UNIVERSITAS INDONESIA

PENGENALAN ENTITAS KESEHATAN PADA FORUM KESEHATAN ONLINE DENGAN MENGGUNAKAN RECURRENT NEURAL NETWORKS

SKRIPSI

WAHID NUR ROHMAN
1306381856

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
JANUARI 2017

UNIVERSITAS INDONESIA

PENGENALAN ENTITAS KESEHATAN PADA FORUM KESEHATAN 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
JANUARI 2017

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	:	**-**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Wahid Nur Rohman
NPM	:	1306381856
Program Studi	:	Ilmu Komputer
Judul Skripsi	:	Pengenalan Entitas Kesehatan pada Forum Kesehatan 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	:	Alfan Farizki Wicaksono S.T., M.Sc.	( )
Penguji	:	-	( )
Penguji	:	-	( )

Ditetapkan di	:	Depok
Tanggal	:	-

KATA PENGANTAR

Segala puji dan syukur bagi Allah Subhanahu wa Ta'ala, Tuhan semesta alam. Semoga keselamatan dan kesejahteraan senantiasa terlimpahkan atas junjungan kita Nabi Muhammad Shallallahu Alaihi Wasallam, sebaik-baik teladan bagi umat manusia.

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, Sumanto dan Suparti yang selalu memberikan dukungan dan doa kepada penulis.
2. Dra. Mirna Adriani, Ph.D. dan Alfan 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. Alfan Nur Fauzan, S.Kom. yang telah memberikan *template* dokumen skripsi yang telah diperbaiki ini, sehingga penulis sangat terbantu dalam melakukan penulisan
7. Dipta Tanaya, 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. Segenap teman-teman angkatan 2013 (Angklung) yang memberi dukungan dan semangat kepada penulis untuk menyelesaikan skripsi ini.
10. Pihak-pihak lain yang tidak dapat penulis sebutkan satu-persatu yang sudah memberikan bantuan dan dukungannya kepada penulis.

Depok, Januari 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

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 Kesehatan 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 pernyatan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : -

Yang menyatakan

(Wahid Nur Rohman)

ABSTRAK

Nama	:	Wahid Nur Rohman
Program Studi	:	Ilmu Komputer
Judul	:	Pengenalan Entitas Kesehatan pada Forum Kesehatan 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 labeling*. Kami mengusulkan penggunaan teknik *Deep Learning* dengan menggunakan *Recurrent Neural Networks* (RNNs), karena RNNs merupakan *state-of-the-art* untuk permasalahan *sequence labeling* seperti permasalahan pada penelitian ini. Penulis mengusulkan fitur kata itu sendiri, kamus kesehatan, *stop word*, POS-Tag, frasa kata (nomina dan verba), kata sebelum dan kata sesudah. Selain itu penulis juga mengusulkan dua arsitektur RNNs, yaitu LSTMs 1 layer dan LSTMs 2 layer. Hasil eksperimen menunjukkan bawah model yang diusulkan mampu memberikan hasil yang cukup baik. Berdasarkan eksperimen dengan kombinasi fitur kata itu sendiri, kamus kesehatan, *stop word*, frasa kata (nomina dan verba), 1 kata sebelum dan 1 kata sesudah dengan arsitektur LSTMs 1 layer mampu mencapai rata-rata *f-measure* 63.06% dan LSTMs 2 layer mampu menghasilkan rata-rata *f-measure* 62.14%. Hasil tersebut lebih baik dibandingkan dengan *baseline* yang digunakan, yaitu penelitian Herwando (2016) dengan *f-measure* 54.09%.

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

Nowadays, anyone can use online health forum to look for disease without meet the doctor. Through the forum, someone just need to post his symptom and question on the form given. A lot of information which can be retrieved from the forum, such as symptom, disease, or treatment. On this research, we try to extract disease, symptom, treatment and drug automatically. We see that this problem as a sequence labeling problem. We propose Deep Learning technique using Recurrent Neural Networks (RNNs), because RNNs is state-of-the-art for sequence labeling problem. We propose some features such as word features, medical dictionary, stop word, POS-Tag, word phrase(verb and noun), word before and word after. Furthermore we propose two RNNs architectures, which are LSTMs in one layer and LSTMs in two layer with multi-input. The result of this experiment shows that the model proposed can give the good result. Based on the experiment using the combination features of its own word, medical dictionary, stop word, word phrase (noun and verb), 1 word before and 1 word after using the first RNNs architecture achieve f-measure 63.06%, and using second RNNs architecture achieve f-measure 62.14%. Thats result is better than the baseline used (Herwando, 2016), f-measure 54.09%.

Keywords:
MER, RNNs, disease, symptom, treatment, drug

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

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 dan lain sebagainya, 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.

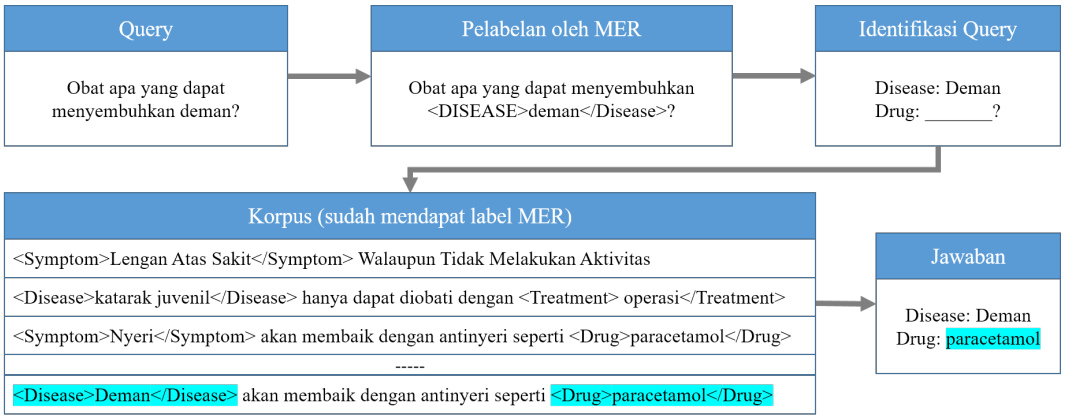
Penelitian dalam rangka mengembangkan sistem MER sudah banyak dilakukan oleh beberapa peneliti. Salah satu penelitian tersebut dilakukan oleh Abacha dan Zweigenbaum (2011) dengan menggunakan dokumen medis rumah sakit berbahasa Inggris. Entitas yang mendapatkan label pada penelitian tersebut adalah *treatment*, *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 dilakukan. 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 Fields* dengan hasil akhir yaitu *precission* 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 memandang permasalahan pelabelan dokumen kesehatan sebagai permasalahan *sequence labeling*. Kami mengusulkan penggunaan teknik *Deep Learning* dengan menggunakan *Recurrent Neural Networks* (RNNs), karena RNNs merupakan *state-of-the-art* untuk permasalahan *sequence labeling* seperti permasalahan pada penelitian ini. Sebelumnya penelitian terkait hal ini pernah dikerjakan oleh Mujiono et al. (2016), dengan jenis entitas yang digunakan adalah entitas *Drug* saja. Peneliti tersebut menggunakan model RNNs untuk melabeli dokumen secara otomatis. Dengan menggunakan fitur vektor kata yang menggunakan *word embedding* saja, *f-measure* yang diberikan mencapai 86.45%. Oleh karena itu, penulis mengusulkan model RNNs 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. Sistem MER juga dapat digunakan pada aplikasi *Question*

Answering (Abacha dan Zweigenbaum, 2011) dengan cara memanfaatkan hasil pelabelan untuk melakukan identifikasi entitas yang ditanyakan. Gambar 1.1 merupakan contoh penggunaan MER pada aplikasi *question answering* dalam bidang kesehatan.



Gambar 1.1: Diagram Penggunaan MER pada aplikasi *Question Answering*

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. Bagaimana performa RNNs dibandingkan dengan CRFs (*baseline*) untuk sistem MER yang dikembangkan?
2. Fitur apa saya yang membuat sistem MER memiliki performa terbaik?

1.3 Tujuan dan Manfaat Penelitian

Penelitian ini bertujuan untuk membangun sistem yang mampu melakukan ekstraksi entitas kesehatan dari forum *online*. Sebenarnya, pada penelitan 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 RNNs 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. Sistem MER juga dapat digunakan pada aplikasi *Question Answering* (Abacha dan Zweigenbaum, 2011) dengan cara memanfaatkan hasil pelabelan untuk melakukan identifikasi entitas yang ditanyakan.

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 pemelajaran dan untuk mendukung penelitian yang penulis lakukan. Literatur yang penulis gunakan memiliki keterkaitan terhadap kasus MER, *Sequence Labeling* dan *Recurrent Neural Networks*.

2. Pengumpulan Data
   Pada tahapan ini, penulis mengumpulkan data percobaan yang diperlukan. Penulis mengumpulkan dokumen teks dari forum kesehatan *online* 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, dan lain sebagainya. 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 seperi 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.

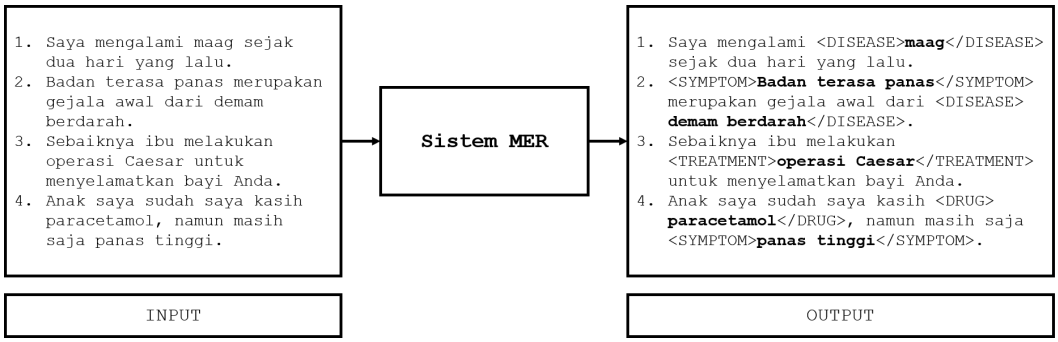
- Bab 5 EKSPERIMEN
  Pada bab ini penulis menjelaskan analisis dari hasil eksperimen yang telah penulis kerjakan pada tahap sebelumnya. Hasil eksperimen penulis sajikan dalam bentuk tabel dan grafik.

- 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 (*Named Entity Recoginition*) atau disingkat 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 entitas yang berada pada domain kesehatan seperti nama penyakit (*disease*), gejala penyakit (*symptom*), obat (*drug*), langkah penyembuhan (*treatment*), nama protein, DNA, RNA dan lain sebagainya. 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 dimasud 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 MER 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-measeure* 63.69% dengan fitur *its own word*, frasa, kamus (*symptom*, *disease*, *treatment* dan *drug*), *window feature (previous word)* 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 *http://health.detik.com/*, *http://detikhealth.com/* dan *http://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* yang model komputasinya terdiri dari beberapa layer. *Deep learning* mampu mempelajari dan mengekstrak representasi data/fitur secara otomatis pada abtraksi tingkat tinggi (LeCun et al., 2015). Model tersebut memberikan hasil yang sangat baik dalam penelitan di berbagai bidang seperti *speech recognition*, *object detection*, *sequence labeling* dan lain sebagainya.

Struktur pembelajaran pada *deep learning* berbentuk hierarki karena termotivasi dari bagaimana neokorteks pada otak maunusia bekerja secara mendalam. Neokorteks tersebut melakukan proses pemelajaran berlayer dan secara otomatis mampu mengketrak fitur dan melakukan abstraksi dari *resource* yang diberikan (Bengio et al., 2007). Struktur tersebut terdiri atas *input layer*, *hidden layer* dan *output layer*. *Input layer* memiliki fungsi sebagai tempat masuknya data yang akan dipelajari oleh model. *Hidden layer* melakukan aproksimasi fungsi untuk mendapatkan target dari data *training* yang diberikan. Disebut *hidden layer* karena

pada layer ini, *output* tidak bisa kita lihat (Goodfellow et al., 2016). *Hidden layer* inilah yang menjadi *key role* dalam *deep learning*. Sedangkan *output layer* merupakan layer untuk mengembalikan target yang diinginkan.

Deep learning ini mampu memberikan model yanng memiliki performa sangat baik dalam *supervised learning* (Goodfellow et al., 2016). Dengan menambahkan lebih banyak layer dan unit di dalam layer, *deep network* dapat merepresentasikan fungsi dengan kompleksitas yang tinggi. Secara umum, *deep learning* memetakan *input vector* ke *output vector*. Walaupun hal ini mudah dilakukan oleh manusia secar manual, namun untuk *dataset* yang sangat besar, tentu hal ini tidak mungkin dilakukan. Ada banyak macam model *Deep Learning* yang sesuai dengan kebutuhan komputasi, seperti *Deep Belief Network* (Hinton et al., 2006), *Recurrent Neural Networks* (Elman, 1990), *Long Short Term Memory* (Hochreiter dan Schmidhuber, 1997), *Restricted Boltzman Machine* (Pennington et al., 2014) dan lain sebagainya.

2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) merupakan 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* sebelumya. 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* terkait *sequence data*, seperti *language model* (Mikolov et al. (2010)) dan *speech recognition* (Graves et al. (2013)).

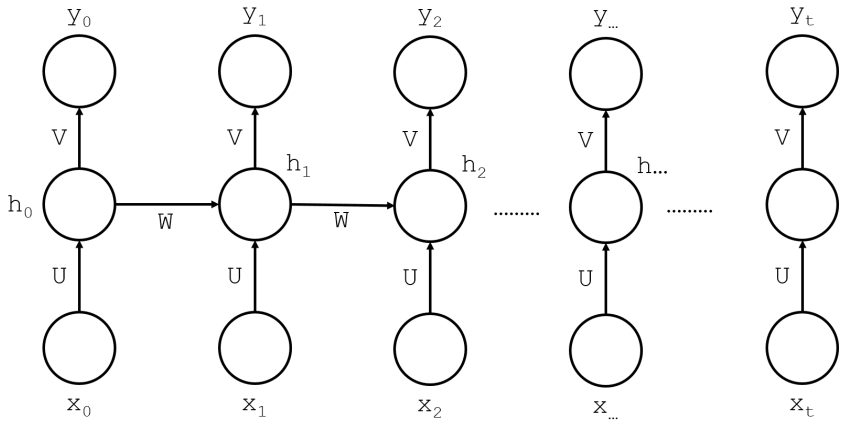
Dibandingkan dengan FNNs, RNNs memiliki beberapa kelebihan (Mikolov et al., 2010), yaitu:

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* $x(\vec{t}) \in \mathbb{R}^{N}$, *hidden layer* $h(\vec{t}) \in \mathbb{R}^{H}$, dan *output layer* $y(\vec{t}) \in \mathbb{R}^{M}$. 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* menggunakaan dan mengestimasi tiga parameter tersebut.

