UNIVERSITAS INDONESIA

PENGEMBANGAN KORPUS ...

SKRIPSI

ADITYA RAMA
1306397854

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
JANUARI 2017

UNIVERSITAS INDONESIA

PENGEMBANGAN KORPUS ...

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	:	Pengembangan Korpus ...

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	:	Rahmad Mahendra, S.Kom., M.Sc.	(	)
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

Alhamdulillahirobbil alamin, 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 dan adik-adik penulis
3. Nadiarani
4. Item 4

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:

Pengembangan Korpus ...

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	:	Pengembangan Korpus ...

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	:	Building Textual Entailment Corpus using Wikipedia Revision History Data with Co-training Method

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

HALAMAN JUDUL **i**

LEMBAR PERNYATAAN ORISINALITAS **ii**

LEMBAR PENGESAHAN **iii**

KATA PENGANTAR **iv**

LEMBAR PERSETUJUAN PUBLIKASI ILMIAH **v**

ABSTRAK **vi**

Daftar Isi **viii**

Daftar Gambar **x**

Daftar Tabel **xi**

Daftar Kode **xii**

1 PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Perumusan Masalah 2
1.3 Tujuan dan Manfaat Penelitian 2
1.4 Ruang Lingkup Penelitian 2
1.5 Metodologi Penelitian 3
1.6 Sistematika Penulisan 3

2 TINJAUAN PUSTAKA **5**
2.1 Textual Entailment 5
2.1.1 Tingkatan Textual Entailment 6
2.1.2 Pemanfaatan Textual Entailment 8
2.2 Korpus 8
2.2.1 Kriteria dan Tahap Pembuatan Korpus 9
2.2.2 Korpus Textual Entailment 9
2.3 Wikipedia 10
2.3.1 Wikipedia Offline 11
2.3.2 Wiki Markup Language 11
2.3.3 Wikipedia Revision History 13
2.3.4 Pemanfaatan Wikipedia 15
2.4 Semi-supervised Learning 15
2.4.1 Co-training 16

2.4.2 Co-training dalam Textual Entailment 17
2.5 Language Model 18
2.5.1 N-gram 18
2.5.2 Word Embedding 19
2.6 Naive Bayes 20
2.7 Recurrent Neural Network 21
2.8 Evaluasi 22
2.8.1 Cross Validation 22
2.8.2 Kappa 23

3 RANCANGAN PENGEMBANGAN KORPUS 25
3.1 Rancangan Pengembangan Korpus 25
3.2 Pembentukan Kandidat T dan H 27
3.3 Anotasi Manual 35
3.4 Co-training 37
3.4.1 Penentuan Classifier Pertama 37
3.4.2 Penentuan Classifier Kedua 41

4 IMPLEMENTASI 42
4.1 Ekstraksi Teks 42
4.2 Pemisahan kalimat 44
4.3 Pemodelan Word Embedding 46
4.4 Pembentukan T dan H 47
4.5 Ekstraksi Fitur 51
4.5.1 Ekstraksi Fitur View Pertama 51
4.5.2 Ekstraksi Fitur View Kedua 54
4.6 Co-training 54

5 EVALUASI DAN ANALISIS HASIL 56
5.1 Pengumpulan Data 56
5.2 Hasil Pengolahan Data 56
5.3 Hasil Anotasi Manual 58
5.4 Pengujian Arsitektur RNN 60
5.5 Pemilihan Fitur View Kedua 61
5.6 Pengujian Classifier View Kedua 62
5.7 Co-training 62
5.7.1 Hasil Co-training 63
5.7.2 Evaluasi Co-training 65
5.7.3 Analisis Co-training 67

6 PENUTUP 71
6.1 Kesimpulan 71
6.2 Saran 72

Daftar Referensi 74

DAFTAR GAMBAR

2.1 Contoh halaman Wikimedia *dumps* Bahasa Indonesia 11
2.2 Berkas XML artikel Wikipedia 12
2.3 Berkas XML Wikipedia Revision History 13
2.4 Arsitektur RNN pada *Textual Entailment* History (Bowman et al., 2015) 21
2.5 Ilustrasi *K-fold cross validation* (Olson dan Delen, 2008) 23

3.1 Rancangan Arsitektur Pengembangan Korpus *Textual Entailment* . . 25
3.2 Contoh isi korpus *Textual Entailment* yang diharapkan 27
3.3 Salah satu artikel Wikipedia beserta daftar revisinya 28
3.4 Contoh hasil pemasangan teks induk dan revisi 29
3.5 Ilustrasi Penyesuaian Paragraf 30
3.6 Contoh Penyesuaian Paragraf 30
3.7 Ilustrasi Penyesuaian Kalimat 31
3.8 Contoh Penyesuaian Kalimat 32
3.9 Ilustrasi Pemasangan Kalimat 33
3.10 Contoh Pemasangan Kalimat 34
3.11 Contoh hasil pasangan T dan H 35
3.12 Contoh hasil anotasi manual 36
3.13 Dua arsitektur RNN yang akan dicoba 38
3.14 Contoh penyesuaian kata antara 2 kalimat 40

4.1 Potongan artikel pada berkas XML Wikipedia 42
4.2 Hasil Ekstraksi pada XML seluruh artikel Wikipedia 43
4.3 Contoh hasil proses pemisahan kalimat 46

5.1 Contoh hasil Co-training pada salah satu iterasi 64
5.2 Jumlah data yang diperoleh pada tiap iterasi 65
5.3 Akurasi pada setiap kelompok iterasi 67
5.4 Contoh kesalahan pelabelan 68
5.5 Contoh kesalahan pelabelan 68
5.6 Contoh kesalahan pelabelan 69
5.7 Contoh pelabelan berhasil 70

DAFTAR TABEL

2.1 Tingkatan Textual Entailment 7
2.2 Contoh N-gram 19
2.3 Skala pengukuran Kappa (Landis dan Koch, 1977) 23
2.4 Perhitungan Cohen's Kappa 24

5.1 Dua jenis data XML Wikipedia 56
5.2 Berkas model *word embedding* 57
5.3 Kappa antara penulis dan masing-masing anotator pada data ujicoba 58
5.4 Hasil anotasi uji coba 59
5.5 Jumlah data per label hasil notasi 60
5.6 Hasil *cross validation* dua arsitektur RNN 60
5.7 Contoh N-gram yang kemunculannya digunakan sebagai fitur 61
5.8 Hasil *cross validation* beberapa *classifier* di Weka 62
5.9 Jumlah data pada setiap iterasi 66

DAFTAR KODE

2.1 Algoritme *bootstrapping* 16
2.2 Algoritme Co-training pada penelitian Blum dan Mitchell (1998) . . 17
2.3 Algoritme Co-training pada penelitian Zanzotto dan Pennacchiotti (2010) 17
4.1 Ekstraksi Wikipedia Revision History 43
4.2 Pemisahan kalimat 45
4.3 Pemodelan *word embedding* 46
4.4 Penghapusan kata-kata yang mengarah ke vandalisme 47
4.5 Pemasangan teks induk dan revisi 47
4.6 Penyesuaian paragraf 48
4.7 Levenstein Distance 49
4.8 Pemasangan kalimat 50
4.9 Ekstraksi Fitur *word embedding* 51
4.10 Penambahan fitur arsitektur RNN kedua 52
4.11 Perhitungan *similarity* kelompok kata 53
4.12 Algoritme Co-training yang akan digunakan 54

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

Task word sense disambiguation merupakan pekerjaan yang mempunyai tujuan utama untuk mengenali *sense*(makna) yang tepat dari sebuah kata. Sebagai manusia, pekerjaan ini termasuk relatif mudah karena kita dapat secara mengenal *sense* dari sebuah kata berdasarkan makna dari konteks yang ada di sekelilingnya. Komputer pada dasarnya hanya dapat mengenali kata sebagai sebuah representasi data, sehingga perlu adanya *modeling* agar komputer dapat mengenal dan membedakan makna antara satu kata dengan kata lainya. Sebuah contoh sederhana dari penentuan makna yang tepat dari sebuah kata dapat dilihat pada kalimat berikut.

(S1) : Setiap orang **bisa** melakukan tindak kriminal.

(S2) : Racun dari **bisa** ular itu sudah menyebar di tubuhnya.

(S3) : Ani sangat senang memakan **cokelat** pemberian Budi.

(S4) : Budi mempunyai sofa **cokelat** yang sangat mahal

Kata yang memiliki makna ganda (polisemi) pada kalimat satu dan kalimat dua adalah "bisa". kata "bisa" pada kedua kalimat tersebut memiliki makna yang berbeda, dimana kata "bisa" dalam kalimat satu menunjukan "kemampuan seseorang untuk melakukan sesuatu", dan kata "bisa" kalimat dua merujuk pada makna "racun yang berasal dari gigitan ular". Pada contoh permasalahan kalimat satu dan dua, penyelesaian relatif cukup mudah jika dilakukan dengan pendekatan *POS Tagging*. Penggunaan bantuan *task POS Tagging* tersebut dapat langsung membedakan bahwa "bisa" pada kalimat satu adalah kata kerja, sementara "bisa" kalimat kedua adalah objek. Kalimat ketiga dan keempat memberikan contoh lain dari makna yang berbeda pada kata yang sama berdasarkan konteks yang berbeda. Cokelat pada kalimat pertama bermakna makanan manis olahan dari hasil pohon cokelat sementara cokelat kalimat keempat merupakan warna. Kedua kata 'cokelat' tersebut sama-sama mempunyai *POS Tag* berupa *noun*, namun demikian makna yang terkandungnya berbeda.

Pengartian makna untuk sebuah kata sendiri merupakan masalah yang kompleks dan tidak mudah. Sebagai manusia, terdapat berbagai macam cara ataupun sudut pandang untuk merepresentasikan kata dari makna sebuah kalimat/paragraf. Sebagai contoh pisau pada bahasan sebelumnya, kita dapat membedakan makna pisau berdasarkan kegunaannya, manfaatnya, dampaknya, dan lain-lain.

Penggunaan dari *word sense disambiguation* ini dapat bermanfaat untuk *sub task* pada *machine translation*, dimana satu buah kata dapat diterjemahkan menjadi beberapa kemungkinan kata yang berbeda berdasarkan makna yang dikandungnya. Mengacu pada contoh kalimat pertama, kata **bisa** dapat diterjemahkan menjadi*can* atau *could*. Berbeda dengan kalimat kedua yang mana kata **bisa** diterjemahkan menjadi *venom*. Mengetahui makna kata (*sense*) yang tepat dapat mempermudah proses penerjemahan otomatis untuk *unit* kata.

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 mencari fitur yang optimal untuk *wsd system* bahasa Indonesia dan memberikan tambahan *sense* kata-kata dalam bahasa Indonesia yang dapat dipindahkan dari *sense* kata bahasa Inggris.

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* saja, yang mana ini berarti *sense* yang terkait pada kata-kata tidak akan dalam (*shallow*).

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 *WSI*.

2. Perumusan Masalah
   Masalah-masalah yang ada dalam penelitian nantinya dianalisis penyelesaiannya pada tahap ini.

3. Perancangan Pengembangan Korpus
   Proses-proses yang akan dilakukan dalam penelitian pengembangan korpus *Textual Entailment* terlebih dahulu dirancang, seperti pengumpulan dan pengolahan data Wikipedia *revision history*, hingga percobaan Co-training.

4. Implementasi
   Tahap ini adalah inti dari penelitian. Implementasi dilakukan untuk mencari tahu jawaban atas rumusan masalah.

5. Analisis dan Kesimpulan
   Hasil eksperimen dianalisis lebih dalam pada tahapan ini. Kemudian disimpulkan sebagai jawaban dari rumusan masalah.

1.6 Sistematika Penulisan

Sistematika penulisan yang ada dalam laporan penelitian ini sebagai berikut:

- Bab 1 PENDAHULUAN
  Pada bab ini, dijelaskan mengenai latar belakang yang menjadi dasar dari penulisan. Selain itu, bab ini juga menjelaskan mengenai perumusan masalah, tujuan penelitian, tahapan penelitian, ruang lingkup penelitian, metodologi penelitian, serta sistematika penulisan dari penelitian ilmiah ini.

- Bab 2 TINJAUAN PUSTAKA
  Pada bab ini, dijelaskan mengenai teori-teori yang relevan berdasarkan hasil studi literatur yang telah dilakukan. Studi literatur ini menjelaskan tentang *Textual Entailment*, Wikipedia *revision history*, algoritme Co-training dan hal-hal mendasar lainnya yang dibutuhkan dalam penelitian.

- Bab 3 RANCANGAN PENGEMBANGAN KORPUS
  Rancangan pengembangan korpus *Textual Entailment* Bahasa Indonesia akan dibahas pada bab ini. Salah satu pembahasan dalam bab ini adalah bagaimana cara mengubah Wikipedia *revision history* menjadi data kandidat isi korpus *Textual Entailment* dan bagaimana menerapkan Co-training pada permasalahan tersebut.

- Bab 4 IMPLEMENTASI
  Bab ini menjelaskan mengenai implementasi penelitian, mulai dari tahap pengolahan data Wikipedia *revision history* hingga percobaan Co-training.

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

- Bab 6 PENUTUP
  Kesimpulan penelitian dan saran terkait penelitian dijelaskan pada bab ini. Kesimpulan akan memberikan jawaban atas pertanyaan dari rumusan masalah penelitian. Saran diberikan agar penelitian selanjutnya dapat memperbaiki kekurangan-kekurangan dalam penelitian 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 Textual Entailment

Textual Entailment adalah penelitian di bidang NLP yang bertujuan untuk mengidentifikasikan apakah terdapat hubungan *entailment* di antara dua buah teks (Dagan et al., 2006). Sebuah teks yang disebut sebagai T dikatakan meng-*entail* teks lain yang disebut hipotesis atau H apabila makna teks H menjadi benar jika T diketahui benar. Hubungan *textual entailment* menggambarkan bahwa dalam suatu bahasa ada beragam cara untuk menyatakan sebuah makna ke dalam bentuk teks.

Textual entailment memiliki keterkaitan yang erat dengan fenomena parafrase. Parafrase adalah bentuk dari *textual entailment* dua arah, yaitu T *entail* H dan sebaliknya H *entail* T. Sebagai contoh, diberikan teks berikut:

(K1): Ibu pergi ke pasar ditemani oleh ayah.
(K2): Ayah dan ibu pergi bersama.
(K3): Ayah dan ibu pergi bersama ke pasar.
(K4): Ayah tidak menemani ibu ke pasar.
(K5): Ayah membawa motor.

K1 dan K2 memiliki hubungan *textual entailment* ketika K1 diposisikan sebagai T dan K2 sebagai H. Sedangkan, hubungan antara K1 dan K3 adalah parafrase karena K1 *entail* K3 dan sebaliknya K3 *entail* K1. Di sisi lain, K1 tidak *entail* K4 maupun K5 karena makna kedua kalimat tidak dapat diperoleh dari informasi yang diberikan K1.

Pada beberapa penelitian *Textual Entailment*, salah satunya yang dilakukan oleh Giampiccolo et al. (2008), hubungan T tidak *entail* H dibagi menjadi dua jenis, yaitu *contradiction* dan *unknown*. *Contradiction* terjadi apabila makna T dan H saling berlawanan, yaitu T benar maka H menjadi salah dan sebaliknya. *Contradiction* tergambar dari hubungan K1 dengan K4 atau K3 dengan K4. Sedangkan, hubungan *unknown* terjadi apabila kebenaran H tidak dapat ditentukan oleh T, seperti saat K1 menjadi T dan K5 menjadi H.

Menurut Androutsopoulos dan Malakasiotis (2010) ada tiga sudut pandang pekerjaan dalam *Textual Entailment*, yaitu *Textual Entailment Recognition*, *Textual Entailment Extraction*, *Textual Entailment Generation*. *Textual Entailment Recognition* bertujuan untuk mengindentifikasi apakah ada hubungan *entailment* antara sepasang teks, salah satu cara menyelesaikan masalah tersebut adalah dengan klasifikasi. *Textual Entailment Extraction* adalah penelitian yang bertujuan untuk menemukan atau mengekstrak pasangan teks yang berpotensi memiliki hubungan *entailment*. Sedangkan, dalam *Textual Entailment Generation* hubungan *entailment* tidak diidentifikasi melainkan dibangun, misalnya dengan menggunakan *template* berupa pasangan-pasangan teks yang memiliki hubungan *entailment*.

2.1.1 Tingkatan Textual Entailment

Fenomena *textual entailment* dapat terjadi dalam beberapa kategori tingkatan. Bentivogli et al. (2010) di dalam penelitiannya menyimpulkan terdapat lima tingkatan *Textual Entailment*, yaitu leksikal, sintaktik, leksikal-sintaktik, *discourse* dan *reasoning*.

Pada tingkatan leksikal, perbedaan T dan H umumnya hanya berupa perubahan kata tertentu saja, seperti menggantikan sebuah kata dengan sinonim kata tersebut atau mensubstitusi suatu istilah dengan akronimnya. Pada tingkatan sintaktik, T dan H mengalami perubahan struktur tata bahasa. Sedangkan, Hubungan T dan H pada tingkatan leksikal-sintaktik digambarkan melalui perbedaan leksikal maupun tata bahasa teks tersebut. Tingkatan *discourse* mencakup hubungan yang lebih luas daripada sebuah kalimat. Hubungan T dan H pada tingkatan *discourse* dipertimbangkan juga berdasarkan teks yang berada sebelum atau sesudah teks tersebut. Pada tingkatan *reasoning*, dibutuhkan informasi atau pengetahuan lain agar dapat mengidentifikasikan hubungan antara T dan H.

Tabel 2.1 menunjukkan contoh pada masing-masing tingkatan. Variasi fenomena tidak mutlak berada pada tingkatan yang tertera pada tabel. Beberapa fenomena juga tidak ditemui pada Bahasa Indonesia.

