UNIVERSITAS INDONESIA

CROSS LANGUAGE WORD SENSE DISAMBIGUATION

SKRIPSI

ADITYA RAMA
1306397854

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
JANUARI 2017

UNIVERSITAS INDONESIA

CROSS LANGUAGE WORD SENSE DISAMBIGUATION

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

ADITYA RAMA
1306397854

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	:	**Aditya Rama**
NPM	:	**1306397854**
Tanda Tangan	:	
Tanggal	:	**13 Januari 2017**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Aditya Rama
NPM	:	1306397854
Program Studi	:	Ilmu Komputer
Judul Skripsi	:	Cross Language Word Sense Disambiguation

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	:	Mirna Adriani, Dra, Ph.D.	(	)
Penguji 1	:	Dra. Mirna Adriani Ph.D.	(	)
Penguji 2	:	Ari Saptawijaya S.Kom., M.Sc., Ph.D.	(	)

Ditetapkan di	:	Depok
Tanggal	:	03 Januari 2017

KATA PENGANTAR

Puji dan syukur penulis ucapkan kepada Allah SWT atas segala rahmat yang telah diberikan, sehingga laporan tugas akhir ini dapat diselesaikan pada waktunya. Tak lupa penulis sanjungkan shalawat serta salam kepada junjungan besar, Nabi Muhammad SAW. Penulis menyadari bahwa laporan ini dapat diselesaikan berkat dukungan beberapa pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada :

1. Kedua Orang Tua penulis
2. Kakak penulis
3. Nadiarani
4. Ilham Kurniawan dan Firza Pratama

Depok, Desember 2016

Aditya Rama

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama	:	Aditya Rama
NPM	:	1306397854
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:

Cross Language Word Sense Disambiguation

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 : 13 Januari 2017

Yang menyatakan

(Aditya Rama)

ABSTRAK

Nama	:	Aditya Rama
Program Studi	:	Ilmu Komputer
Judul	:	Cross Language Word Sense Disambiguation

Textual Entailment adalah penelitian di bidang NLP yang bertujuan untuk mengidentifikasikan apakah terdapat hubungan *entailment* di antara dua buah teks. Penelitian *Textual Entailment* sudah dikembangkan dalam berbagai bahasa, namun *Textual Entailment* untuk Bahasa Indonesia masih sangat minim. Penelitian ini ditujukan untuk mengembangkan korpus *Textual Entailment* Bahasa Indonesia secara otomatis menggunakan metode Co-training, sebuah metode *semi-supervised learning* yang pernah digunakan pada pengembangan korpus *Textual Entailment* Bahasa Inggris. Sumber data yang digunakan untuk Co-training adalah Wikipedia *revision history*. Pada akhir penelitian, terdapat sejumlah 1857 data korpus yang dihasilkan secara otomatis dengan akurasi data sebesar adalah 76%. Hasil tersebut menunjukkan bahwa kombinasi metode Co-training dan data Wikipedia *revision history* berpotensi menghasilkan korpus *Textual Entailment* yang berukuran besar dan baik.

Kata Kunci:
Textual Entailment, Co-training, Wikipedia *revision history*, korpus, Bahasa Indonesia

ABSTRACT

Name : Aditya Rama
Program : Computer Science
Title : Cross Language Word Sense Disambiguation

Textual Entailment is a research in NLP that aims to identify whether there is an entailment relation between two texts. Textual Entailment research has been developed in a variety of languages but it is rare for the Indonesian language. This study aimed to develop a corpus of Indonesian Textual Entailment with Co-training method, a semi-supervised learning method that has been used in the development of English Textual Entailment corpus. Wikipedia revision history is used as the data resources. At the end of the study, the corpus contains 1857 data that is generated automatically with 76% accuracy. The results of this study show that the combination of Co-training method and the Wikipedia revision history data could potentially produce a good corpus of Indonesian Textual Entailment.

Keywords:
Textual Entailment, Co-training, Wikipedia *revision history*, corpus, Indonesian language

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB 1
PENDAHULUAN

Bab ini membahas mengenai latar belakang penelitian, perumusan masalah, tujuan dan manfaat penelitian, ruang lingkup penelitian, metodologi penelitian, serta sistematika penulisan.

1.1 Latar Belakang

Word Sense Disambiguation (WSD) merupakan salah satu tugas untuk menentukan makna terbaik dari sebuah kata. Sebuah kata sendiri dapat memiliki beberapa makna dan bergantung pada konteks dimana kata tersebut muncul. Pendekatan *machine learning* baik itu *supervised*, *semi-supervised*, ataupun *unsupervised* dapat dilakukan untuk menyelesaikan permasalahan ambiguasi kata tersebut.

Pendekatan *supervised* yang digunakan untuk membangun sistem WSD membutuhkan data *training* dan *resource* yang tidak sedikit. Kebutuhan akan data dan *resource* yang besar tersebut merupakan kendala dari bahasa-bahasa tertentu. Bahasa Inggris sebagai salah satu bahasa internasional mempunyai data dan *resource* yang cukup banyak untuk membangun sistem dengan *supervised learning*. Namun demikian, bahasa Indonesia sendiri termasuk dalam *under resource language* dimana sumber daya yang dapat dimanfaatkan untuk sistem WSD masih terbatas. Wordnet pada umumnya dipakai sebagai *resource* utama yang memiliki informasi dari makna kata untuk sistem WSD. Penggunaan Wordnet sebagai *sense inventory* tersebut masih terbatas pada bahasa Indonesia dikarenakan jumlah *inventory* yang masih kurang.

Membangun Wordnet secara manual untuk memenuhi kebutuhannya sebagai inventaris makna kata membutuhkan waktu dan dana yang relatif tidak sedikit. Berdasarkan isu tersebut, metode lain dibutuhkan untuk membangun *supervised WSD system* yang akan mengatasi permasalahan *resource* yang belum memadai. Salah satu metode yang akan dicoba pada penelitian ini adalah pemanfaatan korpus dwibahasa dengan pendekatan *cross language*.

Pendekatan *cross language* dengan bahasa Inggris sebagai pasangan korpus diharapkan dapat memperkaya *resource* yang masih kurang pada bahasa Indonesia.

1.2 Perumusan Masalah

Beberapa pertanyaan yang menjadi rumusan masalah dalam penelitian ini yaitu:

1. Bagaimana cara menerapkan pemindahan makna (*sense transfer*) dari korpus paralel bahasa Inggris - Indonesia?
2. Seberapa baik performa *WSD* yang dibangun untuk bahasa Indonesia tersebut?

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari penelitian yang dilakukan adalah memberikan tambahan inventaris berupa *sense* dari kata-kata bahasa Indonesia untuk wordnet Bahasa Indonesia dan menghitung seberapa baik performa *wsd system* bahasa Indonesia yang dibuat.

1.4 Ruang Lingkup Penelitian

Penelitian berfokus pada pemindahan *sense* dari korpus paralel berbahasa Inggris ke Indonesia, dan melakukan *WSD task* pada hasil pemindahan makna kata tersebut.

Word sense disambiguation yang dilakukan pada penelitian ini hanya pada tingkatan *coarse-grained wsd*.

1.5 Metodologi Penelitian

Ada lima tahapan yang dilakukan pada penelitian ini. Penjelasan dari tiap tahapan adalah sebagai berikut.

1. Studi Literatur
   Tahap ini berfokus pada pencarian informasi mengenai *WSD system* baik secara umum maupun teknik yang digunakan, dan juga *task* lain yang berkaitan dengan *WSD* seperti *word sense induction* (*WSI*).
2. Perumusan Masalah
   Masalah-masalah yang ada dalam penelitian nantinya dianalisis penyelesaiannya pada tahap ini.
3. Rancangan Penelitian
   Proses yang melibatkan seluruh penelitian untuk menyelesaikan permasalahan yang ada.

4. Implementasi
   Tahap ini merupakan implementasi dari rancangan yang sudah dibuat untuk memecahkan permasalahan yang ada.

5. Analisis dan Kesimpulan
   Hasil percobaan dianalisis untuk mendapatkan gambaran seberapa baik performa dari sistem yang dibuat.

1.6 Sistematika Penulisan

Sistematika penulisan yang ada dalam laporan penelitian ini sebagai berikut:

- Bab 1 PENDAHULUAN
  Bab ini akan menjelaskan mengenai latar belakang, perumusan masalah, tujuan penelitian, tahapan penelitian, ruang lingkup, metodologi, dan sistematika penulisan dari penelitian ini.

- Bab 2 TINJAUAN PUSTAKA
  Bab ini akan menjelaskan mengenai konsep dan teori yang relevan dari hasil studi literatur yang telah dilakukan. Teori-teori yang dijelaskan meliputi *Word sense disambiguation*, *Word sense induction*, dan beberapa hal lain yang dibutuhkan pada penelitian.

- Bab 3 RANCANGAN PENELITIAN
  Bab ini akan membahas perihal pelaksanaan dari proses *tagging sense* pada korpus English dan *word alignment* pada kedua korpus (Indonesia - English) beserta evaluasinya.

- Bab 4 IMPLEMENTASI
  Pada bab ini akan dijelaskan mengenai implementasi dari sistem WSD yang dibangun untuk melakukan disambiguasi pada kata-kata yang telah ditentukan.

- Bab 5 HASIL DAN ANALISIS
  Pada bab ini, dijelaskan mengenai hasil penelitian beserta evaluasi dan analisis dari hasil tersebut.

- Bab 6 PENUTUP
  Kesimpulan dan saran dari hasil dan pelaksanaan penelitian akan dijelaskan pada bab ini.

BAB 2
TINJAUAN PUSTAKA

Bab ini membahas mengenai studi literatur yang digunakan selama penelitian. Studi literatur ini menjelaskan tentang hal-hal mendasar yang dibutuhkan dalam penelitian.

2.1 Word Sense Disambiguation

Word Sense Disambiguation merupakan salah satu penelitian di bidang NLP yang bertujuan untuk menentukan makna yang paling tepat dari suatu kata berdasarkan konteks kata tersebut ditemukan. Sebagaimana kata dalam suatu bahasa bisa memiliki makna lebih dari satu (polisemi), *task* ini akan menentukan makna kata mana yang paling tepat.

Penentuan makna kalimat dilakukan dengan pemberian informasi berupa kata yang menjadi *target* dan konteks berupa kalimat. Contoh proses disambiguasi yang dilakukan untuk kata **cokelat**:

K1: Roni memakan **cokelat** yang diberikan ibunya.

K2: Walaupun mobil **cokelat** itu mahal, dia sangat ingin membelinya.

