UNIVERSITAS INDONESIA

OPEN DOMAIN INFORMATION EXTRACTION OTOMATIS DARI TEKS BAHASA INDONESIA

TESIS

YOHANES GULTOM
1506706345

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI MAGISTER ILMU KOMPUTER
DEPOK
JUNI 2017

UNIVERSITAS INDONESIA

OPEN DOMAIN INFORMATION EXTRACTION OTOMATIS DARI TEKS BAHASA INDONESIA

TESIS
Diajukan sebagai salah satu syarat untuk memperoleh gelar
Magister Ilmu Komputer

YOHANES GULTOM
1506706345

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI MAGISTER ILMU KOMPUTER
DEPOK
JUNI 2017

ABSTRAK

Nama	:	Yohanes Gultom
Program Studi	:	Magister Ilmu Komputer
Judul	:	Open Domain Information Extraction Otomatis dari Teks Bahasa Indonesia

Banyaknya jumlah dokumen digital yang tersedia saat ini sudah melebihi kapasitas manusia untuk memprosesnya secara manual. Hal ini mendorong munculnya kebutuhan akan metode ekstrasi informasi (*information extraction*) otomatis dari teks atau dokumen digital dari berbagai domain (*open domain*). Sayangnya, sistem *open domain information extraction* (*open IE*) yang ada saat ini hanya berlaku untuk bahasa tertentu saja. Selain itu belum ada sistem *open IE* untuk bahasa Indonesia yang dipublikasikan. Pada penelitian ini Penulismemperkenalkan sebuah sistem untuk mengekstraksi relasi antar entitas dari teks bahasa Indonesia dari berbagai domain. Sistem ini menggunakan sebuah NLP *pipeline*, pembangkit kandidat *triple* (*triple candidates generator*) dan pengembang token (*token expander*) berbasis aturan serta pemilih *triple* berbasis *machine learning*. Setelah melakukan *cross-validation* terhadap empat kandidat model: *logistic regression*, SVM, MLP dan *Random Forest*, Penulismenemukan bahwa *Random Forest* adalah *classifier* yang terbaik untuk dijadikan *triple selector* denan skor F1 0.58 (*precision* 0.62 dan *recall* 0.58). Penyebab utama skor yang masih rendah ini adalah aturan pembangkitan kandidat yang masih sederhana dan cakupan pola *dataset* yang masih rendah.

Kata Kunci:
information extraction, *open domain*, *natural language processing*, bahasa Indonesia

ABSTRACT

Name	:	Yohanes Gultom
Program	:	Magister Ilmu Komputer
Title	:	Automatic Open Domain Information Extraction from Indonesian Text

The vast amount of digital documents, that have surpassed human processing capability, calls for an automatic information extraction method from any text document regardless of their domain. Unfortunately, open domain information extraction (open IE) systems are language-specific and there is no published system for Indonesian language. This paper introduces a system to extract entity relations from Indonesian text in triple format using an NLP pipeline, rule-based candidates generator, token expander and machine-learning-based triple selector. We cross-validate four candidates: logistic regression, SVM, MLP, Random Forest using our dataset to discover that Random Forest is the best classifier for the triple selector achieving 0.58 F1 score (0.62 precision and 0.58 recall). The low score is largely due to the simplistic candidate generation rules and the coverage of dataset.

Keywords:
information extraction, open domain, natural language processing, Indonesian language

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Di masa sekarang ketersediaan dokumen digital berbahasa natural seperti berita, jurnal dan buku elektronik (*e-book*) sudah sangat banyak dan terus meningkat dengan cepat karena didorong oleh meningkatnya pemanfaatan komputer, *smartphone* dan *internet*. Jumlah dokumen digital tersebut telah melampaui batas kemampuan manusia untuk memproses secara manual sehingga menimbulkan kebutuhan akan proses otomatis untuk melakukannya (Banko et al., 2007). Salah satu proses yang dikembangkan adalah *information extraction* (IE) yang secara selektif menyusun dan mengkombinasikan data yang ditemukan di dalam teks atau dokumen menjadi informasi(Cowie and Lehnert, 1996).

Meskipun *IE* sudah mampu manusia untuk memproses dokumen digital dengan lebih efisien, metode yang digunakan umumnya hanya berlaku untuk kelompok dokumen yang homogen atau berada dalam satu domain (*closed-domain*). Hal ini terjadi karena umumnya teknik yang dipakai dibuat sedemikian rupa untuk memanfaatkan pola tertentu pada teks atau dokumen(Cowie and Lehnert, 1996). Sebagai contoh untuk mengekstraksi nama penulis dari berita elektronik, salah satu cara paling mudah adalah mencari nama orang di awal atau akhir dokumen. Cara yang sama tidak bisa digunakan untuk mencari nama penulis dari dokumen lain seperti jurnal karena struktur dokumen yang berbeda. Hal ini mendorong berkembangnya metode lain yang mampu mengekstraksi informasi dari berbagai domain (*open domain*) yang disebut *open domain information extraction* (*open IE*) (Banko et al., 2007).

