DAFTAR ISI

Daftar Isi ii

Daftar Gambar v

Daftar Tabel vi

Daftar Kode vii

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 . . . 3
- 1.5 Tahapan Penelitian . . . 3
- 1.6 Sistematika Penulisan . . . 4

2 TINJAUAN PUSTAKA 5
- 2.1 Leksikal Semantik . . . 5
  - 2.1.1 Word Features . . . 5
    - 2.1.1.1 Orthographic . . . 6
    - 2.1.1.2 Sense . . . 6
    - 2.1.1.3 Word Embedding . . . 6
    - 2.1.1.4 POS Tag . . . 6
  - 2.1.2 Relasi Kata . . . 6
- 2.2 Leksikal Semantik Resources . . . 8
  - 2.2.1 WordNet . . . 8
  - 2.2.2 Wikipedia . . . 9
- 2.3 Relation Extraction . . . 10
  - 2.3.1 Semantic Relation Extraction . . . 11
- 2.4 Pattern Analysis . . . 12
  - 2.4.1 Textual Pattern . . . 12
  - 2.4.2 Pattern Extraction . . . 13
  - 2.4.3 Pattern Matching . . . 13
- 2.5 Tree Representation . . . 13
  - 2.5.1 Standard Trie . . . 13
  - 2.5.2 Suffix Tree . . . 14
- 2.6 Semi Supervised . . . 15
  - 2.6.1 Bootstrapping . . . 15
  - 2.6.2 Meta Bootstrapping . . . 16
  - 2.6.3 Basilisk . . . 16
- 2.7 Evaluasi . . . 17

2.7.1 Sampling 17
2.7.2 Precision dan Recall 18
2.7.3 Kappa 18
2.7.4 Spearman's Rho 19

3 RANCANGAN METODOLOGI 21
3.1 Rancangan Pengembangan Korpus 21
3.2 Pre-processing Data 23
3.2.1 Pembentukan Kalimat Wikipedia 23
3.2.2 Pengumpulan Seed 25
3.3 Pembentukan Pattern 25
3.3.1 Sentence Tagging 26
3.3.2 Pattern Extraction 26
3.4 Ekstraksi Pair 27
3.5 Cycle Semi-Supervised 28
3.6 Metode Evaluasi 29
3.6.1 Evaluasi Pattern 29
3.6.2 Evaluasi Pair 30

4 IMPLEMENTASI 32
4.1 Pembentukan Korpus Kalimat Wikipedia 32
4.1.1 Sentence Splitting 33
4.1.2 Rule Based Formatter 33
4.1.3 POS Tagging Kalimat Wikipedia 34
4.2 Seed Builder 34
4.3 Sentence Tagging 36
4.4 Pattern Extraction 37
4.4.1 Informasi dalam Pattern 38
4.4.2 Pattern Tree 38
4.4.3 Vektor Pattern 39
4.4.4 Validasi Pattern 40
4.4.5 Pembentukan *Pattern* Unik 40
4.4.6 Pengurutan Pattern 41
4.5 Pattern Matching 42
4.5.1 Vektor Pair 44
4.5.2 Filterisasi Pair 44
4.6 Pemodelan Word Embedding 44

5 EVALUASI DAN HASIL 46
5.1 Pengumpulan Data Wikipedia 46
5.2 Hasil Pengolahan Data Wikipedia 46
5.2.1 Ekstraksi Teks 46
5.2.2 Pembentukan Kalimat 46
5.2.3 Hasil POS Tagging Kalimat 47
5.2.4 Pemodelan Word Embedding 47
5.3 Pengumpulan Seed 47
5.4 Pembentukan Pattern Pertama 48

5.4.1 Sentence Tagging dengan Seed 48
5.4.2 Hasil Pattern Extraction 49
5.5 Hasil Eksperimen 49
5.5.1 Eksperimen 1 49
5.5.2 Eksperimen 2 51
5.5.3 Eksperimen 3 52
5.5.4 Evaluasi Eksperimen 53
5.6 Hasil Evaluasi Pair 54
5.6.1 Tingkat Persetujuan Anotator Pair 54
5.6.2 Analisis Pair 55
5.7 Hasil Evaluasi Pattern 55
5.7.1 Pattern Buatan Manual 55
5.7.2 Hasil Anotasi Pattern 56
5.7.3 Tingkat Persetujuan Anotator Pattern 57
5.7.4 Analisis Pattern 58

Daftar Referensi **60**

LAMPIRAN **1**

Lampiran 1 : Pattern Buatan Manual **2**

DAFTAR GAMBAR

2.1 Contoh Standard Trie 14
2.2 Contoh Suffix Trie 15

3.1 Arsitektur Penelititan 22
3.2 Data XML Wikipedia Dump 24
3.3 Korpus kalimat Wikipedia tanpa *postag* 24
3.4 Korpus kalimat Wikipedia *postag* 25

4.1 Pre-processing Data Wikipedia 32
4.2 Proses Pembentukan *Pattern* 37
4.3 Proses ekstraksi *pair* 42

DAFTAR TABEL

2.1 Skala pengukuran Kappa 19

5.1 Berkas model *word embedding* 47
5.2 Jumlah *seed* hasil ekstraksi 47
5.3 Hasil *sentence tagging* dengan *seed* 48
5.4 Pattern terbaik iterasi pertama 49
5.5 Pattern Hasil Eksperimen 1 50
5.6 Pair Hasil Eksperimen 1 50
5.7 Akurasi Eksperimen 1 50
5.8 Pattern Hasil Eksperimen 2 51
5.9 Pair Hasil Eksperimen 2 51
5.10 Akurasi Eksperimen 2 52
5.11 Pattern Hasil Eksperimen 3 52
5.12 Pair Hasil Eksperimen 3 52
5.13 Akurasi Eksperimen 3 53
5.14 Perhitungan tingkat persetujuan anotator *pair* 54
5.15 Perhitungan tingkat persetujuan anotator *pair* 54
5.16 Kategori *pattern* sistem dibandingkan dengan *pattern* manual 56
5.17 Perbandingan hasil anotasi berdasarkan jumlah *pair* benar 57
5.18 Perbandingan hasil anotasi berdasarkan jumlah *pair* salah 57
5.19 Cohen's Kappa untuk setiap dimensi 57

DAFTAR KODE

4.1 Penggunaan Wiki Extractor 32
4.2 Algoritme pembentukan *seed* 35
4.3 Penambahan *pattern* ke dalam *pattern tree* 39
4.4 Proses *parsing* kalimat menjadi *array* 43

BAB 1
PENDAHULUAN

Bab ini membahas mengenai latar belakang penelitian, perumusan masalah, tujuan dan manfaat dilakukannya penelitian, ruang lingkup penelitan, dan terakhir metodologi yang digunakan.

1.1 Latar Belakang

Relasi kata adalah informasi mengenai hubungan yang dimiliki antar satu kata dengan kata yang lain. Informasi tersebut sering digunakan untuk menunjang berbagai penelitian, salah satunya adalah *Question Answering* atau sistem tanya jawab. Suatu pertanyaan dapat langsung dijawab jika diketahui hubungan antar suatu kata dalam pertanyaan dengan relasi yang ditanyakan. Sebagai contoh jika diberi pertanyaan 'Apa ibu kota Indonesia?' dan dimiliki suatu *resource* yang menyimpan relasi ibukota(Indonesia, Jakarta) maka pertanyaan tersebut langsung dapat dijawab dengan jawaban 'Jakarta'.

Salah satu subbagian dalam proses pengembangan sistem tanya jawab tersebut adalah *answer type detection* yang berusaha untuk mencari tahu tipe jawaban sehingga dapat menurunkan lama proses pencarian (Jurafsky dan James, 2000). Tipe jawaban dapat dibangun dalam bentuk hirarki atau yang dikenal sebagai *answer type taxonomy*. Sebuah kata dalam pertanyaan mengandung informasi yang dapat mengenali tipe jawaban. Mengetahui hipernim dan hiponim kata tersebut dapat membantu mencari tahu tipe jawaban.

Berikut adalah beberapa contoh pertanyaan yang jawabannya dengan memanfaatkan informasi relasi semantik dari *question headwords*-nya.

Q1. What are aquatic vertebrates?

Q2. Which type of cat that has short hair and blue eyes?

Pertanyaan pertama ingin mengetahui apa saja vertebrata yang hidup di air, sementara pertanyaan kedua ingin mengetahui jenis kucing yang berbulu pendek dan bermata biru. Pada pertanyaan pertama, *question headword*-nya adalah 'aquatic vertebrate'. Jika dilihat menggunakan salah satu kamus online populer, yaitu WordNet, dapat langsung diketahui jawabanya adalah 'fish' (ikan) melalui relasi semantik yang dimiliki oleh kata tersebut. Untuk pertanyaan kedua dapat diketahui bahwa domain jawabanya merupakan 'cat' (kucing) sehingga dapat langsung didaftar je-

nis kucing yang memiliki ciri-ciri yang sesuai. Hal tersebut memperlihatkan bahwa relasi semantik antar kata dibutuhkan untuk mempermudah penyelesaia masalah *question answering*.

Resource pasangan kata relasi semantik dalam Bahasa Inggris dapat diambil dari salah satu kamus digital populer yaitu WordNet (Miller, 1995). WordNet tersebut dibangun secara manual oleh berbagai ahli linguistik. Setiap *entry* pada WordNet disimpan dalam bentuk set sinonim atau biasa disebut *synset* dan arti dari *synset* tersebut atau biasa disebut *sense*. *Entry* dalam WordNet juga mengandung informasi mengenai relasi semantik antar *synset*. Penelitian mengenai pembangunan WordNet Bahasa Indonesia telah dilakukan sebelumnya, sayangnya jumlah *entry* dalam WordNet Bahasa Indonesia masih sangat terbatas. Selain itu, relasi semantik dalam WordNet juga belum dapat dikatakan baik. Mengetahui hal tersebut, penelitian ini berusaha membangun korpus pasangan kata relasi Behasa Indonesia.

Penelitian ini berusaha mengekstrak kata berdasarkan relasi tertentu dari suatu dokumen sehingga dihasilkan korpus pasanagan relasi kata. Fokus relasi dalam penelitian ini adalah relasi semantik hiperim-hiponim. Keduanya menyatakan relasi antara kata yang lebih umum (hipernim) dengan kata yang lebih khusus (hiponim). Berbagai penelitian sebelumnya sudah pernah mencoba mengekstrak relasi kata hipernim-hiponim dengan berbagai metode. Pada penelitian ini, metode yang digunakan adalah *pattern extraction* dan *matching* dengan memanfaatkan korpus Wikipedia Bahasa Indonesia. Wikipedia memuat banyak kata dari berbagai domain sehingga dapat dimanfaatkan untuk membuat *pattern* yang general serta menghasilkan korpus berukuran besar.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, pertanyaan yang menjadi rumusan penelitian adalah sebagai berikut.

1. Bagaimana cara membangun korpus pasangan kata relasi hypernim-hiponim berkualitas baik secara otomatis?

2. Seberapa baik metode *pattern extraction* dan *matching* digunakan untuk ekstraksi pasangan kata relasi semantik kata Bahasa Indonesia?

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari penelitian *word relation extraction* ini adalah membangun korpus pasangan relasi semantik kata Bahasa Indonesia berukuran besar dan berkualitas

baik secara otomatis. Selain itu, ingin diketahui pula apakah metode *pattern extraction* dan *matching* baik digunakan untuk mengekstrak relasi kata Bahasa Indonesia. Diharapkan korpus pasangan kata relasi yang dihasilkan dapat menunjang berbagai penelitian selanjutnya.

Penelitian ini juga diharapkan dapat memotivasi adanya penelitian selanjutnya di bidang *Language Resource Development*, terutama pembangunan WordNet Bahasa Indonesia. Penelitian mengenai ekstraksi relasi kata berikutnya dengan berbagai metode lain diharapkan terus dilaksanakan sehingga Bahasa Indonesia memiliki *resource* yang semakin baik.

1.4 Ruang Lingkup Penelitian

Penelitian ini berupaya membangun korpus pasangan kata relasi semantik untuk Bahasa Indonesia. Relasi semantik yang menjadi fokus penelitian adalah hipernim dan hiponim, dengan kelas katanya yaitu *noun* dan *proper noun*. Penelitian ini berusaha mengekstrak tidak hanya kata-kata yang merupakan *single token* seperti 'komputer' dan 'sekolah', namun juga kata-kata yang merupakan *multi token* seperti 'bulu tangkis' serta *noun phrase* seperti 'mamalia laut'.

1.5 Tahapan Penelitian

Proses penelitian dilakukan dalam beberapa tahapan sebagai berikut.

1. Studi Literatur
   Pada tahap ini, dilakukan pembelajaran mengenai penelitian-penelitian sebelumnya yang telah dilakukan di bidang *word relation extraction* sehingga diketahui langkah yang perlu diambil selanjutnya.

2. Perumusan Masalah
   Perumusan masalah dilakukan untuk mendefinisikan masalah yang ingin diselesaikan, tujuan penelitian, dan hasil yang diharapkan sehingga proses penelitian dapat berjalan dengan baik.

3. Rancangan Penelitian
   Setelah diketahui hasil yang ingin dicapai, dirancang tahap-tahap eksperimen secara terstruktur. Hal-hal yang diperhatikan mulai dari pengumpulan korpus awal (*seed*), *pre-processing* dokumen, perancangan implementasi *pattern extraction matching*, hingga proses evaluasi.

4. Implementasi
   Implementasi dilaksanakan sesuai dengan rancangan penelitian untuk menjawab rumusan masalah. Segala hasil yang ditemukan digunakan untuk terus memperbaiki metode dan teknik penelitan sehingga didapatkan hasil yang semakin baik.

5. Analisis dan Kesimpulan
   Tahap terakhir dari penelitian ini adalah menganalisis korpus pasangan kata relasi yang dihasilkan. Pertanyaan dari perumusan masalah dijawab, kemudian ditarik kesimpulan.

1.6 Sistematika Penulisan

Sistematika penulisan laporan ini adalah sebagai berikut.

- Bab 1 PENDAHULUAN
  Pada bab ini, dijelaskan latar belakang topik penelitian. Selain itu, perumusan masalah, tujuan penelitan, ruang lingkup penelitian, serta tahapan penelitan dipaparkan dalam bab ini.

- Bab 2 TINJAUAN PUSTAKA
  Bab ini memaparkan teori-teori dasar yang menjadi pedoman penelitian. Seluruh studi literatur mengenai teknik-teknik yang digunakan seperti *pattern matching* dan *extraction*, arsitektur *semi-supervised*, metode evaluasi dan hal-hal mendasar lain yang berhubungan dengan penelitian ini.