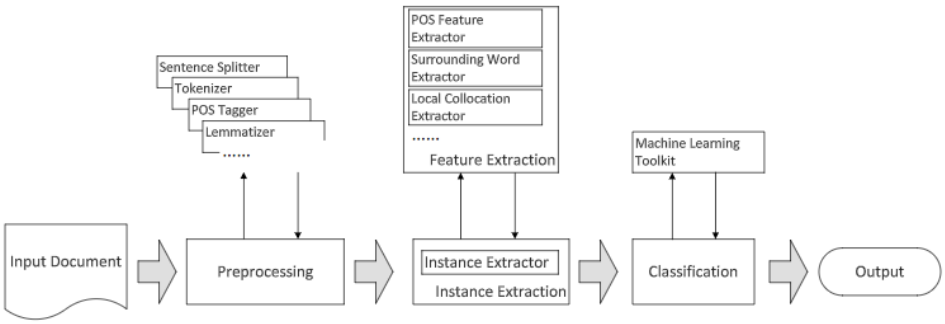
Pada kalimat pertama (K1), **cokelat** yang dimaksud memiliki makna sebagai makanan yang terbuat dari buah *cokelat*. Sementara itu, Kata **cokelat** pada kalimat kedua (K2) memiliki makna yang berbeda, dimana kata tersebut merupakan satu keterangan warna. Penentuan makna yang tepat dapat dilakukan dengan bantuan informasi konteks dari kalimat dimana kata tersebut muncul. Pada K1, kata **memakan** memberikan informasi bahwa *cokelat* yang dimaksud adalah objek yang bisa dimakan. Kata yang memberikan informasi pada kalimat kedua adalah kata **berwarna** yang secara eksplisit menerangkan bahwa **cokelat** yang dimaksud adalah warna. Namun demikian, konteks maupun informasi yang bisa diambil dari kalimat tidak selalu eksplisit. Pada contoh kalimat seperti "Pohon cokelat tua di belakang rumahku sangat besar", cokelat yang dimaksud bisa bermakna "buah cokelat yang sudah tua" atau "berwarna cokelat tua".

Penentuan makna kata yang tepat oleh sistem WSD ditentukan berdasarkan konteks dari kata tersebut berada. Walaupun satu kata dapat memiliki beberapa makna, terdapat kecil kemungkinan bahwa kata yang sama digunakan dalam satu *discourse* untuk menyatakan makna yang berbeda sebagaimana "*one sense per*

discourse" (Gale et al., 1992).

2.1.1 WSD Bahasa Inggris

Salah satu sistem WSD untuk bahasa inggris yang ada adalah "It Makes Sense" (IMS) yang dibuat oleh Zhi Zhong dan Hwee Tou Ng (Zhong dan Ng, 2010). Sistem dibangun menggunakan pendekatan *supervised learning* yang dapat digunakan untuk semua kata bahasa Inggris. Pada dasarnya, *classifier* yang dipilih untuk *task* ini adalah *support vector machine* (SVM). Arsitektur yang dibangun pada IMS dapat dilihat pada gambar berikut:



Gambar 2.1: Arsitektur IMS

Proses *pre-processing* pada IMS dilakukan dengan empat tahapan:

1. Mendeteksi batasan kalimat dengan *sentence splitter*
2. Tokenisasi dengan *tokenizer*
3. POS Tagging untuk semua token
4. Mengubah token menjadi lemma dengan *lemmatizer*

Ekstraksi fitur dilakukan dengan mengombinasikan:

1. POS Tag dari tiga buah kata di kiri dan kanan *target word*, serta kata itu sendiri.
2. Kata-kata sekitar pada konteks kalimat ataupun kalimat tetangganya. Kata-kata yang terkandung di dalam *stopwords* dan memiliki simbol atau angka dibuang dari kalimat tersebut. Kata-kata yang tersisa tersebut kemudian diubah menjadi bentuk kata dasarnya dalam huruf kecil.
3. *Local Collocation* dengan 11 buah *collocation* baik itu sebelum *target word* maupun setelahnya.

Pengujian seberapa baik performa IMS dalam melakukan WSD *task* mendapatkan hasil:

	SensEval-2	SensEval-3	SemEval-2007	
	Fine-grained	Fine-grained	Fine-grained	Coarse-grained
IMS	68.2%	**67.6%**	58.3%	**82.6%**
Rank 1 System	**69.0%**	65.2%	**59.1%**	82.5%
Rank 2 System	63.6%	64.6%	58.7%	81.6%
WNs1	61.9%	62.4%	51.4%	78.9%

Gambar 2.2: Performa IMS (Zhong dan Ng, 2010)

2.2 Word Sense Induction

Word Sense Induction (WSI) adalah sebuah *task* yang mempunyai fungsi utama untuk mendapatkan makna kata dari sebuah korpus atau teks yang belum dianotasi secara otomatis. WSI dapat dilakukan jika penelitian WSD yang ingin dilakukan tidak mempunyai cukup *resource* seperti misalnya Wordnet yang memadai. Terdapat berbagai macam pendekatan dalam melakukan WSI, diantaranya adalah dengan melakukan *clustering* kata (Denkowski, 2009), ataupun menggunakan pendekatan *cross language*.

2.2.1 Pendekatan *Clustering*

Dua kata dianggap dekat secara semantik jika memiliki *co-occurrence* dengan kata-kata tetangganya yang sama (Nasiruddin, 2013). Konsep tersebut mendasari cara WSI mendapatkan *sense* kata secara implisit berdasarkan hasil *cluster* yang terbentuk dari data atau teks mentah (teks yang tidak dianotasi).

Penarikan makna secara implisit dapat dicontohkan pada beberapa kalimat rujukan berikut (Denkowski, 2009):

1. A bottle of tezgüno is on the table.
2. Everyone likes tezgüno.
3. Tezgüno makes you drunk.
4. We make tezgüno out of corn.

Walaupun belum terdapat informasi eksplisit makna dari tezgüno, dapat disimpulkan bahwa tezgüno mengacu pada minuman beralkohol yang

memabukkan. Penarikan kesimpulan ini didapatkan dari kemunculan kata tersebut dengan kata lain pada konteks yang sama.

Pada pendekatan *clustering* ini, makna kata bisa didapatkan secara implisit dari hasil *cluster* yang terbentuk, namun demikian pelabelan yang dilakukan untuk menentukan apa yang direpresentasikan *cluster* tersebut merupakan sebuah *task* tersendiri.

2.2.2 Pendekatan *Cross Language*

Selain pendekatan *clustering*, WSI juga dapat memanfaatkan fitur dimana satu kata dari suatu bahasa, dapat diterjemahkan menjadi beberapa kata di bahasa lain. Contoh kasus tersebut dapat dilihat pada kata "halaman" berikut:

(K1-Indonesia): Aku membaca 10 **halaman** buku Harry Potter

(K1-English): I read 10 **pages** of Harry Potter book

(K2-Indonesia): Ani tinggal di rumah dengan **halaman** yang sangat luas

(K2-English): Ani lives in a house with very large **yard**

Berdasarkan kedua pasangan kalimat tersebut, kata **halaman** dalam bahasa Indonesia dapat diterjemahkan menjadi dua buah kata dalam bahasa Inggris, yaitu *page* ataupun *yard*. Hal ini menunjukan bahwa terjemahan dari suatu kata bergantung pada makna yang dikandung kata tersebut.

2.3 Evaluasi *Word Alignment*

2.3.1 *Word Alignment*

Tugas dari *word alignment* adalah menemukan korespondensi antara kata dan frasa pada teks paralel (Mihalcea dan Pedersen, 2003). Evaluasi ini akan membandingkan antara hasil *alignment* dari sebuah tool *word alignment* dengan hasil *alignment* manusia sebagai *gold standard*. Kasus yang dapat terjadi pada proses *alignment* ini adalah ketika terdapat kata yang tidak memiliki pasangan. Contoh dari kasus tersebut dapat dilihat pada pasangan kaimat berikut:

K1(en) : *He would do it regardless what people say*

K1(id) : Dia akan melakukannya segalanya

Bila melihat bahasa Indonesia sebagai sumber bahasa, maka kata "segalanya" pada kalimat tersebut tidak memiliki pasangan. Pada kasus seperti contoh diatas, kata yang tidak memiliki pasangan akan dipasangkan dengan *token* NULL.

Selain kata yang tidak memiliki pasangan, terdapat juga kasus dimana pasangan adalah berupa frasa. Hal ini dapat dilihat pada contoh berikut:

K2(en) : The victim must be taken to the hospital

K2(id) : Korban tersebut harus di bawa ke rumah sakit

Berangkat dari bahasa asal yaitu Indonesia, kata "rumah sakit" dipasangkan kepada kata "hospital". Hal ini dapat berlaku berkebalikan jika bahasa asal yang digunakan adalah bahasa Inggris seperti kata "untuknya" berpasangan dengan kata "for him".

2.3.2 Tokenisasi

Pada proses evaluasi ini, pemisah kata yang umum digunakan untuk tokenisasi adalah karakter spasi. setiap token dari hasil tokenisasi tersebut kemudian dianggap sebagai satu unit kata. Kata ini akan diindeks dengan angka untuk mempermudah proses *alignment* dan komputasi evaluasi. Contoh tokenisasi dan pemberian indeks pada kalimat "Aku ingin membeli mainan" adalah:

K3(id) : Aku ingin membeli mainan

K3(indeks) : 1 2 3 4

Beberapa *tool* merepresentasikan kalimat dan kata sebagai indeks angka tersebut untuk mempermudah pemrosesan. Suatu *file* dapat berisi indeks dari kalimat dan kata yang ada pada kalimat tersebut seperti:

```
1 4 5 7
2 4 9 2
3 1 8 4
...
```

Dimana angka pertama merepresentasikan kalimat ke n pada korpus, dan angka-angka selanjutnya adalah indeks dari kata pada kalimat tersebut.

2.3.3 Pengukuran Evaluasi

Terdapat empat buah pengukuran berbeda, yaitu *precision*, *recall*, *f-measure*, dan *alignment error rate (AER)* (Mihalcea dan Pedersen, 2003). Diberikan hasil *alignment* dari program berupa A, dan *gold standard alignment* dari *evaluator* (manusia) sebagai G, masing-masing mengandung dua buah *set* yaitu *probable alignment* dan *sure alignment*. Pengukuran evaluasi dapat dilakukan dengan cara berikut:

$$P_T = \frac{|A_T \cap G_T|}{|A_T|} \quad (1)$$

$$R_T = \frac{|A_T \cap G_T|}{|G_T|} \quad (2)$$

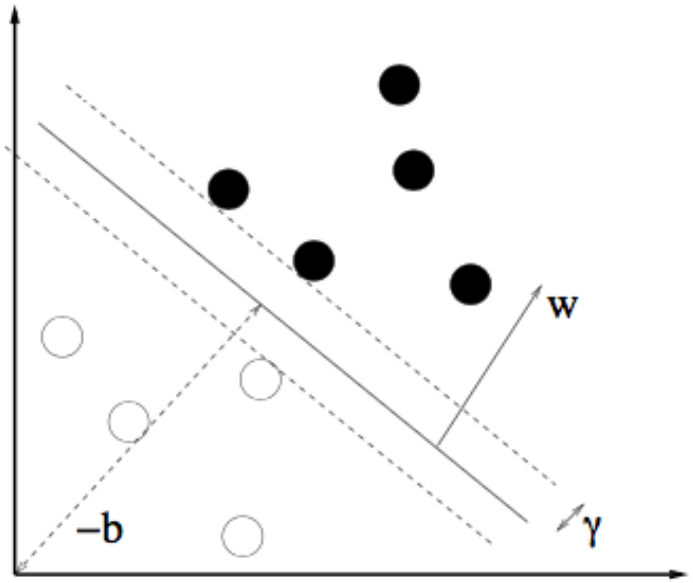
$$F_T = \frac{2P_T R_T}{P_T + R_T} \quad (3)$$

$$AER = 1 - \frac{|A_P \cap G_S| + |A_P \cap G_P|}{|A_P| + |G_S|} \quad (4)$$

Gambar 2.3: Pengukuran Word Alignment

2.4 Support Vector Machine

SVM merupakan salah satu *classifier* yang dapat digunakan untuk permasalahan klasifikasi. SVM termasuk sebagai metode klasifikasi yang populer dan telah digunakan untuk berbagai permasalahan seperti klasifikasi teks, *facial expression recognition*, analisis gen, *word sense disambiguation*, dan lain-lain. SVM dapat dikatakan sebagai salah satu metode yang membangun aturan yang dinamakan sebagai *linear classifier* yang secara teori akan menghasilkan kualitas prediksi dari *unseen data* yang baik (Fradkin dan Muchnik, 2006).