Tabel 1.1: Perbandingan antara *information extraction* tradisional (IE), *open domain extraction* (open IE) dan *knowledge extraction* (KE)

Aspek	**IE**	**Open IE**	**KE**
Domain	Tertutup	Terbuka	Terbuka
Format	Tergantung domain	Triples	RDF Triples
Ontologi	Tidak tersedia	Opsional	Wajib

Metode *open domain information extraction* (*open IE*) mengekstrak informasi dari dokumen dalam format *tuple* (x, r, y) di mana r adalah relasi antara dua buah

argumen/entitas *x* dan *y* (Etzioni et al., 2011). Format informasi dari tiga nilai tersebut, yang disebut juga *triple*, berlaku umum untuk semua dokumen yang berisi bahasa natural sehingga dapat diterapkan pada dokumen dari berbagai domain. Format *triple* yang digunakan open IE memiliki kemiripan dengan format yang lazim digunakan pada *knowledge extraction* (KE), yaitu Resource Data Format (RDF) specification[1] (Auer et al., 2007; Exner and Nugues, 2014). Namun, open IE umumnya tidak mengikuti seluruh spesifikasi RDF dan tidak memiliki set ontologi tetap. Ringkasan perbandingan antara open IE dengan IE tradisional dan KE ditunjukkan pada Tabel 1.1.

Seiring dengan berkembangnya waktu, beberapa sistem open IE sudah dikembangkan (Schmitz et al., 2012) untuk bahasa Inggris. Bahkan penelitian terkait melaporkan kesuksesan aplikasi open IE untuk *task question answering* (Fader et al., 2011) dan *information retrieval* (Etzioni, 2011). Akan tetapi karena sistem open IE menggunakan satu atau lebih *task natural language processing* (NLP) dan aturan/heuristik yang hanya berlaku untuk bahasa tertentu, maka sistem yang berkembang tidak dapat dipakai untuk memproses teks atau dokumen dalam bahasa lain seperti bahasa Indonesia. Oleh karena itu dalam penelitian ini, Penulis memperkenalkan sistem open IE untuk bahasa Indonesia.

Input
”Sembungan adalah sebuah desa yang terletak di kecamatan Kejajar, kabupaten Wonosobo, Jawa Tengah, Indonesia.”

Output
1. (Sembungan, adalah, desa)
2. (Sembungan, terletak di, kecamatan Kejajar)

Gambar 1.1: Contoh input dan output yang diharapkan dari sistem open IE untuk bahasa Indonesia

Sistem open IE yang Penulis ajukan bertujuan untuk mengekstrak sejumlah *triple* dari satu atau lebih teks bahasa Indonesia seperti contoh pada Gambar 1.1. Sistem ini terdiri dari sebuah NLP *pipeline*, pembangkit kandidat *triple* (*triple candidates generator*), pengembang token (*token expander*) dan sebuah model *machine learning* untuk memilih *triple* (*triple selector*). Untuk melatih model *triple selector* tersebut, Penulis juga membuat dataset berisi 1.611 kandidat *triple* bahasa Indonesia yang valid dan yang tidak valid. Sistem ini diharapkan dapat menjadi referensi

[1]Resource Data Format W3C https://www.w3.org/RDF/

dalam pengembangan open IE untuk bahasa Indonesia dan juga digunakan untuk kebutuhan aplikasi yang lebih kompleks seperti pendeteksian plagiarisme, *question answering* dan *knowledge extraction*.

1.2 Permasalahan

Pada bagian ini akan dijelaskan mengenai definisi permasalahan yang ingin diselesaikan pada penelitian ini serta batasan yang ditetapkan.

1.2.1 Definisi Permasalahan

Permasalahan yang ditemukan dan ingin diselesaikan pada penelitian ini:

1. Bagaimana merancang sistem *open IE* yang cocok untuk bahasa Indonesia?
2. Bagaimana implementasi sistem *open IE* tersebut?

1.2.2 Batasan Permasalahan

Batasan permasalahan pada penelitian ini adalah:

1. Proses dibatasi pada dokumen teks bahasa Indonesia yang setiap barisnya hanya berisi satu kalimat. Praproses yang dibutuhkan untuk menggubah dokumen dari format yang berbeda tidak dibahas di penelitian ini.
2. Penelitian ini hanya berfokus untuk menghasilkan *triple* yang eksplisit secara sintaktik. Contoh *triple* yang eksplisit dari kalimat "*Universitas Indonesia berada di Depok, Jawa Barat, Indonesia*" adalah *(Universitas Indonesia, terletak di, Depok)*. Sedangkan *triple* yang implisit seperti *(Depok, terletak di, Jawa Barat)* belum ditangani pada penelitian ini.
3. Penelitian ini tidak berfokus untuk mencapai kinerja sistem yang sebanding dengan sistem open IE untuk bahasa Inggris pada penelitian terkait.