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

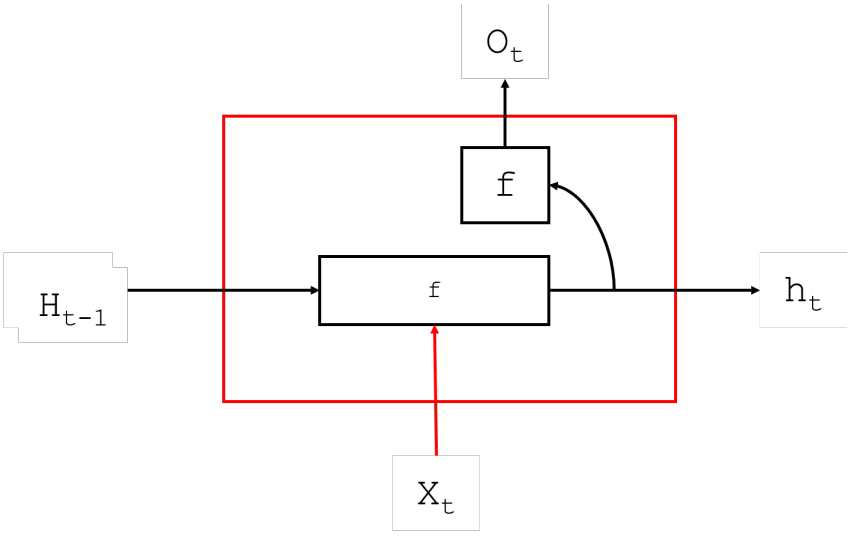
$$y(\vec{t}) = f(V \cdot (\vec{t})) \tag{2.1}$$

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

dimana

$$h(\vec{0}) = f(U \cdot x(\vec{0})) \tag{2.3}$$

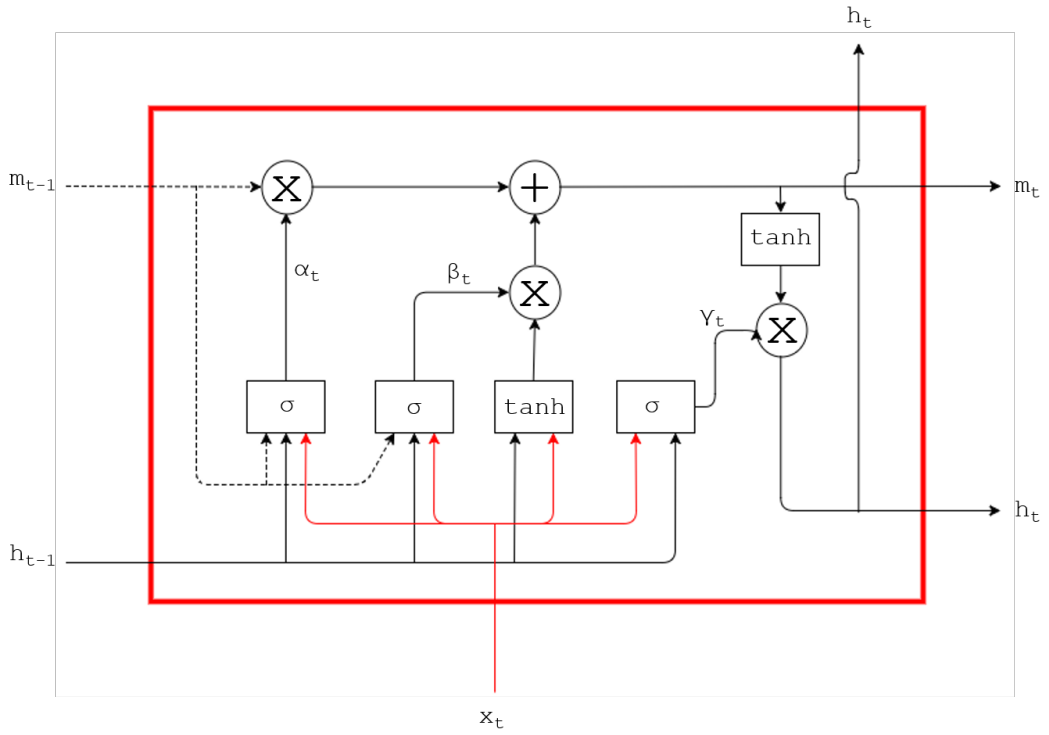
dengan f sebagai *activation function*, misalnya $tanh$ atau $softmax$. Untuk lebih jelasnya, berikut merupakan gambar dari satu buah *timestep* di dalam RNNs.



Gambar 2.3: 1 buah *timestep* dalam RNNs

2.3.1 Long Short Term Memory

Pada penjelasan di atas, RNNs sederhana 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 sederhana 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* (Mozer 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 LSTMs. Pada dasarnya, arsitektur LSTMs mirip dengan RNNs sederhana, namun unit *nonlinear* pada *hidden layer* di dalam RNNs sederhana 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 perah ditimpa sehingga isi memori tidak berubah.

Pada gambar 2.4, kita dapat melihat bahwa 1 blok memori pada LSTMs 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

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

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

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

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

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

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

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

$$h_t = \gamma_t(\times)tanh(m_t) \tag{2.8}$$

dimana

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

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 medis 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% dari *baseline*.

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. Hasil yang didapatkan adalah Char-BiLSTM pada Stockholm EPR corpus mendapatkan *precision* 0.67, *recall* 0.12 dan *f-measure* 0.20 meningkat 60% dibandingkan dengan *baseline*.

2.4 Word Embedding

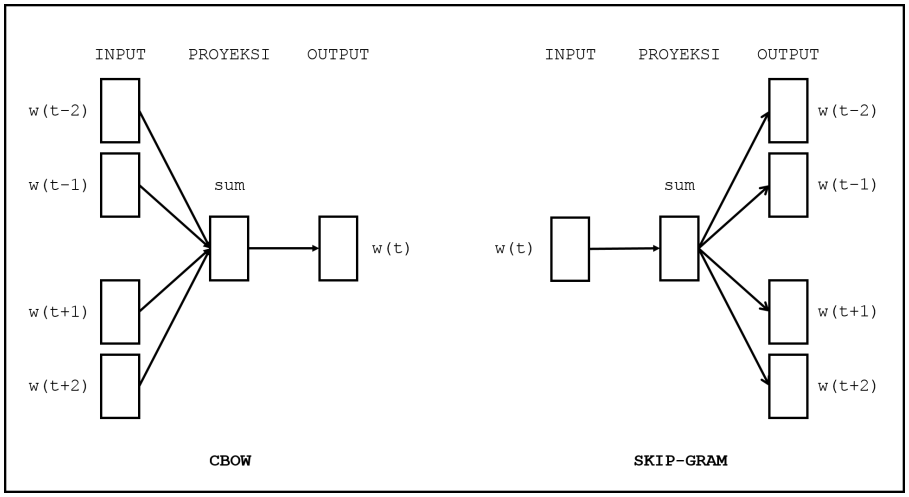
Pada umumnya, pendekatan yang digunakan untuk merepresentasikan sebuah kata sebagai *input* model adalah dengan menggunakan *one-hot-vetor* (Turian et al.,

2010). 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 mngurutkannya 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 sintak maupun semantik 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("obat") = [0.4, -0.9, 0.1, ..., 0.9], dimana W adalah fungsi yang memetakan suatu kata menuju representasi vektor dan W("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 Mikolov et al. (2014) dan berdasarkan pada *neural networks*. 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-occurance* 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

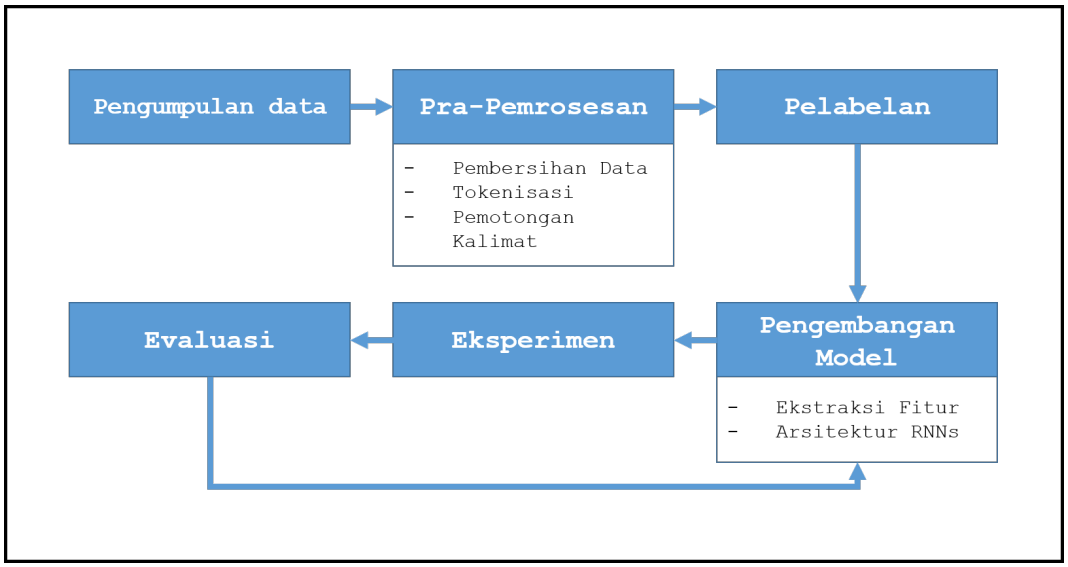
Penelitian ini bertujuan untuk membuat sebuah model yang mampu memberikan label entitas kesehatan pada suatu dokumen. Seperti yang telah dijelaskan pada bab sebelumnya, terdapat banyak entitas kesehatan yang dapat digunakan sebagai target pelabelan. Oleh karena itu, untuk mempermudah penelitian ini penulis 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 yang 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 LSTMs, baru kemudian *output* dari layer tersebut digabung.

Setelah melakukan eksperimen, penulis melakukan evaluasi dari hasil yang

didapatkan dengan menghitung nilai *precission*, *recall* dan *F-measure* dari masing-masing entitas secara keseluruhan. Untuk mendapatkan rata-rata akurasi dari setiap eksperimen, penulis melakukan *10-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 sedangkan 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*) diganti menjadi token "url",
3. mengganti token *email* menjadi kata "email", misalnya sebuah alamat *email* (*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. Gambar 3.2 merupakan contoh pembersihan data pada sebuah teks.

Kalimat sebelum Pembersihan Data	Kalimat setelah Pembersihan Data
Dapat terjadi penurunan detak jantung bayi Â pasca tindakan putar dari luar tersebut.	Dapat terjadi penurunan detak jantung bayi pasca tindakan putar dari luar tersebut.
Jawaban TanyaDok.com di : http://www.tanyadok.com/tanyadokter/183979-2	Jawaban TanyaDok.com di : url
Cepat marah & jantung berdebar	Cepat marah dan jantung berdebar

Gambar 3.2: Ilustrasi Pembersihan Data pada Kalimat

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

Gambar 3.3 berikut merupakan contoh tokenisasi pada sebuah teks.

Kalimat sebelum Tokenisasi
Wah kecil sekali ya Bu.
hampir 1 bulan, napas nya bunyi grok2,
bayi saya sudah berumur enam (6) bulan,

Kalimat setelah Tokenisasi
Wah kecil sekali ya Bu .
hampir 1 bulan , napas nya bunyi grok 2 ,
bayi saya sudah berumur enam (6) bulan ,

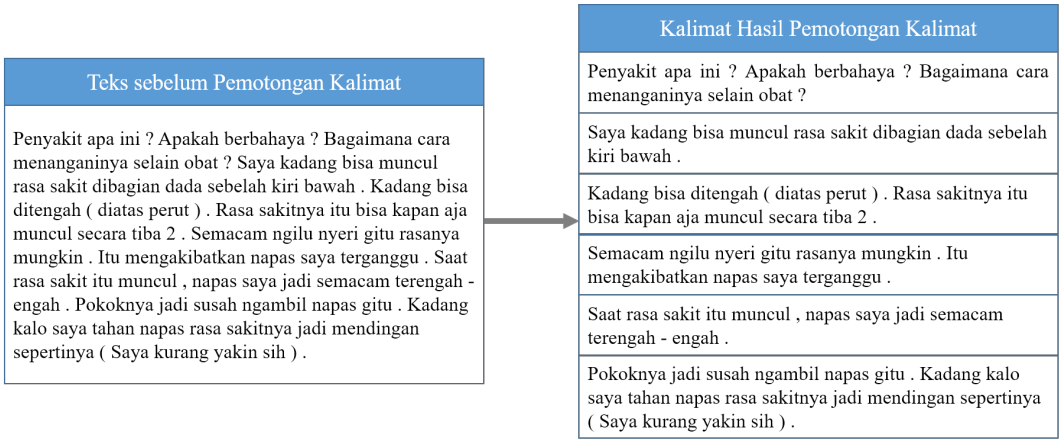
Gambar 3.3: Ilustrasi Tokenisasi pada Kalimat

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.

Gambar 3.4 berikut merupakan contoh dari pemotongan kalimat pada sebuah teks.



Gambar 3.4: Ilustrasi Pemotongan Kalimat pada suatu Teks

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

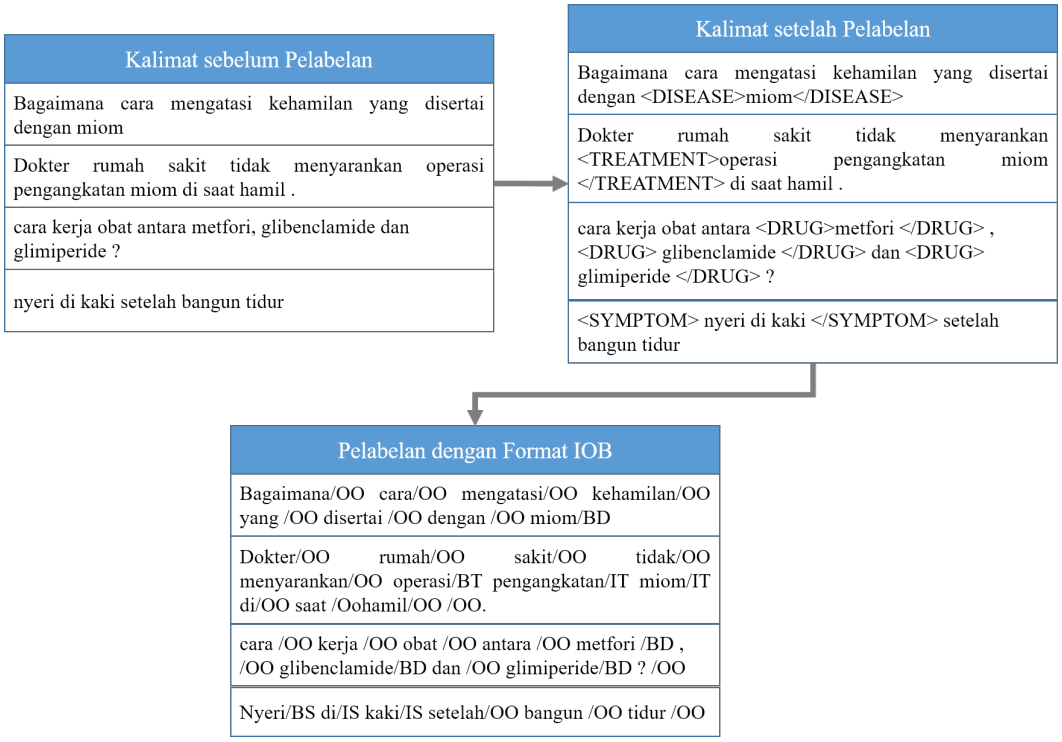
4. *Treatment*
   Entitas *treatment* merupakan cara atau langkah penyembuhan dari suatu penyakit. Contoh dari entitas *treatment* yaitu:

   - Pemeriksaan darah rutin
   - Penilaian denyut kapiler
   - Terapi inhalasi

5. *Other*
   Entitas *other* merupakan suatu entitas selain dari keempat entitas di atas. Contoh dari entitas *other* yaitu:

   - Saya
   - yang
   - Dokter

Setelah proses di atas selesai, label di dalam korpus diubah menjadi format BIO (*begin inside outside*). Gambar 3.5 berikut merupakan ilustrasi dari tahap pelabelan ini.