Gambar 2.4: Hyperplane SVM pada (Fradkin dan Muchnik, 2006)

Konsep dari cara SVM bekerja adalah dengan menemukan sebuah *hyperplane* dengan *margin* (jarak dari *hyperplane* dengan titik kelas terdekat) yang terbesar. Pemilihan *margin* dengan nilai terbesar ini ditujukan agar *classifier* lebih optimal dalam memisahkan objek dengan kelas yang berbeda.

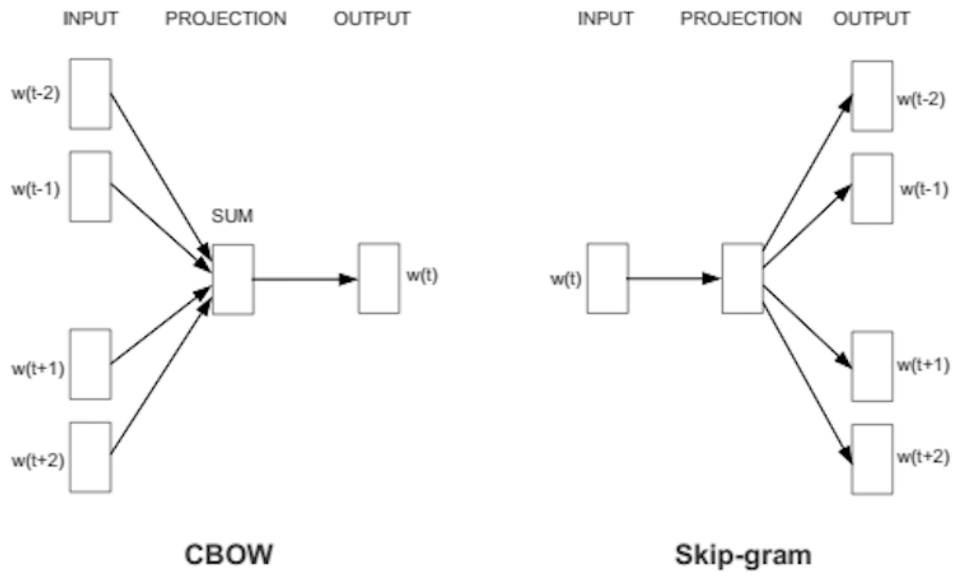
2.5 Word Embedding Language Model

Terdapat beberapa cara untuk merepresentasikan sebuah kata sebagai *input model*. Salah satu cara yang sederhana adalah dengan merepresentasikan kata sebagai *one hot vector*. Pada model ini, setiap kata di dalam sebuah korpus diberikan nomor indeks untuk membangun vektor yang mewakili keberadaan kata tersebut. Jika terdapat sebuah kata yang muncul pada konteks yang ingin direpresentasikan, indeks vektor yang sama dengan indeks kata tersebut akan bernilai 1. Bila terdapat sebuah korpus dengan jumlah kata unik berjumlah 4 dengan kata-kata "Ani", "marah", "kemarin", dan "malam" (sebuah vektor dengan pangjang empat). Representasi *one hot vector* untuk kalimat "Ani marah" dapat ditulis dengan vektor [1,1,0,0].

Representasi *one hot vector* akan mempunyai panjang vektor yang besar jika korpus mempunyai jumlah kata unik yang besar. Terdapat bentuk representasi lain untuk membentuk vektor dari kata, salah satunya adalah dengan *word embedding*. *Word Embedding* menggunakan representasi bilangan *real* pada vektor untuk merepresentasikan sebuah kata berdasarkan hasil *training* dengan

suatu korpus. Contoh representasi dari *word embedding* pada suatu kata "makan" adalah vektor [0.6, -0.3, ..., 0.5] (misalnya). Vektor dari hasil *word embedding* mempunyai karakteristik dimana jarak antara dua buah vektor dari kata yang mirip secara semantik bernilai kecil(dekat). Bila misalkan pada data *training* untuk *word embedding* terdapat banyak kalimat-kalimat berbentuk "... makan Y ..." dimana Y adalah sebuah objek berupa makanan. Maka kata-kata yang mewakili Y seperti misalnya "burger", "apel", "steak", dan lain-lain, akan memiliki vektor yang mirip dan secara implisit dapat saling menggantikan untuk menempati posisi Y tersebut. Berdasarkan keterdekatan vektor tersebut, *word embedding* mampu untuk menangkap semantik dari kata-kata yang ada pada korpus.

Salah satu model *word embedding* yang dapat digunakan adalah Word2Vec (Mikolov et al., 2013). Terdapat dua buah arsitektur Word2Vec tersebut, yaitu skip-Gram dan *Continous bag-of-words* (CBOW). Arsitektur pada CBOW memiliki pendekatan untuk memprediksi setiap kata berdasarkan kata-kata disekelilingnya. Lain halnya dengan arsitektur tersebut, Skip-Gram akan memprediksi kata-kata di sekeliling berdasarkan kata yang diberikan. Gambaran dari skip-gram dan CBOW dapat dilihat pada gambar berikut:



Gambar 2.5: Word2Vec

BAB 3
RANCANGAN PENELITIAN

Bab ini akan menjelaskan gambaran proses penelitian secara keseluruhan yang terdiri dari pembuatan *sense tagged corpus* bahasa Inggris, *word alignment* korpus paralel, peningkatan kualitas dan evaluasi *word alignment*, pemindahan *sense* dari korpus bahasa Inggris, dan sistem WSD yang diimplementasikan.

3.1 Pembuatan *Sense Corpus* Bahasa Inggris

Pembuatan *sense tagged corpus* bahasa Inggris dilakukan dengan menggunakan *tool* IMS untuk mendapatkan makna terbaik yang dapat ditag oleh *tool* tersebut. Makna kata hasil dari proses ini akan dipindahkan ke kata yang bersesuaian pada kalimat yang sama pada bagian *sense transfering*. *File* yang diberikan sebagai masukan dari IMS adalah kalimat-kalimat pada bahasa Inggris yang berasal dari korpus identik.

3.2 Penyelarasan Kata pada Korpus Paralel

Penjajaran kata pada korpus berbahasa Inggris dan Indonesia menggunakan *tools word alignment* bernama Giza++. *Tool* ini merupakan salah satu *word alignment tools* pada *statistical machine translation* (SMT) yang dapat digunakan untuk memasangkan kata-kata pada dua buah korpus atau lebih. Terdapat beberapa *word alignment tools* lain seperti Berkeley *aligner*, anymalign, dan lain-lain. Penyelarasan kata ini digunakan untuk kebutuhan pemindahan *sense* dari kata bahasa Inggris ke kata dalam bahasa Indonesia.

Proses penyelarasan yang dilakukan dengan Giza++ meliputi tahap-tahap berikut:

1. Mempersiapkan kedua buah *file* yaitu korpus bahasa asal (*source*) dan korpus bahasa tujuan (*target*). Kedua *file* ini berpasangan dalam setiap barisnya. Baris pertama dalam *file* pertama berpasangan dengan baris pertama pada *file* kedua sampai akhir baris pada kedua *file*.

2. Menghasilkan *file* perbendaharaan kata dari kedua bahasa dan *list* indeks perbendaharaan kata pada tiap kalimat yang sudah diselaraskan

3. Menghasilkan *cooccurence file* dari kosa kata dan pasangan kalimat tersebut

4. Proses *alignment* yang menghasilkan beberapa macam *output file*

Terdapat satu buah *output file* Giza++ yang berisi pasangan-pasangan kalimat dengan kata-kata yang sudah diselaraskan dengan translasinya dalam bahasa tujuan. Hasil ini merupakan *best viterbi alignment* menurut Giza++.

Pada skenario *alignment* dengan bahasa Indonesia sebagai *source* dan bahasa Inggris sebagai *target*, satu kata dalam bahasa Indonesia akan dipasangkan dengan tepat satu kata dalam bahasa Inggris.

3.3 Evaluasi *Word Alignment*

Word alignment hasil dari *tool* Giza++ dievaluasi dengan menggunakan *anotator* hasil *alignment* dari *anotator* yang akan ditujukan sebagai *gold standard*. Nilai-nilai yang akan dihitung meliputi *precision* (P), *recall* (R), dan F-*score*. Metode evaluasi keseluruhan meliputi:

1. Pemilihan *random sampling* sebanyak seratus buah pasangan kalimat

2. Masing-masing *anotator* memasang-masangkan kata yang tepat pada masing-masing pasangan kalimat, dengan asumsi bahwa anotasi manusia sebagai *gold standard*

3. Hasil anotasi manusia dan keluaran dari *tool* Giza dibandingkan untuk mendapatkan ketiga nilai P, R, dan F-Score.

3.4 Peningkatan Kualitas Hasil *Alignment*

Proses peningkatan kualitas hasil alignment diperlukan untuk meminimalisir kesalahan pemasangan kata-kata pada proses sebelumnya. Permasalahan yang terjadi adalah adanya pasangan-pasangan kata yang tidak benar seperti pada halnya kata "lapangan" yang dipasangkan dengan kata dalam bahasa inggris *field*, *ground*, *involved*, *job*, *program*, dan beberapa kata lainnya. Peningkatan kualitas *alignment* ini dilakukan dengan memanfaatkan hasil *inverse alignment* antara bahasa Indonesia ke Inggris. Pemanfaatkan hasil *alignment* korpus bahasa Inggris ke Indonesia akan menghasilkan pasangan-pasangan kata dengan tingkat kesalahan *alignment* lebih kecil. Metode yang akan dilakukan adalah dengan memeriksa setiap pasangan kata dari bahasa Indonesia yang mana merupakan kata dalam

bahasa Inggris, apakah kata tersebut memiliki pasangan dalam bahasa Indonesia yang sama dengan keluaran dari *inverse alignment* Giza.

Pada kasus kata **lingkungan** dari hasil keluaran Giza memiliki pasangan kata:

1. environment
2. environmental
3. neighborhood
4. within
5. environmentally

Untuk setiap pasangan kata dalam bahasa Inggris tersebut, akan dilakukan pengecekan apa saja pasangan kata bahasa Indonesianya. Bila terdapat kata **lingkungan** dalam pasangan kata bahasa Indonesianya maka kata tersebut dianggap pasangan yang benar.