Tabel 2.1: Tingkatan Textual Entailment

Tingkatan	Variasi Fenomena	Contoh
leksikal	identitas, format, akronim, demonim, sinonim, oposisi semantik, hipernim, pengetahuan geografis	Akronim T: Fasilkom berada di dekat perpustakaan UI H: Fakultas Ilmu Komputer berada di dekat perpustakaan UI Sinonim T: Masakan buatan ibu sangat lezat H: Masakan buatan ibu sangat sedap
sintaktik	*transparent heads*, nominalisasi/verbalisasi, kausatif, parafrase	Kausatif T: Para pekerja melebarkan jalan H: Para pekerja membuat jalan menjadi lebar
leksikal-sintaktik	negasi, *modifier*, realisasi argumen, aposisi, *list*, kordinasi, aktif-pasif, *alternation*.	Aktif-pasif T: Kue tersebut dimakan adik H: Adik memakan kue itu Aposisi: T: Barack Obama menyukai nasi goreng H: Presiden kulit hitam pertama AS menyukai nasi goreng
discourse	*coreference*, aposisi, *zero anaphora*, *ellipsis*, *statement*.	*Coreference* T: Adik menghabiskan kue tersebut karena adik menyukai kue itu H: Adik menghabiskan kue tersebut karena adik menyukainya
reasoning	oposisi, *modifiers*, *genitive*, klausa relatif, *elliptic expressions*, meronim, metonimia, *reasoning on quantities*, semua penurunan kalimat menggunakan pengetahuan dasar.	Oposisi: T: Sebagian orang menggunakan bus H: Sebagian orang tidak menggunakan bus *Elliptic expression*: T: Ayah makan bakso. Ibu makan bakso. H: Ayah makan bakso, begitu juga ibu. Metonimia: T: Ayah membeli Djarum Coklat H: Ayah membeli rokok

2.1.2 Pemanfaatan Textual Entailment

Keragaman dalam menyatakan suatu makna ke dalam teks adalah fenomena yang lumrah terjadi di bidang *Natural Language Processing* (NLP), seperti dalam pengembangan sistem *Question Answering* (QA), *Information Extraction* (IE), *Information Retrieval* (IR), *Machine Translation*, dan *Summarization* (Dagan dan Glickman, 2004). Ide dari pemanfaatan fenomena *textual entailment* dan parafrase dalam penelitian NLP bahkan muncul sebelum penelitian *Textual Entailment* dikembangkan, seperti yang dilakukan Shinyama dan Sekine (2003) untuk IE *system* dan Radev (2000) untuk sistem *Summarization*.

Framework aplikasi *Textual Entailment* dalam penelitian QA yang umum digunakan adalah merepresentasikan pertanyaan sebagai T dan sejumlah teks calon jawaban direpresentasikan sebagai H (Sacaleanu et al., 2008). Jawaban yang akan diberikan oleh sistem tersebut adalah teks pada calon jawaban yang di-*entail* oleh pertanyaan. Pada salah satu penelitian QA yang dilakukan oleh Harabagiu dan Hickl (2006), beberapa variasi dari *QA system* menggunakan *Textual Entailment* dibandingkan dengan *QA system* biasa. Hasil penelitian menunjukkan semua variasi *QA system* menggunakan *Textual Entailment* lebih baik dibandingkan *QA system* biasa.

Salah satu pekerjaan dalam *Information Extraction* adalah melakukan *pattern matching* (Grishman, 1997). Potongan-potongan informasi dimasukkan ke dalam *pattern* yang sudah ditentukan. Oleh karena itu, performa IE *system* dipengaruhi oleh desain dari *pattern* tersebut. Shinyama dan Sekine (2003) menggunakan metode parafrase untuk memperbanyak variasi *pattern* yang mengekspresikan makna yang sama. Dengan metode parafrase, kelompok *pattern* yang sama bisa diperoleh dengan mudah.

Pada penelitian *Summarization*, aplikasi *Textual Entailment* bahkan dapat bervariasi. Salah satu penelitian mencoba memanfaatkan *Textual Entailment* untuk mengevaluasi hasil rangkuman dari sistem *Summarization* (Bhaskar dan Pakray, 2013). Sedangkan, dalam *Summarization* untuk multi-dokumen, pemanfaatan *Textual Entailment* adalah untuk pendeteksian kemunculan teks yang memiliki makna serupa (Radev, 2000). Teks-teks yang *redundant* akan dihilangkan sehingga rangkuman menjadi lebih ringkas.

2.2 Korpus

Korpus adalah sebuah kumpulan teks dengan format standar yang dapat dibaca oleh mesin elektronis dan dibuat berdasarkan kriteria dan tujuan yang tertentu (Atkins

et al., 1992). Berikut adalah penjelasan mengenai kriteria dan tahap pembuatan korpus, serta contoh pembuatan korpus pada penelitian *Textual Entailment*.

2.2.1 Kriteria dan Tahap Pembuatan Korpus

Secara umum, sebuah korpus yang baik adalah korpus yang berisikan data representatif (Xiao, 2010). Menurut Biber (1993), definisi representatif merujuk pada kemampuan korpus dalam mencakup seluruh variasi fenomena tertentu ke dalam beberapa sampel. Representatif diperoleh apabila data yang tersimpan di korpus merupakan *sampling* dari fenomena di kondisi nyata dan setiap variasi fenomena berjumlah seimbang.

Secara garis besar, ada beberapa tahap dalam pembuatan korpus (Atkins et al., 1992), yaitu: spesifikasi dan desain korpus, pemilihan sumber data korpus, perolehan perizinan data korpus, *data capture*, dan pemrosesan korpus. Pada tahap spesifikasi dan desain, dilakukan penentuan terhadap tujuan, jenis, ukuran, dan atribut lain terkait korpus yang akan dibuat. Kemudian sumber data untuk pembuatan korpus ditentukan dan diperoleh izin penggunaannya. Pada tahap *data capture*, data dari sumber diolah agar dapat berubah menjadi bentuk yang diterima oleh komputer, seperti *scanning* pada kumpulan kertas dokumen atau pembuatan transkrip pada data audio. Proses terakhir adalah pemrosesan data hingga data menjadi bentuk korpus yang diharapkan.

2.2.2 Korpus Textual Entailment

Burger dan Ferro (2005) pernah melakukan pengembangan secara otomatis sebuah korpus *Textual Entailment* bahasa Inggris. Namun, korpus tersebut hanya berisikan data *entailment* yang positif, yaitu T *entail* H. Ide pembuatan korpus tersebut menggunakan *headline* dari sebuah berita dan paragraf pertama berita tersebut sebagai pasangan T dan H dengan asumsi *headline* dan paragraf pertama kemungkinan memiliki hubungan parafrase. Selanjutnya, Dagan et al. (2006) mengembangkan korpus *Textual Entailment* bahasa Inggris secara manual dengan bantuan beberapa anotator. Korpus tersebut berisikan daftar pasangan T dan H yang diberi label secara *binary* yaitu *true* dan *false*, *true* jika T *entail* H dan *false* jika T tidak *entail* H. Kemunculan korpus-korpus tersebut mendorong banyak pihak untuk melakukan penelitian *Textual Entailment*.

Perkembangan *Textual Entailment* bahasa Inggris terus berlanjut hingga sekarang. Bowman et al. (2015) merilis korpus Stanford Natural Language Inference (SNLI), yaitu sebuah *Textual Entailment* korpus berisikan 570 ribu data

dengan tiga label yaitu *entailment*, kontradiksi, dan netral. Ukuran korpus tersebut tergolong besar. Proses pembuatannya dilakukan manual oleh 2500 pekerja. Para pekerja diberikan sebuah teks keterangan foto tanpa diperlihatkan foto tersebut. Mereka diminta untuk membuat kalimat yang mungkin menjadi alternatif untuk menggantikan keterangan foto tersebut.

Kemajuan penelitian *Textual Entailment* bahasa Inggris memberi motivasi penelitian *Textual Entailment* untuk bahasa lain, diantaranya Italia, Jerman, dan Spanyol. Pengembangan *Textual Entailment* untuk ketiga bahasa tersebut dilakukan dengan cara yang berbeda. Pada bahasa Italia, dikembangkan korpus *Textual Entailment* yang diberi nama EVALITA. EVALITA dikembangkan menggunakan data dari Wikipedia *revision history* (Bos et al., 2009). Sedangkan, pengembangan korpus *Textual Entailment* Spanyol menggunakan data dari penilaian sebuah QA *system* yang sudah dikembangkan sebelumnya di Spanyol (Peñas et al., 2006). Korpus *Textual Entailment* Jerman dikembangkan dengan menggunakan metode *crowdsourcing* (Zeller dan Padó, 2013) serupa dengan pengembangan korpus SNLI. Korpus *Textual Entailment* Jepang dibuat dengan mengkombinasikan fitur-fitur *entailment* dari T dan H dalam bahasa Jepang dan terjemahan dalam bahasa Inggris dari pasangan T dan H tersebut, tahap penerjemahaan disebut *bilingual enrichment* (Pham et al., 2012).

2.3 Wikipedia

Wikipedia[1] adalah ensiklopedia multi-bahasa yang dapat ditulis secara kolaboratif oleh siapa pun dari seluruh penjuru dunia (Denoyer dan Gallinari, 2006). Sejak kemunculan pertama Wikipedia tahun 2001, Wikipedia memperoleh pengembangan kuantitas yang sangat pesat. Situs yang dikelola oleh organisasi non profit Wikimedia Foundation ini memiliki 374 juta pengunjung unik setiap bulannya berdasarkan data pada bulan September 2015. Ada sekitar 41 juta artikel dari 294 bahasa berbeda yang ditulis oleh 70.000 kontributor aktif pada Wikipedia. Wikipedia memiliki jumlah artikel, suntingan, dan kolaborator yang sangat besar karena sifatnya yang bebas. Selain itu, proses penyuntingan artikel lebih mudah karena menggunakan Wiki *markup language* yang lebih sederhana daripada bentuk HTML.

[1] www.wikipedia.org

2.3.1 Wikipedia Offline

Selain dapat diakses secara *online*, Wikipedia juga tersedia dalam bentuk *offline* dan dapat diunduh secara gratis melalui situs Wikimedia *dumps*[2]. Wikipedia *offline* tersedia dalam format XML dengan berbagai jenis, dua diantaranya adalah halaman seluruh artikel dan *revision history*. Dokumen XML tersebut dimuat dalam berbagai bahasa, sesuai dengan bahasa yang digunakan Wikipedia, termasuk Bahasa Indonesia.

Gambar 2.1: Contoh halaman Wikimedia *dumps* Bahasa Indonesia

Pada gambar 2.1, angka yang terdapat di judul halaman "idwiki dump progress on 20161201" menunjukkan data yang disimpan adalah data Wikipedia terakhir pada tangga 1 Desember 2016. Isi halaman tersebut adalah daftar berkas XML dari Wikipedia dengan jenis tertentu, ada pula keterangan status, tanggal pengambilan, dan ukuran berkas.

2.3.2 Wiki Markup Language

Artikel-artikel yang terdapat pada berkas XML yang tersedia di halaman Wikimedia tertulis dalam bentuk Wiki *markup language* yaitu sebuah format sederhana untuk melakukan *editing* pada halaman Wikipedia. Penjelasan mengenai penggunaan

[2]https://dumps.wikimedia.org

Wiki *markup language* tersedia di salah satu halaman bantuan Wikipedia[3]. Format yang dapat direpresentasikan dengan Wiki *markup language* antara lain, judul *section*, gambar dan media lainnya (seperti animasi dan suara), tabel, daftar (*list*), *link*, serta format-format untuk menebalkan atau memiringkan teks. Gambar 2.2 menunjukkan potongan artikel dalam berkas XML Wikipedia.

```
  <page>
    <title>Asam deoksiribonukleat</title>
    <ns>0</ns>
    <id>1</id>
    <revision>
      <id>11703617</id>
      <parentid>11422826</parentid>
      <timestamp>2016-06-29T14:26:10Z</timestamp>
      <contributor>
        <username>AABot</username>
        <id>175295</id>
      </contributor>
      <minor />
      <comment>Robot: Perubahan kosmetika</comment>
      <model>wikitext</model>
      <format>text/x-wiki</format>
      <text xml:space="preserve">[[Berkas:DNA Structure+Key+Labelled.pn NoBB.png|thumb|right|340px|
[[Berkas:ADN animation.gif|thumb|Gambaran tiga dimensi DNA]]
'''Asam deoksiribonukleat''', lebih dikenal dengan singkatan '''DNA''' ([[bahasa Inggris]]: '''''d'

Dua unting DNA bersifat anti-paralel, yang berarti bahwa keduanya berpasangan secara berlawanan. Pa

Struktur kimia DNA yang ada membuatnya sangat cocok untuk menyimpan [[informasi]] biologis setiap r

Dalam sel, DNA tersusun dalam [[kromosom]]. Semasa [[pembelahan sel]], kromosom-kromosom ini didupl

Para ilmuwan menggunakan DNA sebagai alat molekuler untuk menyingkap teori-teori dan hukum-hukum fi

== Sifat-sifat DNA ==
[[Berkas:DNA chemical structure_id.svg|thumb|300px|Struktur kimia DNA; [[ikatan hidrogen]] ditunjuk
DNA merupakan sebuah [[polimer]] yang terdiri dari satuan-satuan berulang yang disebut [[nukleotida
| url          = http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=mboc4&amp;part=A2
```

Gambar 2.2: Berkas XML artikel Wikipedia

Pada gambar 2.2, terdapat beberapa contoh penggunaan format Wiki *markup language*.

- Judul *section*

  Judul sebuah *section* diapit dengan simbol sama dengan atau '=' sebanyak tingkatan *section* tersebut. Contoh penggunaan judul *section* pada gambar adalah penulisan "==Sifat-sifat DNA==", yang berarti "Sifat-sifat DNA" merupakan judul *section* tingkat kedua.

- Gambar atau media

  Nama dari berkas gambar atau media, seperti animasi maupun suara, ditulis di dalam tanda '[[' dan ']]'. Apabila ada opsi lain untuk mengatur media tersebut, opsi diletakkan di dalam tanda kurung yang sama namun diposisikan setelah nama berkas dan dipisahkan dengan tanda 'I'. Contoh penggunaan format gambar pada gambar di atas adalah "[[Berkas:ADN

[3]https://en.wikipedia.org/wiki/Help:Wiki_markup

animation.gif|thumb|gambaran tiga dimensi DNA]]". Opsi *thumb* berarti media diposisikan di kanan halaman, kemudian teks berikutnya merupakan opsi keterangan gambar.

- Format teks
  Teks yang mengalami penebalan ditandai dengan *triple quote*. Sedangkan, Teks yang dimiringkan ditandai dengan *double quote*.

- *Link*
  Link ditujukan untuk membuat suatu teks frasa dapat merujuk ke halaman yang terkait apabila diklik. Penulisan *link* diapit oleh tanda kurung '[[' dan ']]', contoh pada gambar adalah "[[koromosom]]", "[[informasi]]", dan "[[pembelahan sel]].

2.3.3 Wikipedia Revision History

Wikipedia *revision history* adalah salah satu jenis dari berkas XML yang tersedia di Wikimedia Dumps.

```
<page>
  <title>Asam deoksiribonukleat</title>
  <ns>0</ns>
  <id>1</id>
  <revision>
    <id>1</id>
    <timestamp>2003-11-07T00:43:23Z</timestamp>
    <contributor>
      <username>proxy8.ntu.edu.sg</username>
      <id>0</id>
    </contributor>
    <minor />
    <comment>*</comment>
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text xml:space="preserve">'Asam deoksiribosanukleat'</text>
    <sha1>8z86w02pi069jvecmz9wxy0rlibfnj0</sha1>
  </revision>
  <revision>
    <id>2</id>
    <parentid>1</parentid>
    <timestamp>2003-11-07T00:43:35Z</timestamp>
    <contributor>
      <username>proxy8.ntu.edu.sg</username>
      <id>0</id>
    </contributor>
    <comment>*</comment>
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text xml:space="preserve">'''Asam deoksiribosanukleat'''</text>
    <sha1>1fjnvhloxoz7o9781dofogu5zoczxer</sha1>
  </revision>
```

Gambar 2.3: Berkas XML Wikipedia Revision History

Berkas XML Wikipedia *revision history* memiliki sedikit perbedaan struktur dengan XML artikel Wikipedia umum (lihat gambar 2.3). Sama seperti pada XML Wikipedia umum, halaman artikel ditandai dengan *tag page*. Sedangkan untuk revisi, berkas XML Wikipedia *revision history* tidak hanya menyimpan revisi terakhir saja, melainkan kondisi awal artikel hingga revisi terakhir. Revisi-revisi tersebut ditandai oleh *tag revision*. Setiap revisi memiliki nomor identitas revisi (di dalam *tag id*) dan identitas induk yang menunjuk pada nomor identitas revisi sebelum artikel tersebut sebelum direvisi (berada dalam *tag parent id*). Ada pula keterangan akun kolaborator atau penulis yang berada dalam *tag username*, serta komentar yang penulis tambahkan setelah melakukan revisi yang berada pada *tag comment*. Isi dari artikel itu sendiri berada di dalam *tag text*.

Menurut Zanzotto dan Pennacchiotti (2010), Wikipedia *revision history* adalah contoh sumber data yang baik yang dapat digunakan untuk membuat korpus *Textual Entailment* karena sifat dari pasangan teks sebelum dan sesudah revisi yang alami, tidak bias untuk pasangan teks yang *lexical overlap*-nya tinggi, konsisten, dan seimbang. Teks sebelum dan sesudah revisi dikatakan alami karena dituliskan langsung oleh kolaborator dengan bahasa mereka masing-masing. Kemudian, revisi antara dua teks yang memiliki *lexical overlap* atau kesamaan leksikal yang tinggi umumnya tidak bias ke salah satu jenis hubungan saja (*entail* atau tidak *entail*). Karena merupakan teks dari sumber yang sama, pasangan-pasangan T dan H yang terbentuk akan homogen dan konsisten. Selain itu, perbandingan pasangan teks yang memiliki hubungan *entailment* maupun yang tidak seharusnya seimbang karena dua di antara beberapa penyebab dilakukannya revisi artikel Wikipedia adalah melengkapi informasi yang kurang dan perbaikan atas ketidak-setujuan dengan konten sebelumnya. Revisi yang dilakukan untuk melengkapi suatu informasi cenderung menggambarkan hubungan *entailment*. Sedangkan, revisi atas ketidaksetujuan akan cenderung menggambarkan hubungan *contra-diction* atau *not entail*.

Selain memiliki kelebihan, Wikipedia *revision history* memiliki beberapa kekurangan, yaitu tidak bisa terhindar dari kesalahan pengejaan dan kasus vandalisme. Kasus vandalisme adalah upaya penambahan, penghapusan, maupun perubahan isi artikel Wikipedia yang dilakukan sengaja untuk mengurangi kualitas artikel tersebut. Contoh vandalisme yang kerap dijumpai pada Wikipedia adalah pengosongan artikel, perubahan informasi benar menjadi salah, dan memasukkan kata-kata vulgar, cacian, atau lelucon dalam tulisan. Kedua jenis permasalahan tersebut mungkin sudah tidak muncul di dalam revisi terakhir artikel, namun Wikipedia *revision history* menyimpan riwayat perubahan-perubahan tersebut.