Gambar 3.5: Ilustrasi Pelabelan Entitas pada suatu Kalimat

Setelah pelabelan selesai dilakukan, supaya model RNNs mampu mengenali masing-masing label, penulis menggunakan *one-hot-vector* untuk merepresentasikan masing-masing label. Tabel 3.1 merupakan tabel pemetaan dari label menjadi represebtasinya dalam *one-hot-vector*.

Tabel 3.1: Tabel Pemetaan Label dengan Representasi *One-Hot-Vector*

Label IOB	*One-hot-vector*
BD	[0, 0, 0, 0, 0, 0, 0, 0, 1]
ID	[0, 0, 0, 0, 0, 0, 0, 1, 0]
BS	[0, 0, 0, 0, 0, 0, 1, 0, 0]
IS	[0, 0, 0, 0, 0, 1, 0, 0, 0]
BT	[0, 0, 0, 0, 1, 0, 0, 0, 0]
IT	[0, 0, 0, 1, 0, 0, 0, 0, 0]
BG	[0, 0, 1, 0, 0, 0, 0, 0, 0]
IG	[0, 1, 0, 0, 0, 0, 0, 0, 0]
OO	[1, 0, 0, 0, 0, 0, 0, 0, 0]

Gambar 3.6 merupakan contoh pengubahan kata menjadi *one-hot-vector* dalam suatu kalimat.

Kalimat	Bagaimana	cara	mengatasi	kehamilan	yang	disertai	miom
Label IOB	OO	OO	OO	OO	OO	OO	BD
One-Hot-Vector	[[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1],]

Gambar 3.6: Ilustrasi Pengubahan Label menjadi *One-Hot-Vector*

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 Ekstrasi 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) dan Herwando (2016) 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 2, *word embedding* memberikan hasil yang sangat baik dalam bidang NLP. 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:

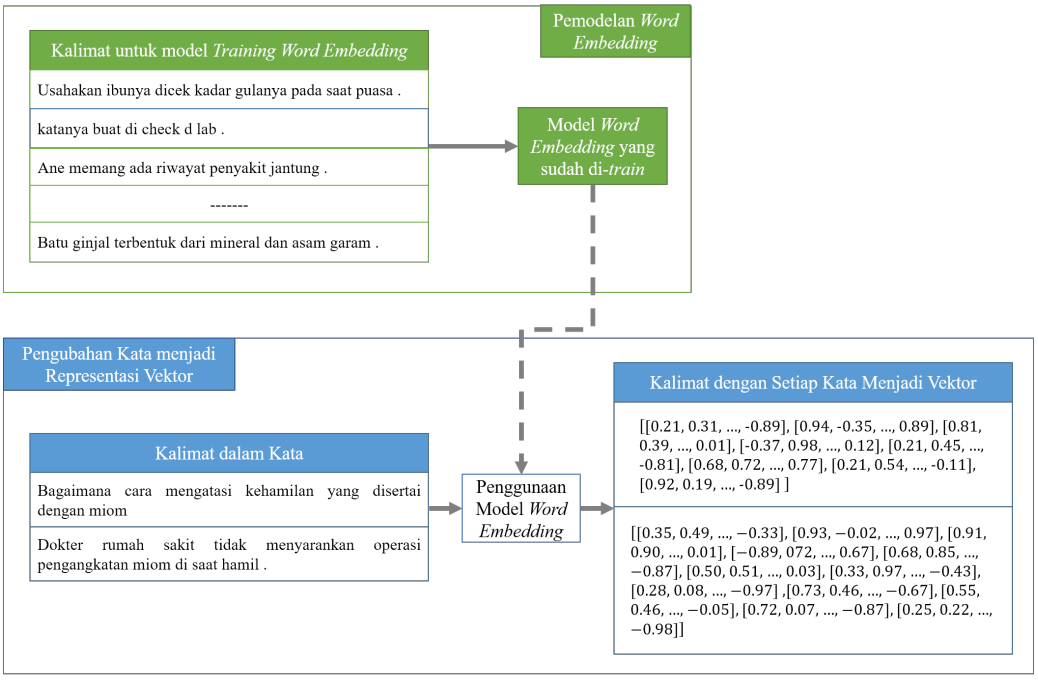
   (a) Pengumpulan data *training* untuk *word embedding*
   Penulis melakukan pengumpulan data teks sebagai *resource* untuk melakukan *training* model *word embedding*. Data teks yang penulis gunakan merupakan data teks dari artikel-artikel kesehatan dan data teks forum kesehatan di kaskus. Penulis menggunakan 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 128 dengan besaran *window* 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 mengubah masing-masing kata dalam kalimat menjadi representasi vektor dengan model yang telah penulis dapatkan pada tahap *training* model *word embedding*.

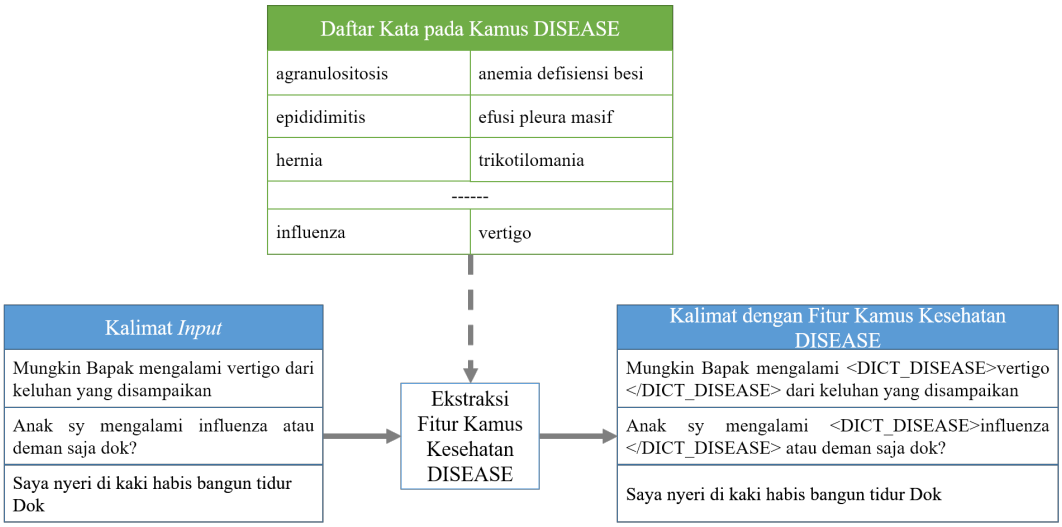
Gambar 3.7 merupakan ilustrasi dari proses ekstraksi fitur kata itu sendiri dalam suatu kalimat dengan menggunakan *Word Embedding*.



Gambar 3.7: Ilustrasi Ekstraksi Fitur Kata pada Suatu Kalimat

2. Fitur 2: Kamus Kesehatan
Fitur kamus kesehatan merupakan fitur yang berisi informasi suatu kata terdapat di dalam kamus kesehatan atau tidak. Fitur ini digunakan dalam penelitian Herwando (2016) dan memberikan kontribusi dalam hasil terbaik. 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. Gambar 3.8 merupakan ilustrasi dari ekstraksi fitur kamus kesehatan dalam suatu kalimat.



Gambar 3.8: Ilustrasi Ekstraksi Fitur Kamus *disease* pada Suatu Kalimat

Supaya model RNNs mengenali fitur ini, penulis menggunakan representasi *one-hot vector*. Untuk suatu kata yang terdapat dalam kamus kesehatan, representasi vektornya adalah $[0, 1]$, sedangkan yang tidak terdapat di dalam kamus kesehatan, representasi vektornya adalah $[1, 0]$. Gambar 3.9 merupakan contoh pengubahan fitur kamus kesehatan menjadi representasi *one-hot-vector*.

Kalimat	Mungkin	Bapak	mengalami	vertigo
Fitur Kamus Disease	-	-	-	√
One-Hot-Vector	[[1, 0],	[1, 0],	[1,0],	[0, 1],]

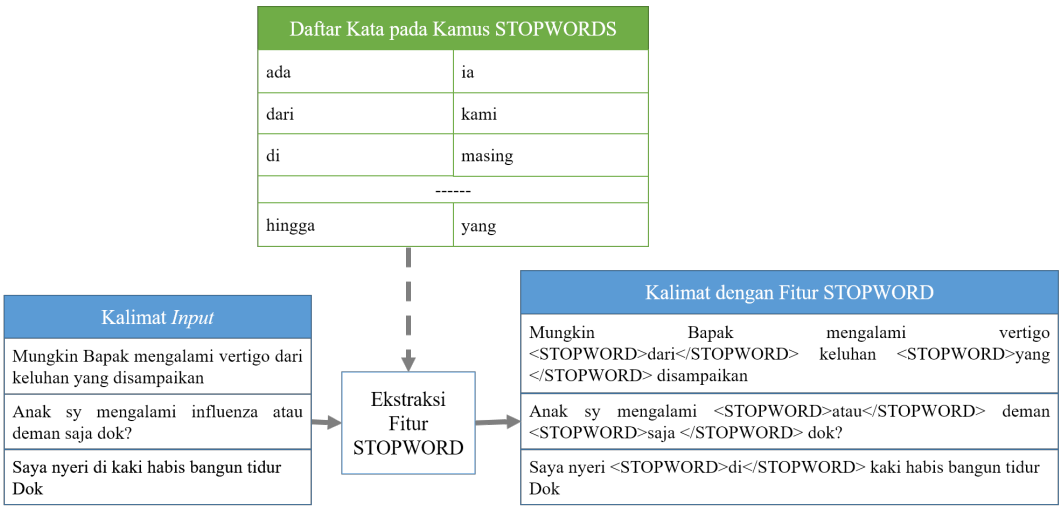
Gambar 3.9: Ilustrasi Pengubahan Label menjadi *One-Hot-Vector*

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. Gambar 3.10 merupakan ilustrasi dari ekstraksi fitur *stopword* dalam suatu kalimat.



Gambar 3.10: Ilustrasi Ekstraksi Fitur *Stopword* pada Suatu Kalimat

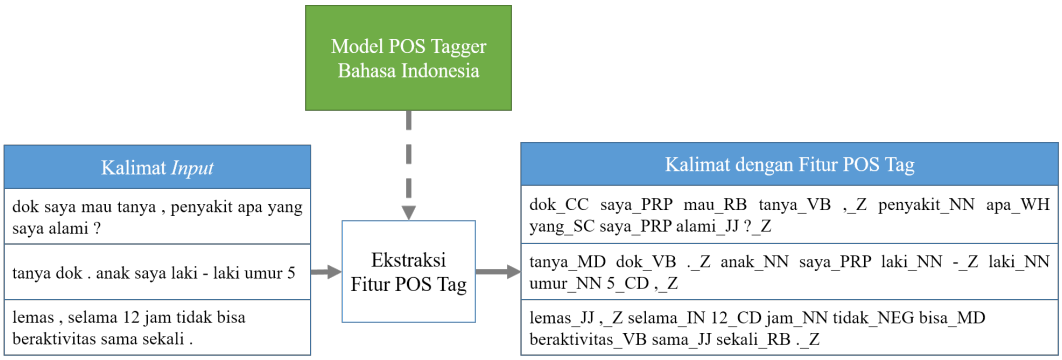
Supaya model RNNs mengenali fitur ini, penulis menggunakan representasi *one-hot vector*. Untuk suatu kata yang merupakan *stopword*, representasi vektornya adalah $[0,1]$, sedangkan yang bukan merupakan *stopword*, representasi vektornya adalah $[1,0]$. Gambar 3.11 merupakan contoh pengubahan fitur *stopword* menjadi representasi *one-hot-vector*.

Kalimat	Saya	nyeri	di	kaki	habus	bangun	tidur
Fitur *Stopword*	-	-	√	-	-	-	-
One-Hot-Vector	[[1, 0],	[1, 0],	[0, 1],	[1, 0],	[1, 0],	[1, 0],	[1, 0]]

Gambar 3.11: Ilustrasi Pengubahan Label menjadi *One-Hot-Vector*

4. Fitur 4: *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. Gambar 3.12 merupakan ilustrasi dari ekstraksi fitur POS-*Tag* dalam suatu kalimat.



Gambar 3.12: Ilustrasi Ekstraksi Fitur POS *Tag* pada Suatu Kalimat

Setelah POS-Tag didapatkan, supaya model RNNs mampu mengenali masing-masing tag, penulis menggunakan *one-hot-vector* untuk merepresentasikan masing-masing label. Tabel 3.2 merupakan tabel pemetaan dari label menjadi representasinya dalam *one-hot-vector*.

Tabel 3.2: Tabel Pemetaan POS-Tag dengan Representasi *One-Hot-Vector*

Label POS-Tag	*One-hot-vector*
JJ	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
NN	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]
NNP	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0]
Z	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0]
IN	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0]
PRP	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]
MD	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0]
VB	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0]
SC	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]
RB	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]
CC	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]
NEG	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0]
WH	[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]
CD	[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]
X	[0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
PR	[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
RP	[0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
FW	[0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
NND	[0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
DT	[0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
OD	[0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
UH	[0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
SYM	[0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
fw	[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
	[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

Gambar 3.13 merupakan contoh pengubahan fitur POS-Tag menjadi *one-hot-vector* dalam suatu kalimat.

Kalimat	tanya	dok	,	anak	saya	laki	-	laki	umur	5
Label POS-Tag	MD	VB	Z	NN	PRP	NN	Z	NN	NN	CD
One-Hot-Vector	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,1,0,0, 0,0,0,0],	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 1,0,0,0, 0,0,0,0],	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 1,0,0,0],	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,1,0],	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,1,0, 0,0,0,0],	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,1,0],	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 1,0,0,0],	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,1,0],	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,1,0],	[0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,1,0, 0,0,0,0, 0,0,0,0]

Gambar 3.13: Ilustrasi Pengubahan Fitur POS-Tag menjadi *One-Hot-Vector*

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. Selain itu, pada penelitian Herwando (2016), fitur ini berkontribusi dalam memberikan hasil terbaik. 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 diujicobakan, 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

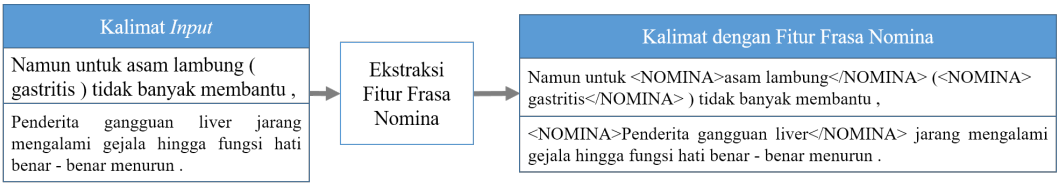
- NP : NN + JJ
- NP : DT + NN
- NP : RB + NN
- NP : CD + NN
- NP : NND + NN

(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

Gambar 3.14 merupakan ilustrasi dari ekstraksi fitur Frasa Nomina dalam suatu kalimat.



Gambar 3.14: Ilustrasi Ekstraksi Fitur Frasa Nomina pada Suatu Kalimat

Supaya model RNNs mengenali fitur ini, penulis menggunakan representasi *one-hot vector*. Untuk suatu kata yang merupakan sebuah frasa, representasi vektornya adalah $[0,1]$, sedangkan yang bukan, representasi vektornya adalah $[1,0]$. Gambar 3.15 merupakan contoh pengubahan fitur frasa nomina menjadi representasi *one-hot-vector*.

