptom, Treatment dan Drug*)

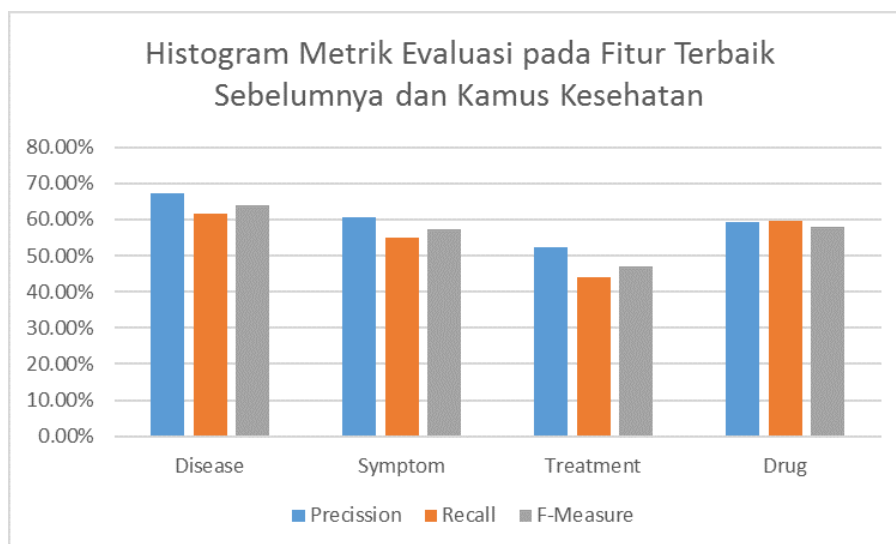
Pada sub-eksperimen ini, penulis menggunakan tambahan fitur Kamus Kesehatan karena berdasarkan penelitian Herwando (2016) fitur ini memiliki kontribusi untuk menambah akurasi pada sistem MER. Selain itu, menurut penulis, informasi suatu kata terdapat dalam sebuah kamus kesehatan mungkin akan memberikan kontribusi untuk meningkatkan akurasi. Oleh karena itu, penulis mencoba untuk menambahkan fitur ini ke dalam model RNNs.

Tabel 5.3 merupakan tabel hasil eksperimen yang didapatkan dengan menggunakan fitur ini.

**Tabel 5.3:** Tabel Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Kamus Kesehatan

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	67.32%	61.78%	64.10%
<i>Symptom</i>	60.55%	55.12%	57.41%
<i>Treatment</i>	52.21%	44.18%	47.02%
<i>Drug</i>	59.42%	59.71%	57.90%

Berikut merupakan grafik yang merepresentasikan Tabel 5.3 dalam bentuk histogram.



**Gambar 5.3:** Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Kamus Kesehatan

Dari tabel dan grafik tersebut, didapatkan informasi bahwa dengan menggunakan tambahan fitur kamus kesehatan terlihat bahwa entitas *Disease*

mengalami kenaikan nilai *precision*, *recall*, dan *f-measure*. Selain itu, entitas *symptom* dan *treatment* mengalami kenaikan nilai *precision* dan *f-measure*. Entitas *drug* mengalami penurunan pada nilai *precision* namun mengalami kenaikan pada nilai *recall* dan *f-measure*-nya. Secara keseluruhan, Sedangkan entitas *drug* memiliki *precision* tertinggi, yaitu 62.86%. Grafik 5.3 menunjukkan perbandingan *precision*, *recall* dan *f-measure* untuk masing-masing entitas.

Dibandingkan dengan hasil eksperimen Herwando (2016), hasil yang dicapai pada eksperimen ini masih lebih rendah pada entitas *symptom* dan *treatment*. Menurut penulis perlu ada informasi tambahan untuk meningkatkan akurasi. Seperti yang kita ketahui bahwa eksperimen Herwando (2016) tidak hanya menggunakan fitur kata itu sendiri dan kamus kesehatan saja. Oleh karena itu, penulis mencoba melakukan eksperimen kembali dengan menggunakan tambahan fitur lain pada sub-eksperimen 5.4.1.3;

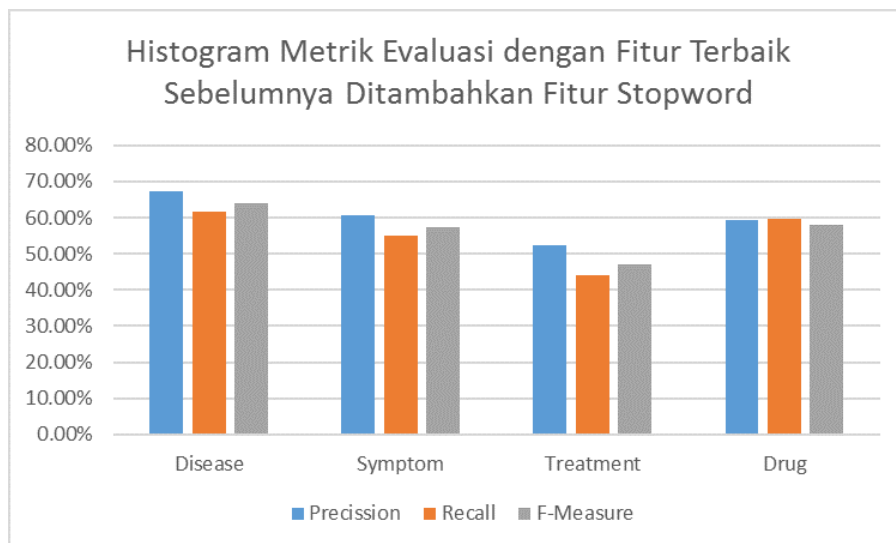
#### 5.4.1.3 Sub-Eksperimen Menguji Fitur Terbaik Sebelumnya Ditambahkan Fitur Stopword