Kata **environment** memiliki pasangan dalam bahasa Indonesia:

1. lingkungan
2. lingkup

Keberadaan kata **lingkungan** dari pasangan kata *environment* mengakibatkan kata *environment* dianggap sebagai pasangan kata yang benar dari *lingkungan*. Proses ini dilakukan untuk setiap kata dalam bahasa Inggris yang merupakan pasangan kata dalam bahasa Indonesia.

3.5 Pemindahan Makna Kata

Pemindahan makna kata dilakukan dengan dua buah *sub-process* yang terdiri dari:
1. Pemasangan antar kalimat yang bersesuaian dengan kata-kata yang berpasangan. Pada contoh kata "halaman" yang berpasangan dengan "courtyard", maka pasangan kalimat "Aku bermain di halaman" akan dipasangkan dengan kalimat "I play at the courtyard".

2. Setelah pasangan kalimat-kalimat yang bersesuaian dipasangkan, kalimat dalam bahasa Inggris tersebut diberikan *sense* dengan menggunakan tool IMS.

3. *Sense* dari kata yang menjadi *target* tersebut kemudian dipindahkan ke kata yang bersesuaian pada bahasa Indonesianya di kalimat tersebut. Bila "courtyard" memiliki *sense* yang artinya adalah "halaman rumah", maka "halaman" pada kalimat "Aku bermain di halaman" memiliki *sense* "halaman rumah".

3.6 Sistem WSD

Sistem WSD yang dibangun adalah dengan menggunakan pendekatan *supervised learning*. Hasil dari pemindahan makna kata akan digunakan sebagai *training* dan *testing* data untuk menguji performa dari sistem yang dibangun. *Classifier* yang digunakan dalam sistem *WSD* ini adalah SVM. Pengujian dilakukan dengan menggunakan beberapa fitur seperti *bag of words*, *POS Tag*, dan *word embedding*. Fitur *bag of words* menggunakan *window* sebanyak dua buah kata kanan dan kiri kata tujuan sebagai kata konteks. *POS Tag* dan vektor *word embedding* juga akan diimplementasikan pada penelitian ini. Performa dari sistem WSD akan dilihat berdasarkan perhitungan F1-score *micro* dari hasil klasifikasi.

BAB 4
IMPLEMENTASI

Bab ini akan menjelaskan perihal implementasi dari pemindahan *sense* dan sistem *WSD* yang dibuat.

4.1 Proses *Word Alignment*

Perlu dilakukan *pre-processing* untuk memisahkan kalimat bahasa Inggris dan bahasa Indonesia menjadi dua buah *file* paralel untuk dapat dimasukan sebagai input dari *tool* Giza.

Perintah berikut digunakan untuk melakukan *alignment* dengan Giza pada penelitian ini:

Kode 4.1: Word Alignment

```
# Lakukan pada direktori Giza
./plain2snt.out [source_language] [target_language]

# proses diatas menghasilkan tiga buah file yaitu dua buah file
   vocabulary yang berisi indeks dengan kata (bahasa asal, dan
   bahasa tujuan), dan satu buah file snt yang berisi \textit{
   alignment} dari kalimat.

./snt2cooc.out [source_language_vcb_file] [
   target_language_vcb_file] [snt_file] > [coocurrence_file]

# proses snt2cooc akan menghasilkan \textit{cooccurence file}

./GIZA++ -S [source_language_vcb] -T [target_language_vcb] -C [
   snt_file] -CoocurrenceFile [cooc_file]
```

Giza mengeluarkan beberapa *file* hasil dari proses tersebut. *Output* yang akan digunakan diantaranya adalah *file* bernama A3.final yang merupakan pasangan kalimat dengan kata-kata yang sudah dipasangkan sesuai dengan prediksi terbaik hasil pemrosesan Giza.

4.2 Peningkatan Kualitas *Alignment*

Hasil dari *alignment* kata yang dilakukan Giza masih menghasilkan pasangan-pasangan kata yang tidak tepat. Untuk mengurangi jumlah pasangan kata yang salah tersebut, dilakukan *enhancement* dengan dua kali proses *alignment* baik itu dari Indonesia ke Inggris maupun sebaliknya.

Konsep dari proses yang dilakukan adalah melakukan validasi terhadap kata-kata yang berpasangan dari kedua korpus. Pertama, setiap kata dalam bahasa Indonesia dikumpulkan terlebih dahulu dengan setiap pasangan kata bahasa Inggrisnya (satu kata bisa memiliki lebih dari satu pasangan). Proses selanjutnya adalah melakukan pengumpulan yang serupa terhadap kata dalam bahasa Inggris dengan pasangan kata dalam bahasa Indonesianya. Proses validasi dilakukan dengan cara:

1. Untuk setiap kata di bahasa Indonesia semisal kata "kali"
2. Lakukan pengecekan terhadap setiap pasangan kata di bahasa Inggris dari "kali" misalkan "time", "river", "fire"
3. Jika pada kamus *enhancement* "time" dipasangkan dengan "kali", dan "waktu" maka kata "time" merupakan pasangan yang dianggap benar. Pada kasus kata "fire", bila pasangan bahasa Indonesianya adalah "api" dan "tungku", maka kata "fire" dianggap bukan pasangan yang tepat dengan "kali" karena tidak terdapat pasangan "fire -> kali".

Kode 4.2: Word Alignment Enhancement

```
dict_id = {}
dict_en = {}

# masukan setiap kosa kata bahasa Indonesia ke dalam dict_id
    sebagai key dan kumpulan pasangan kata bahasa inggrisnya
    sebagai value
# proses yang sama dilakukan untuk dict_en dengan kosa kata
    bahasa Inggris sebagai key dan kumpulan pasangan kata bahasa
    Indonesia sebagai value
# stop adalah list stopword yang didapat dari korpus nltk

# this section is for filtering which english word that has
    corresponding indo translation (bidirectional) from Giza
    output
for indo_word in dict_id.keys():
```

```
if indo_word not in dict_en:
# filtering so no same translation is entered, answer -> answer,
    jawaban -> jawaban
for en_word in dict_id[indo_word].keys():
if en_word in dict_en and indo_word in dict_en[en_word] and
    en_word not in stop:
if indo_word not in final_dictionary:
final_dictionary[indo_word] = { en_word: dict_en[word_en][word_id
    ] }
else:
if en_word not in final_dictionary[indo_word]:
final_dictionary[indo_word][en_word] = dict_en[word_en][word_id]
```

4.3 Sistem WSD

Sistem WSD dibangun dengan menggunakan *supervised learning*. *Machine learning tool* yang digunakan untuk membangun sistem ini adalah Scikit (Pedregosa et al., 2011). Pada sistem ini terdapat tiga buah bagian utama yaitu pemilihan fitur serta *classifier*, ekstraksi fitur, dan evaluasi hasil *classifier*.

4.3.1 Pemilihan Fitur dan *Classifier*

Terdapat tiga buah fitur pada penelitian ini yang terdiri dari:

1. Fitur *bag of words*
2. Fitur *POS tagging*
3. Vektor dari hasil *word embedding*

Classifier yang digunakan pada penelitian ini adalah SVM dengan *library* Python yaitu Scikit dengan parameter *default* berupa kernel linear dan C=1.

4.3.2 Ekstraksi Fitur

Fitur bag of words menggunakan pendekatan kemunculan kata-kata pada konteks kalimat sebagai fitur. Kata yang diambil untuk dijadikan fitur adalah dua buah kata di kiri dan di kanan dari *target word*. Pada penelitian ini, kata-kata yang merupakan bagian dari *stopwords* dalam bahasa Indonesia tidak dimasukan sebagai fitur. Contoh dari fitur ini dapat dilihat pada kalimat dengan kata tujuan "bisa" berikut:

- Ani digigit ular dengan bisa yang berbahaya

Pada contoh kalimat tersebut, *bag of words* yang diambil adalah ["digigit", "ular", "berbahaya", NULL]

Setiap fitur *bag of words* dari setiap kalimat tersebut dikumpulkan untuk menjadi satu fitur besar yang mendeteksi kemunculan kata-kata tersebut pada setiap kalimat.

Proses yang dilakukan dalam pengambilan kata konteks untuk fitur *bag of words* dilakukan dengan tahap-tahap berikut.

Kode 4.3: Fitur Bag of Words

```
bag_of_words []

for each sentence
  split sentence by whitespace into words
  for each word
    if word == target word
      for x in [-2,-1,1,2]
        if word(x) exist and word(x) not in bag_of_words
          add word(x) into bag_of_words

return bag_of_words
```

Fitur POS Tagging menggunakan *tool* dari Stanford bernama "A Part-Of-Speech Tagger" yang dilatih dengan menggunakan model untuk bahasa Indonesia. Proses *tagging* ini dilakukan sebelum memasuki sistem WSD terhadap keseluruhan kalimat dalam korpus identik yang berisi bahasa Indonesia saja. Terdapat *pre-processing* awal pada korpus bahasa Indonesia tersebut untuk menghilangkan beberapa tanda baca seperti titik, koma, tanda tanya, tanda seru, dan beberapa tanda baca lainnya. Setelah proses tersebut, diberikan tanda baca berupa titik pada akhir kalimat sebagai indikator akhir sebagai kebutuhan dari kompabilitas *tool* (tidak semua kalimat pada korpus memiliki tanda baca akhir kalimat). Perintah yang dilakukan untuk melakukan *POS Tagging* tersebut adalah:

Kode 4.4: Stanford POS Tagger

```
java -mx300m -cp 'stanford-postagger.jar:lib/*' edu.stanford.nlp.
   tagger.maxent.MaxentTagger -model <model_bahasa_indonesia> -
   textFile <korpus_bahasa_indonesia>
```

Hasil dari proses tersebut merupakan korpus dengan setiap kata yang sudah memiliki POS Tag dengan format:

```
<kata_1>_<postag_1> <kata_2>_<postag_2> ... <kata_n>_<postag_n>
<kata_x> <postag_x> <kata_y>_<postag_y> ... <kata_z>_<postag_z>
...
```

Kelas kata yang diambil adalah POS Tag dari *target word* kata sebelum, dan kata sesudahnya. Kelas kata dari *target word* diperhitungkan karena pada beberapa kasus kata polisemi dapat dibedakan maknanya berdasarkan POS Tag yang dimilikinya. POS Tag yang digunakan mengacu pada kelas-kelas yang ada pada POS Tag Penn Treebank seperti NN(Noun), NNP(Proper Noun), VB(verb), CC(Coordinating conjunction), dan lain-lain. Pembentukan kelas POS Tag untuk menjadi fitur dilakukan dengan proses yang serupa pada fitur *bag of words* dimana kombinasi kombinasi dari POS Tag yang mungkin direpresentasikan dalam *one hot representation*. Hal ini dapat diilustrasikan dengan proses berikut:

1. Simpan POS Tag dari indeks kata masing-masing (*target word*-1, *target word*, *target word*+1)
2. Untuk masing-masing indeks baik itu -1,0,dan 1
3. Kumpulkan POS Tag yang *distinct*, populasikan dalam *array*
4. *Array* ini nantinya akan merepresentasikan keberadaan POS Tag tertentu pada kata dengan indeks terkait tersebut

Fitur ketiga yang dicoba adalah *word embedding*. Vektor *word embedding* yang dijadikan fitur adalah vektor dari tiga buah kata di kiri dan kanan dari *target word*. Vektor *target word* tidak dimasukkan ke dalam fitur karena akan mempunyai nilai vektor yang sama antara satu konteks dengan konteks lain. Setiap vektor tersebut kemudian disambung (*concat*) sebagai satu buah vektor besar yang merupakan representasi dari vektor fitur pada konteks tersebut.