Kalimat	Namun	untuk	asam	lambung	(	gastritis	)	tidak	banyak	membantu	,
Fitur Frasa Nomina	-	-	√	√	-	√	-	-	-	-	-
One-Hot-Vector	[[1,0],	[1,0],	[0,1],	[0,1],	[0,1],	[0,1],	[0,1],	[1,0],	[0,1],	[1,0],	[1,0]]

Gambar 3.15: Ilustrasi Pengubahan Fitur Frasa menjadi *One-Hot-Vector*

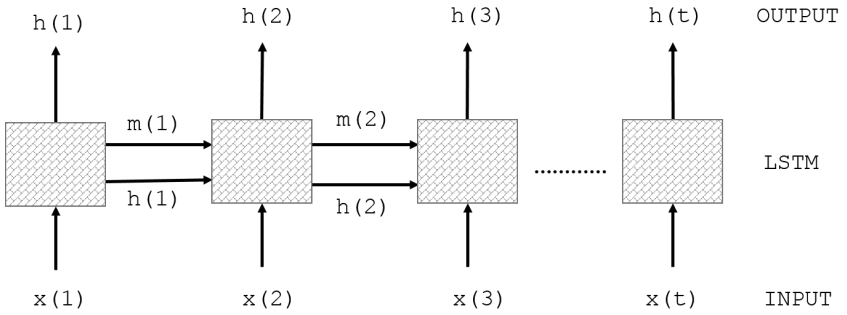
6. Fitur 6: 1 Kata Sebelum
   Fitur ini merupakan fitur yang berisi informasi kata sebelum kata saat ini yang direpresentasikan dalam bentuk vektor untuk masing-masing kata. Fitur ini digunakan pada penelitian penelitian Herwando (2016) yang juga berkontribusi memberikan hasil terbaik pada penelitiannya. 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. LSTMs 1 layer
   Pada LSTMs 1 layer, semua fitur yang menjadi input pada sebuah *timestep* digabung menjadi satu. Untuk menentukan label, penulis menggunakan *feed-forward Neural Networks* pada masing-masing *timestep* di layer terakhir. Berikut merupakan ilustrasi LSTM 1 layer yang penulis gunakan dalam penelitian ini.



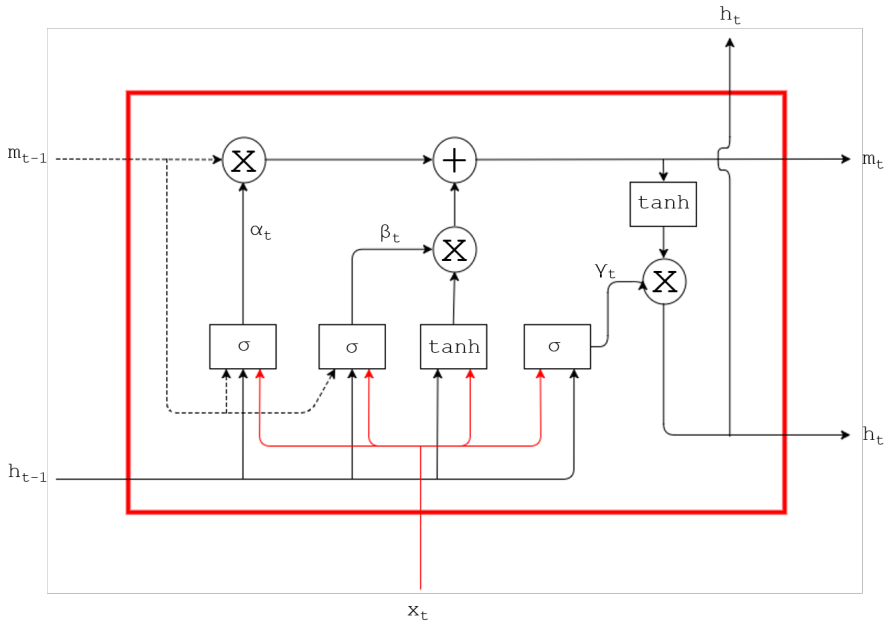
Gambar 3.16: LSTMs 1 layer

Misalnya fitur yang digunakan adalah fitur kata dan frasa. Karena hanya terdapat 1 layer saja, kedua fitur ini digabung terlebih dahulu menjadi 1. Gambar 3.17 merupakan ilustrasi dari proses penggabungan fitur supaya menjadi *input* RNNs, dan *output* serta pemetaan menjadi label pada arsitektur ini.

Kalimat	Bagaimana	cara	mengatasi	kehamilan	yang	disertai	miom
Fitur Kata	[0.36, …, 0.96],	[-0.52, …, 0.26],	[0.13, …, 0.54],	[0.73, …, 0.42],	[0.99, …, -0.63],	[0.01, …, 0.66],	[0.78, …, 0.31],
Fitur Frasa	[1, 0],	[1, 0],	[1, 0],	[0, 1],	[1, 0],	[1, 0],	[0, 1],
Input	[0.36, …, 0.96, 1, 0],	[-0.52, …, 0.26, 1, 0],	[0.13, …, 0.54, 1, 0],	[0.73, …, 0.42, 1, 0],	[0.99, …, -0.63, 1, 0],	[0.01, …, 0.66, 1, 0],	[0.78, …, 0.31, 1, 0],
Output	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
Label IOB	OO	OO	OO	OO	OO	OO	BD
Hasil RNNs	*other*	*other*	*other*	*other*	*other*	*other*	*disease*

Gambar 3.17: 1 buah blok memori dalam LSTM

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



Gambar 3.18: Ilustrasi penggabungan fitur kata dan frasa untuk menjadi *input* LSTMs 1 layer dan *output*

Dari gambar 3.18, 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.

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

$$h_t = \gamma_t(\times)tanh(m_t) \tag{3.2}$$

dimana

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

α_t, β_t dan γ_t merupakan *gates*:

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

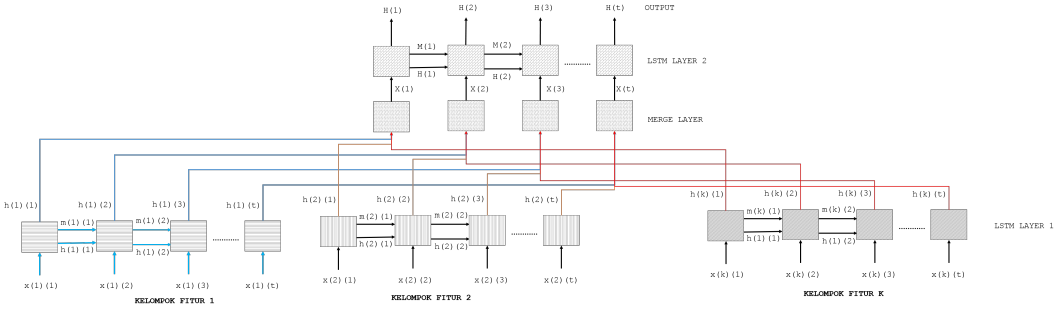
(b) *Input gates*: $\beta_t = \sigma(W_{x\beta} + W_{h\beta} \cdot\ h_{t-1} + W_{m\beta} \cdot\ m_{t-1})$

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

2. LSTMs 2 layer dengan Multi-*Input*

   Pada LSTMs 2 layer dengan multi-*input*, penulis mendefinisikan 2 layer, yang layer terbawah merupakan layer dengan jumlah LSTMs sebanyak n kelompok fitur. Pertama-tama fitur dikelompokkan terlebih dahulu, kemudian

dijadikan *input* untuk masing-masing LSTMs pada tingkat pertama. Setelah itu, hasil dari tingkat pertama tersebut akan digabung menjadi satu, dengan menggunakan layer penggabung (*Merge Layer*). *Output* dari layer penggabung kemudian dimasukkan ke dalam LSTMs layer kedua. Untuk menentukan label, penulis menggunakan *feed-forward Neural Network* pada masing-masing *timestep* di layer terakhir. Berikut merupakan ilustrasi dari LSTMs layer bertingkat yang penulis gunakan.



Gambar 3.19: 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:

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

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

dimana

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

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

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

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

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

Merge layer berfungsi untuk menggabungkan hasil dari *feedforward* pada semua LSTMs layer pertama. Di sini, penulis menotasikan X_t sebagai hasil

dari *merge* di *timestep t* dan $(\cdot)$ merupakan operasi *merging*.

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

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-forwarnya*.

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

$$H_t = \gamma_t(\times)tanh(M_t) \tag{3.9}$$

dimana

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

α_t, β_t dan γ_t merupakan *gates*:

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

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

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

3.6 Eksperimen

Dalam melakukan eksperimen, arsitektur *deep learning* yang penulis gunakan adalah *Recurrent Neural Networks*, dalam hal ini penulis menggunakan LSTMs. Hal ini penulis lakukan karena pada penelitian Mujiono et al. (2016), Jagannatha dan Yu (2016), Limsopatham dan Collier (2016) dan Almgren et al. (2016), penggunaan LSTMs memberikan *output* terbaik dalam MER yang dirancang. Selain itu, LSTMs juga sangat baik dalam masalah *sequence labeling* seperti yang dilakukan oleh Graves et al. (2013) dan merupakan *state-of-the-art* dalam bidang ini. Masih banyak penelitian lain yang membuktikan bahwa LSTMs merupakan arsitektur *deep learning* yang sangat baik dalam masalah *sequence labeling* seperti *Offline Hadwriting 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 *resource* yang penulis miliki. Sebelum melakukan eksperimen,

penulis membagi data *training* menjadi 10 bagian, kemudian melakukan iterasi sebanyak 10 kali yang 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.

3.7 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: Anak saya <Disease>**sakit kepala** sebelah</Disease>
Predicted: Anak saya <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**</Symptom> Predicted : <Sympton>Masuk angin</Sympton> dan <Sympton>**suhu badan tinggi**</Symptom>

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 : Anak saya <Disease>sakit kepala sebelah</Disease> Predicted : Anak saya <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*,

precission dan *recall* untuk masing-masing entitas dengan menggunakan formula:

$$Precission = \frac{TP}{TP + FP} \tag{3.11}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.12}$$

$$F - Measeure = 2 \cdot \frac{Precission \cdot Recall}{Precission + Recall} \tag{3.13}$$

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 dengan cara yang sama.

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 Perangkat Pendukung

Dalam melakukan eksperimen, penulis menggunakan perangkat komputer yang disediakan di lab. Perangkat komputer tersebut memiliki spesifikasi seperti pada tabel 4.1.

Tabel 4.1: Spesifikasi *Hardware*

Processor	i7-4770S
Banyak Core	8 core
Frekuensi Processor	3.1 GHz per core
RAM	8 GB

Berikut merupakan detail dari sistem operasi yang terpasang pada komputer di atas

Tabel 4.2: Spesifikasi Sistem Operasi

Distributor ID	Ubuntu
Deskripsi	Ubuntu 13.10
Versi rilis	13.10
Nama kode	saucy

4.2 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*.

4.3 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:

4.3.1 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;
```

4.3.2 Tokenisasi

Seperti yang dijelaskan pada Bab 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 pemrograman Ruby. Berikut merupakan *pseudocode* untuk melakukan tokenisasi.

4.3.3 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.

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;
```

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 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 atuan pelabelan yang dilakukan oleh Herwando (2016) dalam penelitiannya. Pelabelan ini dilakukan selama 2 minggu.

4.5 Pengembangan Model

4.5.1 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.

4.5.1.1 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 (*https://www.kaskus.co.id/forum/94/health, http://archive.kaskus.co.id/forum/94*). 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;
```

4.5.1.2 Ekstraksi Fitur Kamus Kesehatan

Pada dasarnya implementasi ekstraksi fitur kamus kesehatan mirip dengan implementasi ekstraksi fitur *stop word*. Perbedaanya 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 penulis 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 4.6.

Kode 4.6: *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
```

4.5.1.3 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 ni dapat dilihat pada kode 4.7.

4.5.1.4 Ekstraksi Fitur Part of Speech Tag

Dalam mengimplementasikan ekstraksi fitur POS Tag, penulis menggunakan *tools* Stanford POS Tagger (Toutanova dan Manning, 2000) dan model POS *tagger* yang dikembangkan oleh Dinakaramani et al. (2014). Pertama-tama penulis melakukan

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;
```

pemberian tag pada setiap kaliman di dalam korpus, kemudian mengubah hasil tag tersebut menjadi bentuk *one-hot-vector*. Kode 4.8 merupakan *pseudocode* dalam melakukan ekstraksi fitur POS Tag untuk sebuah kalimat yang penulis lakukan.

Kode 4.8: *Pseudocode* untuk melakukan ekstraksi fitur POS-Tag

```
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;
```

4.5.1.5 Ekstraksi Frasa Kata Benda

Dalam mengimplementasikan ektraksi fitur kata benda, penulis menggunakan *library* NLTK (Bird et al., 2009) 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* yang berisi *array of one-hot-vector* dari masing-masing kata dalam 1 kalimat, yang 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 4.9: *Pseudocode* untuk melakukan ekstraksi fitur frasa kata benda

```
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     writeToFile(corpusChunked)
23 end
```

4.5.1.6 Ekstraksi Frasa Kata Kerja

Sama seperti pada pengimplementasian ektraksi fitur kata benda, penulis menggunakan *library* NLTK (Bird et al., 2009) 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, yang 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 kerja. *Pseudocode* untuk melakukan ekstraksi fitur ini dapat dilihat pada

kode 4.10.

Kode 4.10: *Pseudocode* untuk melakukan ekstraksi fitur frasa kata kerja

```
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
```

4.5.1.7 Ekstraksi Fitur 1 Kata Sebelum

Ekstraksi fitur ini penulis lakukan dengan 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. Kode 4.11 merupakan implementasi dari ekstraksi fitur 1 kata sebelum.

4.5.1.8 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. Kode 4.12 merupakan implementasi dari ekstraksi fitur 1 kata sesudah.

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
```

Kode 4.12: *Pseudocode* untuk melakukan ekstraksi fitur 1 kata sesudah

```
1 Function oneWordBeforeExtract(sentenceVector) is
      Input  : array of vector in a sentence
      Output: array of vector
2     oneWordAfterFeature = [];
3     for i in 0...sentenceVector.length - 1 do
4         oneWordAfterFeature.append(sentenceVector[i+1]);
5     end
6     oneWordAfterFeature.append(zeroth);
7     return oneWordAfterFeature;
8 end
```

4.5.2 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 LSTMs.

4.5.2.1 LSTMs 1 layer

LSTMs 1 layer yang dimaksud adalah model yang digunakan memiliki satu layer LSTMs 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*, LSTMs 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 LSTMs 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 LSTMs 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 4.13 merupakan kode untuk mengimplementasikan model ini.

Kode 4.13: *Pseudocode* untuk arsitektur LSTMs 1 tingkat

```
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;
```

4.5.2.2 LSTMs 2 Layer Multi-Input

LSTMs 2 layer yang dimaksud yaitu terdapat dua layer, layer pertama untuk menerima *input* yang setiap kelompok fitur menjadi *input* bagi LSTMs masing-masing. Misalnya terdapat 3 kelompok fitur, masing-masing kelompok tadi akan menjadi input bagi layer LSTMs masing-masing. Layer kedua sebagai penggabung hasil dari tingkat pertama.