Pada sub-eksperimen ini. penulis mencoba menambahkan informasi lain berupa fitur yang berisi sebuah kata apakah terdapat di dalam kamus *stop word* atau tidak. Penulis berpendapat bahwa dengan adanya informasi *stop word*, adanya kesalahan suatu kata tidak berentitas yang dilabeli sebagai kata berentitas oleh model dapat dikurangi.

Rangkuman hasil sub-eksperimen ini dapat dilihat di Tabel 5.4 dan Gambar 5.4.

**Tabel 5.4:** Tabel Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Stopword

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	65.97%	59.81%	62.28%
<i>Symptom</i>	63.08%	55.20%	58.68%
<i>Treatment</i>	54.73%	46.27%	49.69%
<i>Drug</i>	61.88%	58.99%	59.57%



**Gambar 5.4:** Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Stopword

Dari Tabel 5.4 dan Gambar 5.4 dapat diamati bahwa secara umum, penggunaan fitur Kamus *Stop Word* dapat meningkatkan *precision*, *recall*, dan *f-measure*. Untuk lebih detailnya, entitas *disease* mengalami penurunan nilai *precision* dan *f-measure* tetapi mengalami kenaikan nilai *recall*. Entitas *symptom* dan *treatment* mengalami kenaikan untuk nilai *precision*, *recall* dan *f-measure*. Sedangkan entitas *drug* mengalami kenaikan pada nilai *precision* dan *f-measure* tetapi mengalami penurunan pada nilai *recall*.

Pada sub-eksperimen ini, walaupun secara umum akurasi lebih baik dibandingkan dengan sub-eksperimen sebelumnya, hasil sub-eksperimen ini masih lebih rendah pada entitas *treatment* apabila dibandingkan dengan hasil eksperimen Herwando (2016). Oleh karena penulis mengusulkan fitur tambahan lain yaitu fitur POS-Tag yang akan dijelaskan pada sub-eksperimen 5.4.1.4.

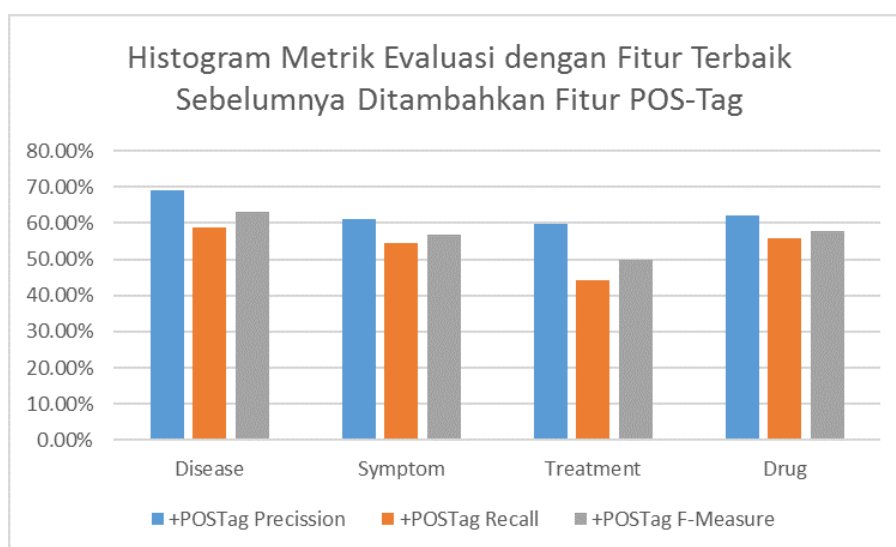
#### 5.4.1.4 Sub-Eksperimen Menguji Fitur Terbaik Sebelumnya Ditambahkan Fitur POS-Tag

Pada sub-eksperimen ini, penulis menambahkan informasi baru pada *resource* yang akan digunakan untuk *training* model yang berupa fitur POS-Tag. Sebelumnya fitur ini telah digunakan pada penelitian Abacha dan Zweigenbaum (2011) pada dokumen berbahasa Inggris dan memberikan kontribusi meningkatkan akurasi dari model MER yang dibangun. Oleh karena itu pada eksperimen ini penulis mencoba menggunakan fitur tersebut dan ingin mengetahui apakah fitur POS-Tag memiliki kontribusi untuk meningkatkan akurasi pada MER dengan dokumen berbahasa Indonesia.

Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.5 dan Gambar 5.5.

**Tabel 5.5:** Tabel Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur POS-Tag

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	69.10%	58.67%	63.22%
<i>Symptom</i>	61.09%	54.43%	57.00%
<i>Treatment</i>	59.73%	44.10%	49.87%
<i>Drug</i>	62.00%	55.74%	57.87%



**Gambar 5.5:** Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur POS-Tag

Dari tabel dan grafik di atas, entitas *disease* dan *treatment* memiliki nilai *precision* dan *f-measure* yang meningkat, tetapi dengan nilai *recall* yang turun. Untuk entitas *symptom*, nilai *precision*, *recall*, dan *f-measure* mengalami penurunan. Sedangkan entitas *drug* mengalami kenaikan hanya pada *precision*-nya saja. Hal ini terjadi karena model POS-Tagger yang digunakan menghasilkan tag yang terkadang tidak konsisten. Selain itu, fitur POS-Tag juga tidak dapat membedakan entitas obat, nama penyakit dan nama orang. Sebagai contoh, nama orang mendapatkan tag "NN" (intan\_NN lusia\_NN), namun nama penyakit juga mendapatkan tag "NN" (Kanker\_NN Otak\_NN).

Oleh karena itu pada sub-eksperimen selanjutnya, penulis mencoba menambahkan fitur lain yang lebih spesifik dibandingkan dengan fitur POS-Tag, yaitu fitur Frasa Kata. Penjelasan lebih lanjut akan dibahas pada sub-eksperimen 5.4.1.5.

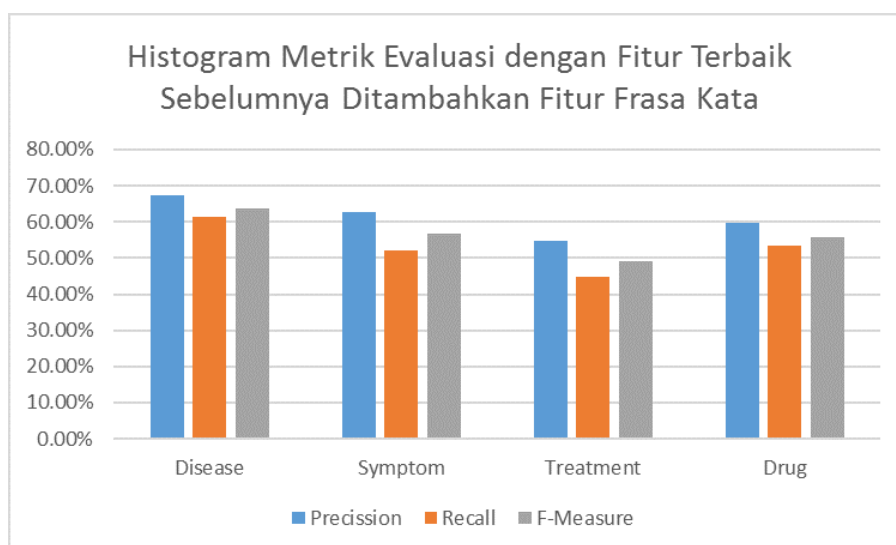
#### 5.4.1.5 Sub-Eksperimen Menguji Fitur Terbaik Sebelumnya Ditambahkan Fitur Frasa Kata

Pada sub-eksperimen ini penulis menambahkan fitur baru yaitu fitur Frasa Kata. Seperti yang telah dijelaskan pada Bab 3, entitas *symptom* dan *treatment* diharapkan akan lebih mudah dikenali karena pada umumnya merupakan frasa kata kerja. Sedangkan entitas *disease* dan *drug* diharapkan juga akan lebih mudah dikenali karena pada umumnya merupakan frasa kata benda.

Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.5 dan Gambar 5.5.

**Tabel 5.6:** Rangkuman Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Frasa Kata

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	67.49%	61.56%	63.81%
<i>Symptom</i>	62.89%	52.27%	56.72%
<i>Treatment</i>	54.87%	44.92%	49.06%
<i>Drug</i>	59.77%	53.37%	55.66%



**Gambar 5.6:** Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur Frasa Kata

Dari tabel dan grafik di atas, entitas *drug* mengalami penurunan untuk nilai *precision*, *recall* dan *f-measure*. Selain itu, entitas *disease* mengalami penurunan pada nilai *precision* tetapi mengalami kenaikan pada nilai *recall* dan *f-measure*. Entitas *symptom* mengalami kenaikan pada nilai *precision* tetapi mengalami penurunan pada nilai *recall* dan *f-measure*. Sedangkan pada entitas *treatment*,

terjadi kenaikan nilai *recall* tetapi nilai *precision* dan *f-measure* mengalami penurunan.