4.3.3 Evaluasi Sistem

Evaluasi dilakukan dengan *cross validation* menggunakan perhitungan F1-score dari hasil klasifikasi yang dilakukan *classifier*. *Crossvalidation* dilakukan dengan interasi sebanyak tiga buah dengan perbandingan antara *training* dan *test set* sebesar 0,7:0,3.

Sebuah algoritma sederhana digunakan sebagai *baseline* untuk pembanding dari sistem WSD yang dibangun. Baseline menggunakan pendekatan *most frequent sense* sebagai cara untuk menentukan makna terbaik dari suatu kata. Bila diberikan

training data untuk kata "bisa" dengan makna "dapat melakukan" sebanyak 4 buah dan makna "racun ular" sebanyak 6 buah, maka algoritma baseline ini akan mengklasifikasikan semua kata "bisa" menjadi "racun ular".

BAB 5
HASIL DAN ANALISIS

Bab ini menjelaskan mengenai hasil yang didapatkan dari eksperimen, serta evaluasi dan analisis terkait hasil tersebut.

5.1 Korpus Identik

Korpus identik berisi pasangan-pasangan kalimat Indonesia-Inggris sebanyak 88.919 kalimat untuk masing-masing bahasa.

5.2 *Word Alignment*

5.3 Pengumpulan Data

Tabel 5.1 menunjukkan spesifikasi dari dua jenis data XML Wikipedia digunakan untuk eksperimen yang diunduh dari situs Wikimedia.

Tabel 5.1: Dua jenis data XML Wikipedia

No	Berkas	Jenis	Tangga Diambil	Ukuran
1	idwiki-20160901-pages-articles-multistream.xml.bz2	*Articles, templates, media/file descriptions*, dan *primary meta-pages*	2016-09-02 13:24:24	381.3 MB
2	idwiki-20160901-pages-meta-history.xml.bz2	All pages with complete page edit history	2016-09-07 18:22:58	2.8 GB

Berkas pertama adalah XML semua artikel Wikipedia tanpa revisi, berkas kedua adalah Wikipedia *revision history*. Kedua berkas merupakan data Wikipedia terakhir pada tanggal 1 September 2016. Jika dilihat ukuran, XML Wikipedia sudah sangat mencukupi kebutuhan data untuk penelitian ini.

5.4 Hasil Pengolahan Data

Ada dua jenis pengolahan data yang dilakukan, yang pertama adalah mengolah data Wikipedia semua artikel menjadi model *word embedding* dan yang kedua adalah pengolahan data utama, yaitu mengubah data Wikipedia *revision history* menjadi pasangan kandidat T dan H. Berikut adalah hasil yang diperoleh pada setiap tahap pengolahan data.

- **Ekstraksi Teks**
  Ekstraksi teks dilakukan pada kedua berkas yang diunduh. Pada berkas pertama yang berukuran 381.3 MB, seluruh artikel dapat diekstraksi. Total artikel yang didapatkan adalah 386.357 artikel. Sedangkan, pada berkas kedua, tidak seluruh artikel yang diekstraksi karena keterbatasan-keterbatasan dalam penelitian ini. Total artikel hasil ekstraksi berkas kedua berjumlah 10.758 artikel yang berisikan 595.136 *revision history*. Contoh keluaran dari proses ekstraksi XML semua artikel Wikipedia dapat dilihat pada gambar **??**; sedangkan, XML Wikipedia Revision History pada gambar **??**.

- **Pemenggalan Kalimat**
  Dari 386.357 artikel, total kalimat setelah dilakukan pemenggalan adalah 3.867.831 kalimat. Semua kalimat yang terdiri lebih dari dua kata akan digunakan untuk membentuk model *word embedding*.

- **Pemodelan Word Embedding**
  Model *word embedding* dibentuk menggunakan data 3.867.831 kalimat bersih dan menghasilkan 3 buah berkas model (lihat pada tabel 5.2).

Tabel 5.2: Berkas model *word embedding*

Nama Berkas	Jenis	Ukuran
word2vec	File	40.987 KB
word2vec.syn0.npy	NPY File	227.472 KB
word2vec.syn1neg.npy	NPY File	227.472 KB

- **Pembentukan T dan H**
  Sebelum membentuk T dan H, teks induk dan revisi dipasangkan terlebih dahulu. Jumlah pasangan yang terbentuk adalah 584.377 pasang. Setelah melalui tahap pemrosesan, dihasilkan sejumlah 67.096 pasang T dan H. Pasangan T dan H yang terbentuk menunjukkan jenis *entailment* yang terjadi adalah tingkat leksikal karena data didominasi dengan pasangan kalimat yang

lexical overlap-nya tinggi, yaitu pasangan yang memiliki banyak kesamaan kata.

- **Anotasi Data Manual**
  Dari 500 data *random* yang digunakan saat pelabelan manual, hanya 400 data saja yang akan digunakan sebagai bibit Co-training. Penjelasan lebih lanjut dapat dilihat pada bagian 5.5.

- **Ekstraksi fitur**
  Kombinasi fitur sepasang T dan H adalah fitur *word embedding* serta fitur tambahan pada masing-masing kalimat T dan H (selanjutnya dibahas pada bagian 5.6). Sedangkan fitur untuk komentar penulis adalah kemunculan N-gram tertentu (selanjutnya dibahas di bagian 5.7). Hanya 15.000 dari 67.096 pasang T dan H serta komentar penulis yg dapat diubah menjadi vektor fitur tersebut karena keterbatasan waktu dan penyimpanan.

5.5 Hasil Anotasi Manual

Sebelum melakukan anotasi, para anotator akan diuji pemahamannya terlebih dahulu. Anotator yang cukup paham mengenai permasalahan *Textual Entailment*, yaitu ketika tingkat persetujuan anotasi penulis dan anotator tersebut melebihi 0,5 berdasarkan perhitungan Cohen's Kappa, akan lanjut ke tahap anotasi data yang sebenarnya. Sebelum uji coba, seluruh anotator diberi panduan anotasi untuk meningkatkan pemahaman mereka terhadap *Textual Entailment*. Ukuran data uji coba adalah 15 data yang dipilih secara khusus agar dapat mencakup berbagai kasus. Berikut adalah tabel hasil anotasi uji coba.

Tabel 5.3: Kappa antara penulis dan masing-masing anotator pada data ujicoba

anotator ke-	1	2	3	4	5	6
agreement	0.867	0.8	0.667	0.8	1	0.8
Kappa	0.865	0.798	0.663	0.798	1	0.798

anotator ke-	7	8	9	10	11	12
agreement	0.933	0.867	0.8	0.933	0.933	0.867
Kappa	0.932	0.865	0.798	0.932	0.932	0.865

Dari tabel 5.3, terlihat bahwa nilai Kappa semua anotator melampaui batas yang ditentukan, sehingga semua anotator dapat melanjutkan ke tahap berikutnya.

Selain menghitung persetujuan antara penulis dan anotator, persetujuan secara keseluruhan perlu dihitung. Persetujuan keseluruhan pada data uji coba akan menjadi *baseline* untuk nilai persetujuan keseluruhan pada data yang sebenarnya. Apabila nilai persetujuan keseluruhan pada data yang sebenarnya lebih baik dari persetujuan keseluruhan di data uji coba, data uji coba benar tergolong representatif.

Tabel 5.4: Hasil anotasi uji coba

	Label E	**Label C**	**Label U**
Data-1	13	0	0
Data-2	13	0	0
Data-3	1	2	10
Data-4	9	4	0
Data-5	13	0	0
Data-6	10	1	2
Data-7	8	1	4
Data-8	3	1	9
Data-9	12	0	1
Data-10	4	0	9
Data-11	13	0	0
Data-12	13	0	0
Data-13	11	2	0
Data-14	13	0	0
Data-15	13	0	0

Isi tabel menunjukkan jumlah anotator yang sepakat melabeli data pada baris tertentu dengan label pada kolom tertentu. Nilai Kappa yang dihasilkan berdasarkan tabel di atas adalah 0.432 dengan *overall agreement* 0.783. Jika merujuk pada pengukuran *agreement* di gambar **??**, nilai tersebut menunjukkan bahwa tingkat persetujuan antar anotator tergolong menengah.

Setelah anotasi, masing-masing datum memiliki 3 label (dapat seragam maupun berbeda-beda), yaitu dari penilaian anotator 1, 2, dan 3. Tingkat persetujuan anotasi dari tiga anotator tersebut kemudian dihitung, *overall agreement* yang dihasilkan sebesar 0.86 dan Kappa sebesar 0.729. Nilai tersebut menunjukkan bahwa pada data yang sebenarnya persetujuan lebih tinggi dibandingkan data uji. Bisa disimpulkan bahwa, data uji yang berukuran kecil tersebut merupakan sampel data yang representatif dan mengandung kasus-kasus yang ambiguitasnya tinggi.

Sebuah datum seharusnya hanya memiliki satu label saja, yaitu E, U, atau C.

Oleh karena itu, label pada setiap datum perlu ditentukan dengan cara memilih label dengan keputusan terbanyak. Apabila label E, C, dan U muncul bersama pada satu datum, langkah yang dilakukan adalah menganalisis lebih dalam datum tersebut dan memutuskan label mana yang lebih tepat, umumnya label akan mengarah ke U. Kemudian, data yang sudah memiliki label tersebut akan kami analisis dari segi jumlahnya. Tujuannya adalah untuk mengetahui beberapa hal diantaranya pola pada pasangan T dan H. Setelah dianalisis kami mendapatkan hasil seperti pada tabel 5.5.

Tabel 5.5: Jumlah data per label hasil notasi

	Label E	Label U	label C
Jumlah	323	54	123

Perbandingan jumlah label pada tabel 5.5 tidak seimbang dan dikhawatirkan dapat mempengaruhi hasil klasifikasi menjadi lebih dominan ke suatu label. Untuk mencegah hal tersebut terjadi, dilakukan langkah-langkah berikut.

- Menggabungkan label C dan U menjadi label baru, yaitu NE yang berarti *not entail*. Langkah ini diambil dengan pertimbangan bahwa data C dan U masih terlalu sedikit. Setelah label digabungkan, hasil data Textual Entailment pada penelitian ini akan *binary*, yaitu E dan NE.
- Memangkas jumlah data berlabel E dari 323 menjadi 223, agar jumlah data berlabel E tidak dominan.

Setelah melakukan langkah-langkah tersebut, bibit yang diperoleh berkurang menjadi 400 data dengan perbandingan label E dan NE adalah 223:177.