2.3.4 Pemanfaatan Wikipedia

Wikipedia merupakan salah satu sumber data yang sering digunakan dalam penelitian NLP. Beberapa contoh *task* NLP yang menggunakan data Wikipedia antara lain, *word sense disambiguation*, kategorisasi teks, permasalahan multi-bahasa, *co-reference resolution*, *semantic relatedness* (Medelyan et al., 2009). Selain *task* tersebut, masih banyak *task* terkait NLP yang memanfaatkan data Wikipedia. Penelitian *Textual Entailment* juga dapat menggunakan data Wikipedia, seperti penelitian yang dilakukan Zanzotto dan Pennacchiotti (2010) untuk memperbesar ukuran korpus *Textual Entailment*.

Selain Wikipedia bahasa Inggris, Wikipedia Bahasa Indonesia kerap digunakan pada penelitian NLP Bahasa Indonesia, beberapa diantaranya adalah penelitian terkait NER oleh Luthfi et al. (2014) dan pembuatan korpus paralel bahasa Indonesia-Jawa oleh Trisedya dan Inastra (2014). Hal ini menunjukkan bahwa data Wikipedia Bahasa Indonesia sudah memadai untuk dijadikan sebagai sumber data penelitian.

2.4 Semi-supervised Learning

Secara konvensional, ada dua jenis *task* untuk menurunkan sebuah fungsi dari sekumpulan data di dalam bidang *machine learning* yaitu *supervised learning* dan *unsupervised learning* (Chapelle et al., 2006). *Supervised learning* menggunakan *training data* atau data berlabel yang sepenuhnya dianotasi manual oleh manusia untuk mencari fungsi pemetaan suatu datum. Sedangkan, *unsupervised learning* sama sekali tidak membutuhkan data berlabel, sehingga fungsi didapatkan hanya dari mempelajari struktur data tidak berlabel.

Ada beberapa jenis pembelajaran dalam *supervised learning* berdasarkan keluaran yang dihasilkan, yaitu klasifikasi, regresi, dan *ranking* (Mohri et al., 2012). Klasifikasi digunakan untuk memetakan data ke dalam kategori atau kelas. Regresi memetakan data ke dalam sebuah nilai kuantitatif (bilangan bulat atau riil). Sedangkan, *ranking* digunakan untuk menghasilkan urutan dari sekumpulan data. Klasifikasi memetakan data menggunakan sebuah model yang dinamakan *classifier*, yaitu model yang didapat setelah proses pelatihan dengan *training data*. *Classifier* tersebut dapat digunakan untuk mengklasifikasikan data baru ke dalam sebuah kelas. *Classifier* memiliki tingkat kepercayaan saat mengklasifikasikan sebuah datum. Tingkat kepercayaan tersebut adalah probabilitas kelas tersebut terpilih dalam proses klasifikasi.

Semi-supervised learning adalah sebuah teknik yang merupakan jalan tengah

atau kombinasi antar *supervised learning* dan *unsupervised learning*. Teknik *semi-supervised learning* hanya membutuhkan sedikit data yang dianotasi manusia dan dapat melabeli secara otomatis data yang tersisa (Chapelle et al., 2006). Jenis pembelajaran dalam *semi-supervised learning* serupa dengan *supervised learning*, salah satunya menggunakan adalah klasifikasi (Mohri et al., 2012).

Ada tiga jenis metode untuk menerapkan *semi-supervised learning*, yaitu *bootstrapping*, *graph regulation*, dan *structure learning* (Blitzer dan Zhu, 2008). Sebagian besar model *semi-supervised learning* menggunakan metode *bootstrapping*.

Kode 2.1: Algoritme *bootstrapping*

```
Masukan:
L = data anotasi manual berjumlah kecil (data berlabel)
U = data tidak berlabel

Selama iterasi belum berhenti:
  Latih classifier C dengan L
  Beri label k buah data dari U menggunakan C
  Tambahkan k data baru ke L
```

Kode 2.1 menunjukkan algoritme umum *bootstrapping*. Masukkan untuk algoritme *bootstrapping* adalah sedikit data berlabel dan data tidak berlabel. Data berlabel akan terus bertambah dalam beberapa iterasi menggunakan data baru yaitu data tidak berlabel yang diklasifikasikan pada tiap iterasi.

2.4.1 Co-training

Metode Co-training adalah variasi metode *semi-supervised learning*, yaitu *bootstraping* yang memanfaatkan beberapa data berlabel sebagai bibit untuk memberi label pada data lainnya (Blum dan Mitchell, 1998);(Chapelle et al., 2006);(Blitzer dan Zhu, 2008). Co-training pertama kali diperkenalkan dan digunakan oleh Blum dan Mitchell (1998) sebagai metode untuk klasifikasi halaman *website*. Perbedaan Co-training dengan metode *bootstrapping* umum terletak pada jumlah *view* yang dipertimbangkan dalam klasifikasi. Data yang digunakan dalam Co-training harus memiliki dua buah *views* yang *conditionally independent* sehingga akan ada dua *classifier* yang saling berdiri sendiri (lihat kode 2.2). Penggunaan *view* kedua tersebut yang menginspirasi pemberian nama Co-training untuk metode ini. Pada penelitiannya, Blum dan Mitchell (1998) menggunakan dua *views*, yaitu kata-kata pada halaman *website* dan kata-kata pada link yang merujuk ke halaman tersebut. Sedangkan, *classifier* yang digunakan adalah Naive Bayes.

Kode 2.2: Algoritme Co-training pada penelitian Blum dan Mitchell (1998)

```
Masukan:
  L = data berlabel
  U = data tidak berlabel

U' = sejumlah u data dari U
Selama masih dalam k iterasi:
  Latih classifier h1 dengan data L dari view pertama
  Latih classifier h2 dengan data L dari view kedua
  Gunakan h1  untuk melabeli p data positif dan n data negatif U'
  Gunakan h2  untuk melabeli p data positif dan n data negatif U'
  Tambahkan data berlabel ke dalam L
  Pilih kembali 2p + 2n data U untuk menggantikan data pada U'
```

Seperti metode *bootstrapping* pada umumnya, ada beberapa iterasi di dalam proses Co-training. Iterasi akan terus berjalan hingga mencapai kondisi berhenti atau *stopping condition*. Pada kode 2.2, kondisi berhenti adalah ketika iterasi sudah berjalan sebanyak *k* kali. Pada setiap iterasi dua buah *classifier* dilatih menggunakan data berlabel. Mula-mula data berlabel yang digunakan hanya sedikit, kemudian data akan terus bertambah seiring dengan banyak iterasi yang berjalan. Penambahan data berlabel diambil dari hasil klasifikasi terbaik pada tiap iterasi.

2.4.2 Co-training dalam Textual Entailment

Zanzotto dan Pennacchiotti (2010) melakukan percobaan menggunakan metode Co-training untuk memperbesar ukuran korpus *Textual Entailment*.

Kode 2.3: Algoritme Co-training pada penelitian Zanzotto dan Pennacchiotti (2010)

```
Masukan:
  L = data berlabel
  U = data tidak berlabel

  L1  = L2 =  L
Selama belum bertemu stopping condition:
  Latih classifier h1 dengan data L dari view pertama
  Latih classifier h2 dengan data L dari view kedua
  Klasifikasi U dengan h1 dapatkan U1
  Klasifikasi U dengan h2  dapatkan U2
  Pilih dan hapus k data terbaik dari U1 sebagai u1
  Pilih dan hapus k data terbaik dari U2 sebagai u2
  Tambahkan u1 ke L2 dan u2 ke L1
```

Kode 2.3 menunjukkan algoritme Co-training yang digunakan oleh Zanzotto dan Pennacchiotti (2010). Metode yang digunakan secara garis besar sama dengan Co-training milik Blum dan Mitchell (1998) (lihat kode 2.2). Namun, ada sedikit penyesuaian, misalnya penentuan *stopping condition*, pengambilan data tidak berlabel sebelum memulai iterasi, serta data berlabel yang dihasilkan dari masing-masing *view* dipisahkan dan digunakan untuk melatih *classifier* pada *view* berlawanan (disilangkan).

Menurut Zanzotto dan Pennacchiotti (2010), ada tiga hal yang dapat digunakan sebagai *stopping condition* dari Co-training, yaitu: jumlah data tidak berlabel yang ditambahkan sudah mencapai batas yang ditentukan (Blum dan Mitchell, 1998), penurunan kualitas data berlabel, dan ketika kedua *classifier* tidak mengklasifikasikan data dengan sesuai (Collins dan Singer, 1999).

Percobaan Zanzotto dan Pennacchiotti (2010) menggunakan sumber data Wikipedia *revision history* karena dianggap cocok sebagai sumber pasangan teks T dan H. Perubahan pada tiap revisi Wikipedia menggunakan bahasa yang alami, tidak bias walaupun *lexical overlap*-nya tinggi, dan memiliki pasangan revisi pada suatu artikel konsisten dan homogen. Ditambah lagi, Wikipedia *revision history* dapat menyediakan dua buah *views* yang menjadi syarat untuk penggunaan metode Co-training.

Kedua *view* yang digunakan pada penelitian Zanzotto dan Pennacchiotti (2010) adalah pasangan teks awal dan revisi sebagai pasangan T dan H, serta komentar penulis saat melakukan revisi. Representasi fitur *view* pertama menggunakan bentuk sintaktik kalimat yang diklasifikasikan menggunakan *classifier SVM-light*. Sedangkan, *view* kedua direpresentasikan dengan fitur *bag-of-word* dari bigram dan unigram pada komentar.

2.5 Language Model

Language model adalah fungsi yang memetakan suatu teks bahasa ke dalam sebuah nilai probabilitas(Manning et al., 2008). Mencari sebuah nilai probabilitas didasari oleh sebuah proses perhitungan atau *counting* (Jurafsky dan Martin, 2014). Beberapa jenis *language model* antara lain N-gram dan *word embedding*.

2.5.1 N-gram

N-gram merupakan salah satu *language model* yang menggunakan teknik perhitungan kemunculan *n* kata terurut dalam sebuah teks. N-gram dengan *n* sama dengan 1, 2 dan 3 berturut-turut disebut dengan istilah unigram, bigram dan trigram

(Jurafsky dan Martin, 2014). Selain pada kata, teknik perhitungan N-gram juga dapat diterapkan pada karakter, POS tag, dan lainnya (Sidorov et al., 2014). Teknik ini yang sering digunakan dalam pengolahan teks penelitian NLP terutama yang menggunakan pendekatan *probabilistic*.

(K1): Fasilkom UI berada di dekat perpustakaan

(K2): Mahasiswa Fasilkom UI berkumpul di dekat perpustakaan

(K3): Dia bukan mahasiswa Fasilkom UI

Berdasarkan kalimat K1, K2, K3 berikut adalah contoh unigram, bigram, dan trigram yang terbentuk.

Tabel 2.2: Contoh N-gram

Unigram	Fasilkom, UI, berada, di, dekat, perpustakaan, mahasiswa, berkumpul. ...
Bigram	Fasilkom UI, UI berada, berada di, di dekat, dekat perpustakaan , ...
Trigram	Fasilkom UI berada, UI berada di, berada di dekat, di dekat perpustakaan, mahasiswa Fasilkom UI, ...

Unigram "Fasilkom" dan "UI" muncul paling sering. Untuk bigram, "Fasilkom UI" muncul paling sering di antara bigram lain. Sedangkan, "mahasiswa Fasilkom UI" dan "di dekat perpustakaan" merupakan trigram yang paling sering muncul.

2.5.2 Word Embedding

Representasi kata adalah objek matematis yang berasosiasi dengan suatu kata, biasanya objek tersebut berbentuk vektor yang memuat elemen berupa nilai dari fitur-fitur tertentu, bisa berupa fitur yang berkaitan dengan semantik maupun gramatikal. *Word embedding* sering pula disebut *distributed representations* adalah salah satu jenis dari representasi kata. Bentuk ini memiliki kelebihan, yaitu padat, berdimensi rendah, dan memiliki nilai yang riil. Fitur yang direpresentasikan biasanya berupa nilai sintaktik dan semantik yang terpendam dalam suatu kata. Berbeda dengan jenis *distributional representations* yang umumnya berdimensi besar namun banyak memiliki elemen vektor berupa nilai *null*, *word embedding* mengurangi kemunculan elemen bernilai *null* tersebut sehingga vektor menjadi padat dan berdimensi rendah. Umumnya bentuk ini digunakan dalam *neural language models* atau model berbasis *neural network* (Turian et al., 2010).

Mikolov et al. (2013) memperkenalkan sebuah prosedur untuk membuat *word embedding* menggunakan korpus teks berjumlah besar yang disebut Word2vec (Mikolov et al., 2013). Setiap vektor representasi kata diperoleh berdasarkan hasil dari pembelajaran model *neural network* terhadap korpus yang diberikan. Ada dua buah arsitektur yang diajukan Mikolov untuk pengembangan *word2vec*, yaitu menggunakan model *continuous bag-of-words* (CBOW) dan *continuous skip-gram* (Skip-gram). Perbedaan dari kedua arsitektur tersebut adalah model CBOW memprediksi kata yang diberikan berdasarkan konteks di sekitarnya, sedangkan Skip-gram memprediksi kata-kata di sekitar menggunakan kata yang diberikan. Selain *word2vec*, terdapat teknik pembuatan model *word embedding* lain, salah satunya Glove (Pennington et al., 2014). Glove merupakan teknik pembentukan model *word embedding* menggunakan pendekatan *unsupervised learning*. Implementasi kedua teknik tersebut sudah dipublikasikan, sehingga siapa saja dapat menggunakan teknik tersebut untuk membuat model *word embedding*: Word2vec[4], Glove[5].

2.6 Naive Bayes

Naive Bayes adalah salah jenis *probabilistic classifier* yang berlandaskan pada teorema Bayes. Dalam Naive Bayes, fitur-fitur yang diperhitungkan diasumsikan berdiri sendiri. Secara umum, di dalam Naive Bayes (Manning et al., 2008), peluang suatu data d untuk masuk ke dalam kelas c dapat diestimasi dengan cara berikut.

$$P(c|d) \propto P(c) \prod_{k=1}^{N_d} P(t_k|c) \tag{2.1}$$

dengan Nd adalah jumlah seluruh dokumen, $P(t_k|c)$ adalah peluang dari suatu token t pada data ke-k yang diketahui termasuk dalam kelas c, dan $P(c)$ adalah peluang suatu data masuk ke dalam kelas c. $P(c)$ dapat dihitung sebagai frekuensi data yang ada pada kelas c dibagi dengan jumlah seluruh data.

$$P(c) = \frac{N_c}{N_d} \tag{2.2}$$

Naive Bayes memiliki beberapa variasi klasifikasi, salah satunya adalah Multinomial Naive Bayes. Pada Multinomial Naive Bayes, nilai $P(t_k|c)$ masing-

[4]https://radimrehurek.com/gensim/models/word2vec.html
[5]http://nlp.stanford.edu/projects/glove/

masing token didapatkan melalui perhitungan berikut.

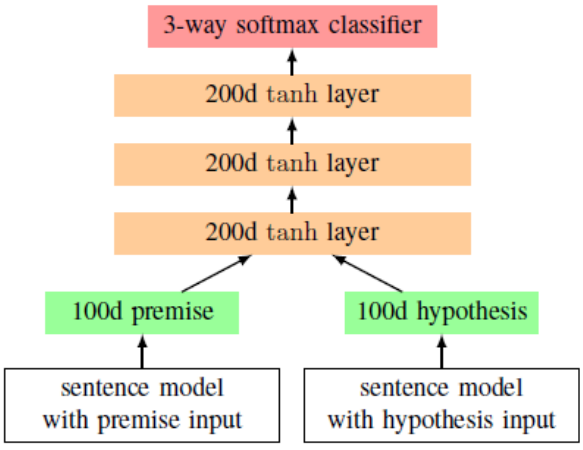
$$P(t_k|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}} \tag{2.3}$$

dengan T_{ct} adalah jumlah kemunculan token t dalam *training data* yang berada di kelas c dan penyebut merupakan jumlah seluruh token t' yang terdapat pada *training data* (disimbolkan dengan V) di kelas c, termasuk token yang muncul lebih dari sekali pada data yang sama.

2.7 Recurrent Neural Network

Recurrent neural networks (RNN) merupakan variasi arsitektur dari neural network yang tergolong dalam jenis *deep learning*. Salah satu hal yang mencirikan RNN adalah bentuknya yang siklik. RNN baik dalam menangani permasalahan dengan data yang berbentuk *sequence* seperti *speech recognition* (Graves et al., 2013) dan *handwriting recognition* (Graves et al., 2009). LSTM merupakan variasi RNN yang diajukan oleh Hochreiter dan Schmidhuber (1997) untuk memperbaiki kelemahan RNN terdahulu, yaitu kemunculan *signal error* saat proses *back-propagation*. Selain menghilangkan permasalahan *error* tersebut. LSTM dapat belajar lebih cepat dibandingkan RNN biasa.

Tahun 2015, Bowman et al. (2015) melakukan percobaan *Textual Entailment* menggunakan *deep learning* dengan korpus SNLI yang mereka kembangkan. Korpus dengan ukuran besar tersebut dianggap mamadai sebagai sumber data *deep learning*. Arsitektur *deep learning* dari percobaan-percobaan tersebut dapat dilihat pada gambar 2.4.



Gambar 2.4: Arsitektur RNN pada *Textual Entailment* History (Bowman et al., 2015)

Ada beberapa variasi percobaan yang Bowman et al. (2015) lakukan, diantaranya arsitektur dikombinasikan dengan RNN biasa dan LSTM-RNN. Hasil percobaan menunjukkan *Textual Entailment* dengan LSTM-RNN lebih unggul dibandingkan dengan RNN biasa, yaitu dengan perbandingan akurasi 77.6% banding 72.2%.

Kemunculan SNLI memicu penelitian *Textual Entailment* dengan menggunakan *deep learning*, antara lain penelitian Rocktäschel et al. (2015), Wang dan Jiang (2016), serta Liu et al. (2016). Tujuan dari penelitian-penelitian tersebut adalah untuk menemukan variasi arsitektur RNN yang dapat meningkatkan akurasi dari penelitian Bowman et al. (2015). Ketiganya mengajukan bentuk arsitektur RNN yang berbeda-beda namun tetap mengujikannya dengan data SNLI.