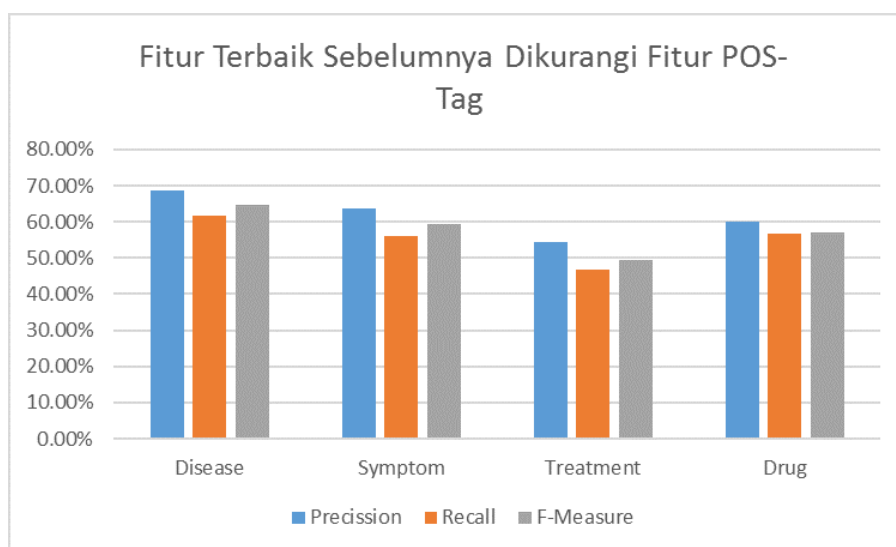
Dari hasil eksperimen ini, menurut penulis hal ini terjadi karena informasi frasa bersifat redundan apabila bergabung dengan fitur POS-Tag. Pada fitur POS-Tag, tidak ada perbedaan antara kata yang merupakan frasa maupun kata yang bukan frasa. Padahal, mayoritas entitas merupakan frasa. Oleh karena itu, pada sub-eksperimen 5.4.1.6, penulis menghilangkan fitur POS-Tag dan tetap mempertahankan fitur Frasa Kata untuk mengetahui hal tersebut.

#### 5.4.1.6 Sub-Eksperimen Menguji Fitur Terbaik Sebelumnya Dikurangi Fitur POS-Tag

Pada sub-eksperimen ini penulis menghilangkan fitur POS-Tag berdasarkan hasil dan analisis pada sub-eksperimen 5.4.1.5. Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.7 dan Gambar 5.7.

**Tabel 5.7:** Rangkuman Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Dikurangi Fitur POS-Tag

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	68.67%	61.80%	64.78%
<i>Symptom</i>	63.79%	56.10%	59.23%
<i>Treatment</i>	54.47%	46.72%	49.58%
<i>Drug</i>	60.08%	56.70%	57.00%



**Gambar 5.7:** Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Dikurangi Fitur POS-Tag

Dari Tabel 5.7 dan Gambar 5.7, terlihat bahwa semua entitas (*disease*, *symptom*, *treatment*,) dan *drug* mengalami kenaikan pada nilai *precision*, *recall*, dan *f-measure*. Seperti yang telah dijelaskan pada sub-eksperimen 5.4.1.5, penggabungan fitur POS-Tag dan Frasa akan memberikan hasil yang lebih rendah. Oleh karena itu, sebaiknya fitur POS-Tag tidak digabung dengan fitur Frasa. Selain itu, penulis lebih memilih fitur Frasa untuk dipertahankan karena fitur ini lebih diskriminatif dibandingkan dengan fitur POS-Tag, dengan melihat bahwa mayoritas kata berentitas merupakan frasa.

Walaupun pada sub-eksperimen ini hasil yang dicapai lebih baik dari sub-eksperimen sebelumnya, hasilnya tetap lebih rendah dari hasil eksperimen Herwando (2016) pada nilai *recall* dan *f-measure* entitas *symptom*. Oleh karena itu penulis mencoba fitur yang lain, yaitu fitur Kata Sebelum. Untuk penjelasan lebih lanjut akan dibahas pada sub-eksperimen 5.4.1.7.