Layer pada tingkat pertama terdiri dari *Masking Layer* dan sebuah Layer LSTMs. Untuk dimensi *input* dan *output Masking Layer* secara otomatis mengikuti dimensi dari data masukan. Dimensi *output* dari Layer LSTMs 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 LSTMs. Keluaran dari *Merge Layer* sesuai dengan total dimensi *output* dari masing-masing LSTMs di tingkat 1. Dimensi keluaran dari masing-masing *Dense Layer* yaitu 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 4.14 merupakan kode untuk mengimplementasikan model ini.

Kode 4.14: *Pseudocode* untuk arsitektur LSTMs layer bertingkat

```
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(LSTM(output = 32));
12     mainModel.add(TimeDistributed(Dense(output = 9)));
13     mainModel.input(groupOfArrTraining);
14     prediction = mainModel.predict(groupOfArrTraining);
15     return prediction;
```

4.6 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

Kode 4.15: *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;
```

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 berkas dengan format JSON yang nantinya akan menjadi *input* pada tahap evaluasi. Berikut merupakan implementasi eksperimen dengan masing-masing arsitektur tersebut.

Kode 4.16: *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);
```

4.7 Evaluasi

Dalam melakukan implementasi pada tahap evaluasi, penulis menghitung nilai *prescision, recall* dan *F-Measure* 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 4.17: *Pseudocode* untuk melakukan evaluasi

```
resultTag = load(resultRNN);
originalTag = load(originalTag);
TP = newHash();
FP = newHash();
FN = newHash();
for i = 0; i < len(resultTag); i++ do
    sentenceResult = resultTag[i];
    sentenceOriginal = originalTag[i];
    for j = 0; j < len(sentenceOriginal); i++ do
        wordResult = sentenceResult[j];
        wordOri = sentenceOriginal[j];
        if wordOri != O then
            if wordResult != O then
                if wordOri == wordResult then
                    TP[wordOri] += 1;
                else
                    FN[wordOri] += 1;
            else
                FN[wordOri] += 1;
        else
            if wordResult != O then
                FP[wordOri] += 1;
prec = newHash();
rec = newHash();
fMeas = newHash();
foreach label in TP do
    prec[label] = TP[label] / (TP[label] + FP[label]);
    rec[label] = TP[label] / (TP[label] + FN[label]);
    fMeas[label] = 2 * (prec[label] * rec[label]) / (prec[label] + rec[label]);
foreach label in prec do
    print "Precission", label, prec[label];
    print "Recall", label, rec[label];
    print "F-Measure", label, fmeas[label];
```

BAB 5
EKSPERIMEN

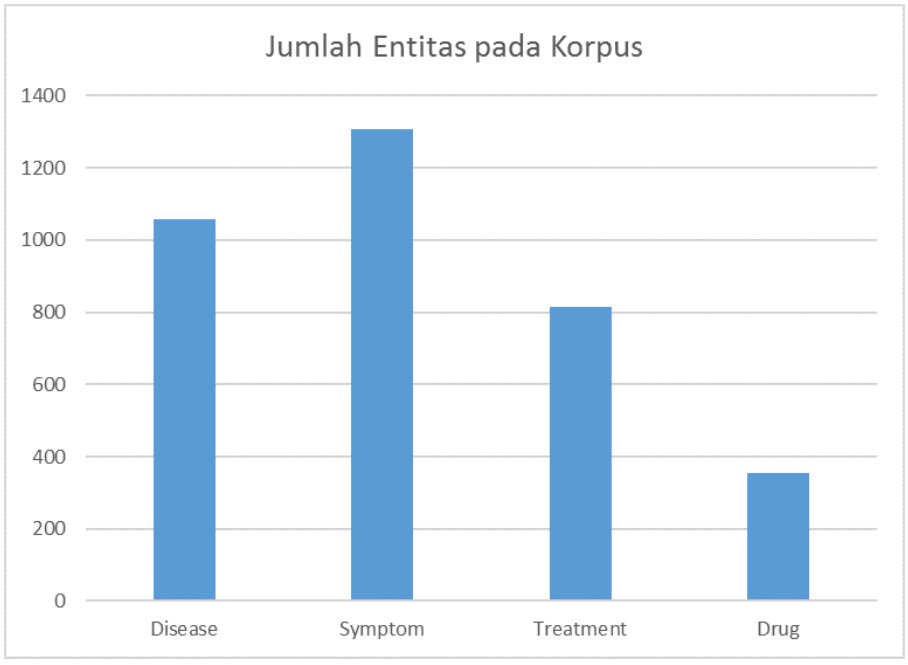
Pada bab ini penulis akan menjelaskan mengenai skeanrio, hasil dan analisis dari eksperimen yang telah dilakukan.

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 Visualisasi Data

Berikut merupakan visualisasi data dari korpus yang penulis miliki. Dapat dilihat dari grafik 5.1, bahwa persebaran jumlah entitas tidak seimbang. Hal ini menjadi kendala penulis dalam melakukan penelitian ini, karena keterbatasan *resource* dan tenaga untuk melakukan pelabelan dokumen secara manual.



Gambar 5.1: Histogram Jumlah Entitas pada Korpus

Tabel 5.1 menunjukkan 39 daftar entitas *disease* teratas yang terdapat di dalam korpus.

Tabel 5.1: Tabel Beberapa Entitas *Disease* pada Korpus dan Jumlahnya

asma (20)	isk (20)	tbc (18)
infeksi (17)	sinusitis (16)	jerawat (15)
alergi (14)	oligomenorea (14)	maag (12)
fam (11)	flu (11)	kanker serviks (11)
hiv (11)	penyakit jantung (11)	varikokel (10)
wasir (10)	infeksi saluran kemih (9)	obesitas (9)
cacar air (9)	diabetes (9)	albino (9)
gerd (8)	hepatitis b (8)	tth (8)
liver (8)	sifilis (8)	tuberkulosis (7)
stroke (7)	sakit maag (7)	anemia aplastik (7)
alergi susu sapi (7)	hepatitis (7)	diare (6)
scabies (6)	kanker otak (6)	jengger ayam (6)
pid (6)	osteoporosis (5)	tifus (5)

Tabel 5.2 menunjukkan 39 daftar entitas *symptom* teratas yang terdapat di dalam korpus.

Tabel 5.2: Tabel Beberapa Entitas *Symptom* pada Korpus dan Jumlahnya

nyeri (60)	sakit kepala (51)	demam (50)
mual (40)	gatal (30)	batuk (27)
pusing (24)	muntah (23)	diare (16)
keputihan (15)	nyeri dada (14)	sesak nafas (14)
kejang (13)	keringat dingin (13)	pilek (12)
nyeri kepala (10)	stres (10)	stress (10)
sesak (8)	jantung berdebar - debar (7)	flu (7)
rasa nyeri (7)	lemas (7)	kesemutan (7)
bersin (6)	sakit gigi (6)	mimisan (6)
nyeri pinggang (6)	depresi (5)	perih (5)
sakit perut (5)	anyang - anyangan (4)	bintilan (4)
kram perut (4)	pingsan (4)	susah tidur (4)
rasa gatal (4)	perubahan mood (4)	kelelahan (4)

Tabel 5.3 menunjukkan 39 daftar entitas *treatment* teratas yang terdapat di dalam korpus.

Tabel 5.3: Tabel Beberapa Entitas *Treatment* pada Korpus dan Jumlahnya

operasi (40)	pemeriksaan fisik (34)	terapi (21)
usg (15)	pemeriksaan penunjang (11)	tes darah (10)
istirahat (9)	pemeriksaan darah (8)	pemeriksaan jantung (7)
ct scan (7)	ekg (6)	x - ray (6)
foto rontgen (5)	biopsi (5)	imunisasi (5)
tes pap (5)	operasi sesar (4)	mri (4)
istirahat cukup (4)	kontrasepsi (4)	dinebu (3)
tes lbc (3)	minum air putih (3)	rontgen (3)
rontgen dada (3)	susu (3)	fisioterapi (3)
tes iva (3)	berolahraga (3)	tes penunjang (3)
hindari memencet (2)	test pack (2)	terapi obat - obatan (2)
makan makanan bergizi (2)	pasang iud (2)	cek darah (2)
kompres dingin (2)	tes hpv (2)	gunakan kondom (2)

Tabel 5.4 menunjukkan 39 daftar entitas *drug* teratas yang terdapat di dalam korpus.

Tabel 5.4: Tabel Beberapa Entitas *Drug* pada Korpus dan Jumlahnya

antibiotik (31)	paracetamol (8)	ibuprofen (7)
parasetamol (5)	whey protein (5)	obat tetes mata (5)
obat batuk (5)	vitamin c (4)	obat herbal (4)
nebulizer (4)	kiranti (4)	salbutamol (4)
salep (4)	obat antibiotik (4)	amoxilin (3)
metronidazol (3)	obat pelangsing (3)	nitrokaf (3)
kortikosteroid (3)	asam mefenamat (3)	tetrahydrolipstatin (3)
steroid (3)	ctm (3)	asam valproat (3)
rifampicin (3)	obat pereda nyeri (2)	habbatussauda (2)
obat depaken (2)	glucovance (2)	obat pereda rasa nyeri (2)
obat nyeri (2)	obat penurun panas (2)	probiotik (2)
sunblock (2)	obat jerawat (2)	bicrolid (2)
plembab (2)	aspirin (2)	acyclovir (2)

5.3 Desain Eksperimen

Pada penelitian ini, penulis melakukan 3 buah skenario utama, yaitu skenario untuk melakukan implementasi dan eksperimen ulang pada *baseline* (penelitian sebelumnya), 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 1: Skenario untuk mengimplementasikan ulang eksperimen sebelumnya
   Skenario ini bertujuan untuk mengimplementasikan ulang model yang diusulkan pada penelitian Herwando (2016), yaitu dengan menggunakan model *Conditional Random Fields* (CRFs). Fitur yang digunakan merupakan fitur yang memberikan hasil terbaik pada penelitian sebelumnya, yaitu fitur kata, fitur kamus, fitur frasa, fitur panjang kata dan fitur kata sebelum. Tujuan dari skenario ini yaitu sebagai *baseline* dan pembanding pada penelitian ini.

2. Skenario 2: 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 LSTMs 1 layer. 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 2.1 Fitur kata

(b) Sub-skenario 2.2 menguji penambahan fitur kamus

(c) Sub-skenario 2.3 menguji penambahan fitur stop word

(d) Sub-skenario 2.4 menguji penambahan fitur POS-Tag

(e) Sub-skenario menguji penambahan fitur frasa kata

(f) Sub-skenario menguji pengurangan fitur POS-Tag

(g) Sub-skenario menguji penambahan fitur 1 kata sebelum

(h) Sub-skenario menguji penambahan fitur 1 kata sesudah

3. Skenario 3: 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 LSTMs 1 layer

   (b) Sub-skenario untuk menguji arsitektur LSTMs 2 layer

5.4 Skenario 1: *Baseline* Eksperimen Herwando (2016)

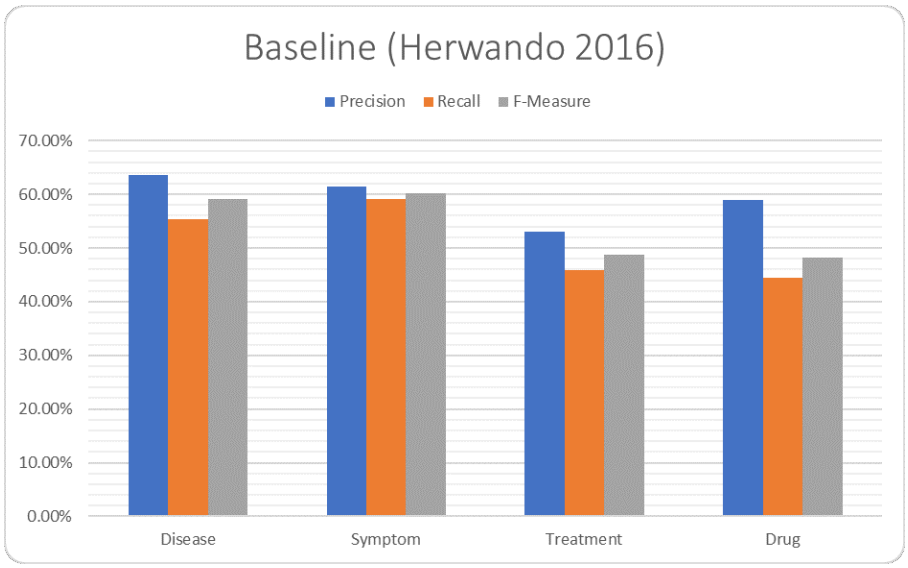
Pada penelitian ini, penulis mencoba melakukan implementasi ulang penelitian yang dilakukan oleh Herwando (2016). Data yang digunakan adalah data yang penulis gunakan dalam penelitian ini supaya perbandingan yang didapatkan *apple-to-apple*. Implementasi dan eksperimen ini bertujuan sebagai *baselaine* eksperimen dan penelitian yang penulis lakukan. Selain itu, juga untuk mengetahui secara singkat fitur yang diskriminatif dalam melakukan *sequence labeling* pada MER ini. Model yang digunakan sama dengan penelitian Herwando (2016), yaitu CRFs dengan penggunaan fitur kata, fitur kamus, fitur frasa, fitur panjang kata dan fitur kata sebelum.

5.4.1 Hasil Eksperimen

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

Tabel 5.5: Tabel Hasil Eksperimen dari Penelitian Herwando (2016) (*Baseline*)

	Precission	*Recall*	*F-Measure*
Disease	63.68%	55.45%	59.13%
Symptom	61.43%	59.21%	60.18%
Treatment	53.10%	45.97%	48.82%
Drug	58.99%	44.46%	48.23%
Overall	**59.03%**	**51.27%**	**54.09%**



Gambar 5.2: Histogram Metrik Evaluasi dari Penelitian Herwando (2016) (*Baseline*)

5.5 Skenario 2: Skenario Pengujian Fitur

5.5.1 Eksperimen 2.1: Fitur Kata

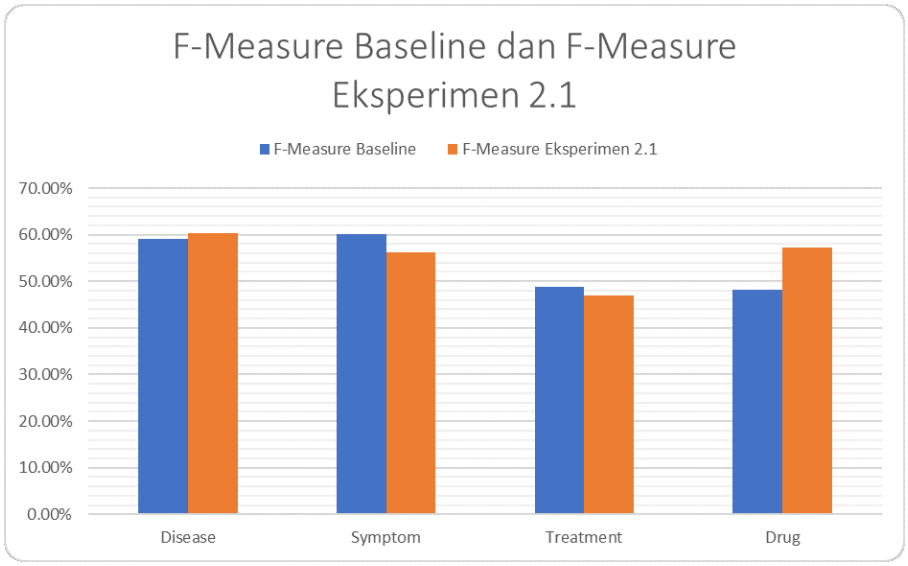
Merujuk pada penelitian Mujiono et al. (2016), penelitian tersebut bertujuan untuk mendapatkan *non-handcrafted feature*, yaitu fitur kata itu sendiri dengan menggunakan *tools Word Embedding*. Oleh karena itu, penulis menguji fitur ini untuk mengetahui pengaruhnya pada program MER di penelitian ini.