1.3 Tujuan dan Manfaat

Tujuan dan manfaat dari penelitian ini adalah:

Tujuan

1. Merancang sistem open IE untuk teks bahasa Indonesia.

2. Mengimplementasikan sistem open IE untuk teks bahasa Indonesia.

Manfaat

1. Menghasilkan sistem *open IE* yang dapat digunakan untuk mengekstrak entitas relasi dan argumen/entitas dalam format *triple* dari teks bahasa Indonesia
2. Memberikan acuan untuk pengembangan sistem *open IE* untuk bahasa Indonesia
3. Memberikan kontribusi terhadap perkembangan sumber daya bahasa (*language resources*) Indonesia

1.4 Sistematika Penulisan

Sistematika penulisan laporan adalah sebagai berikut:

- Bab 1 PENDAHULUAN
  Bab ini akan menjelaskan mengenai latar belakang permasalahan, rumusan masalah, tujuan, manfaat dan batasan penelitian.
- Bab 2 TINJAUAN PUSTAKA
  Bab ini akan menjelaskan landasan teori yang digunakan pada penelitian ini serta memaparkan kajian pustaka terhadap penelitian-penelitian terkait.
- Bab 3 METODE PENELITIAN
  Bab ini akan menjelaskan mengenai tahapan, rancangan & implementasi sistem, pengumpulan & pengolahan data dan teknik evaluasi yang digunakan pada penelitian ini.
- Bab 4 HASIL DAN ANALISIS
  Bab ini akan menjelaskan tentang hasil eksperimen dan analisis hasil eksperimen.
- Bab 5 PENUTUP
  Bab ini akan menjelaskan tentang kesimpulan dari penelitian yang telah dilakukan dan saran untuk penelitian berikutnya.

BAB 2
TINJAUAN PUSTAKA

Pada bab ini dijelaskan mengenai penelitian terkait dan berbagai dasar teori yang menunjang penelitian ini.

2.1 Penelitian Terkait

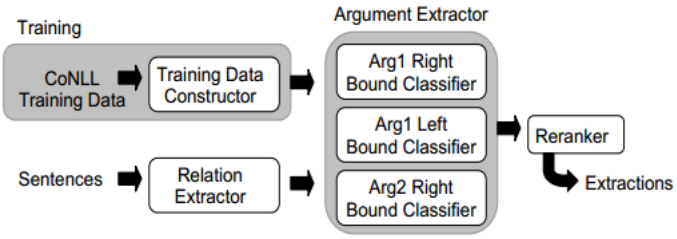
Sejak pertama kali diperkenalkan pada tahun 2007 (Banko et al., 2007), sudah ada beberapa penelitian mengenai *open IE* untuk bahasa Inggris yang dipublikasikan. Sistem *open IE* yang pertama diperkenalkan adalah TEXTRUNNER (Banko et al., 2007). Sistem ini kemudian dikembangkan oleh sistem-sistem dari penelitian berikutnya yaitu (secara berurutan) REVERB (Fader et al., 2011), R2A2 (Etzioni et al., 2011) dan kemudian OLLIE (Schmitz et al., 2012) . Selain itu, salah satu penelitian terbaru juga memperkenalkan sistem *open IE* baru, STANFORD OPEN IE, yang berhasil mengungguli kinerja OLLIE dalam TAC-KBP 2013 *Slot Filling task* (Angeli et al., 2015).

Sistem *open IE* yang pertama diperkenalkan adalah TEXTRUNNER. Sistem ini didesain untuk mengekstrak informasi secara efisien dari halaman-halaman *web* di internet yang jumlahnya sangat besar dan memiliki domain yang berbeda-beda (Banko et al., 2007). Informasi yang diekstrak merupakan *tuple* $t = (e_i, r_{i,j}, e_j)$ di mana $r_{i,j}$ adalah relasi antara entitas e_i dan e_j dalam sebuah kalimat. TEXTRUNNER terdiri dari tiga modul utama (Banko et al., 2007) yaitu: (1) *Self-Supervised Learner*, modul yang melatih sebuah *naive bayes classifier* (NBC) untuk mengenali kandidat *triple* yang valid tanpa memerlukan campur tangan manusia (*self-supervised*), (2) *Single-Pass Extractor*, modul yang mengekstrak sejumlah kandidat *triple* dari setiap kalimat dan menyimpan kandidat yang dianggap valid oleh *classifier*, dan (3) *Redundancy-based Assessor*, modul yang menghitung probabilitas kemunculan *triple* dalam satu dokumen. Sistem ini mampu mengekstrak informasi per kalimat dengan akurasi rata-rata 88% dan mampu memproses 9 juta halaman *web* dalam 68 *CPU hours* (Banko et al., 2007).