2.8 Evaluasi

Berdasarkan fungsinya, ada dua jenis evaluasi menurut Galliers dan Spark Jones (Jones dan Galliers, 1995), yaitu evaluasi intrinsik dan ekstrinsik. Evaluasi intrinsik dilakukan terkait tujuan sistem tersebut dibuat, sehingga evaluasi ini bisa dilakukan hanya dengan menilai hasil keluaran yang sistem itu sendiri. Sedangkan evaluasi ekstrinsik dilakukan terhadap peranan dari sistem tersebut, sehingga butuh dilakukan *task* lain untuk menguji hasil yang sistem keluarkan. Sebagai contoh, sebuah penelitian pengembangan korpus *Textual Entailment* dapat dievaluasi menggunakan metode intrinsik dengan cara menilai akurasi dari hasil korpus yang terbentuk. Sedangkan, jika menggunakan pendekatan ekstrinsik, korpus tersebut bisa dinilai dengan cara mengaplikasikan korpus pada *Textual Entailment task*, seperti identifikasi hubungan *entailment* pada *testing data*.

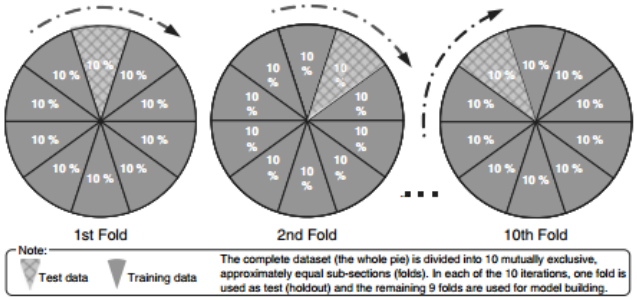
Salah satu cara untuk menghitung suatu nilai evaluasi dari sebuah sistem klasifikasi adalah menggunakan akurasi. Akurasi adalah nilai pembagian dari data yang diklasifikasikan dengan benar terhadap jumlah seluruh data (Manning et al., 2008).

$$akurasi = \frac{jumlah\ data\ diklasifikasikan\ benar}{jumlah\ seluruh\ data} \tag{2.4}$$

2.8.1 Cross Validation

Evaluasi bisa dikombinasikan dengan teknik *cross validation*. *Cross validation* adalah sebuah teknik untuk memanfaatkan sejumlah data yang data tersedia dengan memecah data tersebut menjadi training dan testing data. Untuk mengurangi bias karena proses pemecahan data tersebut, dilakukan *k-Fold cross validation*. *K-*

fold cross validation adalah salah satu jenis penggunaan *cross validation* dengan pengulangan sebanyak *k* kali (Olson dan Delen, 2008).



Gambar 2.5: Ilustrasi *K-fold cross validation* (Olson dan Delen, 2008)

Testing data diambil sejumlah *1/k* dari data total, data selebihnya menjadi *training data*. Lalu pengujian dilakukan dan dihitung akurasinya. Hal tersebut diulangi hingga *k* kali, yaitu ketika semua bagian data pernah menjadi *testing data*. Akurasi keseluruhan adalah rata-rata akurasi pada tiap iterasi.

2.8.2 Kappa

Pada penelitian yang menggunakan anotator, evaluasi perlu dilakukan terhadap tingkat persetujuan antar anotator. Tingkat persetujuan tersebut dihitung menggunakan Kappa (Manning et al., 2008). Berikut adalah persamaan umum suatu kappa.

$$Kappa = \frac{P(A) - P(E)}{1 - P(E)} \tag{2.5}$$

P(A) merupakan proporsi banyaknya nilai yang disetujui (*agreement*) dan P(E) adalah proporsi banyaknya nilai yang disetujui karena ketidaksengajaan.

Landis dan Koch (1977) membagi rentang nilai Kappa menjadi beberapa tingkat persetujuan. Tabel 2.3 menunjukkan pembagian tersebut.

Tabel 2.3: Skala pengukuran Kappa (Landis dan Koch, 1977)

Statistik Kappa	Tingkat persetujuan
<0.00	*poor*
0.00 - 0.20	*slight*
0.21 - 0.40	*fair*
0.41 - 0.60	*moderate*
0.61 - 0.80	*substantial*
0.81 - 1.00	*almost perfect*

Pengukuran Kappa pada tabel di atas kerap digunakan sebagai *benchmark* dalam penelitian-penelitian yang menggunakan anotator.

Kappa memiliki beberapa variasi perhitungan, antara lain Cohen's Kappa dan Fleiss Kappa. Cohen's Kappa digunakan untuk menghitung tingat persetujuan antar dua anotator (Cohen, 1960). Sedangkan, Fleiss Kappa (Fleiss, 1971) bisa digunakan untuk menghitung tingkat persetujuan antara dua atau lebih anotator. Berikut adalah cara perhitungan *P(A)* dan *P(E)* pada Cohen's Kappa.

Tabel 2.4: Perhitungan Cohen's Kappa

		anotator 2			
		label 1	**label 2**	**...**	**label n**
anotator 1	**label 1**	m_{11}	m_{12}	...	m_{1n}
	label 2	m_{21}	m_{22}	...	m_{2n}
	...	...	...	...	...
	label n	m_{n1}	m_{n2}	...	m_{nn}

$$P(A) = \frac{\sum_{k=1}^{n} m_{kk}}{total\,data} \tag{2.6}$$

$$P(E) = \frac{\sum_{k=1}^{n} (\sum_{j=1}^{n} m_{kj} \cdot \sum_{i=1}^{n} m_{ik})}{total\,data} \tag{2.7}$$

dengan n merupakan banyak label, dan m_{ij} merupakan banyaknya data yang diberi label i oleh anotator 1 dan label j oleh anotator 2. Sedangkan, berikut adalah cara perhitungan *P(A)* dan *P(E)* pada Fleiss Kappa.

$$P(A) = \frac{1}{N}\sum_{i=1}^{N} P_i \quad dengan \quad P_i = \frac{1}{n(n-1)}[(\sum_{j=1}^{k} n_{ij}^2) - (n)] \tag{2.8}$$

$$P(E) = \sum_{i=1}^{k} P_j^2 \tag{2.9}$$

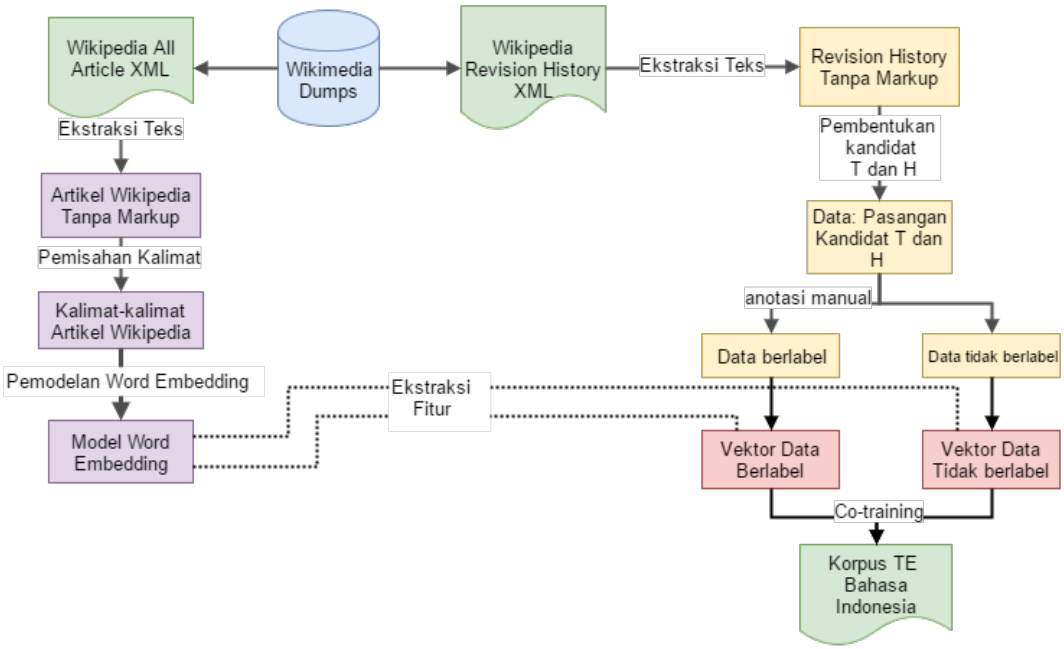
dengan N merupakan jumlah data yang dianotasi, k adalah jumlah label, n adalah jumlah anotator, dan n_{ij} adalah total anotator yang memberi label j pada data ke-i.

BAB 3
RANCANGAN PENGEMBANGAN KORPUS

Pada bab ini, dijelaskan mengenai rancangan dan tahap-tahap pengembangan korpus *Textual Entailment* Bahasa Indonesia.

3.1 Rancangan Pengembangan Korpus

Pengembangan korpus *Textual Entailment* dalam penelitian ini mengikuti arsitektur yang dapat dilihat pada gambar 3.1.



Gambar 3.1: Rancangan Arsitektur Pengembangan Korpus *Textual Entailment*

Seperti yang terlihat pada gambar 3.1, sumber utama data untuk penelitian ini adalah data Wikipedia Bahasa Indonesia yang dapat dilihat di situs Wikimedia[1]. Wikipedia *dumps* adalah bentuk statis dari Wikipedia yang dibuat oleh Wikimedia. Ada dua jenis data yang digunakan untuk penelitian ini, yaitu XML seluruh artikel Wikipedia dan XML Wikipedia *revision history*. Berkas XML Wikipedia seluruh artikel digunakan untuk membuat model *word embedding* yang akan digunakan sebagai salah satu ekstraksi fitur pada penelitian ini. Sedangkan, data Wikipedia

[1]dumps.wikimedia.org/idwiki/

revision hisotry adalah data utama dalam pengembangan korpus *Textual Entailment* menggunakan metode Co-training.

Secara garis besar, tahap dalam pengembangan korpus terbagi menjadi 3 bagian, yaitu pemodelan *word embedding*, pembentukan kandidat pasangan T dan H, serta proses Co-training. Pemodelan *word embedding* adalah proses membuat model *word embedding* menggunakan data artikel Wikipedia. Model ini akan digunakan untuk mengekstraksi fitur utama dari data sebelum diproses dalam Co-training. Pembentukan kandidat pasangan T dan H adalah proses mengubah dan mengekstrak data Wikipedia *revision history* menjadi data yang hanya menyimpan informasi T dan H, serta komentar penulis. Sedangkan, Co-training adalah proses inti dari penelitian ini, yaitu pengolahan data tidak berlabel menjadi data berlabel yang digunakan sebagai penyusun isi korpus *Textual Entailment* Bahasa Indonesia.

Berikut adalah penjelasan singkat mengenai tahapan pada gambar 3.1.

1. Data Wikipedia yang akan digunakan pada penelitian diunduh dari situs Wikimedia *dumps* Bahasa Indonesia.

2. Data Wikipedia diekstrak, yaitu teks dibersihkan dari format Wiki *markup language* dan bagian dari berkas XML yang diambil hanya yang mengandung informasi-informasi yang dibutuhkan, seperti *tag text, revision, id*, dan *comment*.

3. Data artikel Wikipedia tanpa format *markup* diolah menjadi model *word embedding*. Proses pengolahan terdiri dari dua langkah, yaitu memenggal artikel menjadi kalimat dan melatih model *word2vec* dengan kalimat-kalimat untuk membentuk model *word embedding*.

4. Data Wikipedia *revision history* yang bersih diolah menjadi pasangan T dan H. Ada banyak proses yang dilakukan pada tahap ini, proses tersebut dijelaskan pada bagian 3.2.

5. Data kandidat pasangan T dan H dikelompokkan menjadi dua, data berlabel dan data tidak berlabel. Untuk mendapatkan data berlabel perlu dilakukan anotasi manual.

6. Data berlabel dan tidak berlabel diekstrak fitur-fiturnya, salah satu ekstraksi fitur akan berbentuk *word embedding*. Untuk mengubah data menjadi fitur *word embedding*, digunakan model *word embedding* yang sebelumnya sudah dibuat.

7. Data berlabel dan tidak berlabel yang sudah diekstrak ke dalam bentuk vektor fitur kemudian dimasukkan ke dalam proses Co-training. Di proses tersebut, data tidak berlabel akan diberi label secara otomatis. Semua data berlabel yang terbentuk setelah Co-training berhenti disesuaikan formatnya agar menjadi korpus *Textual Entailment* Bahasa Indonesia.

Keluaran yang diharapkan dari penelitian ini adalah korpus *Textual Entailment* dengan format seperti pada gambar 3.2.

```
<pair id=1>
        <T> Ini adalah contoh kalimat T pertama. </T>
        <H> Ini adalah contoh kalimat H pertama. </H>
        <V>E</V>
</pair>
<pair id=2>
        <T>Ini adalah contoh kalimat T kedua.</T>
        <H>Ini adalah contoh kalimat H kedua.</H>
        <V>C</V>
</pair>
<pair id=3>
        <T>Ini adalah contoh kalimat T ketiga.</T>
        <H>Ini adalah contoh kalimat H ketiga.</H>
        <V>U</V>
</pair>
....
```

Gambar 3.2: Contoh isi korpus *Textual Entailment* yang diharapkan

Korpus *Textual Entailment* berisikan pasangan-pasangan T dan H beserta nilai *entailment*-nya. Pada penelitian ini, masing-masing T dan H berbentuk sebuah kalimat. Kalimat T disimpan dalam *tag T*; sedangkan, kalimat H disimpan dalam *tag H*. Nilai *entailment* yang diharapkan adalah salah satu dari tiga jenis label E, U, dan C. Nilai tersebut tersimpan dalam *tag V*. Label E merepresentasikan T *entail* H, label C merepresentasikan hubungan T dan H yang kontradiksi, dan label U untuk T dan H yang tidak berhubungan.

3.2 Pembentukan Kandidat T dan H

Data yang diterima oleh proses Co-training untuk membuat isi korpus *Textual Entailment* adalah kandidat pasangan-pasangan T dan H. Pembentukan kandidat T dan H adalah tahap pemrosesan data Wikipedia *revision history* yang sudah diekstrak agar menghasilkan kandidat pasangan-pasangan T dan H tersebut. Tahap ini terdiri dari sejumlah proses yang akan dijelaskan sebagai berikut.

1. **Penghapusan kata-kata yang mengarah ke vandalisme**
   Wikipedia *revision history* tidak dapat terlepas dari kasus vandalisme. Oleh

karena itu, agar dapat mengurangi jumlah pasangan T dan H yang berbeda hanya karena kemunculan kata-kata *vandal* dilakukan penghapusan kata-kata yang mengandung unsur vandalisme tersebut. Untuk mengimplementasikan penghapusan kata-kata tersebut, daftar kata-kata tidak sopan dibuat secara manual. Kata-kata yang termasuk dalam kata yang akan dihapus adalah kata-kata vulgar, lelucon, atau mengandung unsur cacian, namun tidak tergolong ke dalam kata yang umum dipakai. Contohnya, kata seperti "anjing", kata tersebut sering digunakan untuk mencaci, tetapi tidak dapat dimasukkan ke dalam daftar karena kata tersebut tergolong kata umum apabila konteks pembicaraan yang digunakan sedang membahas mengenai hewan anjing.

Tahap ini tidak ditujukan menghilangkan seluruh kasus vandalisme karena hanya ditujukan untuk menghilangkan kata-kata *vandal* yang muncul pada teks tanpa mengubah makna teks. Contohnya, pada kalimat "Chairil Anwar adalah seorang penulis puisi hahahaha", terdapat kemunculan kata lelucon "hahahaha". Kata tersebut muncul tanpa merusak struktur kalimat, yaitu apabila kata tersebut dihapus, struktur kalimat menjadi benar. Sedangkan, kalimat "Andi Hermanto Ganteng adalah salah satu mantan presiden RI" juga merupakan kasus vandalisme. Kasus tersebut tidak dapat dideteksi di tahap ini. Kasus vandalisme tersebut dapat digunakan menjadi pasangan T dan H bersama dengan kalimat perbaikannya.

2. **Pemasangan teks induk dan teks revisi**

   Teks yang diterima pada tahap ini adalah teks Wikipedia *revision history* yang sudah bersih dari format Wiki *markup languange* seperti pada gambar 3.3.

```
<doc id="1">
    <revision id="0">
        <text> Asam deoksiribonukleat </text>
        <comment></comment>
    </revision>
    <revision id="1" parent_id="0">
        <text> Asam deoksiribonukleat (DNA) </text>
        <comment>copyedit</comment>
    </revision>
    <revision id="2" parent_id="1">
        <text> Asam deoksiribonukleat, lebih dikenal dengan singkatan DNA
        (bahasa Inggris: deoxyribonucleic acid), adalah sejenis biomolekul
        yang menyimpan dan menyandi instruksi-instruksi genetika setiap
        organisme dan banyak jenis virus. </text>
        <comment>tambahan definisi</comment>
    </revision>
    ...
</doc>
```

Gambar 3.3: Salah satu artikel Wikipedia beserta daftar revisinya

Pada gambar 3.3, sebuah dokumen artikel ditandai oleh *tag doc*. Di dalam

artikel terdapat sebuah teks awal, yaitu teks pertama untuk artikel tersebut sebelum mengalami revisi, dan beberapa teks revisi. Perbedaan antara teks awal dan revisi-revisi artikel ditandai dengan kemunculan atribut *parent_id*. Teks awal tidak memiliki *parent_id*; sedangkan, teks revisi memiliki sebuah *parent_id* yang merujuk pada teks dengan *id* tersebut. Teks yang dirujuk merupakan kondisi artikel mula-mula sebelum mengalami revisi, untuk seterusnya teks yang dirujuk tersebut akan disebut sebagai teks induk.

Pada proses ini, masing-masing teks revisi dipasangkan dengan teks induk. Pemasangan teks tidak dilakukan secara transitif. Misalnya, berdasarkan gambar 3.3, teks revisi dengan *id* 1 dipasangkan dengan teks induknya yaitu revisi dengan *id* 0. Sedangkan, teks revisi *id* 2 dipasangkan dengan teks revisi *id* 1. Namun revisi *id* 2 tidak dipasangkan dengan teks induk revisi 1.