- Bab 3 RANCANGAN METODOLOGI

- Bab 4 IMPLEMENTASI

- Bab 5 EVALUASI DAN HASIL

- Bab 6 KESIMPULAN DAN SARAN

BAB 2
TINJAUAN PUSTAKA

Pada bab ini, dijelaskan mengenai studi literatur yang dilakukan. Studi literatur yang dilakukan digunakan sebagai dasar konsep dan teknik penelitian. Dipaparkan pula berbagai istilah dan metode yang digunakan dalam penelitian.

2.1 Leksikal Semantik

Dalam *natural language processing*, terdapat beberapa tingkatan untuk merepresentasikan suatu informasi yaitu kata, sintak, dan semantik. Kata adalah kumpulan simbol yang memiliki arti (*sense*) tertentu. Sintak berarti struktur dari kata yang bila digabung akan membentuk arti baru. Semenatara semantik berarti arti atau makna dari kata itu sendiri. Suatu kata tidak hanya mengandung makna namun juga relasi antar kata serta struktur internal. Studi yang mempelajari sistematik struktur serta relasi semantik disebut Leksikal Semantik (Jurafsky dan James, 2000).

Sebelum melangkah lebih lanjut, perlu diketahui beberapa istilah dasar dalam bidang ini. Kata dapat disebut sebagai lexeme yang mengandung suatu bentuk orthographic dan arti di dalamnya. Kumpulan lexeme tersimpan dalam lexicon atau dapat juga dikenal sebagai kamus. Arti dari suatu lexeme sebenarnya merupakan rangkaian lexeme lain yang mendeskripsikannya.

Bahasa Inggris memiliki kamus digital yang menyimpan segala informasi kata dan strukturnya yang disebut WordNet[1]. WordNet tersebut sering digunakan untuk menunjang berbagai penelitian di bidang *natural language processing*.

2.1.1 Word Features

Suatu kata dapat dilihat dari berbagai bentuk, seperti bentuk orthographic dan phonemic-nya. Suatu kata juga memiliki arti (*sense*) yang merepresentasikan deskripsi terhadap kata tersebut. Sayangnya, bentuk-bentuk tersebut tidak dapat dengan mudah diproses oleh komputer. Melalui berbagai penelitian yang telah dilakukan sebelumnya, kata dapat direpresentasikan ke dalam bentuk yang dapat dibaca oleh mesin. Berikut adalah beberapa perbandingan bentuk representasi kata yang bisa dibaca oleh mesin dan manusia.

[1] wordnet.princeton.edu

2.1.1.1 Orthographic

Orthographic adalah bentuk paling dasar dari suatu kata. Bentuk ini merepresentasikan rangkaian simbol-simbol yang tersusun membentuk suatu kata yang memiliki arti. Studi mengenai bentuk ini banyak digunakan untuk mengetahui perbandingan bentuk kata dasar dengan kata berimbuhan. Bentuk phonemic adalah bagaimana kata tersebut dilafakan. Karena penelitian ini hanya berfokus pada teks, maka hanya diperhatikan bentuk orthographic suatu kata.

2.1.1.2 Sense

Sense adalah makna dari kata tersebut atau definisi kata. Informasi mengenai makna kata disimpan dalam kamus bahasa tersebut. Dalam kamus (lexicon), sense terbentuk dari lexeme-lexeme lain yang mendeskripsikan lexeme tersebut.

2.1.1.3 Word Embedding

Word Embedding digunakan untuk menentukan similarity antar kata relasi yang dihasilkan dari proses *Pattern Matching*.

2.1.1.4 POS Tag

Part-of-speech tagging adalah proses mengelompokan setiap kata dalam suatu kalimat ke dalam kategori yang bersesuaian.

2.1.2 Relasi Kata

Sebelum membahas lebih lanjut mengenai relasi kata, perlu diketahui apa itu relasi. Relasi menggambarkan hubungan atau koneksi yang dimiliki oleh suatu hal dengan yang lain (KBBI). Dalam bidang matematika, relasi memetakan suatu anggota dari himpunan satu ke himpunan lain sesuai dengan hubungan yang didefinisikan. Dalam penelitian ini, relasi yang diperhatikan adalah relasi kata yang berarti satu kata akan dipetakan ke dalam kata lain. Domain untuk kata tersebut adalah kata benda (*noun*) dalam Bahasa Indonesia.

Satu relasi dapat terdiri dari beberapa entitas dan dituliskan dalam bentuk tuple $t = (e_1, e_2, ..., e_n)$ dimana e_i adalah suatu entitas yang memiliki relasi r dalam dokumen D (Bach dan Badaskar, 2007). Relasi sinonim dapat ditulis dalam notasi tersebut. Selain relasi yang mengandung beberapa entitasi, banyak relasi yang hanya menghubungkan antar dua entitas (relasi biner), seperti *terletak-di(Universitas Indonesia, Depok)* atau *ditulis-oleh(Habis Gelap Terbitlah Terang, RA Kartini)*. Re-

lasi kata sendiri dapat didefinisikan secara bebas seperti contoh sebelumnya maupun merupakan relasi khusus seperti relasi semantik.

Semantik adalah arti (*sense*) dari suatu kata. Relasi semantik kata adalah hubungan yang dimiliki antar kata berdasarkan arti atau makna dari kata tersebut. Beberapa relasi semantik adalah sebagai berikut (Miller, 1995).

- Sinonim adalah relasi antar kata dimana dua kata yang berbeda memiliki arti yang sama. Semua kelas kata dapat memiliki relasi sinonim. Dalam WordNet, relasi ini direpresentasikan dalam bentuk *synset* dan bersifat simetris. Sebagai contoh 'makan', 'melahap', dan 'menyantap' memiliki makna yang sama.

- Antonim adalah yang menggambarkan arti yang saling berkebalikan antar kata. Umumnya relasi ini digunakan pada kelas kata sifat (*adverb*) dan kata keterangan (*adjective*). Sama seperti synonymy, relasi ini memiliki sifat simetris. Sebagai contoh kata 'tinggi' memiliki makna yang berkebalikan dengan kata 'pendek'.

- Hiponim adalah relasi yang menyatakan hubungan kata yang lebih khusus. Sementara untuk kata yang lebih umum dikenal dengan relasi hipernim. Kedua relasi ini diperuntukan kelas kata benda (*noun*) dan umumnya satu kata memiliki hanya satu hipernim. Kedua relasi ini bersifat transitif, sehingga dapat digambarkan dalam bentuk hirarki. Sebagai contoh kucing, ikan, kelinci (hiponim) adalah binatang (hipernim). Binatang adalah hiponim dari makhluk hidup. Sehingga dapat dikatakan pula bahwa kucing, ikan, kelinci (hiponim) adalah makhluk hidup (hipernim).

- Meronim dan holonim adalah relasi yang menyatakan hubungan bagian satu dengan yang lain, dimana meronim menyatakan sub-bagian dan holonim menyatakan bagian yang lebih besar. Seperti relasi hyponym-hypernym, relasi meronim-holonim bersifat transitif dan dapat digambarkan dalam bentuk hirarki. Dalam WordNet, relasi ini dibagi ke dalam tiga bagian yaitu *part-meronym*, *member-meronym*, dan *substance-meronym*. Sebagai contoh sebuah sel (holonim) memiliki nukleus, ribosom, mitokondria (meronim).

- *Troponymy* adalah relasi seperti hiponim-hipernim yang khusus untuk kelas kata kerja (*verb*). Dalam Bahasa Inggris, contoh kata yang memiliki relasi ini adalah 'stroll' dan 'walk'.

2.2 Leksikal Semantik Resources

Kebutuhan menyimpan suatu korpus yang merupakan leksikal semantik resource. Informasi mengenai kata, makna, serta relasi di dalamnya tersebut perlu tersimpan secara baik sehingga dapat digunakan. Informasi tersebut juga perlu disimpan sedemikian sehingga dapat dapat dibaca mesin dan digunakan untuk proses komputasi. Bahasa Inggris telah membuat kamus digital yaitu WordNet yang banyak digunakan untuk berbagai penelitian. Sementara untuk Bahasa Indonesia, penelitian yang berusaha membangun WordNet telah dilakukan sebelumnya, namun masih ditemukan beberapa kekurangan dari WordNet Bahasa yang ada.

Leksikal semantik resource yang terkenal diantaranya adalah kamus, thesaurus, ensiklopedia, dan wordnet.

2.2.1 WordNet

WordNet adalah kamus leksikal yang tersimpan secara digital dan digunakan untuk berbagai keperluan komputasi (Miller, 1995). Pembuatan WordNet dilatarbelakangi keperluan mendapatkan *sense* atau arti semantik suatu kata. Informasi tersebut perlu disimpan dan dapat dibaca oleh mesin. WordNet pertama dibuat oleh Miller (1995) berbasis Bahasa Inggris dan sekarang dikenal dengan nama Princeton WordNet (PWN). WordNet menyimpan informasi dalam bentuk database dimana setiap entry-nya adalah pasangan *synset* dan arti semantiknya (*sense*). Set sinonim (*synset*) adalah himpunan kata yang memiliki arti yang sama atau saling berelasi *synonym*.

Kata dalam WordNet dikelompokkan ke dalam beberapa kelas kata yaitu kata benda (*noun*), kata kerja (*verb*), kata sifat (*adjective*), dan kata keterangan (*adverb*). WordNet juga menyimpan informasi mengenai relasi semantik antar *synset*. Relasi yang disimpan adalah sinonim, *antonymy*, hiponim, hipernim, *meronymy*, *holonymy*, *troponymy*, dan *entailment*. Hingga penulisan ini, versi WordNet yang sudah ada adalah versi 3.1 yang dapat diakses melalui situs resminya maupun diunduh datanya. Sementara aplikasi yang sudah terintegrasi dengan sistem UNIX/Linux adalah versi 3.0.

Penelitian mengenai WordNet Bahasa Indonesia pernah dilakukan sebelumnya oleh Putra et al. (2008) serta Margaretha dan Manurung (2008). Indonesian WordNet (IWN) dibangun menggunakan metode mapping antara WordNet yang sudah ada ke dalam Bahasa Indonesia (Putra et al., 2008). WordNet yang digunakan sebagai dasarnya adalah Princeton WordNet. Synset dalam PWN akan dipetakan ke dalam *entry* Kamus Besar Bahasa Indonesia (KBBI), sehingga menghasilkan

hasil yang berkualitas baik secara cepat dan mudah. Penelitian tersebut menghasilkan 1441 synset dan 3074 sense. Relasi semantik antar synset diperoleh dengan memetakan IWN synset dengan PWN synset, sehingga relasi yang dimiliki dalam PWN dapat diturunkan.

Pengembangan WordNet untuk Bahasa Indonesia juga dilakukan oleh Nanyang Technology University (NTU) sejak tahun 2011 dan diberi nama WordNet Bahasa (Noor et al., 2011). WordNet ini telah diintegrasi dengan salah satu *tools* NLP berbasis Python yaitu nltk sehingga dapat dengan mudah digunakan dalam komputasi. WordNet Bahasa juga memanfaatkan PWN untuk mendapatkan relasi semantik antar *synset*. Pada penelitian ini, dimanfaatkan *tools* tersebut untuk mendapatkan *seed* relasi semantik antar kata dalam Bahasa Indonesia.

Walau beberapa penelitian sudah dilakukan sebelumnya, WordNet Bahasa Indonesia masih memiliki beberapa kekurangan. Jumlah kata yang terkandung di dalamnya masih terbatas. Sementara itu, relasi semantik antar kata yang dimiliki oleh WordNet Bahasa Indonesia merupakan hasil turunan dari relasi semantik WordNet Princeton. Hal ini menyebabkan ketergantungan untuk mendapatkan relasi semantik dengan struktur dari PWN. Selain itu, beberapa *synset* Bahasa Indonesia juga tidak dapat dipetakan secara tepat ke *synset* PWN yang menyebabkan beberapa kata kehilangan arti atau mendapat arti yang kurang tepat. Jika berusaha dibentuk ke dalam relasi biner, pasangan kata relasi semantik Bahasa Indonesia yang dihasilkan terlihat kurang baik. Untuk itu, dicetuskanlah penelitian untuk mengekstrak relasi semantik dalam Bahasa Indonesia secara mandiri.

2.2.2 Wikipedia

Wikipedia[2] adalah ensiklopedia terbuka yang memuat berbagai bahasa dan merupakan hasil kolaborasi banyak penulis (Denoyer dan Gallinari, 2006). Wikipedia adalah salah satu korpus teks dokumen terbesar yang disimpan secara *online* dan dapat diakses serta diunduh secara bebas. Wikipedia dikelola oleh organisasi nonprofit bernama Wikimedia Foundation. Pada tahun 2009, jumlah kontributor Wikipedia Bahasa Indonesia telah mencapai 2.502 pengguna aktif. Walau ditulis oleh berbagai narasumber, informasi yang dimuat dalam Wikipedia dibuat secara terstruktur dengan bahasa yang formal. Wikipedia juga memuat informasi umum terbaru (Arnold dan Rahm, 2014).

Hingga Mei 2017, Wikipedia Bahasa Indonesia telah memuat lebih dari 400.000 artikel dari berbagai domain dan terus berkembang secara pesat. Artikel-artikel

[2]www.wikipedia.org

yang disimpan dalam Wikipedia dapat diunduh secara gratis dalam bentuk *dumps*[3] dengan format XML. Beberapa tipe *dump* yang dapat dipilih, diantarnya adalah halaman seluruh artikel, halaman artikel beserta *revision history*, daftar judul artikel, dan lainnya. Secara berkala, Wikimedia membuat *dump* terhadap seluruh artikel terakhir yang disimpan untuk setiap bahasa. Pada situs yang menyediakan pengunduhan data Wikipedia, terdapat tanggal unik yang menyatakan tanggal terakhir data tersebut di-*update*.

Artikel dalam Wikipedia terdiri dari berbagai domain kategori dan memuat berbagai entitas leksikal. Kata-kata tersebut dapat ditemukan dalam berbagai leksikal semantik *resource* yang sudah ada. Zesch et al. (2007) melakukan penelitian yang membandingkan Wikipedia dengan leksikal semantik resource yang sudah ada seperti kamus, thesaurus, wordnet, dan ensiklopedia. Istilah-istilah yang ada dalam Wikipedia lebih mirip dengan ensiklopedia dibanding kamus, thesaurus, maupun wordnet karena memuat banyak kata benda (*noun*) dan beberapa kata sifat (*adjective*) maupun kata kerja (*verb*). Selain itu, Wikipedia juga mengandung banyak kata yang merupakan *named entity* atau *domain specific terms*. Penelitian ini berfokus pada kata-kata yang merupakan kata benda, sehingga Wikipedia cocok digunakan.