5.5.1.1 Hasil Eksperimen

Tabel 5.6 menampilkan hasil pelabelan otomatis dengan menggunakan fitur kata itu sendiri yang direpresentasikan dengan menggunakan vektor *word embedding*.

Tabel 5.6: Tabel Hasil Eksperimen 2.1 dibandigkan dengan *Baseline*

Entitas	*Baseline (Herwando 2016)*			Eksperimen 2.1		
	Precision	*Recall*	*F-Measure*	*Precision*	*Recall*	*F-Measure*
Disease	63.68%	55.45%	59.13%	61.38%	60.42%	60.37%
Symptom	61.43%	59.21%	60.18%	57.05%	56.13%	56.19%
Treatment	53.10%	45.97%	48.82%	49.92%	47.17%	46.96%
Drug	58.99%	44.46%	48.23%	62.86%	53.32%	57.28%
Overall	**59.30%**	**51.27%**	**54.09%**	**57.80%**	**54.26%**	**55.20%**



Gambar 5.3: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 2.1

5.5.1.2 Analisis

Pada tabel 5.6, dapat dilihat bahwa secara umum *recall* dan *F-measure* yang dihasilkan lebih baik dibandingkan dengan *baseline*, walaupun untuk beberapa entitas nilainya lebih rendah (entitas *symptom* dan *treatment*). Selain itu, apabila dilihat pada grafik 5.3, secara umum model ini memberikan hasil yang lebih baik pada entitas *disease* dan *drug*. Kemudian rata-rata *F-measure* yang didapatkan yaitu 55.20%, lebih tinggi dibandingkan *baseline* yang penulis gunakan yaitu 54.09%. Hal ini sangat menarik karena hanya dengan penggunaan fitur kata saja, hasil yang diberikan secara umum lebih baik dibandingkan *baseline*.

Pada eksperimen ini, ada beberapa entitas masih memiliki nilai *precision, recall* dan *f-measure* lebih kecil apabila dibandingkan dengan hasil yang dicapai Herwando (2016). Setelah penulis melakukan analisis terhadap penggunaan penggunaan *tools Word Embedding*, ternyata terdapat 429 kata unik yang tidak terdapat dalam model *word embedding*. Hal ini disebabkan oleh beberapa hal, yaitu:

1. Terdapat kata di dalam korpus yang tidak baku atau salah eja
   Korpus yang didapatkan berasal dari forum kesehatan *online* yang bersifat non-formal. Oleh karena itu baik pasien maupun dokter bebas mengutarakan pendapatnya tanpa adanya aturan bahasa formal. Oleh karena itu banyak ditemukan adanya kata yang tidak baku atau salah eja, misalnya:
   - dllsebaiknya,
   - sekatrang,
   - infeksiny,
   - kliengan.
2. Terdapat istilah sulit yang tidak terkandung di dalam model
   Terdapat beberapa istilah kesehatan pada korpus yang tidak ada di dalam model, hal ini disebabkan karena data untuk *training* model *word embedding* terbatas (model sangat tergantung pada data *training*). Oleh karena terdapat beberapa kata yang tidak terdaftar di dalam model. Contoh beberapa istilah sulit tersebut yaitu:
   - microdermabrians,
   - flixotide,
   - bimaflox,
   - polysiloxanes,
   - scizophrenia.
3. Terdapat kata yang merupakan nama orang
   Adanya kata yang merupakan nama orang tidak bisa dihindari di dalam forum kesehatan *online*. Selain itu, nama orang berbeda untuk setiap orang sehingga sulit mendapatkan vektor kata nama orang tersebut di dalam *word embedding*. Contoh dari kata yang merupakan nama orang di dalam korpus yaitu:
   - novira,
   - risma,

- oktavia,
- sudianto.

Dari beberapa kasus di atas, penulis mengusulkan menambahkan fitur yang memperkaya informasi dari fitur kata itu sendiri, 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 5.5.2.

5.5.2 Eksperimen 2.2: Fitur Kata dan 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 konribusi 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.

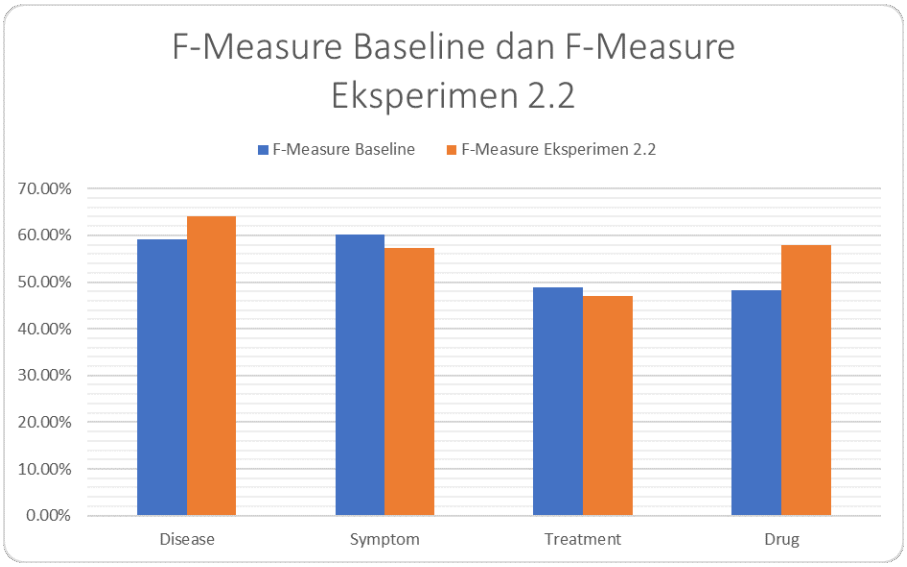
5.5.2.1 Hasil Eksperimen

Tabel 5.7 merupakan tabel hasil eksperimen yang didapatkan dengan menggunakan fitur ini.

Tabel 5.7: Tabel Hasil Eksperimen 2.2 dibandigkan dengan *Baseline*

Entitas	Baseline (Herwando 2016)			Eksperimen 2.2		
	Precision	*Recal*	*F-Measure*	*Precision*	*Recal*	*F-Measure*
Disease	63.68%	55.45%	59.13%	67.32%	61.78%	64.10%
Symptom	61.43%	59.21%	60.18%	60.55%	55.12%	57.41%
Treatment	53.10%	45.97%	48.82%	52.21%	44.18%	47.02%
Drug	58.99%	44.46%	48.23%	59.42%	59.71%	57.90%
Overall	**59.30%**	**51.27%**	**54.09%**	**59.88%**	**55.20%**	**56.61%**

Berikut merupakan grafik yang menunjukkan perbandingan *F-Measure* eksperimen 5.7 dengan *baseline* dalam bentuk histogram.



Gambar 5.4: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 2.2

5.5.2.2 Analisis

Dari tabel dan grafik 5.4, 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 *tratment* 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 *precission* tertinggi, yaitu 62.86%. Grafik **??** menunjukkan perbandingan *precision*, *recall* dan *f-measure* untuk masing-masing entitas.

Dari analisis yang penulis lakukan terhadap korpus dan kamus kesehatan, terdapat beberapa entitas pada korpus yang tidak terdapat pada kamus sehingga mengakibatkan kenaikan hasil tidak besar. Hal ini karena terdapat beberapa penyebab, yaitu:

1. Ada entitas yang merupakan kombinasi atau gabungan dari entitas lain yang dihubungkan dengan kata penghubung. Hal ini banyak penulis temukan pada entitas *treatment* dan *symptom*. Contoh dari kasus ini yaitu:

    - nyeri hebat dibagian ulu hati dan pinggang belakang (gabungan dari "nyeri hebat dibagian ulu hati" dan "nyeri hebat pinggang belakang")
    - kondisi fases berampas , kuning , sedikit berlendir (gabungan dari "kondisi fases berampas", "kondisi fases kuning" dan "kondisi fases sedikit berlendir")

- alis atas dan bibir tidak bisa digerakkan (gabungan dari "alis atas tidak bisa digerakkan" dan "bibir tidak bisa digerakkan")

2. Penggunaan kata ganti orang di dalam entitas. Contoh dari kasus ini yaitu:

   - suara **saya** hilang
   - gusi **saya** berdarah
   - pinggang **saya** sakit

3. Kesalahan eja pada entitas atau penggunaan kata yang tidak baku, misalnya:

   - radang paru 2
   - butawarna
   - jrawatan
   - kanker darah setadium 1

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 5.5.3;

5.5.3 Eksperimen 2.3: Fitur Kata, Kamus Kesehatan dan *Stopword*

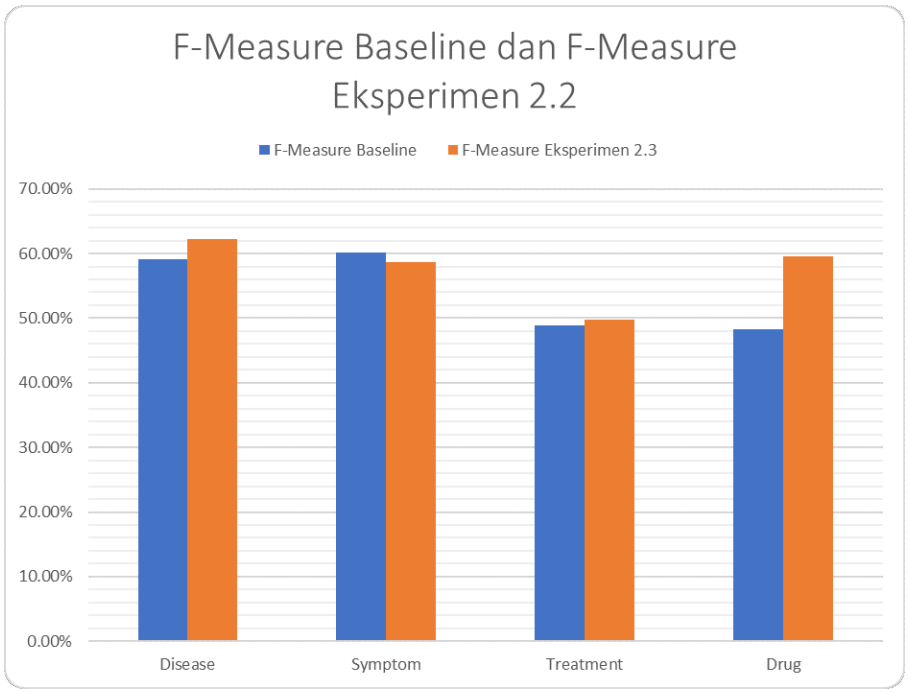
Pada sub-eksperimen ini. penulis mencoba menambahkan informasi lain berupa fitur yang berisi sebuah kata apakah terdapat di dalam kamus *stop word* atau tidak. Penulis berpendapat bahwa dengan adanya informasi *stop word*, adanya kesalahan suatu kata tidak berentitas yang dilabeli sebagai kata berentitas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

5.5.3.1 Hasil Eksperimen

Rangkuman hasil sub-eksperimen ini dapat dilihat di Tabel 5.8 dan Gambar 5.5.

Tabel 5.8: Tabel Hasil Eksperimen 2.3 dibandigkan dengan *Baseline*

Entitas	Baseline (Herwando 2016)			Eksperimen 2.3		
	Precision	*Recall*	*F-Measure*	*Precision*	*Recall*	*F-Measure*
Disease	63.68%	55.45%	59.13%	65.97%	59.81%	62.28%
Symptom	61.43%	59.21%	60.18%	63.08%	55.20%	58.68%
Treatment	53.10%	45.97%	48.82%	54.73%	46.27%	49.69%
Drug	58.99%	44.46%	48.23%	61.88%	58.99%	59.57%
Overall	**59.30%**	**51.27%**	**54.09%**	**61.42%**	**55.07%**	**57.56%**



Gambar 5.5: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 2.3

5.5.3.2 Analisis

Dari Tabel 5.8 dan Gambar 5.5 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 *precission* dan *f-measure* teteapi 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 5.5.4.

5.5.4 Eksperimen 2.4: Fitur Kata, Kamus Kesehatan, *Stopword* dan POS-Tag

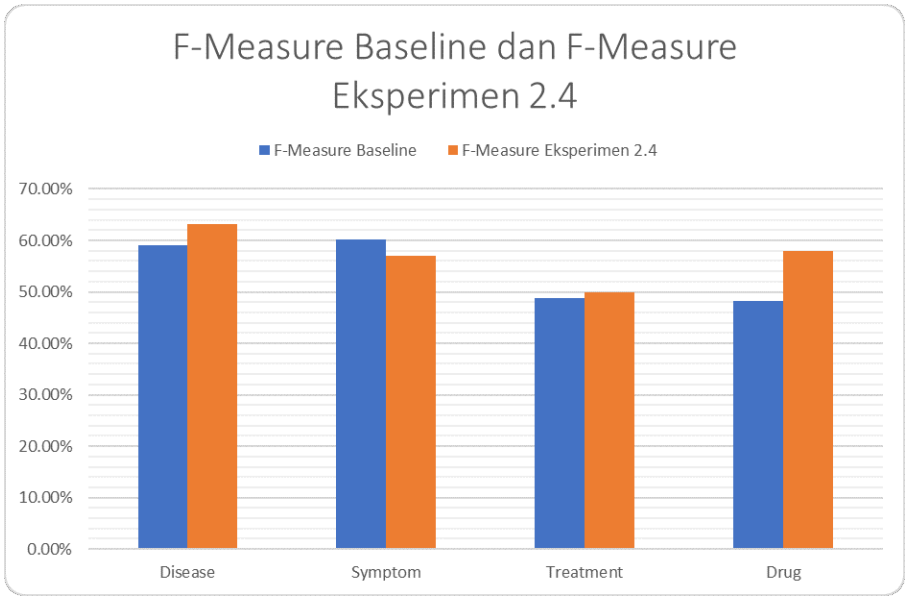
Pada sub-eksperimen ini, penulis menambahkan informasi baru pada *resource* yang akan digunakan untuk *training* model yang berupa fitur POS-Tag. Sebelumnya fitur ini telah digunakan pada penelitian Abacha dan Zweigenbaum (2011) pada dokumen berbahasa Inggris dan memberikan kontribusi meningkatkan akurasi 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.

5.5.4.1 Hasil Eksperimen

Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.9 dan Gambar 5.6.

Tabel 5.9: Tabel Hasil Eksperimen 2.4 dibandigkan dengan *Baseline*

Entitas	Baseline (Herwando 2016)			Eksperimen 2.4		
	Precision	*Recall*	*F-Measure*	*Precision*	*Recall*	*F-Measure*
Disease	63.68%	55.45%	59.13%	69.10%	58.67%	63.22%
Symptom	61.43%	59.21%	60.18%	61.09%	54.43%	57.00%
Treatment	53.10%	45.97%	48.82%	59.73%	44.10%	49.87%
Drug	58.99%	44.46%	48.23%	62.00%	55.74%	57.87%
Overall	**59.30%**	**51.27%**	**54.09%**	**62.98%**	**53.24%**	**56.99%**



Gambar 5.6: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 2.4

5.5.4.2 Analisis

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.