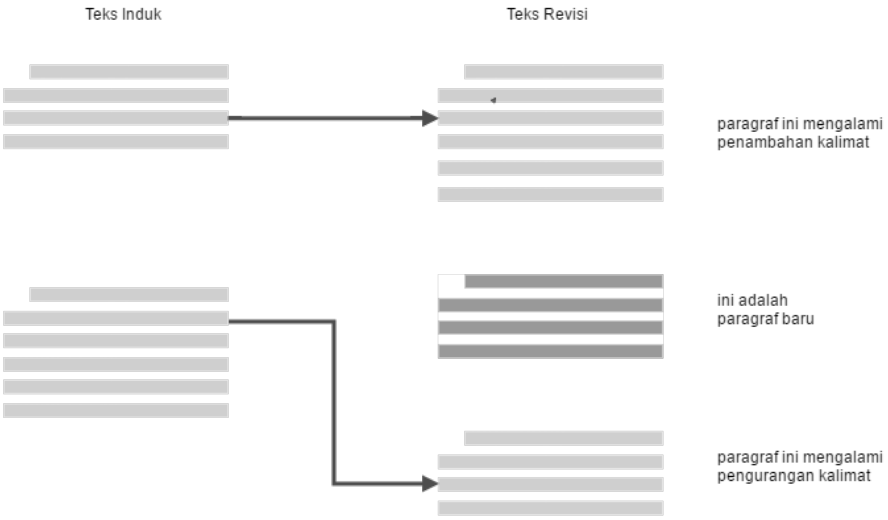
```
<text_pair id=1>
    <teks_induk>
        Asam Deoksiribonukleat
    </teks_induk>
    <teks_revisi>
        Asam Deoksiribonukleat (DNA)
    </teks_revisi>
    <comment>copyedit</comment>
</text_pair>
<text_pair id=2>
    <teks_induk>
        Asam Deoksiribonukleat (DNA)
    </teks_induk>
    <teks_revisi>
        Asam deoksiribonukleat, lebih dikenal dengan singkatan DNA
        (bahasa Inggris: deoxyribonucleic acid), adalah sejenis
        biomolekul yang menyimpan dan menyandi instruksi-instruksi
        genetika setiap organisme dan banyak jenis virus.
    </teks_revisi>
    <comment>tambahan definisi</comment>
</text_pair>
```

Gambar 3.4: Contoh hasil pemasangan teks induk dan revisi

Gambar 3.4 menunjukkan hasil pemasangan teks induk dan revisi dari contoh dokumen di gambar 3.3. Informasi mengenai komentar penulis yang tersimpan dalam sebuah revisi diikutsertakan dalam pemasangan.

3. **Penyesuaian paragraf**

   Proses ini dilakukan pada setiap pasangan artikel induk dan revisi. Paragraf antar kedua artikel yang membicarakan suatu konteks yang sama disesuaikan (ilustrasi lihat di gambar 3.5).



Gambar 3.5: Ilustrasi Penyesuaian Paragraf

Sepasang paragraf yang sesuai tidak selalu memiliki teks yang sama persis atau identik. Paragraf tersebut bisa saja mengalami, penambahan, pengurangan, atau perubahan kalimat. Perbedaan-perbedaan antara kedua paragraf yang sesuai tersebutlah yang akan berpotensi menjadi pasangan T dan H. Gambar 3.6 menunjukkan contoh penyesuaian paragraf.

Teks Induk		Teks Revisi
Universitas Indonesia, biasa disingkat sebagai UI, adalah sebuah peguruan tinggi di Indonesia. Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan. Universitas Indonesia adalah kampus moderm, komprehensif, tebuka, multi budaya, dan humanis yang mencakup disiplin ilmu yang luas. UI telah menghasilkan banyak alumni. Secara umum UI dianggap sebaai salah satu dari tiga perguruan tinggi papan atas di Indonesia bersama dengan Universitas Gadjah Mada da Institur Teknologi Bandung.	→	Universitas Indonesia, biasa disingkat sebagai UI, adalah sebuah perguruan tinggi di Indonesia. Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan, dan kampus lainnya terdapat di daerah Salemba di Jakarta Pusat. Universitas Indonesia adalah kampus moderm, komprehensif, tebuka, multi budaya, dan humanis yang mencakup disiplin ilmu yang luas. UI telah menghasilkan meluluskan 400.000 alumni. Jumlah alumni UI tersebut tergolong banyak. Secara umum UI dianggap sebaai salah satu dari tiga perguruan tinggi papan atas di Indonesia bersama dengan Universitas Gadjah Mada da Institur Teknologi Bandung.
		Sejarah
Sejarah Universitas Indonesia dapat ditelusuri sejak tahun 1849. Ketika itu, pemerintah kolonial Belanda mendirikan sebuah sekolah yang bertujuan untuk menghasilkan asisten dokter tambahan. Lulusannya diberikan sertifikat untuk melakukan perawatan-perawatan tingkat dasar serta mendapatkan gelar Dokter Jawa (Javanese Doctor).	→	Sejarah Universitas Indonesia dapat ditelusuri sejak tahun 1849. Ketika itu, pemerintah kolonial Belanda mendirikan sebuah sekolah yang bertujuan untuk menghasilkan asisten dokter tambahan. Pelajar di sekolah itu mendapatkan pelatihan kedokteran selama dua tahun. Lulusannya diberikan sertifikat untuk melakukan perawatan-perawatan tingkat dasar serta mendapatkan gelar Dokter Jawa (Javanese Doctor).
		Pada tahun 1898, pemerintah kolonial mendirikan sekolah baru untuk melatih tenaga medis, yaitu STOVIA (School tot Opleiding van Indische Artsen). Pendidikan di STOVIA berlangsung selama 9 tahun: 3 tahun setingkat SMP, tiga tahun setingkat SMA, dan tiga tahun lainnya setingkat Diploma.

Gambar 3.6: Contoh Penyesuaian Paragraf

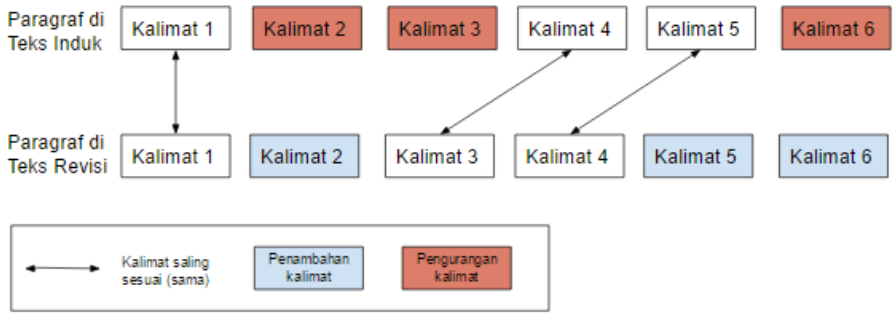
Tujuan dari penyesuaian paragraf adalah untuk mengidentifikasikan pasangan

paragraf yang membicarakan konteks yang sama. Hal ini penting diketahui karena teks T dan H tentu berada dalam konteks pembicaraan yang sama agar dapat memiliki hubungan *entailment*.

Selain dapat menyesuaikan berdasarkan konteks pembicaraan, proses penyesuaian paragraf dapat mengidentifikasikan penambahan maupun penghapusan paragraf. Paragraf pada teks induk yang tidak memiliki pasangan pada teks revisi berarti mengalami penghapusan. Sedangkan, paragraf pada teks revisi yang tidak memiliki pasangan pada teks induk berarti baru ditambahkan. Informasi penambahan dan pengurangan paragraf penting untuk diketahui karena teks pada pengurangan dan penambahan paragraf tidak akan digunakan untuk pembentukan T dan H.

4. **Penyesuaian kalimat**

   Kalimat-kalimat pada pasangan paragraf yang sesuai kemudian dicocokan. Berbeda dengan penyesuaian paragraf, pasangan kalimat yang sesuai antar paragraf adalah kalimat yang sama, yaitu tidak mengalami penambahan, pengurangan, maupun perubahan kata. Namun, pasangan kalimat yang berbeda hanya karena kesalahan penulisan atau tipografi tetap dianggap pasangan kalimat yang sama. Gambar 3.7 menunjukkan ilustrasi penyesuaian kalimat.



Gambar 3.7: Ilustrasi Penyesuaian Kalimat

Tujuan penyesuaian kalimat adalah untuk mengidentifikasi penghapusan dan penambahan kalimat baru. Penghapusan kalimat pada paragraf di teks induk terjadi apabila tidak mendapatkan pasangan di teks revisi. Sebaliknya, penambahan kalimat pada paragraf di teks revisi terjadi jika kalimat tersebut tidak memiliki pasangan di teks induk. Pada proses ini, informasi penambahan dan pengurangan kalimat menjadi penting untuk disimpan. Penambahan dan pengurangan kalimat tersebut yang akan digunakan sebagai

kandidat T dan H.

Paragraf di teks induk	Paragraf di teks revisi
Universitas Indonesia, biasa disingkat sebagai UI, adalah sebuah **peguruan** tinggi di Indonesia. \\ Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan. Universitas Indonesia adalah kampus modern, komprehensif, terbuka, multi budaya, dan humanis yang mencakup disiplin ilmu yang luas. \\ UI telah menghasilkan banyak alumni. \\ Secara umum UI dianggap sebagai salah satu dari tiga perguruan tinggi papan atas di Indonesia bersama dengan Universitas Gadjah Mada dan Institut Teknologi Bandung. \\	Universitas Indonesia, biasa disingkat sebagai UI, adalah sebuah **perguruan** tinggi di Indonesia. \\ Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan, dan kampus utama lainnya terdapat di daerah Salemba di Jakarta Pusat. \\ Universitas Indonesia adalah kampus modern, komprehensif, terbuka, multi budaya, dan humanis yang mencakup disiplin ilmu yang luas. \\ UI telah meluluskan 400.000 alumni. \\ Jumlah alumni UI tersebut tergolong banyak. \\ Secara umum UI dianggap sebagai salah satu dari tiga perguruan tinggi papan atas di Indonesia bersama dengan Universitas Gadjah Mada dan Institut Teknologi Bandung. \\
Penghapusan Kalimat	**Penambahan Kalimat**
Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan.	Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan, dan kampus utama lainnya terdapat di daerah Salemba di Jakarta Pusat.
UI telah menghasilkan banyak alumni.	UI telah meluluskan 400.000 alumni.
	Jumlah alumni UI tersebut tergolong banyak.

Gambar 3.8: Contoh Penyesuaian Kalimat

Gambar 3.8 menunjukkan contoh penyesuaian kalimat. Informasi yang harus disimpan adalah penambahan dan pengurangan kalimat. Kalimat yang berbeda karena kesalahan penulisan akan dianggap sama, seperti kalimat pertama pada contoh.

5. **Pemasangan kalimat**

   Pemasangan kalimat dilakukan dengan menggunakan kalimat akibat penambahan dan pengurangan yang sudah teridentifikasi di proses sebelumnya, serta mengabaikan kalimat-kalimat yang bersesuaian (gambar 3.7). Pemasangan kalimat dibatasi dalam satu kelompok perubahan kalimat yang sama. Sebuah kelompok kalimat terdiri dari kalimat akibat penambahan dan pengurangan yang dibatasi oleh pasangan kalimat sesuai yang sama. Tujuan dari pengelompokan adalah untuk lebih mengecilkan konteks pembicaraan antara kedua teks, dengan asumsi kalimat pada kelompok yang sama mungkin merupakan kalimat-kalimat yang memiliki hubungan substitusi.

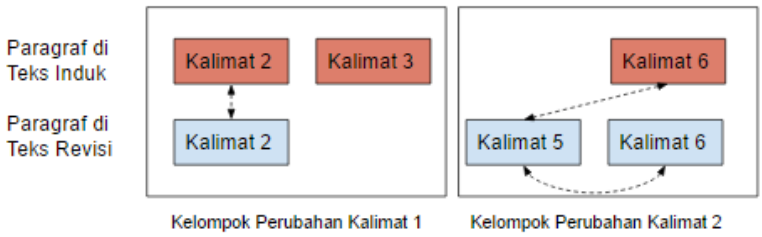
   Masing-masing kalimat pada kelompok yang sama dipasangkan dengan aturan berikut.

   - Kalimat yang dihapus dari teks induk dipasangkan dengan kalimat tambahan di teks revisi dengan posisi yang sama. Hal ini dilakukan

dengan asumsi sepasang kalimat tersebut dapat saling menggantikan (substitusi)

- Apabila kalimat yang dihapus berjumlah lebih banyak daripada kalimat yang ditambahkan, kalimat penghapusan di posisi setelah pemasangan terakhir akan diabaikan. Hal ini dilakukan dengan pertimbangan bahwa kalimat yang dihapus mungkin saja dihilangkan karena tidak penting, vandalisme, atau kasus tidak diinginkan lainnya.
- Apabila kalimat yang ditambah berjumlah lebih banyak daripada kalimat yang dihapus, kalimat penambahan di posisi setelah pemasangan terakhir dipasangankan dengan kalimat sebelumnya. Hal ini dilakukan dengan asumsi bahwa kalimat tersebut merupakan penjelas dari kalimat sebelumnya,

Gambar 3.9 adalah ilustrasi pemasangan kalimat berdasarkan aturan di atas. Ilustrasi ini merupakan lanjutan dari ilustrasi pada gambar 3.7.



Gambar 3.9: Ilustrasi Pemasangan Kalimat

Setelah kalimat dipasangkan, perlu dilakukan penentuan peran kalimat, yaitu menentukan kalimat mana yang berperan sebagai T dan kalimat mana yang berperan sebagai H. Pada penelitian ini, penentuan peran T dan H menggunakan metode yang sederhana yaitu membandingkan jumlah kata pada kalimat (panjang kalimat) antara teks induk dan teks revisi. Pasangan T dan H yang diharapkan adalah ketika T memiliki informasi yang lebih banyak atau sama dengan H. Di antara sepasang kalimat, kalimat yang lebih panjang berperan sebagai T dan selebihnya adalah H. Apabila panjang kalimat sama, maka secara *default* T diperankan oleh kalimat pada teks revisi dengan asumsi revisi umumnya dilakukan untuk menambah informasi baru atau memberi penjelasan terkait teks sebelumnya. Berikut adalah contoh pemasangan kalimat yang merupakan lanjutan dari contoh 3.8.

Kelompok Kalimat 1	
penghapusan	**penambahan**
Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan.	Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan, dan kampus utama lainnya terdapat di daerah Salemba di Jakarta Pusat.
Kelompok Kalimat 2	
penghapusan	**penambahan**
UI telah menghasilkan banyak alumni.	UI telah meluluskan 400.000 alumni.
	Jumlah alumni UI tersebut tergolong banyak.
Kandidat T	**Kandidat H**
Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan, dan kampus utama lainnya terdapat di daerah Salemba di Jakarta Pusat.	Kampus utamanya terletak di bagian Utara dari Depok, Jawa Barat tepat di perbatasan antara Depok dengan wilayah Jakarta Selatan.
UI telah meluluskan 400.000 alumni.	UI telah meluluskan banyak alumni.
Jumlah alumni UI tersebut tergolong banyak.	UI telah meluluskan 400.000 alumni.

Gambar 3.10: Contoh Pemasangan Kalimat

Pada contoh di atas, terdapat dua kelompok kalimat. Pada masing-masing kelompok kalimat, dilakukan pemasangan seperti yang sudah dijelaskan. Setelah pasangan kalimat didapatkan, dilakukan penentuan peran. Ada tiga pasang T dan H pada contoh 3.10. Pasangan pertama didapatkan dari kelompok kalimat pertama dengan T diperankan oleh kalimat yang lebih panjang. Pasangan kedua didapatkan dari kelompok kalimat kedua pada posisi pertama di teks induk dan revisi. Kalimat pada teks revisi berperan sebagai T ketika jumlah kata pada pasangan kalimat tersebut sama. Pasangan ketiga didapatkan dari kelompok kedua dengan menghubungkan kalimat di teks revisi pada posisi kedua dengan kalimat di teks revisi posisi pertama. Hal tersebut dilakukan karena kalimat posisi kedua di teks revisi tidak memiliki pasangan di teks induk. T diperankan oleh kalimat yang lebih panjang.

Pada akhir proses pembentukan kandidat T dan H, komentar penulis ditambahkan pada setiap pasang kandidat. Komentar penulis yang ditambahkan adalah yang disimpan pada penjelasan poin pertama. Selanjutnya, kandidat pasangan T dan H ini akan menjadi masukan pada proses Co-training. Gambar 3.11 menunjukkan contoh hasil dari tahap pembentukan T dan H.

```
<pair id=1>
    <T>Asam deoksiribonukleat atau lebih dikenal dengan singkatan DNA (
    bahasa Inggris : deoxyribonucleic acid ) adalah sejenis asam nukleat yang
    tergolong biomolekul utama penyusun berat kering setiap organisme .</T>
    <H>Asam deoksiribonukleat , lebih dikenal dengan DNA ( bahasa Inggris :
    deoxyribonucleic acid ) , adalah sejenis asam nukleat yang tergolong
    biomolekul utama penyusun berat kering setiap organisme .</H>
    <C>kategori</C>
    <V></V>
</pair>
<pair id=2>
    <T>DNA pertama kali berhasil dimurnikan pada tahun 1868 oleh ilmuwan
    Swiss Friedrich Miescher di Tubingen , Jerman , yang menamainya nuclein
    berdasarkan lokasinya di dalam inti sel .</T>
    <H>DNA pertama kali berhasil dimurnikan pada tahun 1868 oleh ilmuwan
    Swiss Friedrich Miescher , yang menamainya nuclein berdasarkan lokasinya
    di dalam inti sel .</H>
    <C>sejarah</C>
    <V></V>
</pair>
```

Gambar 3.11: Contoh hasil pasangan T dan H

Setiap pasangan T dan H ditandai dengan *tag pair*. Selain pasangan T dan H, terdapat pula informasi berupa komentar saat revisi yang ditandai *tag C* serta *tag V* yang menandai label *entailment* untuk pasangan tersebut.