Walau ditulis secara kolaboratif, Wikipedia memuat artikel secara terstruktur dimana pada paragraf pertama umumnya berisi kalimat-kalimat definisi menganai topik yang sedang dibahas. Bahasa yang ditulis juga cukup formal dan terstruktur. Sudah banyak penelitian ekstraksi relasi semantik dengan metode *pattern* yang menggunakan artikel Wikipedia seperti penelitian yang dilakukan oleh Ruiz-Casado et al. (2005) dan Arnold dan Rahm (2014). Melihat banyaknya manfaat Wikipedia, diputuskan untuk menggunakan Wikipedia sebagai leksikal semantik *resource* dalam penelitian ini.

2.3 Relation Extraction

Relation extraction adalah cabang dari *Information Extraction* (IE) yang berfokus pada proses ekstraksi relasi antar kata. Proses ini berusaha mengekstrak informasi terstruktur dengan definisi yang diinginkan dari teks dokumen atau *resource* yang tidak terstruktur. Beberapa contoh penelitian ini seperti *named entity recognition* (NER) yang mengetahui apakah suatu entitas adalah orang, organisasi, atau lokasi (Bikel et al., 1999).

[3] dumps.wikimedia.org

2.3.1 Semantic Relation Extraction

Semantic relation extraction mengkhususkan pada proses ekstraksi relasi semantik antar kata. Penelitian dalam bidang ini sudah banyak dilakukan dengan berbagai metode. Salah satu metode populer untuk mendapatkan relasi semantik satu domain bahasa adalah menggunakan *pattern extraction* dan *pattern matching* seperti yang telah dilakukan pada penelitian Hearst (1992), Ruiz-Casado et al. (2005), dan Arnold dan Rahm (2014). Penelitian lain memanfaatkan distribusi kata untuk memperoleh *semantic distance* antar kata. *Semantic distance* adalah nilai yang merepresentasikan kedekatan antar kata berdasarkan semantiknya.

Penelitian yang dilakukan oleh Hearst (1992) merupakan salah satu penelitian awal ekstraksi relasi semantik yang menggunakan metode *pattern extraction* dan *matching*. Hearst menggunakan *lexico-syntactic pattern* untuk mendapatkan pasangan kata relasi tanpa membutuhkan pengetahuan sebelumnya dan dapat diaplikasikan dalam teks apapun. Pada awal, Hearst mendifiniskan dua *pattern* berdasarkan hasil observasi dari teks. Selanjutnya, *pattern* baru didapat menggunakan langkah berikut.

1. Tentukan relasi yang akan diamati dan kumpulkan entitas yang menggambarkan relasi tersebut.
2. Dalam teks dokumen, cari lokasi dimana entitas-entitas tersebut berada dan simpan kata-kata diantaranya (lingkungan).
3. Cari kesamaan dari teks yang terekstraksi dan bentuk suatu *pattern* baru.
4. Jika *pattern* baru telah terbukti benar, gunakan *pattern* tersebut untuk mendapatkan entitas baru.

Proses evaluasi dilakukan dengan membandingkan entitas yang dihasilkan dengan synset dalam WordNet.

Penelitian yang dilakukan oleh Ruiz-Casado et al. (2005), memanfaatkan WordNet dan Wikipedia sebagai korpus untuk mendapatkan pasangan kata relasi. Pertama dilakukan *entry sense disambiguation* yang merupakan tahap *pre-processing* untuk memetakan setiap entri dalam Wikipedia dengan *synset* dalam WordNet. Tahap berikutnya adalah ekstraksi *pattern* antara dua konsep. Jika terdapat dua konsep yang saling berhubungan dan memiliki suatu relasi semantik dalam WordNet maka kalimat yang mengandung dua konsep tersebut akan disimpan. Proses tersebut menghasilkan daftar relasi semantik dengan masing-masing memiliki *pattern* di dalamnya. Dari banyak *pattern* yang dihasilkan, proses selanjutnya adalah

pattern generalisation yang bertujuan membuat pattern yang lebih umum. Tahap ini memanfaatkan algoritma *edit-distance* dengan modifikasi. Setelah mendapatkan *pattern*, tahapan terakhir adalah menggunakannya ke dalam korpus untuk mendapatkan entitas baru.

Penelitian yang dilakukan Arnold dan Rahm (2014) juga memanfaatkan korpus Wikipedia dan menggunakan *pattern*. Selain *hypernym-hyponymy* dan *holonym-meronymy*, penelitian ini juga mengidentifikasi relasi sinonim. Kalimat definisi pada Wikipedia di-*parse* menggunakan *Finite State Machine* (FSM) dan konsep-konsep baru diekstrak menggunakan *pattern* yang telah didefinisikan. Penelitian lain yang dilakukan oleh Sumida dan Torisawa (2008) memanfaatkan struktur internal Wikipedia dalam mengekstraksi *pattern* relasi untuk Bahasa Jepang.

2.4 Pattern Analysis

Pattern adalah suatu bentuk yang dapat merepresentasikan kumpulan data berukuran besar. Pada teks dokumen, *pattern* dapat terbentuk secara eksplisit ataupun implisit. Secara eksplisit seperti *lexico-syntactic pattern* yang terbentuk dari kata-kata di dalam dokumen tersebut. Sementara *pattern* terbentuk secara implisit jika dilhat dari kemiripan vektor yang merepresentasika dokumen atau kata-kata di dalam dokumen tersebut.

Lexico-syntactic pattern adalah *pattern* yang hanya memanfatkan kata-kata dalam korpus dokumen dan dimanfaatkan untuk proses *string matching*. Salah satu *pattern* leksikal yang terkenal adalah Hearst Pattern yang merepresentasikan pola-pola untuk mendapatkan relasi hiponim dari dokumen (Hearst, 1992). Menurut Hearst, beberapa syarat yang harus dipenuhi suatu *lexico-syntactic pattern* yang baik adalah sebagai berikut.

- Kemunculannya sering pada teks dalam berbagai domain sehingga dapat mengekstrak banyak entitas.
- Merepresentasikan relasi yang diinginkan sehingga hasil ekstraksi juga benar.
- Dapat dikenali tanpa membutuhkan pengetahuan sebelumnya sehingga dapat dibentuk dalam situasi apapun.

2.4.1 Textual Pattern

Pattern yang dibentuk mengandung tiga bagian penting, yaitu *tag* hiponim, *tag* hipernim, dan kata lainnya. *Tag* hiponim-hipernim merepresentasikan letak kata

yang merupakan kata hiponim-hipernim berada. Sementara kata lainnya adalah kata lainnya yang harus sama untuk proses *pattern matching*. Sebagai contoh terdapat tekstual *pattern* berikut '<hyponym> adalah <hypernym> yang'. *Pattern* tersebut berarti kata sebelum kata 'adalah' merupakan kata hiponim dan kata setelah kata 'adalah' dan sebelum kata 'yang' merupakan kata hipernim. Jika terdapat kalimat 'harimau adalah kucing yang berukuran besar', maka 'harimau' merupakan hiponim dan 'kucing' merupakan hipernim.

2.4.2 Pattern Extraction

Pattern extraction atau *pattern recognition* adalah salah satu cabang dalam *machine learning* yang berusaha mencari pola kemiripan tertentu dari kumpulan data yang diberikan. *Pattern matching* adalah proses untuk mencocokkan suatu *pattern* dengan kumpulan data yang belum dianotasi. Penelitian ekstraksi relsasi semantik menggunakan metode pattern matching telah dilakukan oleh Hearst (1992), Ruiz-Casado et al. (2005), Arnold dan Rahm (2014), dan Sumida dan Torisawa (2008).

Banyaknya penelitian menggunakan metode *pattern matching* dan *extraction* menjadi salah satu alasan penggunaan metode tersebut untuk mengekstrak relasi kata dalam Bahasa Indonesia. Pada penelitian ini, relasi semantik yang diekstrak dibatasi hanya untuk relasi *hypernym-hyponym*.

2.4.3 Pattern Matching

2.5 Tree Representation

Dalam bidang bioinformatik, *suquence analysis* digunakan untuk mengetahui apakah suatu DNA atau RNA mengandung *sequnce* tertentu. Pada penelitian ini, *pattern* yang dihasilkan adalah *pattern* leksikal yang merupakan deretan kata-kata. Algoritma seperti *standard trie* dan *suffix tree* dimanfaatkan secara berurut untuk proses *pattern extraction* dan *matching*.

2.5.1 Standard Trie

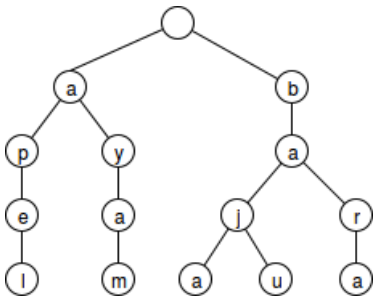
Standard Trie untuk suatu himpunan *string S* adalah *ordered tree* dengan ketentuan berikut.

- Setiap *node*, selain *root*, diberi label sebuah *character*
- *Children* dari sebuah *node* terurut sesuai alphabet

- *Path* dari eksternal *node* hingga *root* membentuk suatu *string* dalam S.

Standard Trie membutuhkan memori sebesar $O(n)$ dimana n adalah total ukuran *string* dalam S. Operasi *insert* membutuh waktu $O(dm)$ dimana m adalah ukuran *string* yang baru dan d adalah ukuran alphabet.

Sebagai contoh berikut adalah *standard trie* yang dibangun dari himpunan *string* apel, ayam, baju, baja, bara.

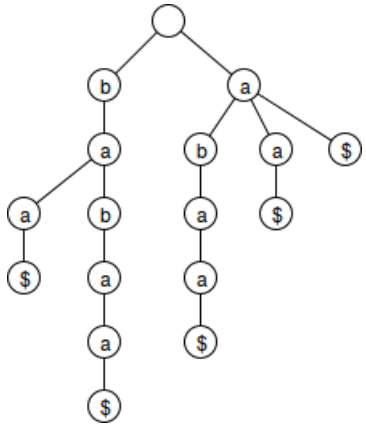


Gambar 2.1: Contoh Standard Trie

2.5.2 Suffix Tree

Suffix tree sering digunakan untuk pencarian sequence yang panjang seperti *genomes* untuk bidang bioinformatik. Pembentukan *suffix tree* mirip seperti *Standard Trie*, namun untuk seluruh *suffix* dalam *string*. Jika diberikan *string* dengan panjang n, dibentuk cabang dengan $n(n-1)/2$ *suffix*. Metode ini banyak dimanfaatkan untuk mempercepat proses pencarian jika diberikan sebuah masukan *query*. Jika terdapat sebuah *pattern* dengan panjang *string* m, maka waktu yang dibutuhkan untuk menjalankan proses *pattern matching* adalah $O(dm)$ dengan d adalah ukuran alfabet. Proses pencarian dilakukan dengan menelusuri *path* dari *root* sesuai dengan *sequence query*. Jika seluruh karakter dalam *query* selesai dijalankan, maka proses pencarian berhasil.

Sebagai contoh *string* 'babaa' menghasilkan *suffix tree* berikut. Jika diberi *query* 'ba' maka akan berhasil terhadap *path* 'babaa' dan 'baa'.



Gambar 2.2: Contoh Suffix Trie

2.6 Semi Supervised

Dalam *machine learning* terdapat dua tipe pendekatan yang umum digunakan yaitu *supervised* dan *unsupervised learning*. Supervised menggunakan data berlabel sebagai data *training* maupun *testing*. Dari kedua data tersebut, dibentuk suatu *classifier* yang dapat memenuhi segala kasus yang mungkin terjadi. Data *testing* digunakan untuk menguji kebaikan *classifier* yang terbentuk. *Unsupervised* menggunakan data yang tidak diberi label sama sekali dan berusaha untuk menemukan pola yang sama untuk suatu kumpulan data tertentu (Prakash dan Nithya, 2014). Pendekatan lain yang merupakan kombinasi antara *supervised* dan *unsupervised learning* adalah *semi supervised learning*.

Semi supervised adalah pendekatan *machine learning* dimana informasi *supervised* data diberikan tidak untuk seluruh data. Sebagian data merupakan data berlabel sementara sebagian lainnya belum memiliki label. Beberapa metode penerapan semi supervised adalah *bootstrapping* (*self training*), *mixture models*, *graph based methods*, *co-training,* dan *multiview learning*.

2.6.1 Bootstrapping

Model *bootstrapping* merupakan salah satu model *semi supervised learning* yang paling umum digunakan. *Bootstrapping* menggunakan data berlabel berukuran kecil dan data tidak berlabel berukuran jauh lebih besar. Proses anotasi data tidak berlabel dilakukan secara bertahap melalui sejumlah iterasi. Dari data *training* berlabel, dibentuk suatu *classifier* yang kemudian digunakan untuk menganotasi data tidak berlabel. Sejumlah *k* data baru yang merupakan hasil pelabelan, dimasukkan ke dalam kelompok data berlabel. Proses tersebut dilakukan secara beru-

lang, sehingga semakin lama iterasi jumlah data berlabel akan bertambah.

Terdapat dua algoritma *bootstrapping* yang pernah digunakan untuk proses *pattern extraction* dan *matching* yaitu *Meta-Bootstrapping* dan *Basilisk* (Riloff et al., 2003). Keduanya digunakan untuk mengelompokan kata ke dalam suatu kategori semantik jika diberikan korpus teks yang belum dianotasi dan suatu *seed*. *Seed* didefinisikan sebagai korpus kata yang sudah diketahui kategori semantiknya. Secara umum, proses ini akan mencari *pattern* berdasarkan seed yang diberikan. Dari *pattern* yang dihasilkan dan teks yang belum dianotasi, diekstrak entitas baru dan dikelompokan berdasarkan kategori semantiknya. Kata-kata tersebut akan digabungkan ke dalam korpus pasangan kata berelasi.

2.6.2 Meta Bootstrapping

Berikut adalah beberapa proses (Riloff et al., 1999) yang dijalankan algoritma *meta bootstrapping* jika diberikan *seed* berukuran kecil yang berasal dari suatu kategori semantik dan korpus yang belum dianotasi.

1. Mengekstraksi *pattern* secara otomatis dengan menerapkan syntactic template.

2. Untuk setiap *pattern* akan diberi bobot berdasarkan jumlah seed yang menghasilkan *pattern*.

3. Diambil *pattern* terbaik dan seluruh seed lama yang merepresentasikan *pattern* maupun *seed* baru yang berhasil diekstrak disimpan.