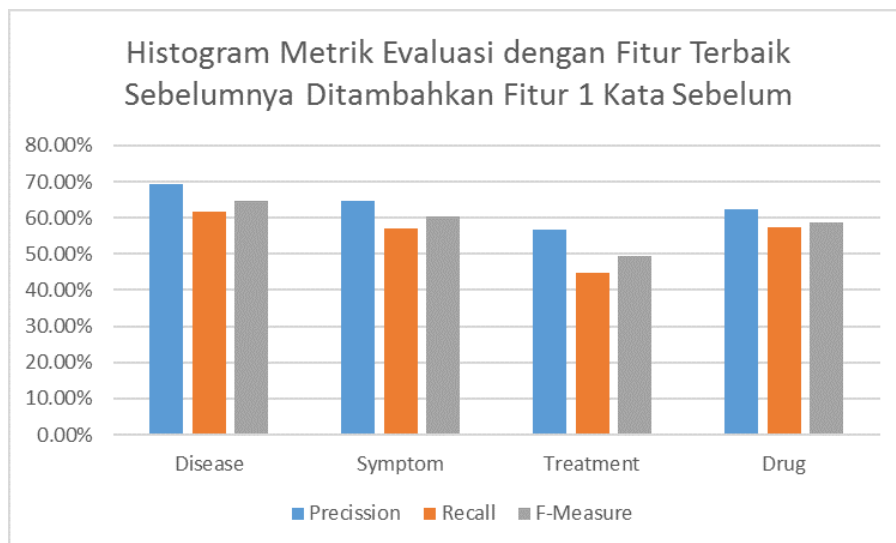
#### 5.4.1.7 Sub-Eksperimen Menguji Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sebelum

Pada sub-eksperimen ini penulis menambahkan fitur baru yaitu fitur 1 Kata Sebelum. Fitur ini digunakan pada penelitian Herwando (2016) yang juga berkontribusi memberikan hasil terbaik pada penelitiannya. Menurut penulis, ada beberapa entitas yang akan lebih mudah diketahui apabila diketahui kata sebelumnya. Misalnya kata "masuk angin", apabila hanya diberikan informasi kata "angin" tanpa kata "masuk", akan lebih sulit menentukan kata tersebut bagian dari suatu entitas *disease* atau bukan. Oleh karena itu, pada sub-eksperimen ini saya mencoba menambahkan fitur tersebut.

Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.8 dan Gambar 5.8.

**Tabel 5.8:** Rangkuman Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sebelum

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	69.49%	61.60%	64.68%
<i>Symptom</i>	64.78%	57.15%	60.23%
<i>Treatment</i>	56.58%	44.71%	49.54%
<i>Drug</i>	62.22%	57.28%	58.76%



**Gambar 5.8:** Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sebelum

Melihat pada Tabel 5.8 dan Gambar 5.8, dapat diketahui bahwa entitas *disease* dan *treatment* mengalami kenaikan pada nilai *precision*, tetapi mengalami penurunan pada nilai *recall* dan *f-measure*. Sedangkan entitas *symptom* dan *drug* mengalami kenaikan pada nilai *precision*, *recall*, dan *f-measure*.

Hasil sub-eksperimen ini masih lebih rendah dibandingkan dengan hasil eksperimen Herwando (2016) pada *recall* dan *f-measure* entitas *treatment*. Oleh karena itu, penulis mencoba menambahkan fitur yang lain yaitu fitur 1 Kata Sesudah, yang akan dibahas lebih lanjut pada sub-eksperimen 5.4.1.8.

#### 5.4.1.8 Sub-Eksperimen Menguji Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sesudah

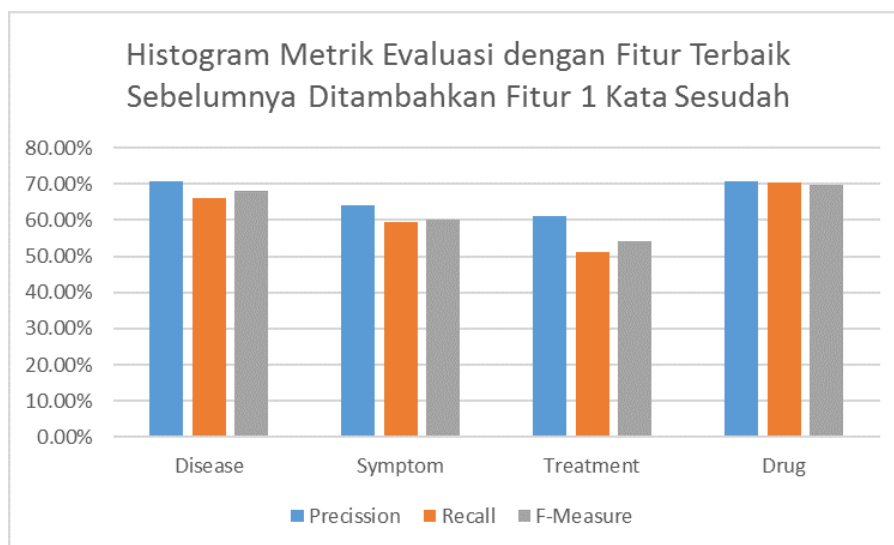
Pada sub-eksperimen ini penulis menambahkan fitur lain yaitu fitur 1 Kata Setelah. Hal ini karena ada beberapa kasus yang mana apabila suatu kata merupakan sebuah entitas, akan lebih mudah dikenali apabila melihat kata atau konteks setelahnya. Sama seperti contoh pada Fitur 1 Kata Sebelum, misal diberikan kata "masuk angin", apabila hanya diberikan informasi "masuk" tanpa "angin", akan lebih sulit mengenali apakah kata tersebut termasuk entitas *disease* atau bukan. Selain itu, fitur ini juga dapat membedakan kata berentitas dengan kata yang bukan, misalnya kata "masuk angin" dengan "masuk rumah". Apabila informasi pada saat tersebut hanya diberikan kata "masuk" saja tanpa kata setelahnya, akan lebih sulit mengenali kata tersebut termasuk kata berentitas atau bukan.

Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.9 dan Gambar 5.9.



**Tabel 5.9:** Rangkuman Hasil Eksperimen dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sesudah

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	70.68%	66.18%	68.17%
<i>Symptom</i>	64.16%	59.55%	61.42%
<i>Treatment</i>	61.02%	51.13%	54.03%
<i>Drug</i>	70.85%	70.33%	69.82%



**Gambar 5.9:** Histogram Metrik Evaluasi dengan Fitur Terbaik Sebelumnya Ditambahkan Fitur 1 Kata Sesudah

Melihat pada Tabel 5.9 dan Gambar 5.9, dapat diketahui bahwa hanya entitas *symptom* yang mengalami penurunan nilai pada *precision*, tetapi nilai *recall* dan *f-measure*-nya naik. Sedangkan entitas lain mengalami kenaikan pada nilai *precision*, *recall* dan *f-measure*. Oleh karena itu, setelah penulis mencoba kemungkinan fitur yang memberikan kontribusi dalam penelitian ini, penulis mencoba arsitektur untuk model RNNs yang lain. Penjelasan lebih lanjut akan dibahas pada eksperimen 5.4.2.

#### 5.4.2 Hasil Ekperimen Pengujian Arsitektur RNNs

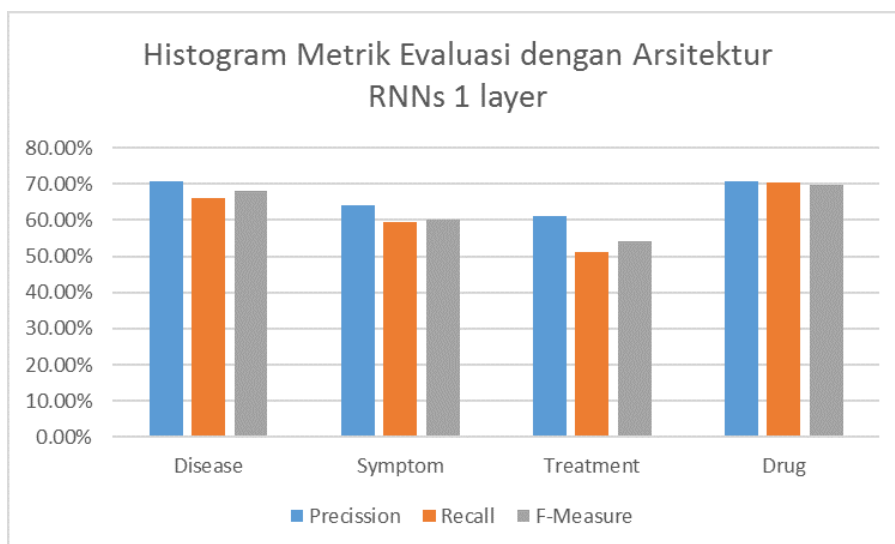
Pada eksperimen ini, penulis mencoba dua buah arsitektur RNNs yang telah penulis usulkan pada Bab3 yaitu RNNs dengan 1 layer dan RNNs dengan 2 layer. Fitur yang digunakan dalam pengujian ini yaitu kombinasi fitur yang menghasilkan akurasi terbaik pada eksperimen pertama, yaitu fitur Kata itu Sendiri, Kamus Kesehatan, *Stop Word*, Frasa Kata, 1 Kata Sebelum dan 1 Kata Sesudah.

#### 5.4.2.1 Sub-Eksperimen Menguji Arsitektur RNNs 1 Layer

Pada sub-eksperimen ini, penulis menggunakan struktur RNNs yang mana semua fitur digabung menjadi satu dalam sebuah *timestep*. Artinya fitur-fitur yang berbeda tersebut akan digabung atau di-*concat* menjadi sebuah vektor yang akan menjadi *input* bagi RNNs ini. RNNs inilah yang digunakan pada eksperimen pertama, sehingga hasilnya sama dengan sub-eksperimen 5.4.1.8.