5.6 Pengujian Arsitektur RNN

Sebelum melakukan Co-training, terlebih dahulu dilakukan pemilihan arsitektur pada *view* pertama. Kedua arsitektur tersebut diuji menggunakan metode 10-fold *cross validation* dengan terhadap data hasil proses anotasi manual atau calon bibit Co-training. Hasil *cross validation* terdapat di tabel 5.6.

Tabel 5.6: Hasil *cross validation* dua arsitektur RNN

Arsitektur RNN tanpa fitur tambahan	0.58
Arsitektur RNN dengan fitur tambahan	0.69

Cross validation menggunakan program dalam bahasa Python yang dibuat dengan menambahkan *library* Keras[1]. *Library* Keras menyediakan berbagai *neural network classifier*, seperti LSTM dan *feed forward neural network*. Pada kedua percobaan, tahap *train* data pada masing-masing RNN menggunakan jumlah *epoch* yang sama, yaitu 200 *epoch*. Jumlah *epoch* dipilih secara heuristik dengan pertimbangan jumlah tersebut tidak terlalu kecil dan juga tidak terlalu besar. Penentuan jumlah *epoch* seharusnya dilakukan dengan beberapa kali percobaan, namun dikarenakan waktu yang tidak mencukupi, *epoch* langsung ditentukan sebesar 200.

5.7 Pemilihan Fitur View Kedua

Program Weka GUI digunakan untuk menentukan fitur pada *view* kedua. Fitur untuk *view* kedua berupa jumlah kemunculan unigram, bigram, dan trigram terbanyak pada komentar penulis dari seluruh data. Oleh karena itu, sebelum mendaftarkan apa saja fitur view kedua, frekuensi kemunculan N-gram pada data terlebih dahulu harus dibuat. Dari 67.000 pasang T dan H serta komentar penulis yang dihasilkan, dihitung kemunculan dari setiap unigram, bigram, dan trigram yang terdapat pada teks komentar. Kemudian kemunculan tersebut diurutkan berdasarkan jumlah terbanyak. N-gram yang dengan frekuensi yang banyak tersebut tidak langsung digunakan sebagai fitur, N-gram tersebut harus terlebih dahulu dianalisis dan dipilih, mana yang merupakan N-gram yang dapat merepresentasikan isi komentar. Pemilihan dilakukan manual dan menggunakan heuristik. Total ada sebanyak 74 buah N-gram yang digunakan sebagai rancangan fitur awal. Tabel 5.7 menunjukkan contoh dari N-gram tersebut.

[1]https://keras.io/

Tabel 5.7: Contoh N-gram yang kemunculannya digunakan sebagai fitur

Unigram	sunting, revisi, kembali, ubah, ganti, menolak, tolak, mengembalikan, batal, rapikan, perbaikan, copyedit, replaced
Bigram	ke versi, versi terakhir, suntingan special, dikembalikan ke, penggantian teks, teks otomatis, bicara dikembalikan, mengembalikan revisi
Trigram	teks terakhir oleh, perubahan terakhir oleh, menolak perubahan teks, menggunakan mesin wikipedia, dengan menggunakan mesin, replaced beliau dia, pada masa di, di tahun pada, di masa pada, masa pada masa

Fitur jumlah kemunculan semua N-gram yang dipilih belum tentu akan menjadi kombinasi fitur yang paling baik. Oleh karena itu, digunakan fitur *select attribute* pada Weka GUI dengan metode *genetic search* untuk memilih kombinasi fitur terbaik. Kombinasi ini didapat berdasarkan hasil training 500 data yang sebelumnya sudah dilabeli secara manual. Setelah melalui proses *select attribute*, jumlah fitur tereduksi menjadi hanya 22 buah fitur jumlah kemunculan N-gram.

5.8 Pengujian Classifier View Kedua

Kombinasi fitur terbaik hasil dari tahap pemilihan fitur menggunakan *select attribute* pada Weka digunakan sebagai fitur untuk percobaan *k-fold cross validation*. *Cross validation* dengan *k=10* tersebut diujikan kepada beberapa *classifier*. *Classifier* untuk *view* kedua belum ditentukan sebelumnya, namun ada beberapa *classifier* yang sudah direncanakan akan dicoba. Semua *classifier* yang akan digunakan tersedia di Weka GUI. Sama seperti saat tahap pemilihan atribut, *cross validation* dilakukan terhadap data hasil anotasi manual. Berikut adalah hasil dari *cross validation*.

Tabel 5.8: Hasil *cross validation* beberapa *classifier* di Weka

Classifier	Akurasi
Decision Tree (J48)	0.61
Multilayer Perceptron	0.626
Bayesian Network	0.61
Naive Bayes	0.626
Multinomial Naive Bayes	0.636

Walaupun nilai akurasi tidak terlihat jauh berbeda, *classifier* Multinomial Naive Bayes memberikan hasil yang baik di antara beberapa percobaan lain, dengan akurasi 0.636. Multinomial Naive Bayes digunakan sebagai *classifier* pada *view* kedua Co-training.

5.9 Co-training

Co-training hanya dilakukan terhadap 15.000 dari 67.096 data yang dimiliki karena keterbatasan pada penelitian ini, diantaranya waktu dan kapasitas penyimpanan untuk melakukan komputasi data. Bibit yang digunakan sejumlah 400 data.

Data hasil klasifikasi hanya diambil yang terbaik untuk ditambahkan pada data berlabel. Untuk mengurangi kemungkinan bertambahnya data yang tidak akurat pada *training data*, ada batasan yang harus dipenuhi oleh masing-masing data yang akan ditambahkan, yaitu data diklasifikasikan ke dalam label yang sama oleh kedua *classifier* pada Co-training dengan tingkat kepercayaan di atas 0.9 untuk *classifier view* pertama dan di atas 0.5 untuk *view* kedua. Angka tersebut dipilih berdasarkan anggapan bahwa *view* pertama akan lebih informatif dibandingkan dengan *view* kedua. Sehingga, nilai kepercayaan RNN dalam melabeli data disyaratkan sangat tinggi.

Setelah sejumlah data terbaik terpilih sebagai calon data tambahan, jumlah label dari data tersebut harus dibandingkan. Apabila jumlah label E:NE berada melebihi 5:4 atau kurang dari 4:5, data yang jumlahnya berlebih harus dipangkas hingga perbandingan E:NE menjadi 1:1. Batasan ini harus ditentukan agar ke depannya tidak terjadi ketimpangan jumlah label pada data. Batasan tersebut diambil berdasarkan perbandingan data berlabel mula-mula yaitu 223:177 yang mendekati dengan perbandingan 5:4.

Pada bagian **??**, disebutkan bahwa *stopping condition* dari Co-training pada penelitian ini adalah ketika kedua *classifier* tidak mampu menyepakati satu pun label yang sama dalam *n* iterasi berurutan. Selain itu, algoritme Co-training pada

gambar **??** menunjukkan ada *k* buah data yang diambil dari data tidak berlabel setiap iterasi. Nilai *n* dan *k* berturut-turut ditentukan sebesar 3 dan 500. Nilai tersebut ditentukan setelah percobaan kombinasi *n* dan *k* pada Co-training dilakukan.

- Percobaan 1 dengan *n*=3 dan *k*=100
  Percobaan 1 terhenti pada iterasi 13, dimana pada iterasi 10, 11, dan 12, data terbaik yang dihasilkan berjumlah 0.

- Percobaan 2 dengan *n*=5 dan *k*=100
  Percobaan 2 terhenti pada iterasi 15. Sama seperti percobaan 1, iterasi 10, 11, 12, 13, dan 14 tidak menghasilkan data terbaik satu pun.

- Percobaan 3 dengan *n*=3 dan *k*=500
  Percobaan 3 masih berjalan hingga iterasi ke 93. Jumlah data terbaik yang dihasilkan pun cukup menjanjikan. Percobaan ini yang kemudian digunakan hingga akhir penelitian ini.

Jika jumlah data tidak berlabel yang digunakan dalam sebuah iterasi terlalu kecil, iterasi akan cepat berhenti karena kemungkinan tidak adanya data terbaik yang diperoleh pada tiap iterasi menjadi lebih besar.

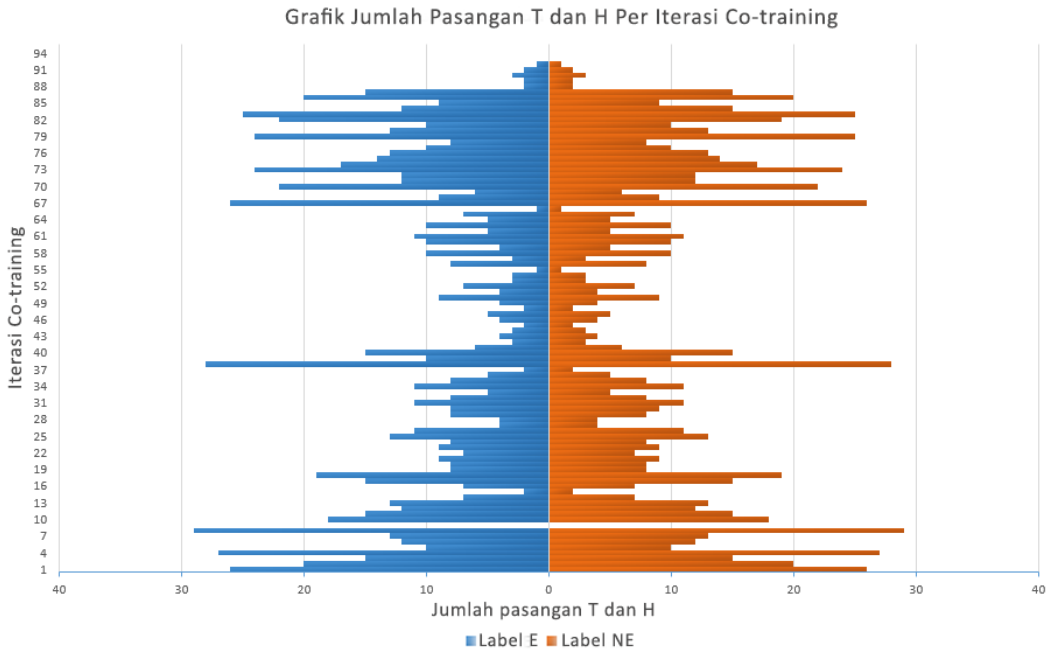
5.9.1 Hasil Co-training

Hasil tiap iterasi dari percobaan Co-training disimpan untuk dievaluasi dan dianalisis. Hasil yang dikeluarkan berupa pasangan T dan H serta komentar penulis yang sudah diberi label. Gambar 5.1 menunjukkan contoh hasil dari Co-training pada salah satu iterasi.

```
<pair id=7263>
    <T>Pada bulan januari 2011 Hary Tanoesoedibjo tidak lagi berada dalam
    kepemilikan saham tvOne .</T>
    <H>Abdul Latief berada dalam kepemilikan saham tvOne dan MNCTV .</H>
    <C>âsuntingan special contributions 125 162 60 25 125 162 60 25 user talk
    125 162 60 25 bicara dibatalkan ke versi terakhir oleh user relly
    komaruzaman relly komaruzaman</C>
    <V>NE</V>
</pair>
<pair id=7208>
    <T>Hanya sedikit pakar Perjanjian Lama di masa kini yang akan mengatakan
    bahwa Mikha menulis keseluruhan kitab ini .</T>
    <H>Hanya sedikit pakar Perjanjian Lama pada masa kini yang akan mengatakan
    bahwa Mikha menulis keseluruhan kitab ini .</H>
    <C>bot penggantian teks otomatis di masa pada masa kosmetik perubahan</C>
    <V>E</V>
</pair>
<pair id=7381>
    <T>Bersama asistennya , Kolonel Colin Mackenzie beliau mengumpulkan dan
    meneliti naskah-naskah Jawa Kuno .</T>
    <H>Bersama asistennya , Kolonel Colin Mackenzie dia mengumpulkan dan
    meneliti naskah-naskah Jawa Kuno .</H>
    <C>penggantian teks otomatis dengan menggunakan mesin wikipedia awb
    autowikibrowser replaced beliau â dia 2</C>
    <V>E</V>
</pair>
```

Gambar 5.1: Contoh hasil Co-training pada salah satu iterasi

Setelah program Co-training terhenti, terdapat 1.857 data yang berhasil dilabeli dari total 15.000 data tidak berlabel yang digunakan, dengan perbandingan jumlah data label E:NE adalah 927:930. Pola jumlah data pada setiap iterasi dapat dilihat pada gambar 5.2.