4. Dilakukan pembobotan ulang untuk setiap *pattern* menggunakan *seed* lama dan baru.

Proses diatas dinamakan *mutual bootstrapping* dan setelah proses tersebut selesai, semua entitas baru hasil ekstraksi dievaluasi. Pembobotan entitas baru diberikan berdasarkan jumlah *pattern* yang mengekstrak kata tersebut. Lima kata terbaik diterima dan dimasukkan ke kamus (korpus) kata berelasi untuk selanjutnya diproses ulang.

2.6.3 Basilisk

Algoritma *Basilisk* (Thelen dan Riloff, 2002) juga memanfaatkan *pattern* dan *seed* dalam membangun korpus untuk suatu kategori semantik tertentu. Beberapa tahapan yang dijalankan adalah sebagai berikut.

1. Secara otomatis membentuk *pattern* dan memberi bobot berdasarkan jumlah seed yang menghasilkan *pattern*. Pattern terbaik dimasukan ke dalam *Pattern Pool*.

2. Untuk setiap entitas baru yang terekstraksi dari *pattern*, dimasukan ke dalam *Candidate Word Pool*. Pemberian bobot dilakukan berdasarkan jumlah *pattern* yang mengekstraksi dan asosiasi kumulatif kata dengan *seed*.

3. Sepuluh kata terbaik diambil dan dimasukan ke dalam kamus (korpus) yang kemudian digunakan untuk iterasi selanjutnya.

Kateogori semantik untuk proses ini bisa lebih dari satu. *Basilisk* memberi bobot berdasarkan informasi kolektif dari kumpulan *pattern* yang mengekstrak kata tersebut. Sementara *Meta-Bootstrapping* hanya mengambil satu *pattern* terbaik dan mengelompokkan seluruh kata yang terekstrak dari *pattern* ke dalam kategori semantik yang sama. Dari hasil penelitian komparatif yang pernah dilakukan (Riloff et al., 2003), didapatkan *Basilisk* mengungguli performa *Meta-Bootstrapping*.

2.7 Evaluasi

Evaluasi dilakukan untuk mengetahui kebaikan hasil penelitian. Evaluasi dapat dilakukan dengan mengukur akurasi data yang dihasilkan. Akurasi adalah nilai perbandingan antara jumlah data yang benar dengan jumlah seluruh data (Manning).

$$akurasi = \frac{jumlah\ data\ benar}{jumlah\ seluruh\ data}$$

Selain menghitung akurasi, proses evaluasi juga menghitung nilai-nilai lainnya. Berikut ada beberapa metode dan teknik evaluasi lain yang digunakan dalam penelitian.

2.7.1 Sampling

Terdapat dua kategori utama dalam *sampling* yaitu *probability* dan *non-probability sampling*. Perbedaan utama keduanya adalah pada *probability sampling*, diambil data secara acak (*random*). Dalam *probability sampling*, terdapat beberapa metode yang dapat digunakan seperti *simple random sampling*, *systematic sampling*, *stratified random sampling*, dan *cluster sampling*.

- *Simple random sampling* perlu mengetahui seluruh data yang ada dan dari data tersebut dipilih secara acak. Hal ini membuat seluruh data memiliki nilai probabilitas terpilih yang sama.

- *Systematic sampling* memilih setiap data ke-n untuk dijadikan *sample*.
- *Stratified random sampling* akan mengelompokan data ke dalam kategori berdasarkan karakteristik tertentu (strata), kemudian data diambil secara acak dari kategori yang ada. Hal ini menyebabkan hasil lebih representatif.
- *Cluster* sampling mirip seperti *stratified sampling* namun dilakukan jika data kelompok yang ingin di-*sampling* sulit berada di lokasi yang terpisah jauh.

Proses *sampling* bermanfaat untuk merepresentasikan data tanpa perlu mengevaluasi seluruh data yang ada. Jika jumlah data yang ingin dievaluasi berukuran besar, proses *sampling* mempercepat pengukuran. Jumlah data yang direpresentasikan oleh satu sample berdasarkan jumlah data asli. Sebagai contoh jika total data adalah 1000 dan jumlah data sample adalah 50, maka satu data *sample* merepresentasikan 20 data asli.

2.7.2 Precision dan Recall

Teknik yang umum digunakan untuk mengevaluasi suatu ekstraksi adalah *precision* dan *recall*. *Precision* adalah nilai yang menyatakan jumlah dokumen benar dan berhasil diambil dibandingkan dengan seluruh jumlah dokumen yang terambil. *Recall* adalah nilai yang menyatakan jumlah dokumen benar dan berhasil diambil dibandingkan dengan jumlah seluruh dokumen yang benar. Semakin banyak dokumen yang diambil maka nilai *recall* akan meningkat sementara nilai *precision* cenderung menurun.

2.7.3 Kappa

Nilai kappa (κ) merepresentasikan tingkat persetujuan antar anotator. Kappa digunakan pada penelitian yang menggunakan bantuan anotator untuk memberi penilaian secara manual. Peniliaian didapatkan menggunakan rumus berikut.

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

- $P(A)$ adalah proporsi penilaian yang setuju (*agreement*)
- $P(E)$ adalah proporsi penilaian yang kebetulan

Landis dan Koch (1977) mendefiniskan tingkat persetujuan berdasarkan nilai Kappa yang diperoleh.

Tabel 2.1: Skala pengukuran Kappa

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*

Beberapa variasi perhitungan untuk Kappa adalah Cohen's Kappa dan Fleiss' Kappa. Cohen's Kappa digunakan untuk mengukur tingkat persetujuan antar dua anotator. Jika diberikan data dengan n label dan $m_i j$ merepresentasikan jumlah data yang diberi label i oleh anotator pertama dan label j oleh anotator kedua, maka proses perhitungan $P(A)$ dan $P(E)$ untuk Cohen's Kappa adalah sebagai berikut.

$$P(A) = \frac{\sum_{k=1}^{n} m_k k}{total\ data}$$

$$P(E) = \frac{\sum_{k=1}^{n} (\sum_{j=1}^{n} m_k j . \sum_{i=1}^{n} m_i k)}{total\ data}$$

Fleiss' Kappa mengukur tingkat persetujuan antar sekelompok anotator berjumlah lebih dari dua. Jika diberikan N data dengan n anotator dimana setiap data diantosi ke dalam salah satu dari k kategori dan $n_i j$ merepresentasikan total anotator yang memberi data i ke label j, proses perhitungan $P(A)$ dan $P(E)$ untuk Fleiss' Kappa adalah sebagai berikut.

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

2.7.4 Spearman's Rho

Spearman's rank correlation coefficient adalah nilai koefisien korelasi antar *ranking* dua parameter. Nilai *Spearman correlation* sama dengan nilai *Pearson correlation* antar dua paramter yang telah di-*ranking*. *Pearson correlation* menggambarkan nilai linear antara dua parameter. *Spearman correlation* berkisar antara -1 hingga $+1$.

Spearman's rho adalah nilai Pearson Correlation Coefficient antar dua variabel yang telah di-*ranking*. Untuk mendapatkan nilai koefisien (r_s), menggunakan rumus

berikut.

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X} \sigma_{rg_Y}}$$

- ρ adalah *Pearson correlation coefficent* yang diaplikasikan pada variabel *ranking*
- $cov(rg_X, rg_Y)$ adalah nilai *covariance* antar variabel *ranking*
- σ_{rg_X} dan σ_{rg_Y} adalah nilai standard deviasi variabel *ranking*

Jika seluruh *ranking* berbeda, proses komputasi dapat dilakukan menggunakan rumus berikut.

$$r_s = 1 - \frac{6\Sigma d_i^2}{n(n^2 - 1)}$$

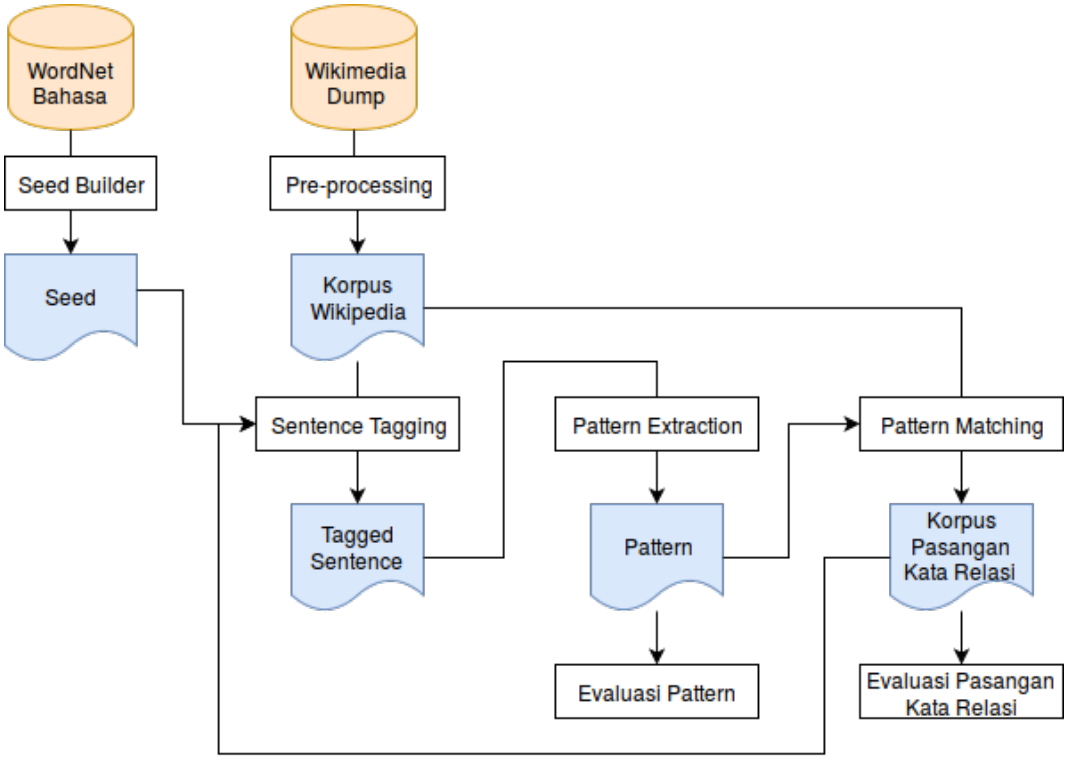
- $d_i = rg(X_i) - rg(Y_i)$ adalah selisih antara dua *ranking*
- n adalah jumlah observasi

BAB 3
RANCANGAN METODOLOGI

Pada bab ini dipaparkan mengenai rancangan dan tahap-tahap proses ekstraksi relasi semantik, mulai dari rancangan pengembangan korpus, pembentukan *seed*, pembentukan *pattern*, ekstraksi *pair*, *cycle semi-supervised*, dan strategi evaluasi yang dilakukan untuk pasangan kata relasi Bahasa Indonesia.

3.1 Rancangan Pengembangan Korpus

Penelitian ini mengusulkan pengembangan korpus pasangan kata berelasi menggunakan arsitektur yang dapat dilihat pada gambar 3.1. Terdapat dua sumber data utama yang digunakan yaitu WordNet Bahasa untuk pembentukan *seed* dan artikel Wikipedia Bahasa Indonesia sebagai korpus teks. Secara garis besar, terdapat enam tahap yang perlu dilakukan yaitu pembuatan *seed*, *pre-processing* data Wikipedia, *sentence tagging*, *pattern extraction*, *pattern matching*, dan terakhir adalah evaluasi. Untuk proses *sentence tagging*, *pattern extraction*, dan *pattern matching* dilakukan secara berulang, sesuai dengan metode *bootstrapping*.



Gambar 3.1: Arsitektur Penelititan

Berikut adalah penjelasan singkat setiap tahapan.

1. *Pre-processing* data

   Sebelum memulai penelitian, terdapat dua hal utama yang perlu dilakukan yaitu pengumpulan *seed* dan pembentukan kalimat Wikipedia. Pengumpulan *seed* dilakukan untuk mendapatkan pasangan kata relasi yang digunakan sebagai dasar penelitian. Proses ini memanfaatkan *resource* yang dimiliki WordNet Bahasa. Selanjutnya, artikel Wikipedia yang diperoleh dalam bentuk *Wikipedia dump* perlu diolah sehingga memiliki representasi sesuai dengan format yang diharapkan. Informasi yang diperlukan hanya bagian isi artikel dan ditulis dalam bentuk kalimat untuk setiap baris.

2. Pembentukan *pattern*

   Menggunakan *seed* dan korpus kalimat Wikipedia, dilakukan *tagging* pasangan kata berelasi terhadap kalimat-kalimat yang ada. Kalimat yang mengandung pasangan kata berelasi akan ditandai dan disimpan sebagai dasar proses selanjutnya. Kalimat-kalimat yang sudah di-*tag* dengan pasangan kata berelasi kemudian akan digunakan untuk proses *pattern extraction*. Hasil dari proses ini adalah sejumlah *pattern* leksikal terbaik dari banyak *pattern* unik yang dihasilkan.

3. Ektraksi pasangan kata relasi
   Pattern yang dihasilkan digunakan untuk mengekstrak pasangan kata relasi baru dengan dibantu korpus Wikipedia dengan *postag*. Proses *pattern matching* dilakukan dengan mencocokan kata-kata bebas dalam *pattern* dengan kalimat, kemudian mengambil bagian yang menempati posisi *tag* hipernim-hiponim. Pasangan kata yang dihasilkan masuk ke dalam korpus pasangan kata relasi hipernim-hiponim.

4. *Cycle semi-supervised*
   Korpus pasangan kata relasi yang terbentuk digunakan untuk iterasi selanjutnya sesuai dengan metode *bootstrapping*. Proses iterasi dilakukan hingga jumlah pasangan kata relasi baru yang dihasilkan jenuh. Setelah satu eksperimen selesai, proses evaluasi dilakukan secara kolektif untuk mengetahui akurasi setiap data yang dihasilkan.

Proses ini diharapkan dapat menghasilkan korpus pasangan kata relasi *hyponym-hypernym* yang berkualitas baik dan berukuran besar. *Pair* yang dihasilkan ditulis dalam bentuk *tuple* $(kata_hyponym; kata_hypernym)$ dengan kedua kata berada dalam kelas kata benda.

3.2 Pre-processing Data

Proses inti dari penelitian ini, *pattern extraction* dan *matching*, memerlukan dua masukan utama yaitu sejumlah pasangan kata hipernim-hiponim dan teks dokumen yang digunakan sebagai korpus. Pasangan kata hipernim-hiponim digunakan untuk proses pembentukan *pattern* sementara teks dokumen digunakan untuk memperoleh pasangan kata baru. Dikarenakan belum ada korpus pasangan kata relasi hipernim-hiponim Bahasa Indonesia, perlu didefinisikan *seed* yang akan digunakan sebagai dasar pasangan kata hipernim-hiponim. Teks dokumen yang digunakan, yaitu Wikipedia, juga memerlukan pemrosesan sebelum menjadi masukan sistem.