3.3 Anotasi Manual

Anotasi manual dilakukan pada sebagian kecil data pasangan kandidat T dan H yang dihasilkan setelah proses pembentukan kandidat T dan H pada bagian 3.2. Data yang dianotasi manual akan menjadi calon data bibit untuk proses Co-training. Sejumlah *n* data yang dipilih secara acak akan diberi label antara E, U, dan C. Label E menyimpan hubungan bahwa T *entail* H, U menyatakan bahwa T dan H tidak memiliki hubungan, sedangkan C menggambarkan hubungan kontradiksi antara T dan H. Berdasarkan gambar 3.11, anotasi dilakukan dengan meletakkan nilai *entailment* E, C, atau U di dalam *tag V*, sehingga hasil yang diperoleh menjadi seperti pada gambar 3.12.

```
<pair>
    <T>Asam deoksiribonukleat atau lebih dikenal dengan singkatan DNA (
    bahasa Inggris : deoxyribonucleic acid ) adalah sejenis asam nukleat
    yang tergolong biomolekul utama penyusun berat kering setiap
    organisme .</T>
    <H>Asam deoksiribonukleat , lebih dikenal dengan DNA ( bahasa Inggris
    : deoxyribonucleic acid ) , adalah sejenis asam nukleat yang
    tergolong biomolekul utama penyusun berat kering setiap organisme .</H>
    <C>kategori</C>
    <V>E</V>
</pair>
<pair>
    <T>DNA pertama kali berhasil dimurnikan pada tahun 1868 oleh ilmuwan
    Swiss Friedrich Miescher di Tubingen , Jerman , yang menamainya
    nuclein berdasarkan lokasinya di dalam inti sel .</T>
    <H>DNA pertama kali berhasil dimurnikan pada tahun 1868 oleh ilmuwan
    Swiss Friedrich Miescher , yang menamainya nuclein berdasarkan
    lokasinya di dalam inti sel .</H>
    <C>sejarah</C>
    <V>E</V>
</pair>
```

Gambar 3.12: Contoh hasil anotasi manual

Agar anotasi tidak subjektif dan lebih mudah pengerjaannya, anotasi tidak dilakukan oleh penulis sendiri, melainkan dilakukan bersama anotator lainnya. Pasangan data kandidat T dan H dianotasi oleh 3 anotator berbeda agar label pada data tersebut tidak subjektif. Agar pekerjaan menjadi lebih mudah, data dibagi sedemikian rupa, sehingga ada beberapa anotator yang terlibat dalam proses ini, salah satunya adalah penulis sendiri. Penulis melakukan anotasi untuk seluruh data, sedangkan anotator lainnya mendapatkan pembagian dengan jumlah yang ditentukan. Berikut adalah langkah-langkah dalam melakukan anotasi data dengan sejumlah anotator:

1. Pembuatan panduan anotasi sebagai pedoman dalam melakukan anotasi terutama pada kasus-kasus yang tingkat ambiguitasnya tinggi. Panduan anotasi dibuat oleh penulis dengan asumsi bahwa permasalahan ini telah dipahami oleh penulis.

2. Perekrutan anotator agar anotasi data tidak menjadi subjektif. Beberapa anotator direkrut untuk membantu pengerjaan anotasi. Sebelumnya, anotator tersebut akan diuji untuk melabeli beberapa data tidak berlabel. Anotator yang tingkat persetujuan-nya baik kemudian akan lanjut untuk menganotasi data yang sebenarnya.

3. Penentuan batas persetujuan untuk tahap uji coba. Batas persetujuan adalah batasan nilai persetujuan terendah yang menentukan apakah anotator tersebut dapat melanjutkan ke tahap anotasi data sebenarnya atau tidak. Batasan tersebut ditentukan berdasarkan tabel 2.3, yaitu mengambil nilai tengah dari

tingkat *moderate* atau menengah, sehingga batas terendah kedua jenis nilai persetujuan adalah 0,5.

4. Uji coba anotasi menggunakan sampel data tidak berlabel. Sampel data tidak berlabel yang dipilih adalah yang representatif dan bervariasi. Uji coba dilakukan terhadap sejumlah kecil data yang sama untuk masing-masing anotator. Setelah uji coba dilakukan, kemudian dilakukan perhitungan persetujuan secara keseluruhan dan individu. Perhitungan persetujuan keseluruhan menggunakan Fleiss Kappa. Sedangkan, nilai persetujuan individu dlakukan antara penulis dan masing-masing menggunakan Cohen's Kappa.

5. Melakukan penyaringan anotator dan pelaksanaan anotasi pada data sebenarnya. Anotator yang nilai persetujuan individu-nya melebihi 0,5 akan terlibat dalam anotasi data tidak berlabel sebenarnya.

6. Melakukan perhitungan Kappa terhadap hasil anotasi. Kappa dihitung untuk menggambarkan nilai persetujuan anotator pada data yang sebenarnya.

Hasil anotasi manual terhadap calon data bibit akan dianalisis terlebih dahulu sebelum data tersebut digunakan ke dalam proses Co-training.

3.4 Co-training

Proses Co-training dilakukan setelah mendapatkan kandidat pasangan T dan H serta komentar penulis, baik yang sudah diberi label maupun yang belum. Pemilihan *view* pada data menggunakan cara yang sama dengan yang penelitian Zanzotto dan Pennacchiotti (2010) karena jenis data yang digunakan sama yaitu Wikipedia *revision history*. *View* pertama adalah pasangan T dan H, sedangkan *view* kedua adalah komentar penulis.

Sebelum proses Co-training dilakukan, data masukan harus diubah ke dalam bentuk vektor fitur melalui tahap ekstraksi fitur, salah satunya akan menggunakan model *word embedding*. Vektor fitur tersebut diklasifikasikan menggunakan dua buah *classifier* untuk masing-masing *view*.

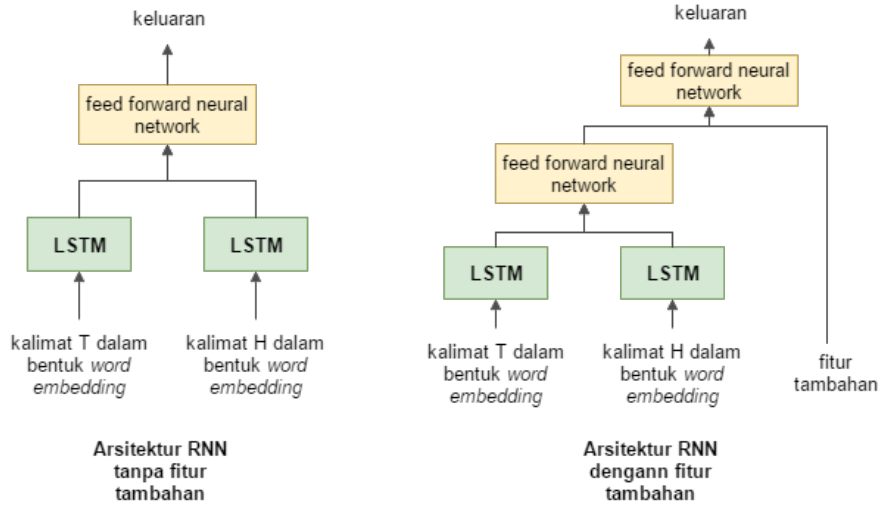
3.4.1 Penentuan Classifier Pertama

View pertama adalah *view* yang cukup mendominasi isi data karena informasi pasangan kalimat yang ingin diprediksi hubungan *entailment*-nya terdapat pada *view* tersebut. Hasil klasifikasi untuk *view* pertama mungkin sangat memengaruhi

hasil pelabelan. Oleh karena itu, penulis sangat mempertimbangkan *classifier* apa yang akan digunakan.

Saat ini, metode *deep learning* sedang populer dalam penelitian dengan pendekatan *machine learning*. RNN adalah salah satu jenis *deep learning* yang paling cocok digunakan untuk memodelkan kalimat. Beberapa penelitian NLP yang menggunakan RNN memberikan hasil yang lebih baik, diantaranya Named Entity Recognition (Hammerton, 2003) dan *Textual Entailment*. Untuk memaksimalkan klasifikasi, *view* ini akan direpresentasikan menggunakan model *word embedding* seperti yang dilakukan pada penelitian *Textual Entailment* dengan RNN yang disebutkan pada bagian 2.7. Selain karena keunggulan RNN, alasan lain mengapa *classifier* pertama adalah RNN dikarenakan NLP *tools* untuk bahasa Indonesia yang dapat menunjang pendeteksian *entailment* masih sulit ditemukan, seperti *tools* untuk mengetahui bentuk sintaktik kalimat yang digunakan dalam penelitian Zanzotto dan Pennacchiotti (2010).

Penggunaan arsitektur dan fitur yang tepat akan mendukung kinerja RNN. Ada dua buah arsitektur RNN yang akan dicoba. Arsitektur pertama adalah RNN dengan menggunakan dua buah LSTM, yaitu untuk T dan H, dengan fitur hanya berupa nilai *word embedding* setiap kata pada teks. Arsitektur ini menyerupai arsitektur umum pada penelitian Bowman et al. (2015). Arsitektur kedua kurang lebih sama dengan arsitektur pertama. Namun ada fitur *non-word embedding* yang ditambahkan pada arsitektur ini.



Gambar 3.13: Dua arsitektur RNN yang akan dicoba

Fitur tambahan pada arsitektur kedua ditujukan untuk memperkuat hubungan leksikal antara T dan H. Salah satu cara untuk melihat hubungan T dan H tersebut adalah dengan menggunakan penyesuaian kata antara kedua kalimat.

Penyesuaian kata adalah proses pencocokan kata yang sama antara T dan H. Kata yang sama dapat digunakan untuk menghitung kesamaan antara teks T dan H. Namun, tidak semua kata di T dan H saling bersesuaian. Kata-kata tidak bersesuaian tersebut juga bisa dimanfaatkan karena kata yang tidak bersesuaian mungkin memiliki hubungan leksikal seperti, sinomim, hipernim, atau antonim. Kata-kata tidak bersesuaian yang berada di antara pasangan kata bersesuaian yang sama kemudian dikelompokkan. Sepasang T dan H dapat memiliki beberapa pasang kelompok kata tidak bersesuaian.

Setelah dilakukan penyesuaian kata, pasangan kelompok kata tidak bersesuaian akan teridentifikasi, begitu pula dengan LCSS antara dua kalimat tersebut. LCSS atau *longest common subsequence* adalah bagian yang sama dan terpanjang antara dua atau lebih *sequence* (Paterson dan Dancík, 1994). LCSS dapat digunakan untuk mengukur tingkat kesamaan antara dua buah *sequence*. LCSS antara T dan H adalah potongan terpanjang antara T dan H yang serupa. Dengan mengetahui informasi-informasi tersebut, berikut adalah fitur tambahan yang diajukan.

1. Rata-rata *similarity* dari pasangan kelompok kata yang tidak bersesuaian di T dan H kecuali yang berpasangan dengan kelompok kosong.

2. Nilai dalam rentang 0 hingga 1 yang menggambarkan jumlah pasangan kelompok kata di T yang berpasangan dengan kelompok kosong ([]) di H. Untuk menghasilkan nilai yang berada dalam rentang 0 hingga 1, jumlah kemunculan pasangan kelompok kata tersebut dapat dimasukkan ke dalam fungsi *sigmoid*.

3. Nilai dalam rentang 0 hingga 1 yang menggambarkan jumlah pasangan kelompok kata di H yang berpasangan dengan kelompok kosong ([]) di T. Sama seperti fitur kedua, jumlah kemunculan pasangan kelompok kata tersebut dimasukkan ke dalam fungsi *sigmoid* agar berada dalam rentang 0 hingga 1.

4. Nilai dalam rentang 0 hingga 1 yang merepresentasikan jumlah kata yang sesuai dari T dan H. Mula-mula kata yang sesuai antara T dan H dihitung, kemudian agar mendapatkan nilai yang berada di antara 0 hingga 1, jumlah kata tersebut harus dibagi dengan jumlah kata pada teks. Nilai fitur ini adalah,

$$(2 \times \textit{jumlah kata yang sama})/(\textit{jumlah kata } T + \textit{jumlah kata } H) \quad (3.1)$$

5. Nilai dalam rentang 0 hingga 1 yang merepresentasikan panjang LCSS dari T dan H. Panjang sebuah teks dihitung berdasarkan jumlah kata pada teks

tersebut. Agar mendapatkan nilai yang berada di antara 0 hingga 1, panjang LCSS harus dibagi dengan panjang teks. Nilai fitur ini adalah,

$$(2 \times panjang\ LCSS)/(panjang\ T + panjang\ H) \tag{3.2}$$

6. Nilai yang merepresentasikan kebenaran bahwa T dan H diawali kata yang sama. Apabila T dan H diawali kata yang sama, nilai fitur ini adalah 1, sebaliknya adalah 0.

Gambar 3.14 menunjukkan contoh penyesuaian kata antara 2 kalimat. Kelompok kata tidak bersesuaian di T dan H adalah: ([dia], [beliau]), ([mengambil], [berkuliah, di]), ([di], []), dan ([], [pada]). Kata-kata yang bersesuaian adalah "sedang", "jurusan", "kedokteran", "Universitas", "Indonesia", "Salemba", "tahun", "1990". Sedangkan, LCSS T dan H adalah "Universitas Indonesia Salemba".

Gambar 3.14: Contoh penyesuaian kata antara 2 kalimat

Berikut adalah perhitungan fitur tambahan untuk contoh 3.14.

1. Pasangan kelompok kata tidak bersesuaian yang tidak berpasangan dengan kelompok kosong adalah ([dia], [beliau]) dan ([mengambil], [berkuliah, di]). Maka, nilai fitur pertama adalah,

$$\frac{(similarity([dia],[beliau]) + similarity([mengambil],[berkuliah,di])}{2} \tag{3.3}$$

2. Pasangan kelompok kata tidak bersesuaian yang berpasangan dengan kelompok kosong di T berjumlah satu, yaitu ([], [pada]). Sehingga nilai fitur kedua adalah $sigmoid(1)$.

3. Pasangan kelompok kata tidak bersesuaian yang berpasangan dengan kelompok kosong di H berjumlah satu, yaitu ([di], []). Sehingga nilai fitur ketiga adalah $sigmoid(1)$.

4. Ada delapan buah kata yang bersesuaian, sehingga nilai fitur keempat adalah,

$$(2 \times 8)/(11 + 12) = 0.59 \tag{3.4}$$

5. Panjang LCSS dari contoh tersebut adalah tiga kata, sehingga nilai fitur kelima adalah,

$$(2 \times 3)/(11 + 12) = 0.26 \tag{3.5}$$

6. Kata pertama dari kedua kalimat tidak sama, sehingga nilai fitur keenam adalah 0.

3.4.2 Penentuan Classifier Kedua

Classifier kedua dipilih dengan menggunakan percobaan 10-fold *cross validation* terhadap fitur-fitur teks komentar pada data hasil anotasi manual. Ada beberapa kandidat *classifier* yang akan digunakan, yaitu Neural Network (Multilayer Perceptron), Decision Tree, Naive Bayes, Multinomial Naive Bayes, dan Bayesian Network. Ekstraksi fitur yang digunakan untuk *view* kedua cukup sederhana, yaitu jumlah kemunculan N-gram tertentu pada teks komentar. Sebelum menentukan apa saja N-gram tersebut, teks komentar terlebih dahulu dianalisis berdasarkan frekuensi N-gram yang terdapat pada teks. Unigram, bigram, dan trigram yang paling sering muncul dijadikan sebagai fitur. Dengan cara tersebut, fitur yang didapat akan berjumlah banyak, namun tidak semua fitur baik untuk klasifikasi. Oleh karena itu, dilakukan pemilihan kombinasi atribut terbaik.

BAB 4
IMPLEMENTASI

Pada bab ini, dijelaskan mengenai implementasi dari pengolahan data hingga percobaan Co-training.

4.1 Ekstraksi Teks

Hal pertama yang harus dilakukan terhadap berkas XML Wikipedia adalah membersihkan berkas tersebut dari format Wiki *markup language*. Ekstraksi dilakukan dengan bantuan *tools* Wikipedia Extractor[1] yang berupa program berbahasa Python untuk mengekstrak dan membersihkan dokumen XML Wikipedia. Program ini dibuat oleh Giuseppe Attardi dan Antonio Fuschetto dari University of Pisa, Italia, dan telah direvisi oleh beberapa kontributor.

Untuk membersihkan berkas XML semua artikel Wikipedia, *tools* Wikipedia Extractor bisa langsung dijalankan dengan perintah berikut pada terminal.

```
WikiExtractor.py [nama berkas masukan] -o [nama berkas keluaran]
```

```
<page>
  <title>Asam deoksiribonukleat</title>
  <ns>0</ns>
  <id>1</id>
  <revision>
    <id>11703617</id>
    <parentid>11422826</parentid>
    <timestamp>2016-06-29T14:26:10Z</timestamp>
    <contributor>
      <username>AABot</username>
      <id>175295</id>
    </contributor>
    <comment>Robot: Perubahan kosmetika</comment>
    <text xml:space="preserve">[[Berkas:DNA Structure+Key+Labelled.pn
    NoBB.png|thumb|right|340px|Struktur [[heliks ganda]] DNA. [[Atom]]-atom pada struktur tersebut
    diwarnai sesuai dengan [[unsur kimia]]nya dan struktur detail dua pasangan basa ditunjukkan oleh
    gambar kanan bawah]]
[[Berkas:ADN animation.gif|thumb|Gambaran tiga dimensi DNA]]
'''Asam deoksiribonukleat''', lebih dikenal dengan singkatan '''DNA''' ([[bahasa Inggris]]:
'''''d'''eoxyribo'''n'''ucleic '''a'''cid''), adalah sejenis biomolekul yang menyimpan dan menyandi
instruksi-instruksi [[genetika]] setiap [[organisme]] dan banyak jenis [[virus]]. Instruksi-instruksi
genetika ini berperan penting dalam pertumbuhan, perkembangan, dan fungsi organisme dan virus. DNA
merupakan [[asam nukleat]]; bersamaan dengan [[protein]] dan [[karbohidrat]], asam nukleat adalah
[[makromolekul]] esensial bagi seluruh [[makhluk hidup]] yang diketahui. Kebanyakan molekul DNA terdiri
dari dua unting [[biopolimer]] yang berpilin satu sama lainnya membentuk [[heliks ganda]]. Dua unting DNA
ini dikenal sebagai [[polinukleotida]] karena keduanya terdiri dari [[monomer|satuan]]-satuan molekul yang
disebut [[nukleotida]]. Tiap-tiap nukleotida terdiri atas salah satu jenis [[basa nitrogen]] ([[guanina]]
(G), [[adenina]] (A), [[timina]] (T), atau [[sitosina]] (C)), gula [[monosakarida]] yang disebut
[[deoksiribosa]], dan gugus [[fosfat]]. Nukleotida-nukelotida ini kemudian tersambung dalam satu rantai
[[ikatan kovalen]] antara gula satu nukleotida dengan fosfat nukelotida lainnya. Hasilnya adalah rantai
punggung gula-fosfat yang berselang-seling. Menurut kaidah [[pasangan basa]] (A dengan T dan C dengan G),
[[ikatan hidrogen]] mengikat basa-basa dari kedua unting polinukleotida membentuk DNA unting ganda
```

Gambar 4.1: Potongan artikel pada berkas XML Wikipedia

[1] http://medialab.di.unipi.it

Gambar 4.1 menunjukkan potongan isi berkas masukan yang berupa XML semua artikel Wikipedia. Sedangkan, gambar 4.2 menunjukkan hasil yang dikeluarkan program Wikipedia Extractor setelah melakukan ekstraksi pada potongan isi artikel di gambar 4.1.

```
<doc id="1" url="?curid=1" title="Asam deoksiribonukleat">
Asam deoksiribonukleat

Asam deoksiribonukleat, lebih dikenal dengan singkatan DNA (bahasa Inggris:
"deoxyribonucleic acid"), adalah sejenis biomolekul yang menyimpan dan menyandi
instruksi-instruksi genetika setiap organisme dan banyak jenis virus.
Instruksi-instruksi genetika ini berperan penting dalam pertumbuhan, perkembangan,
dan fungsi organisme dan virus. DNA merupakan asam nukleat; bersamaan dengan
protein dan karbohidrat, asam nukleat adalah makromolekul esensial bagi seluruh
makhluk hidup yang diketahui. Kebanyakan molekul DNA terdiri dari dua unting
biopolimer yang berpilin satu sama lainnya membentuk heliks ganda. Dua unting DNA
ini dikenal sebagai polinukleotida karena keduanya terdiri dari satuan-satuan
molekul yang disebut nukleotida. Tiap-tiap nukleotida terdiri atas salah satu jenis
basa nitrogen (guanina (G), adenina (A), timina (T), atau sitosina (C)), gula
monosakarida yang disebut deoksiribosa, dan gugus fosfat. Nukleotida-nukelotida ini
kemudian tersambung dalam satu rantai ikatan kovalen antara gula satu nukleotida
dengan fosfat nukelotida lainnya. Hasilnya adalah rantai punggung gula-fosfat yang
berselang-seling. Menurut kaidah pasangan basa (A dengan T dan C dengan G), ikatan
hidrogen mengikat basa-basa dari kedua unting polinukleotida membentuk DNA unting
ganda
```

Gambar 4.2: Hasil Ekstraksi pada XML seluruh artikel Wikipedia

Setelah melalui proses ekstraksi, hanya teks di dalam *tag text* saja yang diambil dan dibersihkan. Format Wikipedia *markup language* yang dibersihkan, antara lain dengan penghapusan tabel dan gambar (teks di dalam kotak biru pada gambar 4.1), serta pembersihan format *hyperlink* (teks yang diberi garis merah pada gambar 4.1).