Gambar 5.2: Jumlah data yang diperoleh pada tiap iterasi

Jumlah data yang diperoleh di tiap iterasi sangat acak dan beragam, mulai dari yang sangat kecil hingga sangat besar, bahkan pada iterasi ke-9, Co-training tidak mengeluarkan data berlabel satu pun. Jumlah data tersebut berada di rentang 0 hingga 58. Dengan mempertimbangkan pola yang acak dan beragam, evaluasi dilakukan tidak per iterasi, melainkan per 5 iterasi. Bila dilihat dari segi perbandingan jumlah label E dan NE, Co-training yang dirancang dapat mengatasi masalah ketidakseimbangan yang dikhawatirkan di awal. Kemungkinan besar data yang diperoleh dipangkas jumlahnya karena terlihat dari hasil perbandingan label pada tiap iterasi yang hampir semuanya 1:1. Hasil perbandingan antara E dan NE yang hampir sama tersebut menjadi salah satu kekurangan pada penelitian ini. Seharusnya pemangkasan tidak berdasarkan perbandingan jumlah data hasil pelabelan, namun berdasarkan perbandingan jumlah data berlabel keseluruhan.

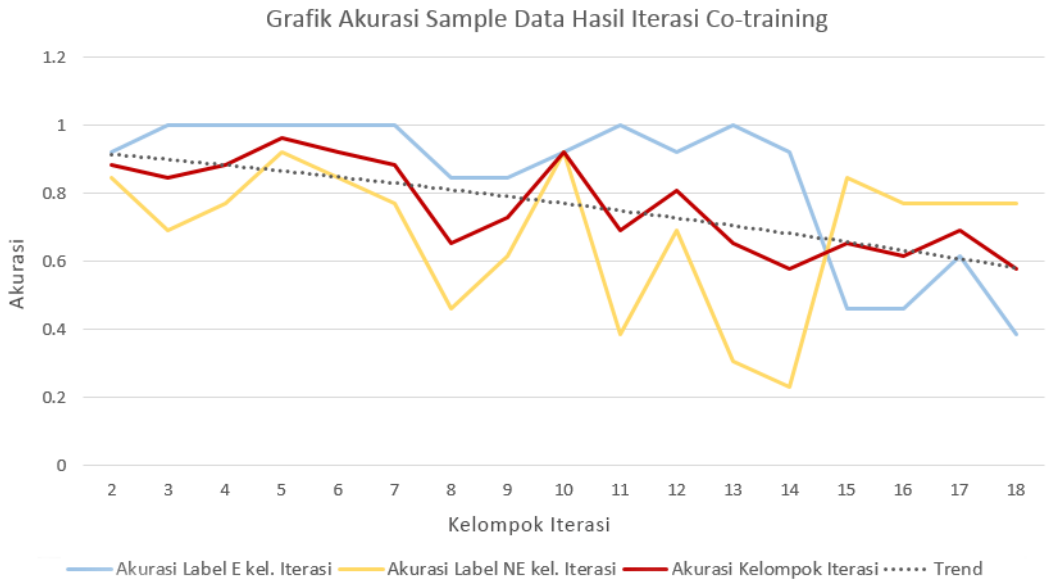
5.9.2 Evaluasi Co-training

Evaluasi dilakukan setiap 5 iterasi sekali (untuk selanjutnya disebut kelompok iterasi) dengan cara mengambil sampel acak sebanyak 13 data berlabel E dan 13 data berlabel NE dari total data dalam sebuah kelompok iterasi. Data yang terpilih akan dievaluasi secara manual. Tabel 5.9 menunjukkan akurasi setiap kelompok iterasi setelah evaluasi.

Tabel 5.9: Jumlah data pada setiap iterasi

No	Label E	Label NE	Jumlah	Sampel E Tepat	Sampel NE Tepat	Akurasi Label E	Akurasi Label NE	Rata-rata Akurasi
1	98	98	196	11	9	0.85	0.69	0.77
2	72	72	144	12	11	0.92	0.85	0.88
3	49	49	98	13	9	1.00	0.69	0.85
4	57	57	114	13	10	1.00	0.77	0.88
5	46	46	92	13	12	1.00	0.92	0.96
6	35	36	71	13	11	1.00	0.85	0.92
7	43	43	86	13	10	1.00	0.77	0.88
8	60	60	120	11	6	0.85	0.46	0.65
9	18	18	36	11	8	0.85	0.62	0.73
10	24	24	48	12	12	0.92	0.92	0.92
11	18	18	36	13	5	1.00	0.38	0.69
12	35	36	71	12	9	0.92	0.69	0.81
13	38	38	76	13	4	1.00	0.31	0.65
14	64	64	128	12	3	0.92	0.23	0.58
15	79	79	158	6	11	0.46	0.85	0.65
16	68	69	137	6	10	0.46	0.77	0.62
17	78	78	156	8	10	0.62	0.77	0.69
18	42	42	84	5	10	0.38	0.77	0.58

Nilai akurasi total pada label E adalah 0.82, akurasi total label NE adalah 0.67, dan akurasi untuk data keseluruhan adalah 0.76. Jika hasil akurasi pada kelompok iterasi di-plot pada sebuah grafik (lihat grafik 5.3), dapat disimpulkan bahwa akurasi Co-training cenderung mengalami penurunan. Hasil evaluasi juga menunjukkan kecenderungan pelabelan label E lebih akurat dibandingkan NE.



Gambar 5.3: Akurasi pada setiap kelompok iterasi

Evaluasi dengan data metode *random sampling* ini sebaiknya dilakukan berkali-kali hingga nilai akurasi konvergen. Kemudian, nilai akurasi total dihitung dengan cara mencari nilai rata-rata dari seluruh akurasi *sampling*.

5.9.3 Analisis Co-training

Jumlah data pada korpus yang dihasilkan cukup besar apabila dibandingkan dengan jumlah bibit yang dimasukkan yaitu 400 data. Namun, jumlah hasil data tersebut masih terbilang kecil bila dibandingkan dengan ukuran data tidak berlabel semula, yaitu belum mencapai 13% dari total data tidak belabel. Hal tersebut menunjukkan bahwa Co-training yang dibangun masih belum percaya diri untuk mampu melabeli data yang tersisa.

Data yang diekstrak dengan Wikipedia cenderung didominasi oleh data *textual entailment* tingkat leksikal dan data yang tingkat *lexical overlap*-nya tinggi. Data *textual entailment* tingkat leksikal yang diekstrak pun kemungkinan besar berlabel E. Sedangkan, label NE umumnya diberikan pada data yang teks antara T dan H nya jauh berbeda. Oleh karena itu, Co-training dapat lebih mudah mengidentifikasikan sebagian besar label data tersebut sehingga akurasi yang dihasilkan menjadi baik.

Setelah melakukan evaluasi, ditemukan beberapa kekurangan dari Co-training yang diajukan, di antaranya:

1. Co-training membuat kesalahan pelabelan akibat data penggunaan bahasa asing pada data.

```
<pair id=6923>
    <T>Dari tinjauan Geografis , dua area terbesar dari Belgia
    adalah Belanda - yang merupakan area dari Flandria yang ada di
    utara , dengan 59 % dari populasi secara keseluruhan , dan
    Perancis - terletak di bagian selatan dari daerah Walonia ,
    dengan populasi sebesar 31 % Daerah Ibu Kota Brussel .</T>
    <H>Dari tinjauan Geografis , dua area terbesar dari Belgia
    adalah Belanda - yang merupakan area dari Flanders yang ada di
    utara , dengan 59 % dari populasi secara keseluruhan , dan
    Perancis - terletak di bagian selatan dari area Wallonia ,
    dengan populasi sebesar 31 % Brussels-Capital Region .</H>
    <C>beberapa terjemahan</C>
    <V>NE</V>
</pair>
```

Gambar 5.4: Contoh kesalahan pelabelan

Gambar di atas menunjukkan contoh data yang mengalami kesalahan pelabelan untuk suatu label yang seharusnya E. Penyebabnya adalah beberapa kata pada kalimat pada H menggunakan bahasa Inggris.

2. Co-training memprediksi label E dikarenakan *lexical overlap* antara T dan H cukup tinggi. Padahal perbedaan leksikal antara T dan H tersebut mengakibatkan perubahan makna pada teks.

```
<pair id=4973>
    <T>RRT ialah negara terbesar ke-4 di dunia setelah Rusia,
    Kanada, dan Amerika Serikat dan wilayahnya mencakup daratan yang
    sangat luas di bekas Peradaban Lembah Sungai Kuning .</T>
    <H>RRT ialah negara dengan datara terbesar ke-2 di dunia setelah
    Rusia,  dan wilayahnya mencakup daratan yang sangat luas di
    bekas Peradaban Lembah Sungai Kuning .</H>
    <C>suntingan special contributions 120 161 1 60 120 161 1 60
    user talk 120 161 1 60 bicara dibatalkan ke versi terakhir oleh
    user rotlink rotlink</C>
    <V>E</V>
</pair>
<pair id=7279>
    <T>Abdul Latief berada dalam kepemilikan saham tvOne dan TPI (
    sekarang MNCTV ) .</T>
    <H>Abdul Latief tidak lagi berada dalam kepemilikan saham tvOne
    .</H>
    <C>diganti ke semula</C>
    <V>E</V>
</pair>
```

Gambar 5.5: Contoh kesalahan pelabelan

Gambar di atas menunjukkan contoh kesalahan pelabelan untuk suatu data yang *lexical overlap*-nya tinggi namun seharusnya berlabel NE. Co-training belum bisa mendeteksi perbedaan angka pada (data pertama) dan penggunaan kata negasi (pada data kedua).