**Tabel 5.10:** Rangkuman Hasil Eksperimen dengan Arsitektur RNNs 1 Layer

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	70.68%	66.18%	68.17%
<i>Symptom</i>	64.16%	59.55%	61.42%
<i>Treatment</i>	61.02%	51.13%	54.03%
<i>Drug</i>	70.85%	70.33%	69.82%



**Gambar 5.10:** Histogram Metrik Evaluasi dengan Arsitektur RNNs 1 Layer

Pada eksperimen ini, hasil yang sudah lebih baik apabila dibandingkan dengan hasil yang dicapai Herwando (2016) di entitas. Namun, dari eksperimen sebelumnya, ada akurasi yang turun, yaitu nilai *precision* untuk entitas *symptom*. Menurut penulis hal ini terjadi karena informasi fitur yang berbeda-beda dijadikan satu, sehingga ada kemungkinan hilangnya informasi dari masing-masing fitur tersebut. Oleh karena itu, untuk mengatasi permasalahan tersebut penulis mengusulkan arsitektur yang mana masing-masing kelompok fitur yang berbeda dipisahkan dan menjadi *input* bagi masing-masing LSTMs. Untuk penjelasan eksperimen ini akan dijelaskan pada sub-eksperimen 5.4.2.2

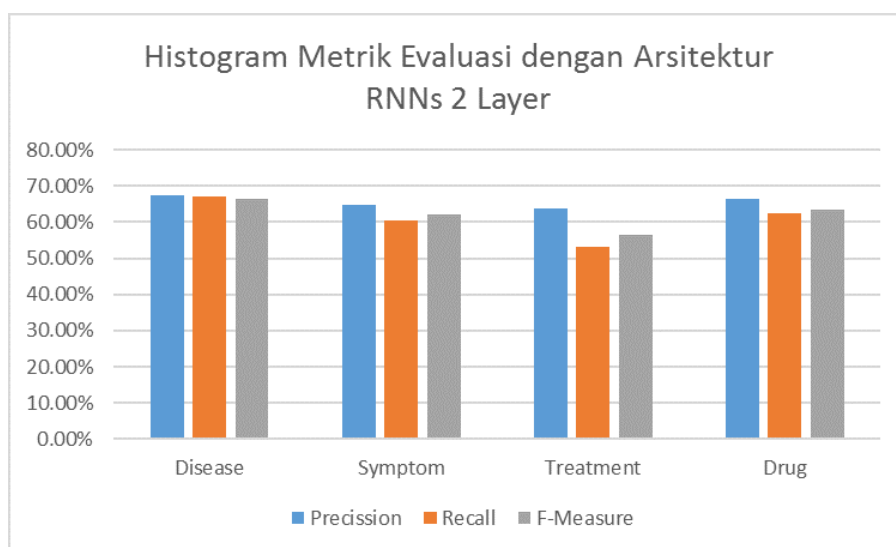
### 5.4.2.2 Sub-Eksperimen Menguji Arsitektur RNNs 2 Layer

Pada sub-eksperimen sebelumnya, fitur-fitur yang berbeda digabung menjadi satu, sehingga ada kemungkinan hilangnya informasi dari fitur tersebut. Oleh karena itu, penulis mengusulkan adanya layer tambahan setelah masing-masing fitur tersebut masuk ke dalam model. Penulis mengusulkan bahwa masing-masing kelompok fitur menjadi *input* RNNs secara terpisah. Setelah masuk di RNNs, *output* dari masing-masing RNNs tersebut di-*merge* ke dalam sebuah layer, lalu masuk kembali ke RNNs untuk melihat konteks fitur-fitur sebelumnya. Dengan diusulkannya arsitektur RNNs ini penulis berharap bahwa masing-masing fitur terjaga informasinya dan tidak terganggu dengan informasi lain.

Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.11 dan Gambar 5.11.

**Tabel 5.11:** Rangkuman Hasil Eksperimen dengan Arsitektur RNNs 2 Layer

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>Disease</i>	67.47%	67.19%	66.31%
<i>Symptom</i>	64.90%	60.63%	62.13%
<i>Treatment</i>	63.92%	53.13%	56.51%
<i>Drug</i>	66.39%	62.33%	63.61%



**Gambar 5.11:** Histogram Metrik Evaluasi dengan Arsitektur RNNs 2 Layer

Pada eksperimen ini, hasil yang sudah lebih baik apabila dibandingkan dengan hasil yang dicapai Herwando (2016) di masing-masing entitas dan lebih baik dibandingkan eksperimen 5.4.2.1 pada identifikasi entitas *symptom* dan *treatment*. Oleh

karena itu, sub-eksperimen ini merupakan eksperimen dengan hasil yang terbaik dari penelitian ini.

## BAB 6

### KESIMPULAN DAN SARAN

#### 6.1 Kesimpulan

Penelitian ini bertujuan untuk membangun sistem yang mampu melakukan ekstraksi entitas kesehatan dari forum *online* secara otomatis. Fitur yang digunakan pada penelitian ini yaitu fitur Kata itu Sendiri, Kamus Kesehatan, *Stop Word*, POS-Tag, Frasa kata, 1 kata sebelum dan 1 kata sesudah. Penelitian ini juga menggunakan dua arsitektur RNNs yang diusulkan, yaitu RNNs dengan 1 layer dan RNNs dengan 2 layer.