REVERB adalah sistem *open IE* yang dikembangkan untuk memperbaiki dua masalah pada pendahulunya, TEXTRUNNER. Masalah yang ingin diselesaikan oleh REVERB adalah inkoherensi hasil ekstraksi *incoherent extractions* dan hasil ekstraksi yang tidak informatif *uninformative extractions* (Fader et al., 2011). Un-

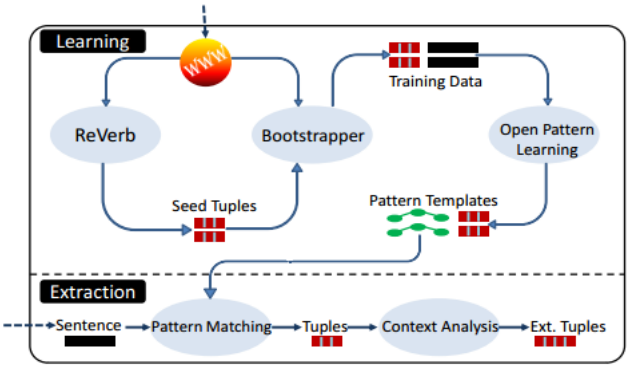
tuk mengekstrak *triple* $t = (e_i, r_{i,j}, e_j)$, sistem ini menggunakan dua algoritma utama, yaitu (1) *Relation Extraction*, algoritma yang mengekstrak relasi $r_{i,j}$ menggunakan pembatasan sintaksis dan leksikal yang menyelesaikan dua masalah tersebut, dan (2) *Argument Extraction*, algoritma yang mencari entitas e_i dan e_j yang dihubungkan oleh relasi $r_{i,j}$ menggunakan heuristik. REVERB menerima *input* berupa kalimat yang telah dianotasi POS-nya % potongan frase kata bendanya (NP *chunk*) dan menghasilkan *output* sejumlah *triple*. Dari hasil pengujian yang dilakukan, REVERB mencapai *precision* dan *recall* yang hampir dua kali lebih baik dari TEXTRUNNER (Fader et al., 2011).

Jika REVERB memperbaiki masalah pada ekstraksi relasi, R2A2 berfokus untuk memperbaiki ekstraksi argumen/entitas (Etzioni et al., 2011). Jika REVERB hanya menggunakan aturan atau heuristik untuk mengekstraksi argumen (Fader et al., 2011), R2A2 menggunakan modul berbasis *machine learning*, ARGLEARNER. Modul ini menerima relasi dan kalimat sebagai *input* dan mengembalikan dua buah argumen sebagai *output*. Modul ini menggunakan tiga buah *classifier* berbasiskan REPTREE (Hall et al., 2009) dan *sequence labeling* CRF (McCallum, 2002) untuk mengekstrak argumen dari kalimat melalui proses yang ditunjukkan pada Gambar 2.1 (Etzioni et al., 2011).



Gambar 2.1: Proses pelatihan dan ekstraksi ARGLEARNER

Penelitian berikutnya memperkenalkan OLLIE (*Open Language Learning for Information Extraction*) (Schmitz et al., 2012) yang menjadikan REVERB sebagai salah satu modulnya. OLLIE menggunakan REVERB untuk mencari (*open pattern template*) sebagai panduan untuk mengekstrak triple relasi dan argumen dari kalimat. Selain itu OLLIE juga menambahkan modul untuk melakukan analisis dan penambahan informasi kontekstual pada hasil ekstraksi sehingga presisi lebih tinggi (Schmitz et al., 2012). Proses pelabelan (*labeling*) data latih dan ekstraksi OLLIE ditunjukkan pada Gambar 2.2.



Gambar 2.2: Proses *labeling* dan ekstraksi pada OLLIE

One of the most research proposes new open IE system that replaces the usage of large open patterns in Ollie (Schmitz et al., 2012) with a set of fewer patterns for canonically structured sentences and a classifier that learns to extract self-contained clauses from a sentence (Angeli et al., 2015). This system is implemented in **Stanford OpenIE** which is also integrated in the populer open source suites, Stanford Core NLP.

2.2 *Open Domain Information Extraction*

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2.3 *Natural Language Processing*

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2.3.1 CONLL-U

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

2.3.2 *Part of Speech Tagging*

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

2.3.3 *Named-Entity Recognition*

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi

fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

2.3.4 *Dependency Parsing*

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

2.4 *Supervised Learning*

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2.4.1 *Logistic Regression*

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

2.4.2 *Support Vector Machine*

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

2.4.3 *Multi-Layer Perceptron*

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

2.4.4 *Random Forest*

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

2.4.5 *Cross Validation*

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem.

Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

BAB 3
METODE PENELITIAN

Pada bab ini dijelaskan mengenai tahapan, rancangan & implementasi sistem, pengumpulan & pengolahan data dan teknik evaluasi yang digunakan pada penelitian ini.