Tools Wikipedia Extractor hanya menerima struktur XML semua artikel Wikipedia, sehingga tidak dapat mengolah XML Wikipedia *revision history*. Oleh karena itu, ada sedikit variasi yang dilakukan agar tetap bisa menggunakan *tools* ini pada data Wipedia *revision history*. Variasi tersebut ditunjukkan pada kode 4.1.

Kode 4.1: Ekstraksi Wikipedia Revision History

```
#Sediakan berkas template XML artikel Wikipedia yang kosong
template = #berkas template XML artikel Wikipedia
revision = #berkas XML Wikipedia Revision History

for page in revision: #semua artikel
  page_revision = []
  page_id = #id artikel
  for r in page: #semua teks revisi
    comment = #teks di dalam tag comment
    r_id = #id teks revisi
    p_id = #id induk teks revisi
```

```
    #masukan teks revisi ke dalam template XML
    template.insert_text(r)
    #panggil program WikiExtractor dan ekstrak template
    extracted = run_command(WikiExtractor.py template)
    page_revision.add([extracted, comment, r_id, p_id])
    #bersihkan teks template
    template.clear()
  #cetak page_revision
```

Berdasarkan kode 4.1, isi dari artikel Wikipedia *revision history* pada setiap versi revisi dimasukkan ke dalam *template* yang memiliki struktur serupa dengan Wikipedia *all article*. Kemudian, *template* yang sudah diganti isinya diekstrak menggunakan *tools* Wikipedia Extractor dan disimpan hasil keluarannya bersama dengan beberapa informasi lain yang dibutuhkan, seperti informasi di dalam *tag comment* dan *id*. Setelah program ini dijalankan, hasil yang diperoleh akan seperti contoh pada gambar 3.3.

4.2 Pemisahan kalimat

Hasil ekstraksi Wikipedia seluruh artikel dipisahkan menjadi kalimat per kalimat, lalu masing-masing kalimat dilakukan normalisasi. Dalam sebuah teks, kalimat dipisahkan dengan menggunakan tanda baca titik, seru, dan tanya. Namun ada beberapa kondisi khusus yang menunjukkan bahwa kemunculan tanda tersebut bukan berarti sebagai pemisah kalimat, yaitu:

1. Kemunculan tanda baca berada di dalam suatu kutipan dan tanda kurung. Contoh: "Pertumbuhan penjualan tahun ini (lihat Tabel 9.1) menunjukkan kenaikan yang pesat.".

2. Kemunculan tanda baca berada dalam suatu istilah (seperti alamat situs) atau bilangan. Contoh: istilah www.google.com dan bilangan 1.000.000.

3. Kemunculan tanda baca berada dalam sebuah singkatan umum Indonesia atau sebuah gelar. Contoh: dll., dsb., tgl., hlm., S.Kom., S.Pd., Dr.

Pada tahap ini juga dilakukan normalisasi. Ada dua jenis normalisasi yang dilakukan, yaitu normalisasi gelar dan singkatan serta normalisasi tanda baca.

1. Normalisasi gelar dan singkatan.
   Normalisasi ini dilakukan untuk kata-kata yang mengandung tanda baca titik di dalamnya, seperti singkatan dan gelar. Kata-kata tersebut rawan mengalami kesalahan penulisan, misalnya "S.Kom." ditulis "S.Kom" atau

juga singkatan seperti "dll." ditulis "dll" (tanpa titik). Untuk mengimplementasikan normaliasasi ini, kemungkinan-kemungkinan bentuk kesalahan penulisan yang umum terjadi didaftarkan terlebih dahulu. Kemudian tiap kesalahan tersebut dipetakan pada penulisan yang benar.

2. Normalisasi tanda baca
   Kalimat-kalimat yang terbentuk kemudian mengalami normalisasi tanda baca, yaitu pemisahan tanda baca yang menempel pada kata-kata di kalimat kecuali kata yang memang mengandung tanda baca tersebut (tidak memiliki arti jika tidak dengan tanda baca, contohnya singkatan, gelar, alamat situs). Contoh kalimat yang sudah mengalami normalisasi tanda baca: "Chairil meninggal dalam usia muda di Rumah Sakit CBZ (sekarang Rumah Sakit Dr. Cipto Mangunkusumo) , Jakarta pada tanggal 28 April 1949 ; penyebab kematiannya tidak diketahui pasti , menurut dugaan lebih karena penyakit TBC ."

Kode 4.2 menunjukkan alur dalam melakukan pemisahan kalimat. Mula-mula semua istilah singkatan dan gelar ditandai di dalam sebuah *tag*. Hal tersebut ditujukan agar tanda baca pemisah kalimat, seperti titik, yang digunakan di dalam singkatan atau gelar dapat dibedakan dari tanda baca pemisah kalimat yang sebenarnya. Kemudian, tanda baca pemisah kalimat selain yang terdapat dalam *tag* singkatan dan gelar ditandai dengan *tag* pemisah kalimat. Setelah itu, lakukan proses penghapusan kasus-kasus yang bukan merupakan pemisahan kalimat dengan cara menghilangkan *tag* pemisah kalimat pada tanda bacanya. Terakhir, lakukan pemisahan berdasarkan *tag* pemisah kalimat.

Kode 4.2: Pemisahan kalimat

```
def get_sentences(text) :
  #tandai semua kata pada daftar singkatan dan gelar dalam tag <
     ABBR><\ABBR>
  #selagi menandai singkatan, cek apakah kata dalam tag sudah
     normal
  normal_text = abbreviations_tag(text)

  #tandai karakter . ? dan ! ke dalam tag <EOS><\EOS>
  marked_text = first_sentence_breaking(normal_text)

  #hapus tag <EOS><\EOS> untuk kasus yang bukan akhir kalimat
  fixed_marked_text = remove_false_end_of_sentence(marked_text)

  #pisahkan kalimat berdasarkan tag <EOS><\EOS>
```

```
  sentences = fixed_marked_text.split(<EOS>*<\EOS>)

  #normalisasi tanda baca
  for s in sentences:
    s = punctuation_normalization(s)

  return sentences
```

Gambar 4.3 menunjukkan contoh hasil pemisahan kalimat.

```
1 Asam deoksiribonukleat
2 Asam deoksiribonukleat , lebih dikenal dengan singkatan DNA ( bahasa
  Inggris : deoxyribonucleic acid ) , adalah sejenis biomolekul yang
  menyimpan dan menyandi instruksi-instruksi genetika setiap organisme
  dan banyak jenis virus .
3 Instruksi-instruksi genetika ini berperan penting dalam pertumbuhan
  , perkembangan , dan fungsi organisme dan virus .
4 DNA merupakan asam nukleat ; bersamaan dengan protein dan
  karbohidrat , asam nukleat adalah makromolekul esensial bagi seluruh
  makhluk hidup yang diketahui .
5 Kebanyakan molekul DNA terdiri dari dua unting biopolimer yang
  berpilin satu sama lainnya membentuk heliks ganda .
6 Dua unting DNA ini dikenal sebagai polinukleotida karena keduanya
  terdiri dari satuan-satuan molekul yang disebut nukleotida .
```

Gambar 4.3: Contoh hasil proses pemisahan kalimat

4.3 Pemodelan Word Embedding

Pembentukan model *word embedding* dilakukan dengan bantuan *library* Gensim-Word2Vec[2] yang tersedia untuk bahasa Python. Seperti yang disebutkan pada bagian 2.5.2, salah satu cara mengaplikasikan *word embedding* adalah menggunakan prosedur *word2vec*. *Library* Gensim-Word2Vec adalah *library* bahasa Python yang dapat digunakan untuk mengimplementasikan prosedur *word2vec*. *Library* tersebut diaplikasikan pada program Python yang memproses masukan berupa hasil pemisahan kalimat. Berikut adalah kode program tersebut.

Kode 4.3: Pemodelan *word embedding*

```
from gensim.models import word2vec

file = #file berisi kalimat-kalimat
sentences = []
for sentence in file:
  words = sentence.split()
  if (len(words) > 1):
```

[2]https://radimrehurek.com/gensim/models/word2vec.html

```
    sentences.append(words)

model = word2vec.Word2Vec(sentences
            , size = #panjang vektor
            , window = #ukuran window
            , min_count = #jumlah kata minimum
            )
model.save(model_name)
```

Kode 4.3 menerima masukan berupa *list of* kalimat. Kemudian setiap kalimat ditokenisasi menjadi *list of* kata. *Word2vec* menerima *list of list* kata untuk membentuk model *word2embedding*. Kemudian, yang perlu dilakukan adalah mengatur parameter lain pada *word2vec*, yaitu *size* yang merupakan panjang dari vektor kata yang dibuat, *window* adalah jumlah kata yang diprediksi dari suatu kata, *min_count* adalah panjang minimum dari *list* kata yang dimasukkan.

4.4 Pembentukan T dan H

Seperti yang telah dijelaskan pada bagian 3.2, ada lima proses yang harus dilakukan. Pada bagian ini akan dijelaskan cara mengimplementasikan proses tersebut.

1. Penghapusan kata-kata yang mengarah ke vandalisme
   Kode 4.4 menunjukkan proses penghapusan kata-kata yang mengarah pada kasus vandalisme. Kata-kata yang mengarah pada vandalisme tersebut terlebih dahulu didaftarkan dan disimpan dalam sebuah *list*.

Kode 4.4: Penghapusan kata-kata yang mengarah ke vandalisme

```
text = #teks yang sedang diproses
vandal_words = #daftar kata-kata yang tergolong vandal

for word in vandal_words:
  text = text.remove(word)
return text
```

2. Pemasangan teks induk dan teks revisi
   Kode 4.5 menunjukkan proses pemasangan kalimat. Setiap teks revisi dipasangkan dengan teks induknya yaitu teks dengan *id* yang sama dengan *parent id* pada teks revisi.

Kode 4.5: Pemasangan teks induk dan revisi

```
TH_pairs = []
for article in file: #semua artikel dalam file
```

```
    for r in article: #semua teks revisi di artikel, kecuali
        teks pertama
      p_id = #parent_id
      TH_pairs.append([r, article[p_id]]) #ambil artikel
        induk
  return TH_pair
```

3. Penyesuaian paragraf

   Kode 4.6 menunjukkan proses penyesuaian paragraf. Paragraf pada teks induk dan revisi terlebih dahulu diidentifikasi. Salah satu cara identifikasi adalah menggunakan tanda *newline*. Untuk setiap paragraf di teks induk, pasangkan dengan salah satu paragraf yang sesuai di teks revisi. Pemasangan tersebut dilakukan secara terurut. Pada penelitian ini, paragraf kami anggap sesuai apabila paling tidak memiliki sebuah teks yang sama.

Kode 4.6: Penyesuaian paragraf

```
def paragraph_alignment(parent_text, revisi_text) :
  #inisialisasi pasangan paragraf
  par_pairs = []
  #pisahkan paragraf di teks induk
  parent_pars = parent_text.split(newline)
  #pisahkan paragraf di teks revisi
  revision_pars = revisi_text.split(newline)

  i = 0
  j = 0
  while (i < len(parent_pars)):
    moving_j = j
    while (moving_j < len(revision_pars) and i < len(
        psarent_pars)):
      if same_paragraph(revision_pars[i], parent_pars[
          moving_j]):
        #simpan pasangan paragraf yang sesuai
        par_pairs.append(parent_pars[moving_j],
            revision_pars[i])
        i++
        j = moving_j + 1
      moving_j++
    i++
  return par_pairs
```

4. Penyesuaian kalimat

   Kalimat-kalimat dari pasangan paragraf yang sesuai kemudian dibandingkan

agar dapat diketahui kalimat mana yang sama antar kedua paragraf tersebut. Dua kalimat dianggap sama jika:

- Kedua kalimat sama persis
- Perbedaan antara dua kalimat hanya berupa keterangan tambahan menggunakan tanda kurung (). Contoh: "Asam Deoksiribonukleat (DNA) adalah ..." dan "Asam Deoksiribonukleat adalah ..."
- Perbedaan antara kedua kalimat hanya perbaikan kata-kata salah ejaan. Contoh: "Antropologi adalah salah satu cabang ilmu social ..." dan "Antropologi adalah salah satu cabang ilmu sosial ...". Untuk kasus ini, kata-kata salah ejaan diidentifikasi menggunakan algoritme Levenstein Distance (Levenshtein, 1966). Dimana jika *distance* yang diberikan lebih kecil dari 3, kata dianggap sama.

Kode 4.7: Levenstein Distance

```
def levenshteinDistance(word_1, word_2):
  if len(word_1) > len(word_2):
    word_1, word_2 = word_2, word_1

  distances = range(len(word_1) + 1)
  for i, char_2 in enumerate(word_2):
    distances_ = [i+1]
    for j, char_1 in enumerate(word_1):
      if char_1 == char_2:
        distances_.append(distances[j])
      else:
        distances_.append(1 + min((distances[j],
           distances[j + 1], distances_[-1])))
    distances = distances_
  return distances[-1]
```

- Kalimat hanya berbeda dari segi penggunaan huruf besar dan kecil
- Kalimat hanya berbeda dari segi penggunaan tanda baca

5. Pemasangan Kalimat

   Kelompok kalimat penambahan di teks induk dan kelompok kalimat pengurangan di teks revisi yang berada yang berada di antara kalimat yang sama dipasangkan sesuai dengan aturan berikut.

   - Apabila jumlah kalimat yang dikurangi lebih banyak, kalimat pengurangan di posisi akhir (yang tidak mendapat pasangan) tidak akan digunakan.

- Apabila jumlah kalimat yang ditambahkan lebih banyak, kalimat tambahan di posisi akhir dipasangkan dengan kalimat yang berada sebelum kalimat tersebut di dalam teks akhir. Hal ini dilakukan dengan pertimbangan bahwa, sebuah kalimat yang ditambahkan bisa saja berupa rincian atau penjelas dari kalimat sebelumnya.
- Selebihnya, kalimat yang dikurangi dipasangkan dengan kalimat tambahan dengan urutan yang sesuai.

Kode 4.8 menunjukkan implementasi dari aturan di atas.

Kode 4.8: Pemasangan kalimat

```
def T_H_pairing(sen_group_1, sen_group_2):
  pairs = []
  #aturan poin pertama
  if len(sen_group_1) > len(sen_group_2):
    sen_group_1 = sen_group_1[:len(sen_group_2)] #potong
       kalimat yang berlebih
  #aturan poin kedua
  elif len(sen_group_2) > len(sen_group_1):
    for i in range(len(sen_group_1) .. len(sen_group_2)-1)
       :
      #pasangkan dengan kalimat sebelumnya
      pairs.append([sen_group_2[i], sen_group_2[i+1]])

  #aturan poin ketiga
  for i in range(0 .. min(len(sen_group_1), len(
     sen_group_2))):
    #pasangkan kalimat dengan posisi yang sesuai
    pairs.append([sen_group_1[i], sen_group_2[i]])
  return pairs
```

Proses pemasangan belum berakhir, pasangan kalimat tersebut perlu ditentukan posisinya yaitu kalimat mana yang berperan sebagai T dan mana yang menjadi H. Untuk menentukan hal tersebut kami hanya menggunakan perbandingan panjang kalimat (dihitung dari jumlah kata) sebagai pertimbangan. Kalimat yang lebih panjang akan menjadi T, selebihnya menjadi H. Terakhir, pasangan T dan H diberi pelengkap yaitu komentar penulis yang diambil dari komentar pada teks revisi.

4.5 Ekstraksi Fitur

Pada bagian ini akan dijelaskan ekstraksi fitur yang dilakukan pada kedua *view*. Masing-masing *view* diekstrak dengan cara yang berbeda.

4.5.1 Ekstraksi Fitur View Pertama

Fitur utama dari *view* pertama adalah fitur *word embedding* masing-masing teks T dan H. Untuk mengetahui vektor *word embedding* masing-masing kata, diperlukan model *word embedding* yang sebelumnya telah dibuat. Kode kode-ekstraksi-WE menunjukkan proses ekstraksi fitur *word embedding*.