Hasil dari penggunaan fitur ini kurang baik, karena beberapa hal, yaitu:

1. Tag yang dihasilkan tidak konsisten. Pada beberapa entitas, terkadang suatu kata mendapatkan tag "A", namun di entitas yang lain untuk kata yang sama mendapatkan tag yang berbeda. Contoh dari kasus ini yaitu:
    - "antibiotik" memiliki tag "vb", sedangkan pada entitas lain "antibiotik" memiliki tag "NN"
    - "sakit kepala" memiliki beberapa tag pada entitas berbeda, yaitu "CD NN", "JJ NN", dan "NN NN"
    - "nyeri" memiliki beberapa tag pada entitas berbeda, yaitu "NN", "VB", "IN", "WH", dan "IN".
2. Tidak ada perbedaan tag antara kata berentitas dengan tidak, misalnya nama orang mendapatkan tag "NN" (intan_NN lusia_NN), namun nama penyakit juga mendapatkan tag "NN" (Kanker_NN Otak_NN).
3. Model POS-Tagger yang digunakan merupakan model untuk kalimat dengan topik umum, tidak dikhususkan pada topik kesehatan.

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 5.5.5.

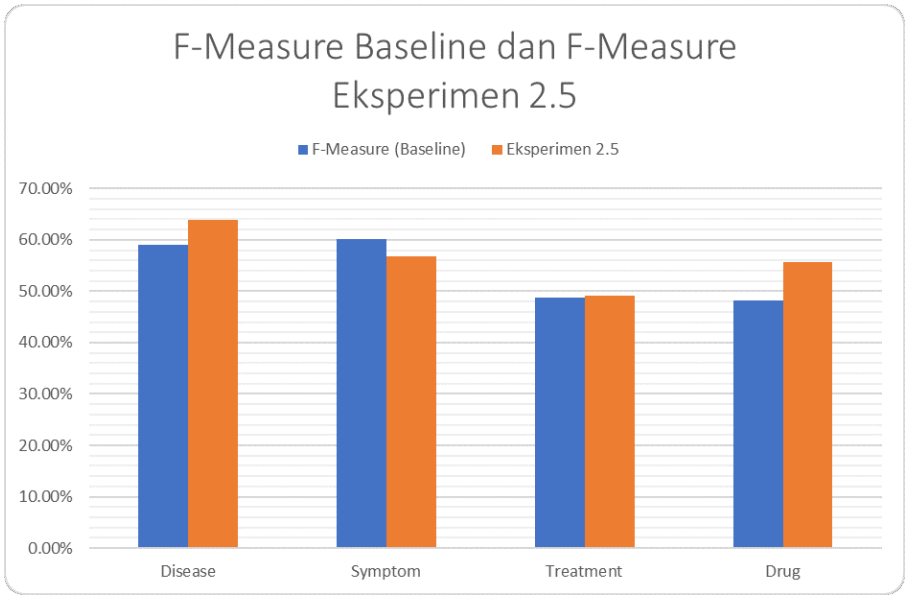
5.5.5 Eksperimen 2.5: Fitur Kata, Kamus Kesehatan, *Stopword*, POS-Tag dan Frasa Kata

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 5.9 dan Gambar 5.6.

Tabel 5.10: Tabel Hasil Eksperimen 2.5 dibandigkan dengan *Baseline*

Entitas	Baseline (Herwando 2016)			Eksperimen 2.5		
	Precision	*Recall*	*F-Measure*	*Precision*	*Recall*	*F-Measure*
Disease	63.68%	55.45%	59.13%	67.49%	61.56%	63.81%
Symptom	61.43%	59.21%	60.18%	62.89%	52.27%	56.72%
Treatment	53.10%	45.97%	48.82%	54.87%	44.92%	49.06%
Drug	58.99%	44.46%	48.23%	59.77%	53.37%	55.66%
Overall	**59.30%**	**51.27%**	**54.09%**	**61.26%**	**53.03%**	**56.31%**



Gambar 5.7: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 2.5

Dari tabel dan grafik di atas, entitas *drug* mengalami penurunan untuk nilai *precission, recall* dan *f-measure*. Selain itu, entitas *disease* mengalami penurunan pada nilai *precission* 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 korpus yang penulis miliki, berikut merupakan informasi statistik dari penggunaan fitur frasa kata kerja:

1. Untuk entitas *disease*, sebanyak 442 entitas merupakan frasa kata benda, 31 entitas merupakan bagian dari frasa kata benda dan 583 entitas bukan merupakan frasa.

2. Untuk entitas *symptom*, sebanyak 486 entitas merupakan frasa kata benda, 194 entitas merupakan bagian dari frasa kata benda dan 626 entitas bukan merupakan frasa.

3. Untuk entitas *treatment*, sebanyak 338 entitas merupakan frasa kata benda, 76 entitas merupakan bagian dari frasa kata benda dan 401 entitas bukan merupakan frasa.

4. Untuk entitas *drug*, sebanyak 152 entitas merupakan frasa kata benda, 8 entitas merupakan bagian dari frasa kata benda dan 194 entitas bukan merupakan frasa.

Sedangkan berikut merupakan informasi statistik dari penggunaan fitur frasa kata benda:

1. Untuk entitas *disease*, sebanyak 943 entitas merupakan frasa kata benda, 43 entitas merupakan bagian dari frasa kata benda dan 70 entitas bukan merupakan frasa.

2. Untuk entitas *symptom*, sebanyak 842 entitas merupakan frasa kata benda, 363 entitas merupakan bagian dari frasa kata benda dan 101 entitas bukan merupakan frasa.

3. Untuk entitas *treatment*, sebanyak 561 entitas merupakan frasa kata benda, 201 entitas merupakan bagian dari frasa kata benda dan 53 entitas bukan merupakan frasa.

4. Untuk entitas *drug*, sebanyak 318 entitas merupakan frasa kata benda, 14 entitas merupakan bagian dari frasa kata benda dan 22 entitas bukan merupakan frasa.

Dari 2 informasi di atas, dapat diambil informasi bahwa sebagian besar entitas *disease* dan *drug* merupakan frasa kata benda dan entitas *symptom* dan *treatment* merupakan frasa kata kerja. Hal ini menjadi seharusnya menjadi informasi pembeda dengan kata yang bukan merupakan entitas. Namun apabila dilihat dari hasil eksperimen ini, performa penggunaan fitur ini tidak terlalu bagus atau bahkan turun di entitas *disease* dan *symptom* tersebut. Penulis berpendapat hal ini terjadi karena penggunaan fitur ini sudah cukup mewakili informasi fitur POS-Tag, akrena untuk menentukan suatu kata atau kumpulan kata merupakan frasa adalah dengan menggunakan POS-Tag. Selain itu, pada fitur POS-Tag, tidak ada perbedaan antara kata yang merupakan frasa maupun kata yang bukan frasa. Padahal, mayoritas entitas seperti yang telah dijelaskan di atas merupakan frasa. Oleh karena itu, pada sub-eksperimen 5.5.6, penulis menghilangkan fitur POS-Tag dan tetap mempertahankan fitur frasa kata untuk mengetahui hal tersebut.

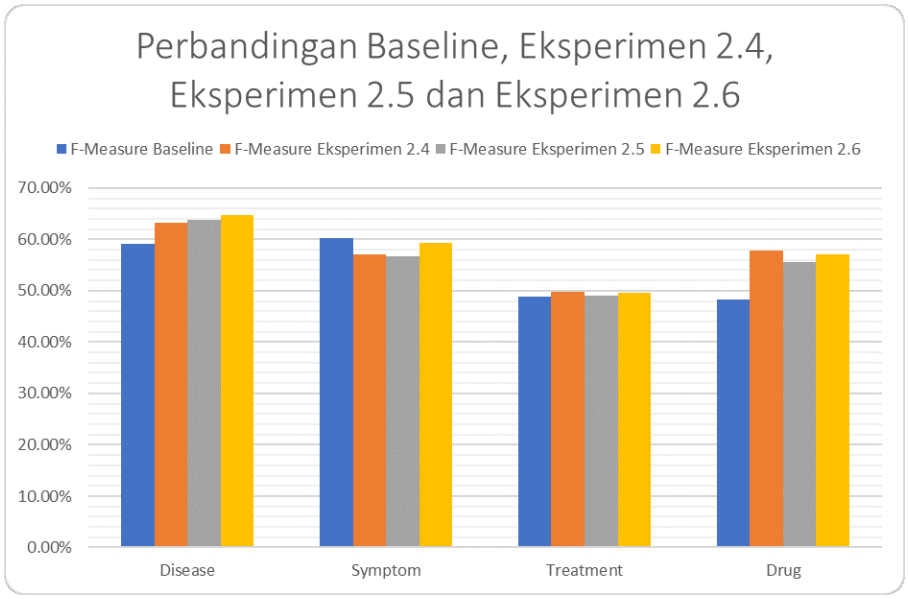
5.5.6 Eksperimen 2.6: Fitur Kata, Kamus Kesehatan, *Stopword* dan Frasa Kata

Pada sub-eksperimen ini penulis menghilangkan fitur POS-Tag berdasarkan hasil dan analisis pada sub-eksperimen 5.5.5. Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.11 dan Gambar 5.8.

5.5.6.1 Hasil Eksperimen

Tabel 5.11: Tabel Hasil Eksperimen 2.6 dibandigkan dengan *Baseline*

Entitas	Baseline (Herwando 2016)			Eksperimen 2.6		
	Precision	*Recall*	*F-Measure*	*Precision*	*Recall*	*F-Measure*
Disease	63.68%	55.45%	59.13%	68.67%	61.80%	64.78%
Symptom	61.43%	59.21%	60.18%	63.79%	56.10%	59.23%
Treatment	53.10%	45.97%	48.82%	54.47%	46.72%	49.58%
Drug	58.99%	44.46%	48.23%	60.08%	56.70%	57.00%
Overall	**59.30%**	**51.27%**	**54.09%**	**61.75%**	**55.33%**	**57.65%**



Gambar 5.8: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 2.4, 2.5 dan 2.6

5.5.6.2 Analisis

Dari Tabel 5.11 dan Gambar 5.8, 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 5.5.5, penggabungan fitur POS-Tag dan frasa akan memberikan hasil yang lebih rendah. Oleh karena itu, sebaiknya fitur POS-Tag tidak digabung dengan fitur frasa.

Untuk mempermudah dalam membandingkan hasil eksperimen 2.4, 2.5 dan 2.6, penulis menyajikan grafik 5.8 untuk membandingkan nilai *F-Measure* pada masing-masing eksperimen. Dapat dilihat bahwa apabila fitur POS-Tag dan Frasa digunakan secara bersama-sama, hasil yang diberikan lebih rendah apabila kedua

fitur tersebut dipisah. Namun, apabila dengan menggunakan fitur POS-Tag tanpa Frasa, hasil pada entitas *treatment* dan *drug* lebih bagus. Sedangkan apabila dengan menggunakan fitur Frasa tanpa POS-Tag, hasil pada entitas *disease* dan *symptom* lebih baik. Oleh karena itu penulis memilih salah satu dari kedua fitur tersebut. Apabila dilihat dari rata-rata *F-Measure*, penggunaan fitur Frasa tanpa POS-Tag memberikan hasil yang paling tinggi (57.65%) apabila dibandingkan dengan penggunaan fitur POS-Tag tanpa Frasa (56.99%). Oleh karena itu, penulis mempertahankan fitur Frasa dan tidak menggunakan fitur POS-tag pada eksperimen selanjutnya.

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* pada entitas *symptom*. Oleh karena itu penulis mencoba fitur yang lain, yaitu fitur Kata Sebelum. Untuk penjelasan lebih lanjut akan dibahas pada sub-eksperimen 5.5.7.

5.5.7 Eksperimen 2.7: Fitur Kata, Kamus Kesehatan, *Stopword*, Frasa Kata dan Kata Sebelum

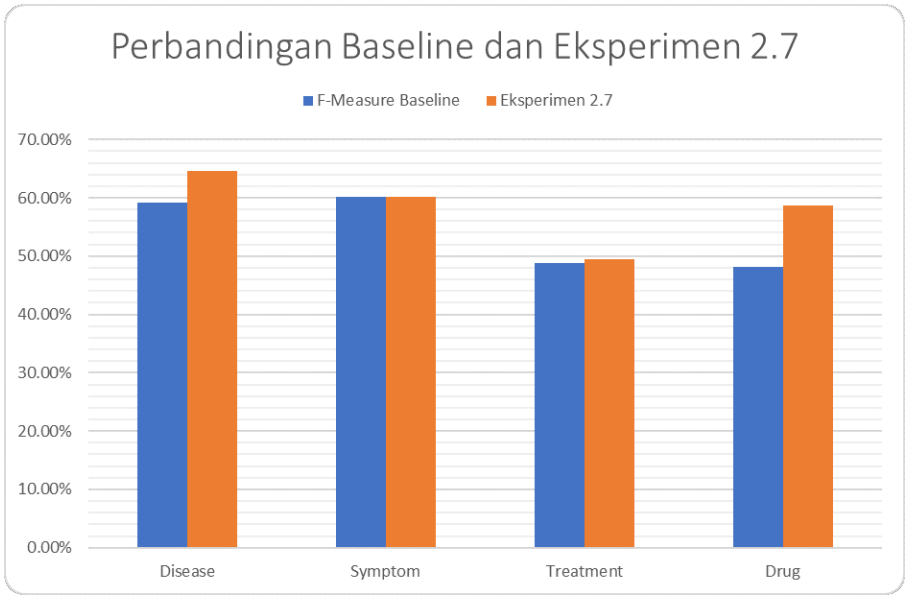
Pada sub-eksperimen ini penulis menambahkan fitur baru yaitu fitur 1 kata sebelum. Fitur ini digunakan pada penelitian Herwando (2016) yang juga berkontribusi memberikan hasil terbaik pada penelitiannya. Menurut penulis, ada beberapa entitas yang akan lebih mudah diketahui apabila diketahui kata sebelumnya. 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 penulis mencoba menambahkan fitur tersebut.

5.5.7.1 Hasil Eksperimen

Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.12 dan Gambar 5.9.

Tabel 5.12: Tabel Hasil Eksperimen 2.7 dibandigkan dengan *Baseline*

Entitas	Baseline (Herwando 2016)			Eksperimen 2.7		
	Precision	*Recall*	*F-Measure*	*Precision*	*Recall*	*F-Measure*
Disease	63.68%	55.45%	59.13%	69.49%	61.60%	64.68%
Symptom	61.43%	59.21%	60.18%	64.78%	57.15%	60.23%
Treatment	53.10%	45.97%	48.82%	56.58%	44.71%	49.54%
Drug	58.99%	44.46%	48.23%	62.22%	57.28%	58.76%
Overall	**59.30%**	**51.27%**	**54.09%**	**63.27%**	**55.19%**	**58.30%**



Gambar 5.9: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 2.7

5.5.7.2 Analisis

Melihat pada Tabel 5.12 dan Gambar 5.9, dapat diketahui bahwa entitas *disease* dan *treatment* mengalami kenaikan pada nilai *precission*, tetapi mengalami penurunan pada nilai *recall* dan *f-measure*. Sedangkan entitas *symptom* dan *drug* mengalami kenaikan pada nilai *precision, recall,* dan *f-measure*.