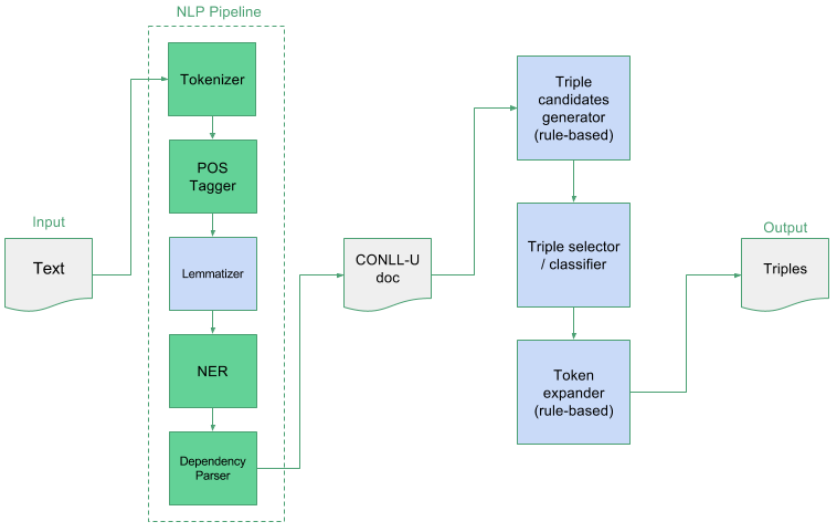
3.1 Tahapan Penelitian

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

3.2 Rancangan dan Implementasi Sistem

As shown in the flowchart Figure 3.1, our system is composed of four main components: **NLP pipeline**, **triple candidate generator**, **triple selector** and **token expander**. Each of them are explained further in following subsections.



Gambar 3.1: Indonesian open domain information extraction flowchart

3.2.1 NLP Pipeline

The NLP pipeline is a series of NLP tasks that annotates one or more sentences and saves them in CONLL-U[1] format, a token-based sentence annotation format containing lemma, POS tag, dependency relation and a slot for additional annotation. The pipeline assumes that each sentence in the input document is separated by new line so preprocessing may be required. The detail of each model the pipeline are described below:

1. Tokenizer

   We use default tokenizer provided by Stanford Core NLP, `PTBTokenizer` (Manning et al., 2014), which mimics Penn Treebank 3 tokenizer[2]. While this tokenizer provides many options to modify its behavior, we stick to default configuration that split sentence by whitelines to get the tokens.

2. Part of Speech Tagger

   We trained default Stanford Core NLP `MaxentTagger` (Toutanova et al., 2003) with Indonesian universal POS tag dataset which we convert from dependency parsing dataset[3]. This POS tagger uses Max Entropy (multi-class logistic regression) classifier which yields **93.68%** token accuracy and

[1]CONLL-U format description http://universaldependencies.org/format.html
[2]Penn Treebank 3 https://catalog.ldc.upenn.edu/LDC99T42
[3]UD Indonesian dataset https://github.com/UniversalDependencies/UD_Indonesian

63.91% sentence accuracy when trained using 5,036 sentences and tested with 559 sentences from the dataset.

3. Lemmatizer

   The lemmatizer used in this pipeline, `IndonesianLemmaAnnotator`, is implemented based on an existing Indonesian rule-based Lemmatizer (Suhartono, 2014) with some improvements:

   - Reimplementation in Java language
   - Usage of in-memory database to speed up dictionary lookup
   - Integration with Stanford Core NLP annotator API for reusability

   This lemmatizer yields **99%** accuracy when tested using dataset of 5,638 token-lemma pairs[4]. We use lemma as one of the features for NER classifier.

4. Named-Entity Recognizer (NER)

   Stanford NLP `CRFClassifier` (Finkel et al., 2005), a linear chain Conditional Random Field (CRF) sequence models, is trained using a dataset containing 3,535 Indonesian sentences with 5 entity class: Person, Organization, Location, Quantity and Time. When tested using 426 sentences, this models achieves 0.86 precision, 0.85 recall and **0.86** F1-score. The dataset itself is a combination between dataset from Faculty of Computer Science, University of Indonesia and a public dataset[5].

5. Dependency Parser

   We relied on Stanford NLP `nndep.DependencyParser` (Chen and Manning, 2014), to annotate dependency relation of each token in the sentence. We train this transition-based neural network model using a Indonesian universal dependencies dataset of 5,036 sentences and 3,093 Indonesian word embedding[6] (vector representation of words). Tested with 559 sentences, this model scores **70%** UAS (Unlabeled Attachment Score) and **46%** LAS (Labeled Attachment Score).

[4]Indonesian Lemmatizer https://github.com/davidchristiandy/lemmatizer

[5]Indonesian NER https://github.com/yusufsyaifudin/indonesia-ner

[6]Indonesian word embedding https://github.com/yohanesgultom/id-openie/blob/master/data/parser-id.embed

The output of the pipeline is a CONLL-U document containing annotated sentence such as Figure 3.2. The document becomes an input for next model, the triple candidate generator which is described in Section 3.2.2. Since the annotations that are directly used by following process are POS tag, named entity and dependency relation, we estimate that the accuracy of this NLP pipeline is **65.30%** which comes from the average of POS tagger sentence accuracy, NER F1-score (in percent) and dependency parser LAS. Additionally, this pipeline is built by extending Stanford Core NLP classes and packaged as single Java program (JAR) to improve reusability.