Setelah dilakukan penelitian, secara garis besar didapatkan bahwa fitur Kata itu Sendiri, Kamus Kesehatan, *Stop Word*, Frasa Kata, 1 Kata Sebelum dan 1 Kata Sesudah memberikan hasil yang terbaik, yaitu **angka yang manakah yang perlu ditampilkan? Secara overall atau diperici bagian terbaik untuk entitas *disease, symptom, treatment, drug, atau overall*.**

Selain itu, arsitektur yang penulis usulkan dalam penelitian ini, yaitu arsitektur RNNs dengan 2 layer memberikan hasil yang terbaik pada penelitian ini. Selain itu, hasil yang diberikan juga lebih baik dibandingkan dengan hasil penelitian Herwando (2016). **angka yang manakah yang perlu ditampilkan? Secara overall atau diperici bagian terbaik untuk entitas *disease, symptom, treatment, drug, atau overall*.**

#### 6.2 Saran

Setelah melakukan eksperimen dan menganalisis hasilnya, ada beberapa saran untuk penelitian selanjutnya, antara lain sebagai berikut.

1. Penelitian ini hanya menggunakan 305 *post* forum kesehatan *online* sehingga perlu penambahan data *training* dan *testing* mengingat *deep learning* membutuhkan data yang besar dalam melakukan *training*.
2. Perlu dibuat korpus dengan jumlah masing-masing entitas yang seimbang, sehingga hasil yang diberikan tidak bias.

## DAFTAR REFERENSI

- Abacha, A. B. dan Zweigenbaum, P. (2011). Medical entity recognition: A comparison of semantic and statistical methods. In *Proceedings of BioNLP 2011 Workshop*, pages 56–64. Association for Computational Linguistics.
- Almgren, S., Pavlov, S., dan Mogren, O. (2016). Named entity recognition in swedish health records with character-based deep bidirectional lstms. *BioTxtM 2016*, page 30.
- Bakker, B. (2001). Reinforcement learning with long short-term memory. In *NIPS*, pages 1475–1482.
- Bengio, Y., LeCun, Y., et al. (2007). Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5).
- Bengio, Y., Simard, P., dan Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.
- Chollet, F. (2015). Keras. <https://github.com/fchollet/keras>.
- conference on neural information processing systems, I., Mozer, M. C., Jordan, M. I., dan Petsche, T. (1997). *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference*. Mit Press.
- Dinakaramani, A., Rashed, F., Luthfi, A., dan Manurung, R. (2014). Designing an indonesian part of speech tagset and manually tagged indonesian corpus. In *IALP*, pages 66–69.
- Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.
- Graves, A. (2012). Neural networks. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 15–35. Springer.
- Graves, A., Mohamed, A.-r., dan Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

- Graves, A. dan Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552.
- Hachey, B., Radford, W., Nothman, J., Honnibal, M., dan Curran, J. R. (2013). Evaluating entity linking with wikipedia. *Artificial intelligence*, 194:130–150.
- Haykin, S. S., Haykin, S. S., Haykin, S. S., dan Haykin, S. S. (2009). *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, NJ, USA:.
- Herwando, R. (2016). Pengenalan entitas kesehatan pada forum kesehatan online berbahasa indonesia menggunakan algoritma conditional random fields. Master's thesis, Universitas Indonesia, Kampus UI Depok.
- Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, page 91.
- Hochreiter, S., Bengio, Y., Frasconi, P., dan Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.
- Hochreiter, S. dan Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Hs, W. (2005). *Bahasa Indonesia: mata kuliah pengembangan kepribadian di perguruan tinggi*. Gramedia Widiasarana Indonesia.
- Huang, Z., Xu, W., dan Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Jagannatha, A. N. dan Yu, H. (2016). Bidirectional rnn for medical event detection in electronic health records. In *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting*, volume 2016, page 473. NIH Public Access.
- Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine.
- Lang, K. J., Waibel, A. H., dan Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural networks*, 3(1):23–43.
- LeCun, Y., Bengio, Y., dan Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

- Limsopatham, N. dan Collier, N. (2016). Learning orthographic features in bi-directional lstm for biomedical named entity recognition. *BioTxtM 2016*, page 10.
- Mikolov, T., Chen, K., Corrado, G., dan Dean, J. (2014). word2vec.
- Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., dan Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.
- Mujiono, S., Fanany, M. I., dan Basaruddin, C. (2016). A new data representation based on training data characteristics to extract drug named-entity in medical text. *arXiv preprint arXiv:1610.01891*.
- Pennington, J., Socher, R., dan Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.
- Řehůřek, R. dan Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. <http://is.muni.cz/publication/884893/en>.
- Schmidhuber, J., Wierstra, D., Gagliolo, M., dan Gomez, F. (2007). Training recurrent networks by evoluno. *Neural computation*, 19(3):757–779.
- Seki, K. dan Mostafa, J. (2003). A probabilistic model for identifying protein names and their name boundaries. In *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 251–258. IEEE.
- Sutskever, I., Vinyals, O., dan Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., dan Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.
- Suwarningsih, W., Supriana, I., dan Purwarianti, A. (2014). Inner indonesian medical named entity recognition. In *Technology, Informatics, Management, Engineering, and Environment (TIME-E), 2014 2nd International Conference on*, pages 184–188. IEEE.