3.2.1 Pembentukan Kalimat Wikipedia

Data Wikipedia yang diperoleh dari Wikimedia *dumps* masih mengandung banyak *tag* yang tidak digunakan pada penelitian ini seperti *tag id* dan *revision*. Selain itu simbol-simbol khusus (*markup*). Penelitian ini ingin mengekstrak *pattern* dari *free text*, sehingga format-format khusus tersebut perlu dibersihkan. Gambar 3.2 menunjukan contoh data XML yang diperoleh dari Wikimedia Dump. Setelah

data Wikipedia dibersihkan dari simbol-simbol tersebut, langkah selanjutnya adalah merepresentasikan korpus dalam bentuk kalimat.

```
  <page>
    <title>Asam deoksiribonukleat</title>
    <ns>0</ns>
    <id>1</id>
    <revision>
      <id>12184290</id>
      <parentid>12009030</parentid>
      <timestamp>2017-01-23T04:32:50Z</timestamp>
      <contributor>
        <username>HsfBot</username>
        <id>820984</id>
      </contributor>
      <minor />
      <comment>Bot: 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|Struktur [[heliks ganda]] DNA. [[Atom]]-atom pada struktur ters
ebut diwarnai sesuai dengan [[unsur kimia]]nya dan struktur detail dua pasangan b
asa 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 biomoleku
l yang menyimpan dan menyandi instruksi-instruksi [[genetika]] setiap [[organisme
]] dan banyak jenis [[virus]]. Instruksi-instruksi genetika ini berperan penting
```

Gambar 3.2: Data XML Wikipedia Dump

Artikel-artikel Wikipedia dibentuk ke dalam format yang telah didefinisikan dengan satu kalimat dipisahkan untuk setiap barisnya. Selanjutnya, data akan di format ulang sesuai definisi untuk mempermudah pemrosesan selanjutnya. Beberapa aturan yang diberikan adalah menghilangkan frasa di dalma tanda kurung, menghilangkan simbol, serta memberi *tag start* dan *end* pada awal dan akhir kalimat.

```
<start> Asam deoksiribonukleat , lebih dikenal dengan singkatan DNA , adalah seje
nis biomolekul yang menyimpan dan menyandi instruksi-instruksi genetika setiap or
ganisme dan banyak jenis virus . <end>
<start> Instruksi-instruksi genetika ini berperan penting dalam pertumbuhan , per
kembangan , dan fungsi organisme dan virus . <end>
<start> DNA merupakan asam nukleat ; bersamaan dengan protein dan karbohidrat , a
sam nukleat adalah makromolekul esensial bagi seluruh makhluk hidup yang diketahu
i . <end>
```

Gambar 3.3: Korpus kalimat Wikipedia tanpa *postag*

```
<start>_X Asam_NNP deoksiribonukleat_NNP ,_Z lebih_RB dikenal_VB dengan_IN singka
tan_NN DNA_FW ,_Z adalah_VB sejenis_NND biomolekul_NN yang_SC menyimpan_VB dan_CC
 menyandi_VB instruksi-instruksi_NN genetika_NN setiap_CD organisme_NN dan_CC ban
yak_CD jenis_NND virus_NN ._Z <end>_X
<start>_X Instruksi-instruksi_NN genetika_NN ini_PR berperan_VB penting_JJ dalam_
IN pertumbuhan_NN ,_Z perkembangan_NN ,_Z dan_CC fungsi_NN organisme_NN dan_CC vi
rus_NN ._Z <end>_X
<start>_X DNA_FW merupakan_VB asam_NN nukleat_NN ;_Z bersamaan_VB dengan_IN prote
in_NN dan_CC karbohidrat_NN ,_Z asam_NNP nukleat_NNP adalah_VB makromolekul_NN es
ensial_NN bagi_IN seluruh_CD makhluk_NN hidup_NN yang_SC diketahui_VB ._Z <end>_X
```

Gambar 3.4: Korpus kalimat Wikipedia *postag*

Korpus kalimat yang dihasilkan kemudian digunakan untuk proses *part-of-speech tagging*. Hasil dari *pre-processing* data Wikipedia adalah dua korpus besar yaitu korpus kalimat tanpa *postag* (Gambar 3.3) yang digunakan sebagai masukan *pattern extraction* dan korpus kalimat dengan *postag* (Gambar 3.3) yang digunakan sebagai masukan *pattern matching*.

3.2.2 Pengumpulan Seed

Pasangan kata relasi hipernim-hiponim diambil dari data yang dimilki oleh WordNet Bahasa yang dikembangkan oleh NTU. Pemanfaatan WordNet Bahasa dilatarbelakangi jumlahnya yang lebih besar dibanding Indonesian WordNet (IWN). Relasi semantik antar kata pada WordNet Bahasa memanfaatkan WordNet Princeton versi 3.0. *Synset* pada WordNet Bahasa dipetakan ke *synset* WordNet Princeton, sehingga relasi semantik yang dimiliki oleh WordNet Princeton ikut diwarisi. Alasan lain penggunaan WordNet Bahasa adalah karena telah terintegrasi dengan *tools* nltk sehingga dapat langsung digunakan untuk membentuk *seed* secara mudah.

Seluruh lema Bahasa Indonesia yang ada di dalam korpus WordNet Bahasa diambil. Untuk setiap lema, akan dicari hipernimnya sehingga dapat dibentuk menjadi pasangan kata relasi hipernim-hiponim. Sayangnya, tidak semua pasangan kata yang dihasilkan adalah benar akibat beberapa kekurangan yang dimiliki WordNet Bahasa. Untuk itu dilakukan proses filterisasi untuk meminimalisir *error* yang mungkin terjadi. Seluruh pasangan kata yang lolos proses filterisasi dibentuk menjadi *tuple* biner. Hasil akhir dari proses ini adalah himpunan *tuple* yang berisi dua elemen dengan format $(k_1;k_2)$ dimana k_1 pertama merupakan kata hiponim dan k_2 merupakan kata hipernimnya.

3.3 Pembentukan Pattern

Pattern leksikal yang akan digunakan ingin seluruhnya dibentuk secara otomatis oleh sistem. Terdapat dua masukan utama untuk proses ini yaitu pasangan kata re-

lasi hipernim-hiponim yang telah diketahui dan korpus dokumen. Pada tahap awal, pasangan kata relasi masukan adalah *seed* yang dibentuk menggunakan WordNet Bahasa. Untuk tahap selanjutnya, pasangan kata relasi menggunakan korpus *pair* hasil proses ekstraksi. Terdapat dua tahapan utama dalam pembentukan *pattern* yaitu *sentence tagging* dan *pattern extraction*.

3.3.1 Sentence Tagging

Sentence tagging adalah proses *intermediate* sebelum sistem dapat membentuk sebuah *pattern* leksikal. Proses ini akan memberi *tag* hipernim dan hiponim terhadap suatu kata di dalam kalimat. Masukan untuk proses ini adalah pasangan kata relasi dan korpus Wikipedia tanpa *pos tag*. Untuk setiap kalimat dalam korpus Wikipedia, dicek apakah kalimat tersebut mengandung pasangan kata relasi. Jika mengandung, maka kata dalam kalimat yang merupakan pasangan kata akan di-*tag* hipernim dan hiponim. Satu kalimat dicocokan dengan seluruh pasangan kata relasi karena terdapat kasus dimana satu kalimat mengandung lebih dari satu pasangan kata relasi. Proses ini menghasilkan kalimat-kalimat Wikipedia yang telah diberi *tag* hipernim atau hiponim.

Sebagai contoh, terdapat pasangan relasi kata '(sepak bola;olahraga)' serta kalimat '<start> sepak bola adalah cabang olahraga yang menggunakan bola yang umumnya terbuat dari bahan kulit dan dimainkan oleh dua tim . <end>'. Korpus kalimat dengan tag hipernim-hiponim yang dihasilkan adalah '<start> <hyponym>sepak bola<hyponym> adalah cabang <hypernym>olahraga<hypernym> yang menggunakan bola yang umumnya terbuat dari bahan kulit dan dimainkan oleh dua tim . <end>'. Dengan adanya penanda kata mana yang merupakan hipernim dan hiponim, proses ekstraksi *pattern* akan lebih mudah dilaksanakan.

3.3.2 Pattern Extraction

Setelah didapatkan kumpulan kalimat yang sudah diberi *tag* hipernim dan hiponim, dicari barisan kata yang mirip dan dapat dijadikan sebuah *pattern*. *Pattern extraction* adalah proses untuk mendapatkan *pattern* leksikal yang kemunculannya sering dalam korpus. Masukan dari proses ini adalah kalimat-kalimat Wikipedia yang telah diberi *tag* hipernim dan hiponim. Tidak seluruh kata dalam kalimat digunakan untuk membentuk *pattern* leksikal. Bagian dalam kalimat yang diperhatikan adalah kata diantara dua *tag* hipernim-hiponim, *n* kata sebelum *tag* pertama, dan *n* kata setelah *tag* terakhir. Bagian kata yang diambil akan dimasukkan ke dalam *pattern tree* untuk mengetahui kemunculannya yang paling sering. Seluruh *pattern* direp-

resentasikan menjadi vektor dan dirutkan berdasarkan bobot tertingginya. Proses ini akan menghasilkan kumpulan *pattern* unik yang terurut berdasarkan bobotnya. Dari *pattern* unik tersebut, akan diambil sejumlah *pattern* terbaik untuk kemudian digunakan sebagai dasar pembentukan *pair* baru.

Sebagai contoh, proses *sentence tagging* menghasilkan tiga kalimat yang telah diberi *tag* hipernim-hiponim. Kalimat berikut digunakan untuk mencari *pattern* yang merepresentasikan relasi hipernim-hiponim.

- <start> <hyponym>kelinci<hyponym> adalah <hypernym>binatang<hypernym> yang gemar memakan wortel . <end>
- <start> <hyponym>mobil<hyponym> adalah <hypernym>kendaaraan<hypernym> beroda empat . <end>
- <start> selain sepak bola, <hyponym>basket<hyponym> adalah <hypernym>olahraga<hypernym> yang disukai masyarakat . <end>

Dari kalimat di atas *pattern* unik yang dihasilkan yaitu '<hyponym> adalah <hypernym>'. Lebih detil tahapan pembentukan *pattern* dapat dilihat pada bab 4.4.

3.4 Ekstraksi Pair

Tujuan utama dari penelitian ini adalah membentuk korpus pasangan kata relasi, sehingga proses ekstraksi *pair* adalah tahapan utama dari keseluruhan penelitian. Pada proses ini, dimanfaatkan *pattern* yang telah terbentuk sebelumnya untuk mengekstrak pasangan kata baru. Pasangan kata tersebut kemudian difilter sebelum digabung ke dalam korpus pasangan kata relasi. Ektraksi *pair* menggunakan teknik *pattern matching*.

Pattern matching adalah proses mencocokan suatu *pattern* ke dalam teks dokumen. Proses ini dilaksanakan untuk mendapatkan pasangan kata relasi (*pair*) baru. Dua masukan utama untuk proses ini adalah pattern dan korpus Wikipedia *pos tag*. Penggunaan korpus Wikipedia dengan *pos tag* untuk membatasi *pair* yang terekestrak hanya berasal dari kelas kata *noun* atau *proper noun*. Namun, tidak seluruh *pair* yang dihasilkan benar memiliki relasi hipernim-hiponim. Untuk itu, dihitung nilai bobot untuk seluruh *pair* yang dihasilkan. Nilai bobot digunakan untuk dibandingkan dengan nilai threshold yang didefinisikan. Hanya *pair* yang bobotnya melebihi nilai *threshold* yang dapat masuk ke korpus pasangan kata relasi.

Sebagai contoh, terdapat *pattern* leksikal '<hyponym> adalah <hypernym> yang'. Berikut adalah beberapa kalimat Wikipedia dengan *postag*.

- <start>_X sepak_NNP bola_NNP adalah_VB olahraga_NN paling_RB populer_JJ di_IN Britania_NNP ._Z <end>_X
- <start>_X Berikut_VB adalah_VB daftar_NN penghargaan_NN yang_SC diberikan_VB ._Z <end>_X
- <start>_X Gado-gado_NNP adalah_VB makanan_NN yang_SC berasal_VB dari_IN Betawi_NN ._Z <end>_X

Pair yang dihasilkan setelah melalui tahap *pattern matching* hanya '(gado-gado;makanan)'. Pasangan kata '(berikut;daftar penghargaan)' tidak termasuk karena kata 'berikut' bukan merupakan kata benda (*noun*). Sementara walau '(sepak bola;olahraga)' adalah pasangan kata hipernim-hiponim yang benar, kalimat tersebut tidak cocok dengan *pattern* leksikal yang diberikan menyebabkan *pair* tidak terekstrak.

3.5 Cycle Semi-Supervised

Pembelajaran menggunakan pendekatan *semi supervised learning* dilatarbelakangi ketersediaan pasangan kata relasi semantik yang telah diketahui (*seed*) berukuran terbatas dan korpus berukuran besar yang belum dianotasi. Ingin didapatkan *pair* baru dari korpus yang belum diantoasi tersebut. Metode *semi supervised* yang diterapkan adalah *Bootstrapping*. Untuk lebih spesifiknya, algoritma *bootstrapping* yang digunakan gabungan antara Meta-Bootstrapping dan Basilisk dengan beberapa modifikasi. Pemilihan metode tersebut didasari proses ekstraksi *pair* baru yang memanfaatkan *pattern*. Secara umum, proses *bootstrapping* dikelompokan ke dalam dua tahap yaitu pra-iterasi dan iterasi *semi supervised*.

Pra-iterasi atay dapat disebut sebagai iterasi ke-0 adalah proses pembentukan *pattern* pertama dengan memanfaatkan *seed* yang berasal dari WordNet Bahasa. Tahap *pre-processing* menghasilkan kumpulan *seed* dan korpus kalimat Wikipedia. *Seed* digunakan untuk *tagging* kalimat Wikipedia sehingga menghasilkan kalimat yang telah di-*tag* hipernim-hiponim. Kalimat-kalimat tersebut digunakan untuk membentuk *pattern* unik yang digunakan untuk memulai iterasi *semi supervised*. *Pattern* yang dibentuk dari *seed* lema sama maupun *strict* digabung dan diurutkan berdasarkan bobot. Lima *pattern* terbaik diambil untuk selanjutnya digunakan dalam proses ekstraksi *pair*. Sementara *seed* yang membentuk kelima *pattern* tersebut langsung dimasukkan ke dalam korpus pasangan relasi kata. Hal tersebut guna memfilter *seed* salah yang terbentuk akibat *error* dari *resource* WordNet Bahasa.