1	Sembungan	sembung	PROPN	_	_	4	nsubj	_	_
2	adalah	adalah	VERB	_	_	4	cop	_	_
3	sebuah	buah	DET	_	_	4	det	_	_
4	desa	desa	NOUN	_	_	0	root	_	_
5	yang	yang	PRON	_	_	6	nsubj:pass	_	_
6	terletak	letak	VERB	_	_	4	acl	_	_
7	di	di	ADP	_	_	8	case	_	_
8	kecamatan	camat	PROPN	_	_	6	obl	_	LOCATION
9	Kejajar	jajar	PROPN	_	_	8	flat	_	LOCATION
10	,	,	PUNCT	_	_	4	punct	_	_
11	kabupaten	kabupaten	NOUN	_	_	4	appos	_	_
12	Wonosobo	Wonosobo	PROPN	_	_	11	flat	_	LOCATION
13	,	,	PUNCT	_	_	11	punct	_	_
14	Jawa	Jawa	PROPN	_	_	11	appos	_	LOCATION
15	Tengah	tengah	PROPN	_	_	14	amod	_	LOCATION
16	,	,	PUNCT	_	_	11	punct	_	_
17	Indonesia	Indonesia	PROPN	_	_	11	appos	_	_
18	0	0	PUNCT	_	_	4	punct	_	_

Gambar 3.2: Example of CONLL-U sentence annotation format

3.2.2 Triple Candidates Generator

Triple candidates generator is used to extract relation triples candidates from CONLL-U document produced by NLP pipeline. It uses a set of rules listed in Table 3.1 to extract relations (predicates) and arguments (subjects and predicates) from the sentence. The results of triples extraction are not always the positive or valid relation triples so, unlike TextRunner (Banko et al., 2007), we cannot use them directly as training data for triple selector/classifier.

For example, applying the rules to an annotated sentence in Figure 3.2 will generate these 17 triples candidates where only five of them are valid triples (checkmarked):

- (Sembungan, adalah, desa) ✓
- (Sembungan, adalah, terletak)
- (Sembungan, adalah, kecamatan)
- (Sembungan, adalah, kabupaten)

- (Sembungan, adalah, Jawa)
- (Sembungan, adalah, Tengah)
- (Sembungan, adalah, Indonesia)
- (Sembungan, terletak, kecamatan) ✓
- (Sembungan, terletak, kabupaten) ✓
- (Sembungan, terletak, Jawa) ✓
- (Sembungan, terletak, Tengah)
- (Sembungan, terletak, Indonesia) ✓
- (desa, terletak, kecamatan)
- (desa, terletak, kabupaten)
- (desa, terletak, Jawa)
- (desa, terletak, Tengah)
- (desa, terletak, Indonesia)

In order to build a training data for the triple selector, we used triple candidates generator to generate 1,611 triple candidates from 42 sentences. As part of the label step, we manually label **132 positive** and **1,479 negative** triples which we use to train binary classifier as triple selector in the learn step.

During the extraction step, triple candidates generator is used in the system to extract unlabeled candidates from CONLL-U document. These unlabeled triples will be labeled by trained triple selector as described in (referring to flowchart in Figure 3.1).

3.2.3 Triple Selector

Triple selector is a machine learning classifier trained using manually labeled dataset of valid and invalid relation triples. For example, given the input of 17 candidates in Section 3.2.2, the selector will label the five check-marked triples as true and label the rest as false.

We use Random Forest (Breiman, 2001), an ensemble methods that aggregate classification results from multiple decision trees, as the model for the classifier. We

Tabel 3.1: Triple candidate generation rules

Type	Condition
Subject	Token's POS tag is either PROPN, NOUN, PRON or VERB
	Token is not "yang" nor "adalah"
	Token's dependency is neither "compound" nor "name"
	Token's dependency is either "compound" or "name" but separated by more than 2 tokens from its head
Predicate	Token's position is after Subject
	Token's POS tag is either VERB or AUX
Object	Token's position is after Subject and Predicate
	Token's POS tag is either PROPN, NOUN, PRON or VERB
	Token is not "yang" nor "adalah"
	Token's dependency is neither "compound" nor "name"
	Token's dependency is either "compound" or "name" but separated by more than 2 tokens from its head

use the Scikit-Learn[7] implementation of Random Forest with following configuration:

- Decision tree criterion: Gini Impurity
- Minimum number of samples to split tree node: 5 samples
- Maximum features used in each tree: 4 (square root of the number of features)
- Maximum trees depth: 8
- Number of trees: 20
- Class weight: balanced (prediction probability is multiplied by the ratio of training samples)

[7]scikit-learn: machine learning in Python http://scikit-learn.org

Tabel 3.2: Triple selector features

#	Triple Features
1	Subject token's POS tag
2	Subject token's dependency relation
3	Subject token's head POS tag
4	Subject token's named entity
5	Subject token's distance from predicate
6	Subject token's dependency with predicate
7	Predicate token's POS tag
8	Predicate token's dependency relation
9	Predicate token's head POS tag
10	Predicate token's dependents count
11	Object token's POS tag
12	Object token's dependency relation
13	Object token's head POS tag
14	Object token's named entity
15	Object token's dependents count
16	Object token's distance from predicate
17	Object token's dependency with predicate

We discover the configuration by using Grid Search (Wasserman, 2015), an exhaustive search algorithm to find optimal hyper-parameters, to find the best F1 score for Random Forest classifier using dataset described in Section 3.2.2.