3. Co-training membuat kesalahan pelabelan pada data yang seharusnya mudah dideteksi nilai *entailment*-nya, seperti data yang memiliki *lexical overlap* tinggi dan seharusnya berlabel E.

```
<pair id=6855>
    <T>Penggunaan informasi dalam beberapa macam bidang , seperti
    bioinformatika , informatika kedokteran & informatika medis ,
    dan informasi yang mendukung ilmu perpustakaan , merupakan
    beberapa contoh yang lain dari bidang informatika .</T>
    <H>Penggunaan informasi dalam beberapa macam bidang , seperti
    bioinformatika , informatika medis , dan informasi yang
    mendukung ilmu perpustakaan , merupakan beberapa contoh yang
    lain dari bidang informatika .<</H>
    <C>suntingan special contributions 203 89 18 22 203 89 18 22
    user talk 203 89 18 22 bicara dikembalikan ke versi terakhir
    oleh user borgx borgx</C>
    <V>NE</V>
</pair>
```

Gambar 5.6: Contoh kesalahan pelabelan

Gambar 5.6 menunjukkan tingkat kesamaan antara T dan H yang tinggi, namun diberi penilaian NE oleh Co-training. Kesalahan seperti ini sulit untuk diprediksi penyebabnya.

4. Semakin lama Co-training dilakukan, data yang dihasilkan akan semakin jenuh. Pada kelompok iterasi-iterasi terakhir, data berlabel yang diperoleh semakin jenuh, yaitu memiliki jenis revisi yang serupa, misalnya hanya merevisi kata "dia" menjadi "beliau", "Cina" menjadi "Tiongkok", atau "di" menjadi "pada". Hal ini bisa disebabkan karena koleksi data berlabel yang tersisa sudah tidak bervariasi lagi.

Selain memiliki kasus-kasus kesalahan, ada pula kasus ketika Co-training bisa memberikan label yang tepat pada kasus yang terbilang sulit diprediksi oleh komputer. Contohnya terdapat pada gambar berikut.

```
<pair id=14313>
    <T>Bandara ini terletak di timur laut Palembang , melayani baik
    penerbangan domestik maupun internasional ( sejak runway di
    perpanjang ) .</T>
    <H>Bandara ini terletak di barat laut Palembang , melayani baik
    penerbangan domestik maupun internasional .</H>
    <C>menolak 10 perubahan teks terakhir dan mengembalikan revisi
    6996852 oleh relly komaruzaman</C>
    <V>NE</V>
</pair>
<pair id=14648>
    <T>Sedangkan wilayah Bandung Raya ( Wilayah Metropolitan Bandung
    ) merupakan metropolitan terbesar ketiga di Indonesia setelah
    Jabodetabek dan Gerbangkertosusila ( Grebangkertosusilo ) .</T>
    <H>Sedangkan wilayah Bandung Raya ( Wilayah Metropolitan Bandung
    ) merupakan metropolitan terbesar kedua di Indonesia setelah
    Jabodetabek .</H>
    <C>menolak 3 perubahan terakhir oleh pengguna henry jonathan
    henry jonathan dan pengguna 118 97 186 217 118 97 186 217 dan
    mengembalikan revisi 4791889 oleh luckan bot diragukan</C>
    <V>NE</V>
</pair>
```

Gambar 5.7: Contoh pelabelan berhasil

Contoh pada gambar di atas bertentangan dengan kasus yang ditemukan pada poin kedua pembahasan kekurangan Co-training. Pada gambar 5.7, Co-training menunjukkan kehandalannya dalam memberikan label NE untuk data yang *lexical overlap*-nya tinggi namun mengandung makna yang berbeda. Hal tersebut bisa disebabkan karena dukungan dari *view* komentar.

Pola yang ditunjukkan oleh hasil dari proses Co-training tidak dapat diprediksi. Beberapa contoh kasus yang sama atau serupa, diberikan label yang berbeda oleh Co-training. Penyebabnya kemungkinan besar adalah karena setiap iterasi Co-training dilatih oleh data berlabel tambahan. Sehingga sulit untuk mengetahui pola pelabelan Co-training. Salah satu penyebab kekurangan Co-training ini adalah penggunaan data bibit yang penulis akui masih terbilang kecil, apalagi untuk penelitian *deep learning*. Namun, jika dilihat dari akurasi, Co-training cukup memberikan hasil yang baik.

BAB 6
PENUTUP

6.1 Kesimpulan

Terdapat berbagai cara untuk membangun korpus *Textual Entailment*, salah satu cara yang cukup efisien adalah dengan pendekatan *semi-supervised learning*. Keunggulan pendekatan *semi-supervised learning* adalah kemampuannya dalam mengurangi usaha manual manusia. Co-training merupakan salah satu contoh metode *semi-supervised learning*. Metode Co-training pernah dicoba untuk memperbesar ukuran korpus *Textual Entailment* bahasa Inggris, dengan menjadikan isi korpus semula menjadi bibit dalam Co-training, kemudian Co-training akan memperbanyak isi korpus dengan melabeli data tidak berlabel yang dimasukkan.

Penelitian ini berusaha mencoba menggunakan metode Co-training untuk membangun dari awal korpus *Textual Entailment* Bahasa Indonesia. Untuk mengaplikasikan metode tersebut, dibutuhkan data dengan dua *view* yang saling lepas. Wikipedia *revision history* Bahasa Indonesia, yaitu data riwayat revisi dari artikel Wikipedia dalam Bahasa Indonesia, merupakan salah satu sumber data yang memiliki kriteria tersebut. *View* pertama adalah pasangan teks sebelum dan sesudah revisi (bisa disebut juga sebagai pasangan T dan H) dan *view* kedua adalah komentar penulis setelah melakukan revisi.

Masukan untuk proses Co-training berupa data pasangan T dan H serta komentar penulis yang sebagian kecil telah dilabeli (bibit Co-training) dan selebihnya belum diberi label. Data tersebut diperoleh dari Wikipedia *revision history* yang melalui beberapa tahap pengolahan, yaitu ekstraksi teks Wikipedia, pembentukan kandidat T dan H, anotasi manual, serta ekstraksi fitur. Dengan memasukkan sedikit data berlabel sebagai bibit, Co-training akan melabeli data tidak berlabel secara otomatis menggunakan dua *classifier* yang bekerja terpisah pada masing-masing *view*. Percobaan Co-training yang dilakukan pada penelitian ini, menunjukkan hasil yang cukup baik. Co-training hanya diberi bibit 400 data, namun dapat memperbesar ukuran data dengan menambahkan 1857 data baru. Akurasi dari hasil yang dikeluarkan juga cukup baik untuk ukuran penelitian pionir *Textual Entailment* Bahasa Indonesia, yaitu 76%.

Data berlabel terakhir setelah Co-training berhenti dijadikan data isi korpus. Data tersebut merupakan pasangan kalimat pada artikel Wikipedia sebelum dan

sesudah direvisi. Data yang dihasilkan cenderung mengarah ke *Textual Entailment* tingkat leksikal karena perubahan yang terjadi hanya perbedaan penggunaan kata, seperti sinonim. Walaupun akurasi cukup baik, data yang dihasilkan cukup jenuh dan kurang bervariasi. Revisi yang terjadi umumnya adalah parafrase, sehingga nilai label *entail* yang dihasilkan cukup mendominasi. Hal ini mungkin disebabkan karena revisi yang terjadi di Wikipedia didominasi dengan kasus yang seragam, contohnya revisi dengan bot mengubah kata dengan sinonimnya. Namun, jika dilihat dari segi ukuran, Wikipedia *revision history* cukup memadai.

Penulis berharap penelitian ini dapat menjadi motivasi untuk pengembangan *Textual Entailment* Bahasa Indonesia selanjutnya. *Textual Entailment* Bahasa Indonesia harus terus berkembang agar penelitian bidang NLP lain untuk Bahasa Indonesia dapat merasakan manfaat *Textual Entailment*.

6.2 Saran

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

1. Co-training pada penelitian ini menggunakan salah satu jenis *classifier* metode *deep learning*, namun data untuk melatih *classifier* tersebut data yang digunakan berukuran kecil, yaitu hanya 400 data. Sebaiknya, ukuran data berlabel sebagai bibit Co-training diperbesar. Hasil dari penelitian ini juga bisa digunakan kembali untuk memperbesar bibit pada penelitian selanjutnya.

2. Pada penelitian ini, beberapa parameter pada saat menjalankan proses Co-training ditentukan secara heuristik tanpa melakukan percobaan, seperti batasan tingkat kepercayaan *classifier* dalam melabeli data untuk menentukan apakah data berlabel tersebut baik, perbandingan jumlah data yang seimbang antara label E dan NE, serta cara pemangkasannya. Oleh karena itu, diharapkan penelitian selanjutnya melakukan percobaan terhadap penentuan parameter tersebut.

3. Metode Co-training yang digunakan dapat dicoba dengan menggunakan kombinasi *classifier* selain RNN atau Multinomial Naive Bayes.

4. Arsitektur RNN yang digunakan bisa lebih dikembangkan, misalnya dengan menambahkan lebih banyak fitur tambahan yang lebih menggambarkan hubungan T dan H atau desain arsitektur RNN baru yang lebih baik dan menentukan jumlah *epoch* melalui proses percobaan.

5. Amati lebih dalam mengenai fitur-fitur yang berpotensi memberikan informasi *entailment* dari view komentar penulis. Pada penelitian ini *view* komentar baru hanya menggunakan fitur-fitur yang sederhana. Tambahkan lagi fitur yang lebih relevan, misalnya menggunakan POS-Tag.

6. Menjadikan nama akun penulis (kolaborator Wikipedia) sebagai salah satu pertimbangan dalam klasifikasi. Hal ini disarankan atas dasar adanya kemungkinan seorang penulis yang sama melakukan revisi pada beberapa artikel atau penulis yang sama lainnya kerap melakukan tindakan vandalisme. Permasalahan ini belum dipertimbangkan pada penelitian ini.

7. Gunakan atau tambahkan sumber data lain selain Wikipedia Revision History. Wikipedia Revision History lebih cocok digunakan untuk RTE tingkat leksikal.

8. Jika menggunakan evaluasi *sampling*, baiknya evaluasi dilakukan dalam beberapa kali. Evaluasi pada penelitian ini hanya sempat dilakukan sekali karena keterbatasan waktu. Sebaiknya evaluasi jenis *sampling* dilakukan berkali-kali hingga akurasi konvergen.

DAFTAR REFERENSI

Denkowski, M. (2009). A survey of techniques for unsupervised word sense induction. *Language & Statistics II Literature Review*, pages 1–18.

Fradkin, D. dan Muchnik, I. (2006). Support vector machines for classification. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 70:13–20.

Gale, W. A., Church, K. W., dan Yarowsky, D. (1992). One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language*, pages 233–237. Association for Computational Linguistics.

Mihalcea, R. dan Pedersen, T. (2003). An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond-Volume 3*, pages 1–10. Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., dan Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nasiruddin, M. (2013). A state of the art of word sense induction: A way towards word sense disambiguation for under-resourced languages. *arXiv preprint arXiv:1310.1425*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., dan Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Zhong, Z. dan Ng, H. T. (2010). It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83. Association for Computational Linguistics.