Iteriasi pertama dimulai dengan mengekstraksi *pair* menggunakan kelima *pattern* terbaik. *Pair* yang dihasilkan dapat direpresentasikan ke dalam bentuk vektor. Nilai-nilai dalam vektor *pair* digunakan untuk menghitung bobot suatu *pair*. Seluruh *pair* yang nilai bobotnya melebihi *threshold* bergabung dengan *pair* dalam korpus pasangan kata relasi. Iterasi kedua hingga selanjutnya dimulai dengan *sentence tagging* menggunakan korpus pasangan kata relasi sebagai *seed*, kemudian dilanjutkan dengan proses *pattern extraction*. *Pattern* yang dihasilkan akan digabung dengan seluruh *pattern* lama kemudian diurutkan. Seperti pada metode *Basilisk*, jumlah *pattern* yang digunakan pada iterasi berikutnya akan bertambah. Satu *pattern* terbaik dari hasil pengurutan bergabung dengan *pattern* terpilih lama untuk digunakan dalam proses *pattern matching*. Hal ini membuka kemungkinan adanya *pair* baru terekst rak. Sama seperti proses sebelumnya, *pair* yang bobotnya melebihi nilai *threshold* digabung ke dalam pasangan kata relasi.

Iterasi dilakukan hingga korpus pasangan kata relasi jenuh atau dapat dikatakan *pair* baru yang masuk ke dalam korpus berjumlah sedikit. Pada penelitian ini, jika *pair* baru berjumlah kurang dari lima puluh maka iterasi akan berhenti. Evaluasi *pattern* dan *pair* dilakukan secara kolektif diakhir eksperimen. Anotator melakukan evaluasi secara manual untuk mengetahui kualitas *pattern* dan *pair* yang dihasilkan.

3.6 Metode Evaluasi

Penelitian ini tidak hanya menghasilkan korpus pasangan kata relasi hipernim-hiponim, namun juga *pattern* leksikal Bahasa Indonesia yang merepresentasikan relasi tersebut. Setelah suatu eksperimen selesai, evaluasi dilakukan terhadap *pattern* dan *pair* yang dihasilkan. Proses evaluasi dilakukan dengan bantuan anotator.

3.6.1 Evaluasi Pattern

Evaluasi *pattern* dilakukan dengan bantuan dua anotator dan melalui beberapa tahap. Berikut adalah proses yang dilakukan untuk evaluasi *pattern*.

1. Anotator membuat *pattern* secara manual yang diyakini dapat mengekstrak kata-kata relasi semantik sesuai dengan format *pattern* yang didefinisikan.

2. Anotator melakukan penilaian terhadap *pattern* yang dihasilkan oleh sistem. Suatu *pattern* dinilai berdasarkan jumlah *pair* benar maupun salah yang mungkin terekstrak dengan nilai antara 1 (sedikit), 2 (sedang), dan 3 (banyak).

3. Anontator melakukan *ranking* dari *pattern* hasil ekstraksi.

4. Nilai *precision* dan *recall* diperoleh dengan membandingkan *pattern* yang dibentuk oleh anotator dan *pattern* yang dihasilkan sistem.

5. Nilai Spearman's Rho diperoleh dengan membandingkan *ranking pattern* yang dilakukan anotator dengan *ranking pattern* yang dihasilkan sistem.

Pattern manual dibandikan dengan *pattern* buatan sistem dengan melihat seberapa cocok keduanya. Suatu *pattern* dapat dikategorikan ke dalam tiga kelompok yaitu, *exact match*, *partial match*, atau *no match*. *Pattern* dikatakan *exact match* jika seluruh token dan urutannya dalam satu *pattern* adalah tepat sama dengan *pattern* yang lain. *Pattern* dikatakan *partial match* jika token dalam satu *pattern* adalah subbagian dari keseluruh token untuk *pattern* lainnya, dimana urutan kemunculan diperhatikan. Sebagai contoh, <*hypernym*> *adalah* <*hyponym*> dan <*hypernym*> *adalah sebuah* <*hyponym*> dapat dikatakan *partial match*. *Pattern* dikatakan *no match* jika tidak memenuhi kriteria *exact match* maupun *partial match*.

Pattern hasil sistem diberi nilai anotasi berdasarkan dua dimensi, yaitu banyaknya *pair* benar yang dihasilkan dan banyaknya *pair* salah yang dihasilkan. Anotator memberi nilai anotasi dengan skala 1 hingga 3, dimana nilai 1 berarti *pair* berjumlah sedikit, nilai 2 berarti *pair* berjumlah sedang, dan nilai 3 berarti *pair* berjumlah banyak. Dari nilai anotasi, dibuat *confussion-matrix* untuk diketahui banyaknya *pattern* yang berkualitas tinggi. Sementara *ranking* yang digunakan anotator dibandingkan dengan *ranking* yang dibuat sistem dan hasilnya dibandingkan.

3.6.2 Evaluasi Pair

Evaluasi *pair* dilakukan menggunakan teknik *random sampling*. Sejumlah *pair* yang dihasilkan diambil secara acak dan dianotasi dengan nilai 'benar' atau 'salah' oleh anotator. Berikut adalah proses dalam evaluasi *pair* hasil ekstraksi:

1. Terdapat tiga anotator berbeda yang menganotasi data yang sama.

2. Anotator memberi nilai benar atau salah terhadap suatu *pair* serta kategori yang didefinisikan. Untuk *pair* benar dapat termasuk kategori *pair* adalah *instance-class* atau *pair* adalah *class-class*. Untuk *pair* salah dapat termasuk kategori *pair* tanpa relasi, *pair* dengan relasi semantik lain, atau *pair* yang posisi hypernym-hyponym-nya terbalik.

3. Nilai Kappa dihitung untuk mengetahui tingkat persetujuan antar anotator. Perhitungan dilakukan menggunakan Fleiss' Kappa.

4. Hasil anotasi digunakan untuk menghitung akurasi *pair* yang dihasilkan sistem.

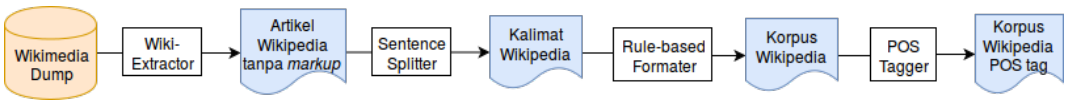
Dari setiap nilai anotasi, *pair* dimasukan ke dalam kategori yang lebih spesifik. Untuk data yang dianotasi benar, *pair* dapat dimasukan ke dalam dua kategori yaitu *instance-class* atau *class-class*. *pair* dimasukkan ke dalam kategori instanace-class jika kata hyponym merupakan suatu instance, sementara kategori class-class jika kata hyponym merupakan suatu class. Untuk data bernilai salah, dapat dimasukan ke kategori salah tanpa hubungan, salah akibat posisi hiponim-hipernim terbalik (*false posistion*), atau salah dengan relasi semantik lain (*false relation*). Kategori salah akibat posisi hiponim-hipernim terbalik terjadi saat kedua kata yang terekstrak benar memiliki relasi semantik hiponim-hipernim namun kata yang menempati posisi hiponim seharusnya adalah hipernim, begitu pula sebaliknya. Suatu *pair* dapat digolongkan ke dalam kategori salah dengan relasi semantik lain jika kedua kedua kata dalam *pair* tidak berhubungan hipernim-hiponim, namun memiliki relasi lain. Dalam penelitian ini, relasi lain hanya khusus dalam relasi semantik sinonim, antonim, atau meronym-holonym.

BAB 4
IMPLEMENTASI

Bab ini menjelaskan secara detail proses implementasi dari pengolahan data, proses *pattern extraction* dan *matching*, pembobotan dan *ranking* baik *pattern* maupun *pair*.

4.1 Pembentukan Korpus Kalimat Wikipedia

Untuk dapat mengubah data XML Wikipedia menjadi kalimat dengan format yang diinginkan perlu melalui beberapa tahap. Gambar 4.1 memperlihatkan secara detil setiap tahapan dalam pemrosesan data Wikipedia. Pertama, data akan dibersihkan dari format *markup language* menggunakan WikiExtractor. Kemudian setiap baris yang merepresentasika satu paragraf dipisahkan menjadi kalimat perbaris menggunakan program *sentence splitter*. Kalimat yang dihasilkan akan diformat menggunakan *rule-based formatter* sehingga sesuai dengan keinginan. Terakhir, POS Tagger akan memberikan *postag* untuk setiap token dalam kalimat.



Gambar 4.1: Pre-processing Data Wikipedia

Data XML yang diperoleh dari Wikipedia Dump diekstrak secara otomatis menggunakan WikiExtractor[1]. WikiExtractor adalah program berbasis Python yang dibuat oleh Giuseppe Attardi dan Antonio Fuschetto. *Tools* ini akan membersihkan artikel Wikipedia dari format MediaWiki *markup language* sehingga dihasilkan korpus yang hanya berisi konten artikel saja. Program ini dapat diunduh dari Github dan dijalankan pada sistem operasi berbasis UNIX/Linux menggunakan perintah berikut.

Kode 4.1: Penggunaan Wiki Extractor

```
$ WikiExtractor.py xml-dump-file -o output-file
```

Jika tidak memberi spesifikasi opsi apapun, artikel yang dihasilkan membersihkan seluruh *markup language* dan hanya menyimpan isi artikel tanpa disertakan informasi seperti kategori, riwayat, dan versi artikel.

[1] github.com/attardi/wikiextractor/wiki

4.1.1 Sentence Splitting

Korpus yang dihasilkan menghasilkan baris-baris yang merepresentasikan suatu paragraf dalam artikel Wikipedia. Pada penelitian kali ini, ingin dilihat relasi hipernim-hiponim antar dua kata pada kalimat yang sama. Untuk itu, perlu dilakukan proses *sentence splitting* yang dapat memisahkan setiap kalimat dalam paragraf. Hasil dari proses tersebut adalah dokumen yang terdiri dari baris-baris yang merepresentasikan satu kalimat.

Proses ini dilakukan dengan menggunakan program berbasis Perl yang telah dibuat sebelumnya oleh Ken Nabila Setya dari Fasilkom UI, Indonesia. Ditambah satu program yang dapat secara otomatis melakukan *splitting* untuk seluruh dokumen dalam direktori. Berikut adalah contoh sebuah paragraf dalam artikel Wikipedia yang telah dibersihkan menggunakan WikiExtractor.

Charles Anthony Johnson (3 Juni 1829 - 17 Mei 1917), kemudian dikenal sebagai Charles Brooke memerintah Sarawak sebagai Raja Putih kedua dari 3 Agustus 1868 hingga meninggal dunia. Dia menggantikan pamannya, James Brooke sebagai raja.

Kalimat di atas akan dimasukan ke program *sentence splitter*. Berikut adalah hasil yang diberikan dari proses tersebut.

Charles Anthony Johnson (3 Juni 1829 - 17 Mei 1917), kemudian dikenal sebagai Charles Brooke memerintah Sarawak sebagai Raja Putih kedua dari 3 Agustus 1868 hingga meninggal dunia.
Dia menggantikan pamannya, James Brooke sebagai raja.

4.1.2 Rule Based Formatter

Kalimat-kalimat yang dihasilkan dari proses sebelumnya, dimasukkan ke dalam program *rule based formatter* sehingga membentuk korpus dengan format yang diingingkan. Penambahan aturan juga untuk mengurangi ambiguitas dan bentuk usaha generalisasi *pattern*. Berikut adalah beberapa aturan tambahan untuk yang diberikan pada korpus Wikipedia.

1. Menghilangkan frasa yang berada di dalam tanda kurung.
   Frasa yang terletak di dalam tanda kurung dapat dianggap sebagai penjelas kata atau frasa sebelumnya. Proses ini merupakan salah satu upaya generalisasi *pattern*.

2. Memisahkan simbol-simbol yang berhimpit pada awal dan akhir kata.
   Beberapa token yang dipisahkan oleh spasi dalam kalimat merupakan kata yang berhimpit dengan tanda baca. Untuk mempermudah proses selanjutnya yaitu *sentence tagging*, dilakukan *pre-processing* tambahan yaitu memisahkan simbol-simbol *non-alphanumerical*.

3. Memberi penanda awal kalimat dengan '<start>" dan akhir kalimat dengan '<end>'.
   Pemberian simbol awal dan akhir kalimat memperjelas isi kalimat dan juga menunjang proses *pattern extraction* dan *pattern matching*.

Dari contoh kalimat di atas, setelah melalui proses *formatting* yang didefinisikan menghasilkan kalimat berikut.

<start> Charles Anthony Johnson , kemudian dikenal sebagai Charles Brooke memerintah Sarawak sebagai Raja Putih kedua dari 3 Agustus 1868 hingga meninggal dunia . <end>
<start> Dia menggantikan pamannya , James Brooke sebagai raja . <end>

4.1.3 POS Tagging Kalimat Wikipedia

Proses *part-of-speech tagging* dilakukan pada korpus Wikipedia yang telah berbentuk kalimat dengan format yang didefinisikan. Pada penelitian ini, kelas kata yang menjadi pengamatan adalah *noun* (NN) dan *proper noun* (NNP), sehingga proses *pos tagging* dilakukan guna mengidentifikasi kata-kata tersebut. Proses *pos tagging* dijalankna dengan program Stanford POS Tagger, dengan model yang digunakan merupakan hasil penelitian sebelumnya menggunakan Bahasa Indonesia. Setelah selesai melalui proses *tagging*, dilakukan penyesuaian agar format korpus lebih rapi dan terstruktur. Berikut adalah contoh kalimat yang sudah melalui tahap *pos tagging*. Korpus ini digunakan untuk proses *pattern matching*.