Kode 4.9: Ekstraksi Fitur *word embedding*

```
from gensim.models import word2vec
vector_null = #vektor default [0, 0, ...]

def sentence_to_vec (sentence):
  vec = []
  model = word2vec.Word2Vec.load(model_name) #nama model
  words = sentence.split()
  for word in words:
    try:
      vec.append(model[word].tolist()) #mencari vektor kata
          menggunakan model
    except:
      vec.append(vector_null) #jika kata tidak pernah dimasukkan
          ke model, gunakan vektor default
  return vec
```

Fitur yang terbentuk adalah sebuah vektor dengan elemen berupa *word embedding* dari semua penyusun teks. Apabila kata yang penyusun teks tersebut tidak ada di dalam model *word embedding*, kata akan diganti dengan vektor *default*. Panjang vektor yang digunakan harus sama, agar memiliki panjang yang sama, kami melakukan *padding* terhadap seluruh vektor kalimat T dan H. *Padding* adalah proses penambahan elemen pada vektor yang panjangnya tidak mencapai batas yang ditentukan. *Padding* dilakukan dengan menambahkan vektor *null* di bagian akhir vektor yang panjangnya masih di bawah batas. Semua vektor di-*padding* hingga panjangnya mencapai panjang vektor kalimat dengan kata terbanyak.

Pada arsitektur RNN kedua, ada 6 buah fitur tambahan yang perlu diimplementasikan, yaitu:

1. Rata-rata *similarity* dari pasangan kelompok kata yang tidak bersesuaian di T

dan H kecuali yang berpasangan dengan kelompok kosong.

2. Nilai dalam rentang 0 hingga 1 yang menggambarkan jumlah pasangan kelompok kata di T yang berpasangan dengan kelompok kosong ([]) di H.

3. Nilai dalam rentang 0 hingga 1 yang menggambarkan jumlah pasangan kelompok kata di H yang berpasangan dengan kelompok kosong ([]) di T.

4. Nilai dalam rentang 0 hingga 1 yang merepresentasikan jumlah kata yang sesuai dari T dan H.

5. Nilai dalam rentang 0 hingga 1 yang merepresentasikan panjang LCSS dari T dan H.

6. Nilai yang merepresentasikan kebenaran bahwa T dan H diawali kata yang sama.

Kode 4.10 menunjukkan implementasi dari fitur tambahan di atas.

Kode 4.10: Penambahan fitur arsitektur RNN kedua

```
#masukan adalah kalimat T dan H
def addOtherFeatures(text, hipo):
  OtherFeature = []
  T = text.split()
  H = hipo.split()
  (same, diff)  = calculate_diff(T, H)
  feature1 = avg(diff) #mencari rata-rata nilai dalam list 'diff'
      kecuali yang bernilai 1 dan -1
  feature2 = sigmoid(count(-1, diff)) #hitung jumlah -1 di 'diff'
  feature3 = sigmoid(count(1, diff)) #hitung jumlah 1 di 'diff'
  feature4 = (2 * same) / (len(T) + len(H))
  feature5 = (2 * (LCSS(T, H)) / (len(T) + len(H))
  feature6 = T[0] == H[0] ? 1 : 0
  OtherFeature = [feature1, feature2, feature3, feature4,
     feature5, feature6]
  return OtherFeature
```

Fungsi LCSS yang digunakan adalah LCSS pada umumnya. Sedangkan, untuk menghitung nilai *similarity*, digunakan fungsi *n_similarity()* yang disediakan olen Gensim Word2vec, yaitu fungsi yang menghitung *cosine similarity* dari dua buah *list* kata. Pada kode 4.11, ditampilkan cara menghitung kemunculan kata bersesuaian, mengelompokkan kata, dan menghitung *similarity* kelompok kata tersebut.

Kode 4.11: Perhitungan *similarity* kelompok kata

```
from gensim.models import word2vec

def calculate_diff(T, H):
  model = #load model word embedding
  same = 0 #inisalisasi jumlah kata sesuai
  diff = group_1 = group_2 = [] #inisialisasi hasil, kelompok
     perubahan kata di T, dan kelompok perubahan kata di H
  #membatasi akhir dan awal kalimat
  Tword = ['<start>'] + T + ['<end>']
  Hword = ['<start>'] + H + ['<end>']
  s = s_new = 0
  for i, t in enumerate(Tword):
    found = False
    moving_s = s
    while (moving_s < len(Hword)):
      if (t == Hword[moving_s]):
        same += 1
        found = True
        s_new = moving_s
        break
      else:
        moving_s = moving_s + 1

    if (found == False):
      group_1.append(Tword[i])
    else:
      for k in range(s+1, s_new):
        group_2.append(Hword[k])
      #kelompok kata T kosong
      if (len(group_2) != 0 and len(group_1)==0):
        diff.append(-1)
      #kelompok kata H kosong
      elif (len(group_2) == 0 and len(group_1)!=0):
        diff.append(1)
      else:
        #similarity antar kelompok kata
        diff.append(model.n_similarity(group_1, groupt_2))
      group_1 = group_2 = []
    s = s_new
  return (same, diff)
```

Fungsi di atas akan menerima masukan dua buah teks dan mengembalikan nilai *similarity* dari tiap kelompok perubahan kata pada teks. Hasil yang diberikan berupa jumlah kata bersesuaian antara kedua teks dan daftar nilai *similarity*, seperti

[similarity 1, similarity 2, ...].

4.5.2 Ekstraksi Fitur View Kedua

Program Weka GUI[3] digunakan untuk menentukan fitur pada *view* kedua. Fitur untuk *view* kedua berupa jumlah kemunculan unigram, bigram, dan trigram terbanyak yang muncul 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. Kemudian, N-gram diurutkan berdasarkan frekuensinya, beberapa N-gram terbanyak akan dipilih sebagai fitur *view* kedua. Akan tetapi, jumlah kemunculan N-gram pada daftar tersebut belum tentu akan menghasilkan kombinasi fitur yang paling baik. Oleh karena itu, kami menggunakan fitur *select attribute* pada Weka GUI dengan metode *genetic search* untuk memilih kombinasi fitur terbaik.

4.6 Co-training

Co-training akan berjalan dalam beberapa iterasi hingga mencapai *stopping condition*. Pada penelitian ini, *stopping condition* yang digunakan adalah ketika kedua *classifier* tidak mengklasifikasikan data dengan sesuai. Sedikit variasi untuk kondisi tersebut, algoritme Co-training akan berhenti ketika kedua *classifier* tidak mampu menyepakati satu pun label yang sama dalam *n* iterasi berurutan.

Kode 4.12: Algoritme Co-training yang akan digunakan

```
Masukan:
L = data berlabel
U = data tidak berlabel

U' = sejumlah k data dari U
iterasi_gagal = 0
Selama iterasi_gagal < n:
  Latih classifier h1 dengan data L dari view pertama
  Latih classifier h2 dengan data L dari view kedua
  Klasifikasi U' dengan h1 dapatkan U1'
  Klasifikasi U' dengan h2 dapatkan U2'
  Ambil data terbaik dari U1' dan U2'
    Jika jumlah data terbaik = 0, iterasi_gagal++
    Jika jumlah data terbaik > 0, iterasi_gagal = 0
  Ambil k data dari U untuk menggantikan isi U'
```

[3]http://www.cs.waikato.ac.nz/ml/weka/

Kode 4.12 menunjukkan algoritme Co-training yang diajukan untuk penelitian ini. Data hasil klasifikasi terbaik pada setiap iterasi adalah data yang diklasifikasikan oleh masing-masing *classifier* yang menghasilkan label yang sama dan dengan tingkat kepercayaan (lihat bagian 2.4) untuk label tersebut melebihi batas yang ditentukan. Semakin tinggi tingkat kepercayaan sebuah kelas terpilih menunjukkan bahwa *classifier* semakin yakin dalam melakukan klasifikasi. Sedangkan, untuk tetap menjaga keseimbangan data berlabel, jumlah data hasil klasifikasi yang diambil akan diperhatikan perbandingannya.

BAB 5
EVALUASI DAN ANALISIS HASIL

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

5.1 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.

5.2 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 4.2; sedangkan, XML Wikipedia Revision History pada gambar 3.3.

- **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 5.3.

- **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 5.4). Sedangkan fitur untuk komentar penulis adalah kemunculan N-gram tertentu (selanjutnya dibahas di bagian 5.5). 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.

5.3 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 2.3, 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.

5.4 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.

[1]https://keras.io/

Penentuan jumlah *epoch* seharusnya dilakukan dengan beberapa kali percobaan, namun dikarenakan waktu yang tidak mencukupi, *epoch* langsung ditentukan sebesar 200.

5.5 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.

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.

5.6 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.

5.7 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 4.6, 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 4.12 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.

5.7.1 Hasil Co-training

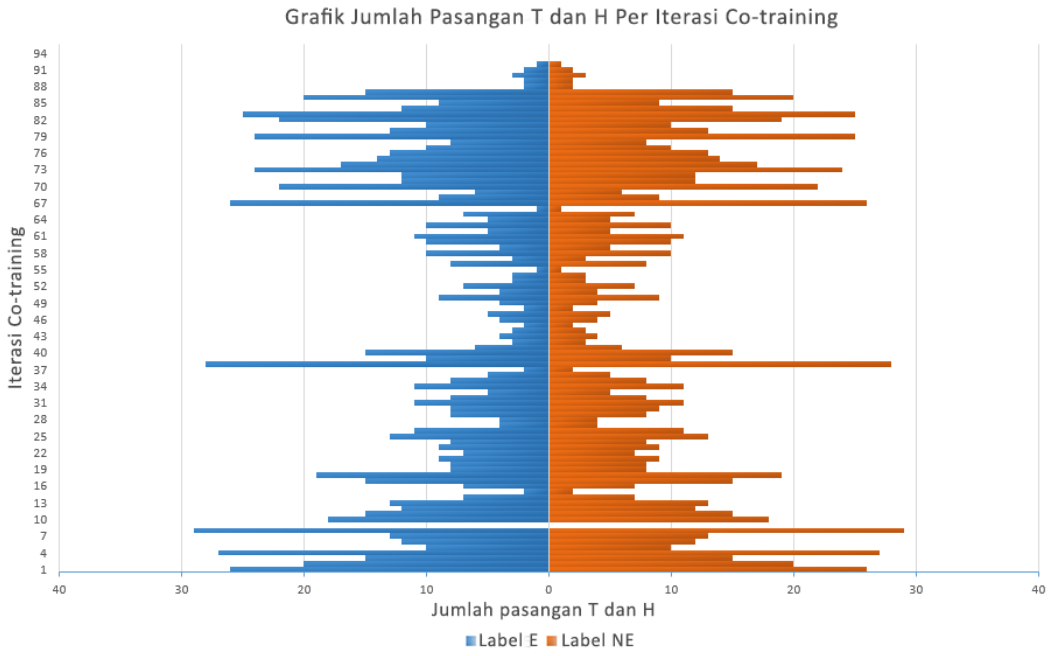
Hasil tiap iterasi dari percobaan Co-training disimpan untuk dievaluasi dan dianalisis. Hasil yang dikeluarkan berupa pasangan T dan H serta komentar penulis

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.

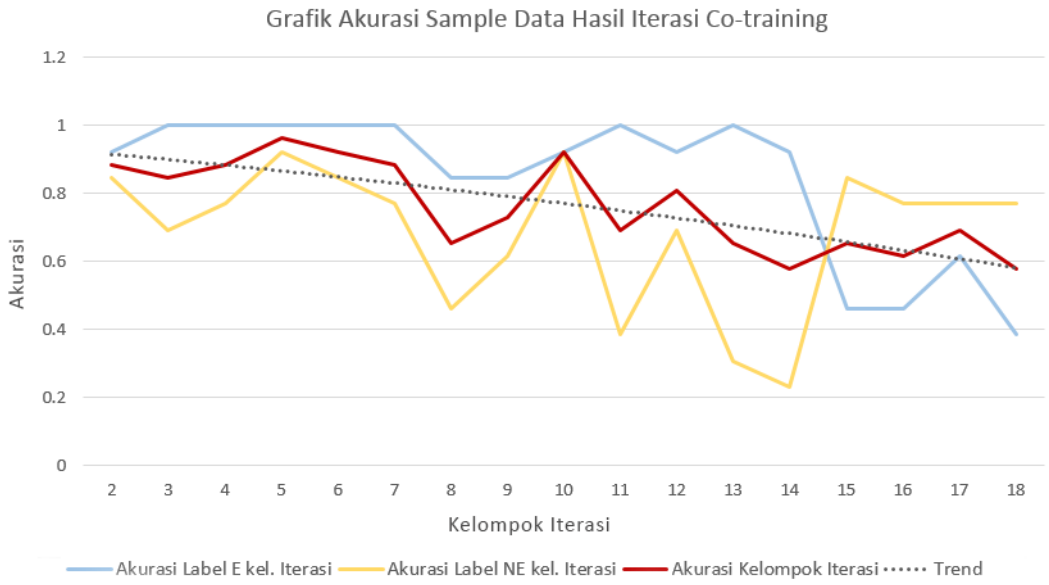
5.7.2 Evaluasi Co-training

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*.

5.7.3 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

Androutsopoulos, I. dan Malakasiotis, P. (2010). A survey of paraphrasing and textual entailment methods. *J. Artif. Int. Res.*, 38(1):135–187.

Atkins, S., Clear, J., dan Ostler, N. (1992). Corpus design criteria. *Literary and Linguistic Computing*, 7(1):1–16.

Bentivogli, L., Cabrio, E., Dagan, I., Giampiccolo, D., Leggio, M. L., dan Magnini, B. (2010). Building textual entailment specialized data sets: a methodology for isolating linguistic phenomena relevant to inference. In Chair), N. C. C., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., dan Tapias, D., editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Bhaskar, P. dan Pakray, P. (2013). Automatic evaluation of summary using textual entailment. In *Proceedings of the Student Research Workshop associated with RANLP 2013*, pages 30–37, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Biber, D. (1993). Representativeness in corpus design. *Literary and linguistic computing*, 8(4):243–257.

Blitzer, J. dan Zhu, X. J. (2008). Semi-supervised learning for natural language processing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Tutorial Abstracts*, pages 3–3. Association for Computational Linguistics.

Blum, A. dan Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, Madison, WI.

Bos, J., Zanzotto, F. M., dan 'Pennacchiotti', M. (2009). 'textual entailment at evalita 2009'. In *'Proceedings of EVALITA 2009'*, pages 1–7.

Bowman, S. R., Angeli, G., Potts, C., dan Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015*

Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics.

Burger, J. dan Ferro, L. (2005). Generating an entailment corpus from news headlines. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, EMSEE '05, pages 49–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chapelle, O., Schölkopf, B., Zien, A., et al. (2006). *Semi-supervised learning*, volume 2. MIT press Cambridge.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Collins, M. dan Singer, Y. (1999). Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.

Dagan, I. dan Glickman, O. (2004). Probabilistic Textual Entailment: Generic Applied Modeling of Language Variability. In *Learning Methods for Text Understanding and Mining*.

Dagan, I., Glickman, O., dan Magnini, B. (2006). The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pages 177–190, Berlin, Heidelberg. Springer-Verlag.

Denoyer, L. dan Gallinari, P. (2006). The wikipedia xml corpus. *SIGIR Forum*, 40(1):64–69.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Giampiccolo, D., Dang, H. T., Magnini, B., Dagan, I., Cabrio, E., dan Dolan, B. (2008). The fourth pascal recognizing textual entailment challenge. *TAC 2008 Proceedings*.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., dan Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–68.

Graves, A., Mohamed, A.-R., dan Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649.

Grishman, R. (1997). Information extraction: Techniques and challenges. In *Information extraction a multidisciplinary approach to an emerging information technology*, pages 10–27. Springer.

Hammerton, J. (2003). Named entity recognition with long short-term memory. pages 172–175.

Harabagiu, S. dan Hickl, A. (2006). Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 905–912.

Hochreiter, S. dan Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jones, K. dan Galliers, J. (1995). *Evaluating natural language processing systems: An analysis and review*, volume 1083 of *Volumer24, Number 2*. Springer Verlag.

Jurafsky, D. dan Martin, J. H. (2014). *Speech and language processing*. Pearson.

Landis, J. dan Koch, G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

Liu, Y., Sun, C., Lin, L., dan Wang, X. (2016). Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Luthfi, A., Distiawan, B., dan Manurung, R. (2014). Building an indonesian named entity recognizer using wikipedia and dbpedia. pages 19–22.

Manning, C. D., Raghavan, P., dan Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Medelyan, O., Milne, D., Legg, C., dan Witten, I. H. (2009). Mining meaning from wikipedia. *Int. J. Hum.-Comput. Stud.*, 67(9):716–754.

Mikolov, T., Chen, K., Corrado, G., dan Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mohri, M., Rostamizadeh, A., dan Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

Olson, D. L. dan Delen, D. (2008). *Advanced data mining techniques*. Springer Science & Business Media.

Paterson, M. dan Dancík, V. (1994). Longest common subsequences. In *Proceedings of the 19th International Symposium on Mathematical Foundations of Computer Science 1994*, MFCS '94, pages 127–142, London, UK, UK. Springer-Verlag.

Peñas, A., Rodrigo, A., dan Verdejo, F. (2006). Sparte, a test suite for recognising textual entailment in spanish. In *Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'06, pages 275–286, Berlin, Heidelberg. Springer-Verlag.

Pennington, J., Socher, R., dan Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Pham, Q. N. M., Nguyen, L. M., dan Shimazu, A. (2012). An empirical study of recognizing textual entailment in japanese text. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 438–449. Springer.

Radev, D. R. (2000). A common theory of information fusion from multiple text sources step one: Cross-document structure. In *Proceedings of the 1st SIGdial Workshop on Discourse and Dialogue - Volume 10*, SIGDIAL '00, pages 74–83, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kociský, T., dan Blunsom, P. (2015). Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Sacaleanu, B., Orasan, C., Spurk, C., Ou, S., Ferrandez, O., Kouylekov, M., dan Negri, M. (2008). Entailment-based question answering for structured data. In *Coling 2008: Companion volume: Demonstrations*, pages 173–176, Manchester, UK. Coling 2008 Organizing Committee.

Shinyama, Y. dan Sekine, S. (2003). Paraphrase acquisition for information extraction. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 65–71. Association for Computational Linguistics.

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., dan Chanona-Hernández, L. (2014). Syntactic n-grams as machine learning features for natural language processing. *Expert Syst. Appl.*, 41(3):853–860.

Trisedya, B. D. dan Inastra, D. (2014). Creating indonesian-javanese parallel corpora using wikipedia articles. pages 239–245.

Turian, J., Ratinov, L., dan Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wang, S. dan Jiang, J. (2016). Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451, San Diego, California. Association for Computational Linguistics.

Xiao, R. (2010). Corpus creation.

Zanzotto, F. M. dan Pennacchiotti, M. (2010). Expanding textual entailment corpora fromwikipedia using co-training. In *Proceedings of the 2nd Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 28–36, Beijing, China. Coling 2010 Organizing Committee.

Zeller, B. dan Padó, S. (2013). A Search Task Dataset for German Textual Entailment . In *Proceedings of the 10th International Conference on Computational Semantics (IWCS)*, pages 288–299, Potsdam.