Seperti pada penelitian Herwando (2016), fitur ini berhasil meningkatkan performa dari beberapa entitas, karena fitur ini memberikan informasi tambahan kata sebelumnya, misalnya:

- "penyakit", "penderita", "mengalami" dan "mengalami" dapat memberikan informasi mengenai entitas *disease*
- "mengandung", "minum", "pemberian", "obat", "menggunakan" dapat memberikan informasi mengenai entitas *drug*

- "dengan", "melakukan", "dilakukan" dapat memberikan informasi mengenai entitas *treatment*
- "mengalami", "disertai", "sering", "keluhan", "penyebab" dapat memberikan informasi mengenai entitas *symptom*

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 5.5.8.

5.5.8 Eksperimen 2.8: Fitur Kata, Kamus Kesehatan, *Stopword*, Frasa Kata, Kata Sebelum dan Kata Sesudah

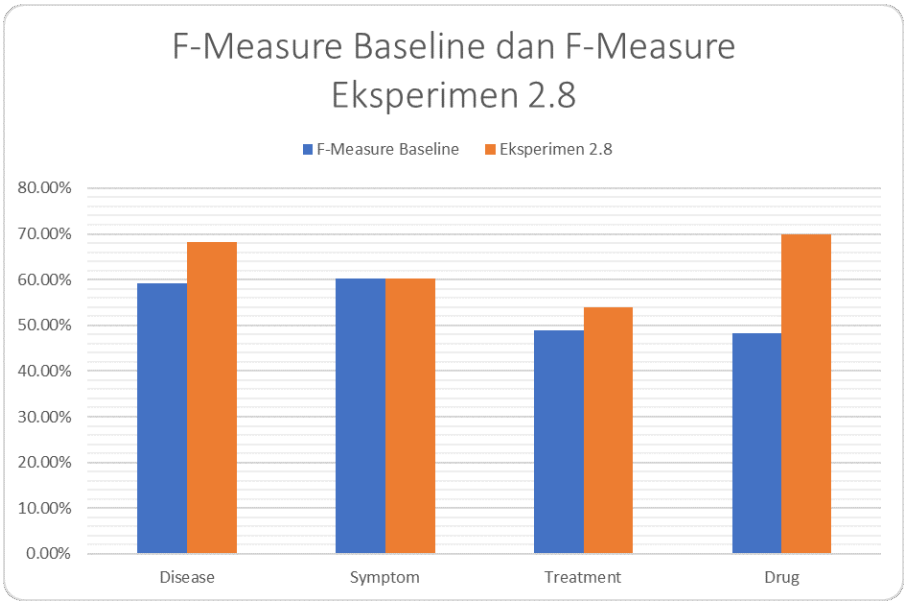
Pada sub-eksperimen ini penulis menambahkan fitur lain yaitu fitur 1 Kata Setelah. Hal ini karena ada beberapa kasus yang mana apabila suatu kata merupakan sebuah entitas, akan lebih mudah dikenali apabila melihat kata atau konteks setelahnya. Sama seperti contoh pada fitur 1 kata sebelum, misal diberikan 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.

5.5.8.1 Hasil Eksperimen

Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.13 dan Gambar 5.10.

Tabel 5.13: Tabel Hasil Eksperimen 2.8 dibandigkan dengan *Baseline*

Entitas	Baseline (Herwando 2016)			Eksperimen 2.7		
	Precision	*Recall*	*F-Measure*	*Precision*	*Recall*	*F-Measure*
Disease	63.68%	55.45%	59.13%	70.68%	66.18%	68.17%
Symptom	61.43%	59.21%	60.18%	64.16%	59.55%	60.23%
Treatment	53.10%	45.97%	48.82%	61.02%	51.13%	54.03%
Drug	58.99%	44.46%	48.23%	70.85.%	70.33%	69.82%
Overall	**59.30%**	**51.27%**	**54.09%**	**65.29%**	**61.80%**	**63.06%**



Gambar 5.10: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 2.8

5.5.8.2 Analisis

Melihat pada Tabel 5.13 dan Gambar 5.10, 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 5.6.

5.6 Skenario 3: Skenario Pengujian Arsitektur RNNs

Pada eksperimen ini, penulis mencoba dua buah arsitektur RNNs yang telah penulis usulkan pada Bab 3 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.

5.6.1 Eksperimen 3.1: Menguji Arsitektur LSTMs 1 Layer

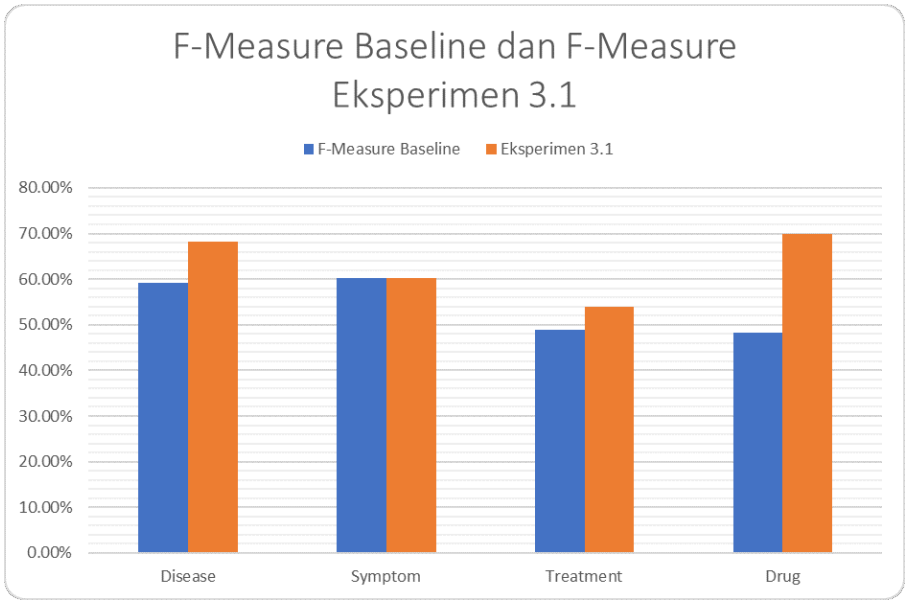
Pada sub-eksperimen ini, penulis menggunakan struktur RNNs yang mana semua fitur digabung menjadi satu dalam sebuah *timestep*. Artinya fitur-fitur yang berbeda tersebut akan digabung atau di-*concat* menjadi sebuah vektor yang akan menjadi

input bagi LSTMs ini. LSTMs inilah yang digunakan pada eksperimen pertama, sehingga hasilnya sama dengan sub-eksperimen 5.5.8.

5.6.1.1 Hasil Eksperimen

Tabel 5.14: Tabel Hasil Eksperimen 3.1 dibandigkan dengan *Baseline*

Entitas	Baseline (Herwando 2016)			Eksperimen 2.7		
	Precision	*Recall*	*F-Measure*	*Precision*	*Recall*	*F-Measure*
Disease	63.68%	55.45%	59.13%	70.68%	66.18%	68.17%
Symptom	61.43%	59.21%	60.18%	64.16%	59.55%	60.23%
Treatment	53.10%	45.97%	48.82%	61.02%	51.13%	54.03%
Drug	58.99%	44.46%	48.23%	70.85.%	70.33%	69.82%
Overall	**59.30%**	**51.27%**	**54.09%**	**65.29%**	**61.80%**	**63.06%**



Gambar 5.11: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 3.1

5.6.1.2 Analisis

Pada eksperimen ini, hasil yang sudah lebih baik apabila dibandingkan dengan hasil yang dicapai Herwando (2016) di entitas. Namun, dari eksperimen sebelumnya, terdapat 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-masng kelompok fitur yang berbeda dipisahkan dan menjadi *input* bagi masing-masing LSTMs. Untuk penjelasan eksperimen ini akan dijelaskan pada sub-eksperimen 5.6.2

5.6.2 Eksperimen 3.2: Menguji Arsitektur LSTMs 2 Layer Multi-Input

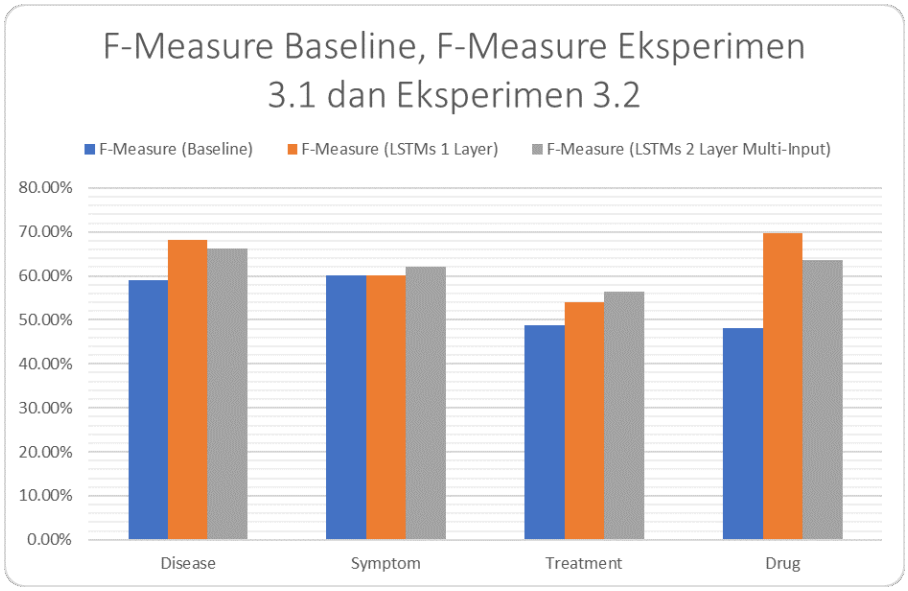
Pada sub-eksperimen sebelumnya, fitur-fitur yang berbeda digabung menjadi satu, sehingga ada kemungkinan hilangnya informasi dari fitur tersebut. Oleh karena itu, penulis mengusulkan adanya layer tambahan setelah masing-masing fitur tersebut masuk ke dalam model. Penulis mengusulkan bahwa masing-masing kelompok fitur menjadi *input* LSTMs secara terpisah. Setelah masuk di RNNs, *output* dari masing-masing LSTMs tersebut di-*merge* ke dalam sebuah layer, lalu masuk kembali ke LSTMs 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.

5.6.2.1 Hasil Eksperimen

Rangkuman hasil sub-eksperimen ini dapat dilihat pada Tabel 5.15 dan Gambar 5.12.

Tabel 5.15: Tabel Hasil Eksperimen 3.2 dibandigkan dengan *Baseline*

	Precision	*Recall*	*F-Measure*
Disease	67.47%	67.19%	66.31%
Symptom	64.90%	60.63%	62.13%
Treatment	63.92%	53.13%	56.51%
Drug	66.39%	62.33%	63.61%
Overall	**65.67%**	**60.82%**	**62.14%**



Gambar 5.12: Histogram Perbandingan *F-measure Baseline* dengan Eksperimen 3.1, dan 3.2

5.6.2.2 Analisis

Pada eksperimen ini, hasil yang sudah lebih baik apabila dibandingkan dengan hasil yang dicapai Herwando (2016) di masing-masing entitas dan lebih baik dibandingkan eksperimem 5.6.1 pada identifikasi entitas *symptom* dan *treatment*.

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. Data yang digunakan bersumber dari *post* pada forum kesehatan *online* yang telah didapatkan oleh Herwando (2016) dan penulis. 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 LSTMs dengan 1 layer dan LSTMs dengan 2 layer dengan multi-*input*.

Terkait dengan rumusan masalah pertama, setelah dilakukan penelitian secara garis besar didapatkan kesimpulan bahwa model RNNs yang dihasilkan mampu memberikan performa yang lebih baik dibandingkan dengan model CRF (*baseline*) pada penelitian Herwando (2016). Dari penggunaan fitur kata itu sendiri saja, model RNNs sudah memberikan performa yang lebih baik dengan nilai *F-Measure* dan *recall* yang lebih tinggi.

Terkait dengan rumusan masalah kedua, didapatkan kesimpulan lain bahwa fitur kata itu sendiri, kamus kesehatan, *stop word*, frasa Kata, 1 kata sebelum dan 1 kata sesudah memberikan hasil yang terbaik, yaitu dengan rata-rata *f-measure* 63.06% (*disease* 68.17%, *symptom* 61.42%, *treatment* 68.17% dan *drug* 68.17%).

Dua arsitektur yang diusulkan memiliki kelebihan masing-masing. Untuk arsitektur LSTMs dengan 1 layer, *f-measure* sama dengan percobaan untuk mendapatkan fitur terbaik, karena eksperimen tersebut menggunakn arsitektur LSTMs yang sama. Sedangkan untuk arsitektur kedua (LSTMs 2 layer), rata-rata *f-measure* yang didapatkan adalah 62.14%. LSTMs pertama memiliki nilai *f-measure* pada entitas *disease* dan *drug* yang lebih bagus, yaitu masing-masing 68.17% dan 69.82%. Sedangkan LSTMs kedua memiliki nilai *f-measure* pada entitas *symptom* 62.13% dan *treatment* 56.51%. Namun, apabila dilihat dari waktu komputasi, LSTMs pertama lebih baik dibandingkan LSTMs kedua.

LSTMs pertama tidak bisa dikatakan lebih baik dibandingkan LSTMs kedua dan begitu pula sebaliknya, karena hasil yang diperoleh mengatakan bahwa masing-masing arsitektur memiliki hasil yang lebih baik di beberapa macam entitas.

Namun, arsitektur ini mampu memberikan hasil yang lebih baik dari hasil penelitian Herwando (2016). Hal ini akan menarik apabila *resource* semakin diperbesar, apakah tetap LSTMs 1 layer lebih baik, karena LSTMs 2 layer memiliki parameter lebih banyak, sehingga mampu menyimpan informasi yang lebih besar.

6.2 Saran

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

1. Penelitian ini hanya menggunakan 309 *post* forum kesehatan *online* sehingga perlu penambahan data *training* dan *testing* mengingat *deep learning* membutuhkan data yang besar dalam melakukan *training* untuk mendapatkan model yang baik.
2. Perlu dibuat korpus dengan jumlah masing-masing entitas yang seimbang, sehingga hasil yang diberikan tidak bias.
3. Sebaiknya, pelabelan dokumen secara manual melibatkan pihak yang ahli di bidangnya (dalam hal ini dokter, perawat, apoteker, atau mahasiswa di bidang kesehatan) supaya label yang diberikan lebih tepat.
4. Sama seperti pada penelitian Herwando (2016), sebaiknya dibuat model POS-Tagger yang khusus di bidang kesehatan, sehingga pemberian tag pada dokumen kesehatan lebih tepat.

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.

Bird, S., Klein, E., dan Loper, E. (2009). Nltk book.

Chollet, F. (2015). Keras. https://github.com/fchollet/keras.

Dinakaramani, A., Rashel, 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.

Goodfellow, I., Bengio, Y., dan Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

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. Bachelor's thesis, Universitas Indonesia, Kampus UI Depok.

Hinton, G. E., Osindero, S., dan Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

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 parallellism 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., Cernockỳ, J., dan Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Mozer, M. C., Jordan, M. I., dan Petsche, T. (1997). *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference*. Mit Press.

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 evolino. *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). Imner indonesian medical named entity recognition. In *Technology, Informatics, Management,*

Engineering, and Environment (TIME-E), 2014 2nd International Conference on, pages 184–188. IEEE.

Toutanova, K. dan Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70. Association for Computational Linguistics.

Turian, J., Ratinov, L., dan Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.