<start>_X Charles_NNP Anthony_NNP Johnson_NNP ,_Z kemudian_CC dikenal_VB sebagai_IN Charles_NNP Brooke_NNP memerintah_VB Sarawak_NNP sebagai_IN Raja_NNP Putih_NNP kedua_CD dari_IN 3_CD Agustus_NNP 1868_CD hingga_IN meninggal_VB dunia_NN ._Z <end>_X

4.2 Seed Builder

Proses pengumpulan *seed* dibantu dengan *resource* yang dimiliki WordNet Bahasa menggunakan *tools* nltk. Proses pengumpulan diawali dengan mengambil seluruh

lema Bahasa Indonesia yang dimiliki oleh korpus nltk. Setelah itu, ambil seluruh *synset* yang mengandung lemma tersebut. Dari setiap *synset*, ambil relasi hipernimnya. Dari setiap *synset* hipernim, ambil lema Bahasa Indonesianya. Dilakukan pula filterisasi *synset* ataupun lema untuk mengurangi ambiguitas. Untuk setiap *synset* maupun lema yang diambil pada setiap tahapan, hanya boleh berasal dari kelas kata kerja (*noun*). Setelah didapatkan, bentuk ke dalam pasangan *tuple* biner. Berikut adalah

Kode 4.2: Algoritme pembentukan *seed*

```
buildSeed:
  ls = get_all_indonesian_lemma()
  for l in ls:
    syns = get_all_synsets(l)
    for s in syns:
      hs = get_all_hypernyms(s)
      for h in hs:
        hls = get_lemmas(h)
        for hl in hls:
          filter(hl)
          print_seed(l, hl)
```

Filterisasi dilakukan dengan tujuan mengurangi ambiguitas, namun tetap berusaha mendapatkan *seed* sebanyak mungkin. Filterasisasi juga dilakukan untuk mendapatkan *seed* awal yang diyakini benar dan berkualitas. Salah satu bagian terpenting proses ini adalah memasangkan hanya lema yang merupakan *noun* ke lema yang juga adalah *noun*. Jika kemungkinan lema tersebut tergolong ke dalam kelas kata bukan *noun*, lema tidak diikutsertakan sebagai *seed* awal.

Tantangan dalam proses ini adalah banyak ditemukan kasus dimana satu lema dikandung oleh lebih dari satu *synset* atau satu *synset* memiliki lebih dari satu *synset hypernym*. Ambiguitas dalam kasus tersebut dapat mengurangi kualitas *seed* yang dihasilkan. Mengatahui hal tersebut, dibuatlah dua pendekatan berbeda untuk proses filterisasi *seed*.

Pendekatan pertama adalah tetap mengambil lemma yang sama pada *synset hypernym* yang berbeda. Hal ini dilatarbelakangi adanya lema yang berasal dari *synset* berbeda namun memiliki lema *hypernym* yang sama. Pada contoh (i), satu lema yang sama dimiliki oleh dua *synset* yang berbeda namun kedua *synset* tersebut memiliki *synset* hypernym yang sama. Lema untuk kedua *synset* hypernym juga sama sehingga tetap diikutsertakan sebagai *seed*. Pada contoh (ii), satu lema berasal dari dua *synset* yang berbeda dan dua *synset hypernym* berbeda, namun ada lema yang sama yaitu 'laluan'. Sehingga pasangan '(pintu_masuk,laluan)' tetap

diikutsertakan sebagai *seed*.

Pendekatan lainnya adalah dengan metode filterisasi yang *strict*. Jika satu lema memiliki lebih dari satu *synset hypernym*, maka lema tersebut dianggap ambigu dan langsung tidak diikutsertakan ke dalam *seed* awal. Berdasarkan tabel contoh lema, *synset*, dan hipernimnya, hanya contoh (i) yang diterima sebagai *seed* karena *synset hypernym* untuk lema tersebut sama. Sementara (ii) ditolak karena *synset hypernym* berbeda.

i.	(paruh, Synset(beak.n.02)) => ([bibir, kuala, muara], Synset(mouth.n.02)) (paruh, Synset(beak.n.01)) => ([bibir, kuala, muara], Synset(mouth.n.02))
ii.	(pintu_masuk, Synset(entrance.n.01)) => ([akses, capaian, laluan], Synset(access.n.03)) (pintu_masuk, Synset(orifice.n.01)) => ([koridor, laluan, lorong], Synset(passage.n.07))

Format penulisan $(l_1, S_1) => (ls_2, S_2)$ berarti l_1 adalah lema hiponim, S_1 adalah *synset* hiponim, ls_2 adalah himpunan lema hipernim, dan S_2 adalah *synset* hipernim.

4.3 Sentence Tagging

Pada Gambar 4.2 diperlihatkan bahwa tahap awal pembentukan *pattern* adalah melakukan *tagging* pasangan kata relasi ke dalam kalimat-kalimat Wikipedia. Data yang digunakan untuk proses ini adalah korpus Wikipedia tanpa *postag*. Beberapa tahapan dilakukan pada proses *tagging sentence* dengan pasangan kata relasi adalah sebagai berikut.

1. Dibaca seluruh pasangan kata relasi hipernim-hiponim.
2. Untuk setiap kalimat pada korpus Wikipedia, di cek apakah kalimat tersebut mengandung kedua kata dalam pasangan kata relasi.
3. Pengecekan dilakukan secara berulang untuk seluruh pasangan kata relasi karena terdapat kemungkinan satu kalimat mengandung lebih dari satu pasang kata relasi.
4. Kata-kata yang merupakan bagian dari pasangan kata relasi kemudian diberi *tag* sesuai relasinya dan disimpan ke dalam korpus berisi kalimat yang sudah memiliki *tag* hiponim dan hipernim.

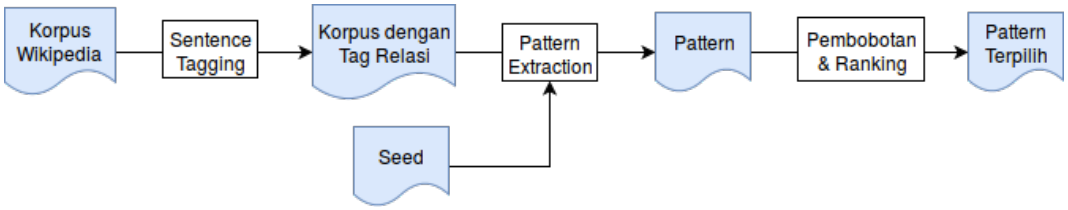
Pada penelitian ini, satu kalimat yang telah di-*tag* hanya mengandung tepat satu pasangan kata relasi. Untuk kasus khusus dimana suatu pasangan kata relasi terdiri dari suatu kata yang merupakan sub kata pasangannya, maka pasangan kata tersebut tidak diikutsertakan untuk menghindari ambiguitas. Contoh pasangan kata yang tidak diikutsertakan adalah $(ikan\ gurame; ikan)$ dimana kata 'ikan' terkandung dalam kedua kata relasi.

Berikut adalah contoh kalimat yang terbentuk dari proses *sentence tagging*. Diberikan pasangan kata relasi hipernim-hiponim $(fermion; partikel)$ dan $(boson; partikel)$ serta kalimat '<start> seluruh partikel dasar adalah boson atau fermion . <end>'. Hasil proses *sentence tagging* adalah sebagai berikut.

- <start> seluruh <hypernym>partikel<hypernym> dasar adalah boson atau <hyponym>fermion<hyponym> . <end>
- <start> seluruh <hypernym>partikel<hypernym> dasar adalah <hyponym>boson<hyponym> atau fermion . <end>

4.4 Pattern Extraction

Setelah mendapatkan kalimat-kalimat yang telah di-*tag* dengan kata relasi, ingin dicari *pattern* yang dapat digunakan untuk menambah jumlah relasi kata. Gambar 4.2 menunjukan bahwa kalimat-kalimat tersebut selanjutnya masuk ke dalam proses *pattern extraction*. Pada penelitian ini, diusulkan pembuatan *pattern* menggunakan *standard trie* dengan beberapa modifikasi. Proses ini diimplementasi secara mandiri menggunakan program Java dengan mengikuti algoritma pembuatan *Trie* sederhana.



Gambar 4.2: Proses Pembentukan *Pattern*

Suatu *node* merepresentasikan kata dalam kalimat dan dari satu kalimat terbentuk sebuah cabang dalam *tree*. *Node* menyimpan beberapa informasi seperti nama *node*, *parent*, *childs*, dan informasi identitas tambahan seperti apakah *node* tersebut merupakan relasi (*hypernym* atau *hyponym*) dan apakah *node* tersebut merupakan *leaf*. Untuk kata yang merupakan kata relasi, *node* menyimpan informasi jenis relasi beserta *list* dari kata yang merupakan bagian dari relasi tersebut.

4.4.1 Informasi dalam Pattern

Satu *pattern* tidak hanya menyimpan informasi *sequence* kata yang merepresentesaikan *pattern* tersebut, namun juga beberapa informasi tambahan lainnya. Informasi-informasi lain tersebut adalah jumlah kemunculan dalam korpus, *seed* unik yang membentuk *pattern*, dan kalimat unik yang membentuk *pattern*. Informasi-informasi tersebut digunakan untuk pembentukan vektor *pattern*, pemberian bobot *pattern*, dan melakukan *sorting* untuk mendapatkan *pattern* terbaik.

4.4.2 Pattern Tree

Pattern Tree adalah sebuah *tree* yang menyimpan seluruh *pattern* yang dihasilkan dari korpus. Dalam pembuatan *pattern tree*, tidak perlu menyimpan seluruh kata dalam kalimat. Hanya *sequence* kata tertentu saja yang dianggap dapat menghasilkan *pattern* yang baik untuk diikutsertakan. Maka dari itu, perlu diambil *sequence* kata dalam kalimat yang digunakan sebagai *pattern*. Terdapat tiga pendekatan dalam proses ini, diantaranya adalah sebagai berikut.

1. Hanya memperhatikan kata yang berada diantara pasangan kata relasi.
   Pada kasus ini, hanya ingin dilihat kata-kata yang berada diantara kata yang merupakan hipernim-hiponim atau hipernim-hiponim. Kata-kata diantara dua relasi dapat dianggap paling dekat jika ingin mencari *memisahkan* setiap relasi tersebut. Pada contoh diatas, *sequence* kata yang dihasilkan adalah '<hyponym>singa<hyponym> adalah <hypernym>kucing<hypernym>'

2. Mengikutsertakan *n* kata sebelum kata relasi pertama.
   Beberapa kata berbasis Perl sebelum program dapat memberikan informasi untuk yang dapat meningkatkan kualitas satu program. Pada contoh diatas dengan $(n = 1)$, *sequence* kata yang dihasilkan adalah '<start dalam direktori> <hyponym>singa<hyponym> adalah <hypernym>kucing<hypernym>'

3. Mengikutsertakan *n* kata setelah kata relasi terakhir.
   Tipe ini sama dengan sebelumnya, namun dilihat pengaruh kata-kata yang mengikuti kata relasi. Pada contoh diatas dengan $(n = 1)$, *sequence* kata yang dihasilkan adalah '<hyponym>singa<hyponym> adalah <hypernym>kucing<hypernym> yang'

Suatu kalimat akan di-*parse* ke dalam bentuk *array*. Proses *parsing* dilakukan berdasarkan spasi antar kata. Dari *array* yang terbentuk, dicari bagian-bagian yang akan dimasukan ke dalam *pattern tree* sesuai dengan pendekatan yang dipilih. Jika

hanya memperhatikan kata diantara pasangan kata relasi maka bagian-bagian lain tidak masuk ke dalam *pattern tree*. Begitu jika dipilih pendekatan dengan *n* kata sebelum atau setelah pasangan kata. Dari *pattern tree* yang terbentuk, dapat didaftarkan seluruh *pattern* serta bobot untuk *pattern* tersebut. Kode 4.3 menunjukan secara detil proses penambahan suatu *pattern* ke dalam *pattern tree*.

Kode 4.3: Penambahan *pattern* ke dalam *pattern tree*

```
sentence = #kalimat baru yang akan dimasukan
type = #pendekatan yang dipilih
ptree = #pattern tree yang sudah terbentuk

addNewPattern(sentence, type, ptree):
  sentence_arr = split(sentence, `` '')
  index = getIndexToken(sentence_arr, type)
  addPattern(ptree, sentence_arr, index)

getIndexToken(sentence_arr, type):
  sentence_arr = tokenize_sentence(sentence)
  index = (start, end)
  if (type == inbetween):
    for(i = 0..sentence_arr.size()):
      if (token == relation):
        if (!start) start = i
        else end = i
  else if (type == n before):
    index = getIndexToken(sentence_arr, inbetween)
    if (index.start - n >= 0)
      index.start -= n
    else index.start = 0
  else if (type == n after):
    index = getIndexToken(sentence_arr, inbetween)
    if (index.end + n < sentence_arr.size())
      index.end += n
    else index.end = sentence_arr.size()
  return index
```

4.4.3 Vektor Pattern

Suatu *pattern* dapat direpresentasikan menjadi vektor berdasarkan nilai-nilai yang dimilikinya. Vektor ini dimanfaatkan untuk melakukan pengurutan terhadap seluruh *pattern* yang dihasilkan. Selain itu, nilai-nilai pada vektor *pattern* juga digunakan untuk melakukan filterisasi. Fitur utama pada vektor *pattern* diambil dari informasi

yang disimpan *pattern*, yaitu total kemunculan *pattern*, jumlah *seed* unik, dan jumlah kalimat unik yang membentuk *pattern* tersebut. Fitur lainnya adalah hasil kombinasi perbandingan antar nilai-nilai utama, yaitu nilai perbandingan antara jumlah *seed* unik dibagi jumlah kalimat unik, nilai perbandingan antara jumlah *seed* untuk dibagi total kemunculan, dan nilai perbandingan antara jumlah kalimat unik dibagi total kemunculan.

4.4.4 Validasi Pattern

Setelah terbentuk *pattern tree*, perlu didaftar seluruh *pattern* yang dihasilkan. Proses pembentukan *pattern* cukup dengan menelusuri *path* dari *node leaf* hingga *root*. *Pattern* yang dihasilkan berjumlah banyak, sayangnya tidak seluruh *pattern* yang dihasilkan berkualitas baik. Untuk mengurangi jumlah *pattern* yang kurang baik, dilakukan proses filterisasi. Beberapa aturan yang harus dipenuhi agar suatu *pattern* diterima adalah sebagai berikut.

- Harus ada minimal satu kata diantara dua kata relasi.
  Banyak kasus dimana dua kata relasi hanya dipisahkan oleh spasi. *Pattern* yang hanya mengandung spasi tidak memberikan informasi apapun karena simbol spasi dalam Bahasa Indonesia digunakan sebagai pemisah antar kata. Sebagai contoh kalimat hasil tagging '<start> semua jenis <hypernym>ular<hypernym> <hyponym>beludak<hyponym> memiliki taring yang panjang <end>' tidak akan menghasilkan *pattern* yang *valid*.