We extract 17 features described in Table 3.2 from each triple candidates. These features are based on POS tag, named-entity and dependency relation, instead of shallow syntactic features used by TextRunner or ReVerb (Banko et al., 2007) (Etzioni et al., 2011). Every nominal features are also encoded and normalized along with the whole dataset by removing the mean and scaling to unit variance in order to improve the precision and recall of the classifier.

During the train step, we use the dataset to train triple selector and save the best model as binary file. This model is included in the system to be use during the extraction step.

3.2.4 Token Expander

Instead of using lightweight noun phrase chunker (Banko et al., 2007), our system uses rule-based token expander to extract relation or argument clauses. While having different objective and approach, this token expander works similarly to Clause Selector in Stanford Open IE (Angeli et al., 2015) where the algorithm starts from a token then decides whether to expand to its dependents. Instead of using machine learning model like Clause Selector, it uses simple heuristics based on syntactical features (POS tag, dependency relation and named-entity) described in Table 3.3 and Table 3.4 to determine whether to: (1) expand a token to its dependent, (2) ignore the dependent or (3) remove the token itself. For example, token expander will expand check-marked triples in Section 3.2.2 into:

- (Sembungan, adalah, desa)
- (Sembungan, terletak di, kecamatan Kejajar)
- (Sembungan, terletak di, kabupaten Wonosobo)
- (Sembungan, terletak di, Jawa Tengah)
- (Sembungan, terletak di, Indonesia)

During the label step, token expander is used to make manual annotation process easier. We label a triple candidate as valid only if it makes sense after being expanded to clause. For example, *(Sembungan, terletak, kecamatan)* doesn't seem to make sense before expanded to *(Sembungan, terletak di, kecamatan Kejajar)*.

Tabel 3.3: Token expansion rules for Subject or Object token

#	Condition for Subject or Object Token	Action
1	If dependent's relation to the token is either compound, name or amod	Expand
2	If dependent has same named entity as the token	Expand
3	If dependent and the token are wrapped by quotes or double quotes	Expand
4	If the head is a sentence root	Ignore
5	If dependent's POS tag is CONJ or its form is either , (comma) or / (slash)	Ignore
6	If dependent's POS tag is either VERB or ADP	Ignore
7	If dependent has at least one dependent with ADP POS tag	Ignore
8	If the first or last token in expansion result has CONJ or ADP POS tag	Remove
9	If the first or last index of expansion result is an incomplete parentheses symbol	Remove
10	If the last index of expansion result is yang	Remove
11	Else	Ignore

Tabel 3.4: Token expansion rules for Predicate token

#	Condition for Predicate Token	Action
1	If dependent is tidak	Expand
2	Else	Ignore

BAB 4
HASIL DAN ANALISIS

Pada bab ini dijelaskan hasil evaluasi dan analisis dari penelitian ini.

4.1 Evaluasi

In this research, we report two experiments. The first one shows the performance comparison of four classifiers in selecting valid triples from given candidates. While the second one shows the scalability of our system (using the best classifier) extracting triples from documents (unannotated). Both experiments are run on an Ubuntu 15.04 64-bit, Intel Core i7 5500U (dual cores), DDR3 8 GB RAM, SSD 250 GB machine.

In the first experiment, we chose four classifiers each representing unique characteristics:

1. Linear Logistic RegressionFan et al. (2008) (linear model)
2. Polynomial Support Vector Machine (SVM)Chang and Lin (2011) (nonlinear model)
3. Multi-Layer Perceptron (MLP)Hinton (1989) with 2 hidden layers (20 and 10 ReLUNair and Hinton (2010) neurons)
4. Random ForestWasserman (2015) (ensemble decision trees)

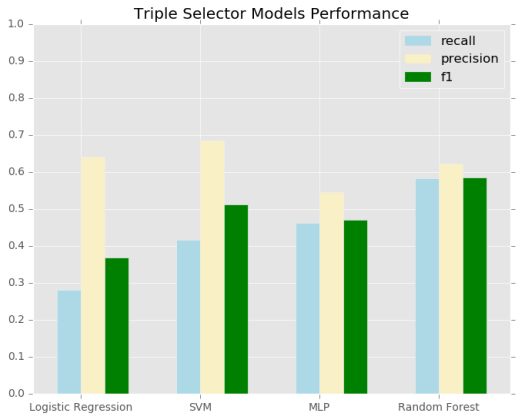
We use the manually annotated triple selector dataset described in Section 3.2.2 to cross-validateKohavi et al. (1995) (k-Fold with $k = 3$) the four classifiers. Since open IE systems requires both precision and recallAngeli et al. (2015), we choose F1 score to determine the best classifier for triple selector. The result of this experiment is shown by Figure 4.1 and Table 4.1 where Random Forest achieves the highest F1 score 0.58.

Tabel 4.1: Triple selector models performance

Model	P	R	F1
Logistic Regression	0.64	0.28	0.36
SVM	**0.68**	0.41	0.51
MLP	0.54	0.46	0.47
Random Forest	0.62	**0.58**	**0.58**

Tabel 4.2: System end-to-end extraction time

Sentences	Triples Extracted	Total Time (s)	Time per Sentence (s)
2	7	6.1	0.800
138	429	11.3	0.082
5,593	19,403	78.6	0.014



Gambar 4.1: Triple selector models performance comparison chart

In the second experiment, we evaluate the performance of our system by extracting triples from three documents with different number of sentences, measuring the total execution time and calculating the average execution time per sentence. The result in Table 4.2 shows that the lowest execution time (or fastest execution time) is 0.014 seconds when processing document of 5,593 sentences.

4.2 Analisis

The first experiment shows that all classifiers are still having problem learning the pattern of triples when cross-validated using $k = 3$ which means two thirds of our dataset is insufficient to cover the patterns in other one third part. The dataset also suffers unbalance 1:11 ratio of positive and negative samples which is caused by lack of efficiency in triple candidates generator. To solve this issue, we plan to annotate more sentences to increase the coverage and improve the efficiency of triple candidates generator. The low performance of linear logistic regression indicates that this problem is not linearly separable. The random forest performs better than other nonlinear models (SVM and MLP) because it is easily tuned to balance the precision and recall by changing the number and the depth of decision trees.

We are also aware that the heuristics used in triple candidates generator and token expander are still limited to explicit pattern. For instance, triple candidate generator can not extract relations *(kecamatan Kejajar, terletak di, Jawa Tengah)* and *(Jawa Tengah, terletak di, Indonesia)* from the sentence in Figure 1.1 yet. In the future research, we plan to improve the model to extract implicit patterns while keeping the number of negative candidates. The token expander is having problem in expanding token to implicitly expected clauses such as *"seorang pelatih sepak bola"* from *"seorang pelatih dan pemain sepak bola"* or *"satu buah torpedo"* from *"satu atau dua buah torpedo"*. We expect there will be more patterns that need to be considered in order to properly expand the token so further research on effective model to achieve this is required. Also, in order to properly evaluate the performance of these components, we need to create test datasets for both triple candidates generator and token expander.

Additionally, through the second experiment, we also find that our system average extraction performance is 0.014 seconds/sentence (for 5,593 sentences document) which is still comparable to TextRunnerBanko et al. (2007). Therefore, in contrast to the argument proposed in the related workBanko et al. (2007)Etzioni et al. (2011), this experiment shows that the heavy linguistic tasks such as dependency parsing doesn't cause performance drawback in big document, assuming the average number of sentences in document do not exceed 5,593.

BAB 5
PENUTUP

Pada bab ini dijelaskan kesimpulan penelitian ini dan saran untuk pengembangan penelitian di masa depan.

5.1 Kesimpulan

This paper introduces an open domain information extraction system for Indonesian text using basic NLP pipelines and combination of heuristics and machine learning models. The system is able to extract meaningful domain-independent relations from Indonesian sentences to be used as document representation or document understanding task. Additionally, the source code and datasets are published openly[1] to improve research reproducibility.

5.2 Saran

In the future, we plan to improve the performance of our system finding better heuristics for triple candidates generator to reduce the negative samples. We also plan adding more training data for triple selector to improve the precision and recall score. We also need to create dataset for triple candidates generator and token expander in order to properly evaluate further improvement of both components. We also consider adding confidence level in the output of every phases (NLP pipelines, candidate generator, triple selector, token expander) and including them as features and/or heuristics may also improve the overall performance of the system.

[1]Paper source code https://github.com/yohanesgultom/id-openie

DAFTAR REFERENSI

Angeli, G., Premkumar, M. J., and Manning, C. D. (2015). Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Cowie, J. and Lehnert, W. (1996). Information extraction. *Communications of the ACM*, 39(1):80–91.

Etzioni, O. (2011). Search needs a shake-up. *Nature*, 476(7358):25–26.

Etzioni, O., Fader, A., Christensen, J., Soderland, S., et al. (2011). Open information extraction: The second generation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Exner, P. and Nugues, P. (2014). Refractive: An open source tool to extract knowledge from syntactic and semantic relations. In *LREC*, pages 2584–2589.

Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial intelligence*, 40(1-3):185–234.

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA.

Manning, C., Grow, T., Grenager, T., Finkel, J., and Bauer, J. (2014). Ptbtokenizer.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Schmitz, M., Bart, R., Soderland, S., Etzioni, O., et al. (2012). Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Suhartono, D. (2014). Lemmatization technique in bahasa: Indonesian. *Journal of Software*, 9(5):1203.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Wasserman, D. (2015). Grid search optimization.

LAMPIRAN

LAMPIRAN 1