- Sebuah *pattern* harus memenuhi nilai *threshold* yang didefinisikan.
  Nilai perbandingan antara jumlah *seed* unik dibagi jumlah kalimat unik lebih dari 0.5. Nilai perbandingan antara jumlah *seed* unik dibagi total kemunculan lebih dari 0.2. Nilai perbandingan antara jumlah kalimat unik dibagi total kemunculan harus lebih dari 0.7. Ketiga nilai *threshold* tersebut didefinisikan mandiri berdasarkan pengamatan dari nilai-nilai pada vektor *pattern*. Nilai tersebut mengeliminasi sejumlah *pattern* yang terbentuk dari banyak kalimat namun variasi *seed* yang menghasilkannya sedikit.

4.4.5 Pembentukan *Pattern* Unik

Pattern yang dihasilkan dari tahap ini harus unik sehingga tidak terjadi ambiguitas jika hendak digunakan. Pada masa awal pengembangan, masalah yang muncul pada *pattern* yang dihasilkan adalah adanya *pattern* yang posisi hipernim-hiponimnya saling berkebalikan. Sebagai contoh beberapa *pattern* ambigu yang dihasilkan

seperti (i) <hyponym> adalah <hypernym>, (ii) <hypernym> adalah <hyponym>, (iii) <hypernym> dan <hyponym>, dan (iv) <hyponym> dan <hypernym>. Kata-kata pada kalimat yang menempati posisi hipernim-hiponim tersebut nantinya digantikan dengan kata yang kemunculannya sesuai dengan *pattern* yang diberikan. Sehingga perlu ada strategi tambahan untuk mengatasi masalah tersebut.

Strategi yang diusulkan dalam penelitian ini untuk mengatasi masalah ambiguitas antar *pattern* adalah membangun membangun suatu arsitektur yang dapat mengidentifikasi dan menyelesaikan. Berikut adalah tahapan yang dilakukan untuk memastikan bahwa *pattern* yang dihasilkan adalah unik.

1. Dicari *pattern* dengan pendekatan hanya memperhatikan kata-kata diantara pasan gan ka a relasi.

2. *Pattern* yang dihasilkan dievaluasi satu dengan yang lain. Jika terdapat *pattern* yang saling terbalik, *pattern* yang bersangkutan dikeluarkan dari *list* dan disimpan untuk dievaluasi ulang.

3. Proses evaluasi ulang dilakukan menggunakan pendekatan pembuatan *pattern* lainnya yaitu dengan memperhatikan *n* kata sebelum dan setelah kata hipernim-hiponim. Hasil dari kedua pendekatan digabung dan dicek apakah *pattern* yang dihasilkan dibutuhkan. Suatu *pattern* dinyatakan dibutuhkan jika *substring* dari *pattern* tersebut tergolong dalam *list pattern* yang membutuhkan evaluasi ulang.

4. Proses evaluasi dilakukan secara berulang dari dengan *n* yang terus bertambah dari 1 (satu). Pada penelitian ini Kalimat di atas akan dimasukan ke program *sentence splitter*. Berikut adalah hasil yang diberikan dari proses tersebut. dua kata sebelum kata relasi pertama atau dua kata setelah kata relasi terakhir.

Setelah melakukan tahapan di atas, tidak ada lagi kasus posisi relasi saling tertukar dari *pattern* yang dihasilkan. *List pattern* yang dihasilkan kemudian diurutkan sebelum ditampilkan.

4.4.6 Pengurutan Pattern

Setelah terbentuk *pattern* yang sesuai, dilakukan proses pengurutan (*sorting*) untuk mengetahui *pattern* mana yang terbaik berdasarkan vektor *pattern* yang dimiliki. Proses pengurutan dilakukan dengan membandingkan satu *pattern* dengan yang lain, dengan tahapan sebagai berikut.

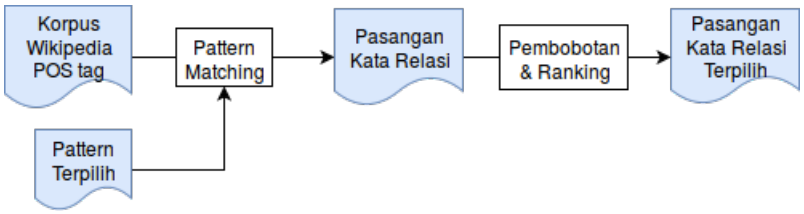
1. Semakin besar jumlah kalimat unik yang membentuk *pattern*.

2. Semakin besar nilai perbandingan antara jumlah *seed* unik yang membentuk *pattern* dengan jumlah kalimat unik yang membentuk *pattern*.

3. Semakin sedikit jumlah token dalam *pattern* jika di-*parse* menggunakan spasi.

4.5 Pattern Matching

Proses ekstraksi *pair* baru dilakukan dengan menggunakan metode *pattern matching* seperti yang dapat dilihat pada gambar 4.3. *Pattern* yang terbentuk dari proses *pattern extraction* digunakan untuk menambah jumlah pasangan kata relasi dengan dilakukan proses *pattern matching* terhadap korpus Wikipedia. Proses *pattern matching* dilakukan menggunakan algoritma *Suffix Tree* dengan modifikasi. Implementasi dilakukan secara mandiri menggunakan program berbasis Java.



Gambar 4.3: Proses ekstraksi *pair*

Mirip seperti pembentukan *suffix tree* mirip seperti pembentukan *pattern tree* yaitu melakukan *parsing* sehingga satu kalimat terepresentasi dalam bentuk *array*. Dari *array* yang dihasilkan, dibentuk sebuah *suffix tree* yang merepresentasikan kalimat tersebut. Sebuah *node* dalam *tree* merepresentasikan satu kata dalam kalimat. *Suffix tree* selanjutnya dicocokan dengan *pattern* yang diberikan. Jika terdapat *path* dalam *tree* yang sesuai dengan *pattern*, dibentuk suatu *pair* yang merepresentasikan pasangan kata relasi hipernim-hiponim yang dihasilkan. Selain kata hipernim-hiponim, *pair* juga menyimpan informasi seperti total memisahkan setiap total dokumen unik, daftar kalimat unik dan *pattern* unik yang menghasilkan *pair*.

Berikut adalah tahapan yang dilakukan program jika diberikan satu kalimat dan satu *pattern*.

1. Kalimat masukan dibentuk menjadi suatu *suffix tree*.

2. *Pattern* masukan ditokenisasi ke dalam bentuk *list* kata. *Pattern* pasti mengandung token <hypernym> dan <hyponym>, selanjutnya disebut token relasi.

3. Jika ditemukan token relasi pada *list pattern* yang sedang dievaluasi, maka *node* yang dikunjungi disimpan sementara sesuai dengan relasinya.

4. Jika token bukan token relasi, maka di evaluasi apakah token sama denga *node* yang dikunjungi. Jika sama maka proses evaluasi dilanjutkan, namun jika berbeda maka *pattern* tidak cocok.

5. Jika seluruh token *pattern* telah dievaluasi dan tidak mengalami kegagalan, makan dianggap berhasil dan kata yang terekstrak disimpan dalam bentuk *pair*.

Pada masa awal pengembangan, masalah pertama yang ditemukan adalah banyaknya *pair* yang salah satu atau kedua kata relasinya tidak teramasuk dalam kelas kata benda. Beberapa *pair* yang salah diantaranya $(Menoitios; salah)$ dimana 'salah' adalah *adjective* dan $(saya, gitaris)$ dimana 'saya' adalah preposi. Untuk itu diputuskan menggunakan korpus Wikipedia yang sudah melalui tahap POS Tagging. *Postag* pada setiap kata dalam kalimat membatasi hanya akan mengekstrak pasangan kata yang keduanya merupakan kata benda (*noun* atau *proper noun*).

Masalah lain yang muncul adalah jika kata yang ingin diekstrak merupakan *multi token*. *Pair* kurang baik yang dihasilkan sebelum mengatasi masalah ini diantaranya $(bola; olahraga)$ yang seharusnya $(sepak\ bola; olahraga)$, $(Serikat; negara)$ yang seharusnya $(Amerika\ Serikat; negara)$, dan $(Monterrey, ibu)$ yang seharusnya $(Monterrey; ibu\ kota)$. Solusi yang digunakan untuk mengatasi masalah ini adalah dengan mengasumsikan kata-kata berurutan yang memiliki kelas kata sama dalam suatu kalimat merupakan *multi token*. *Multi token* disimpan dalam satu *node* pada pembentukan *suffix tree*.

Untuk dapat mengidentifikasi *multi token*, penyesuaian dilakukan pada tahap *parsing* kalimat menjadi *array*. Satu token akan bergabung dengan token sebelumnya jika memiliki *postag* yang sama. Proses ini menghasilkan satu *array* yang merepresentasik kalimat dimana setiap elemen dalam *array* dapat merupakan *single token* maupun *multi token*. Kode 4.4 memaparkan proses pembentukan *array* yang digunakan untuk membangun *suffix tree*.

Kode 4.4: Proses *parsing* kalimat menjadi *array*

```
sentence = #kalimat yang akan dievaluasi

getSentenceArray(sentence):
  tmp_arr = split(sentence, `` '')
  sentence_arr = []
  prev = null
  i = 0
```

```
for s in sentence_arr:
  if (sentence.tag == prev.tag):
    sentence_arr[i-1] += `` '' + s
  else
    sentence_arr[i].add(s)
    i++
  prev = s
return sentence_arr
```

4.5.1 Vektor Pair

Suatu *pair* dapat direpresentasikan ke dalam bentuk vektor berdasarkan nilai-nilai yang dimilikinya. Nilai-nilai fitur yang dimiliki oleh sebuah *pair* adalah total kemunculan *pair*, total dokumen yang membentuk *pair*, jumlah *pattern* unik, dan jumlah kalimat unik. Untuk memperkaya fitur *pair*, dilakukan pula Word Embedding. Nilai *similarity* antar dua kata relasi ditambahkan sebagai sebagai salah satu fitur.

4.5.2 Filterisasi Pair

Pair baru yang dihasilkan untuk proses ini berjumlah sangat banyak, namun tidak semua *pair* yang dihasilkan memenuhi adalah benar pasangan kata relasi hipernim-hiponim. Beberapa *pair* kebetulan terekstrak akibat memenuhi *pattern* leksikal yang sama dengan salah satu *pattern* yang digunakan untuk proses *pattern matching*. Untuk mengeliminasi data yang tidak diyakini benar, dilakukan filterisasi sederhana untuk setiap *pair* yang dihasilkan. *Pair* dinyatakan benar jika terdapat lebih dari satu *pattern* yang mengekstrak *pair* tersebut.

Setelah mengeliminasi *pair* yang hanya terbentuk dari satu *pattern*, selanjutnya dilakukan pembobotan menggunakan rumus Tidak semua *pair* yang dihasilkan masuk ke dalam direktori dalam korpus kata relasi. Bobot satu *pair* dihitung menggunakan rumus 4.1. Jika nilai bobot melebihi *threshold*, maka *pair* dimasukan ke dalam korpus pasangan kata relasi.

$$Bobot = (\frac{jumlah\ pattern\ pembentuk\ pair}{jumlah\ pattern\ digunakan} + similarity\ score)/2 \tag{4.1}$$

4.6 Pemodelan Word Embedding

Untuk menambah fitur pada vektor *pair* yang dapat menunjang proses evaluasi, dilakukan Word Embedding. Implementasinya menggunakan model Word2Vec

berbasis Python. Proses ini dibuat secara otomatis dengan hanya mengmasukan dokumen Bahasa Indonesia berukuran besar. Dalam penelitianini, dokumen Bahasa Indoensia yang digunakan adalah korpus Wikipedia yang telah di proses.

Model dibuat menggunakan korpus Wikipedia yang telah melalui proses POS Tagging. Korpus tersebut diolah sedemikian sehingga kata-kata yang dianggap *multi word*, memiliki kelas kata sama berurutan, digabung dengan simbol garis awah ('_'). Hal ini dilatarbelakangi atas hasil *pair* yang banyak merupakan*multi word*. Jika hal ini tidak dilakukan, maka akan banyak kata yang tidak ditemukan dalam *dictionary* yang dihasilkan *model word embedding*. Parameter pembentukan model adalah $min_count = 1$, $size = 128$, dan $window = 5$.

Model yang terbentuk, digunakan untuk memberi nilai *similarity* antara kata hipernim-hiponim dalam satu *pair*. Proses pencarian nilai *similarity* dilakukan secara kolektif untuk setiap iterasi. Model yang terbentuk dibaca kemudian dicari *similarity* untuk setiap pasang kata relasi hipernim-hiponim. Hal ini diharapkan dapat memberi informasi lebih terhadap kualitas *pair* yang dihasilkan.

BAB 5
EVALUASI DAN HASIL

Bab ini menjelaskan mengenai hasil untuk setiap tahapan penelitian, deskripsi percobaan yang telah dilakukan, serta evaluasi dan hasil terkait percobaan tersebut.

5.1 Pengumpulan Data Wikipedia

Korpus teks dokumen utama dalam penelitian ini adalah artikel Wikipedia. Korpus diunduh dari situs Wikimedia. Berkas yang digunakan adalah idwiki-20170201-pages-article-multistream.xml.bz2, diunduh pada 20 Februari 2017. Berkas tersebut berukuran 398.6 MB dan merupakan data artikel Wikipedia Bahasa Indonesia hingga tanggal 1 Februari 2017. Setelah di-*extract*, ukuran asli berkas XML tersebut adalah 1.9 GB. Berkas tersebut mengandung seluruh *tag* identitas Wikipedia dan ditulis dalam format metadata WIkipedia.

5.2 Hasil Pengolahan Data Wikipedia

Sebelum digunakan sebagai korpus masukan untuk proses *pattern matching* dan *extraction*, data Wikipedia terlebih dahulu diproses.

5.2.1 Ekstraksi Teks

Proses ekstraksi teks dilakukan karena tidak seluruh bagian teks digunakan sebagai data penelitian. Data wikipedia yang ingin digunakan hanya isi artikel. Korpus Wikipedia diekstrak menggunakan WikiExtractor untuk menghilangkan *tag* yang tidak digunakan. Selain itu, artikel Wikipedia juga perlu dibersihkan dari simbol-simbol metadata. Setelah dilakukan proses ini, total ukuran berkasi Wikipedia menjadi 424 MB.

5.2.2 Pembentukan Kalimat

Data hasil ekstraksi memisahkan satu paragraf untuk setiap baris, sementara format yang diinginkan adalah satu baris merepresentasikan satu kalimat. Digunakan program *sentence splitter* untuk memisahkan kalimat dalam berkas Wikipedia. Kemudian, dilanjutkan ke dalam proses *rule-based formatter* sehingga memberi hasil

kalimat yang sudah sesuai format yang ditentukan. Proses tersebut menghasilkan 3.696.339 kalimat dengan total ukuran berkas 431 MB. Berkas ini digunakan sebagai masukan proses *pattern extraction.*

5.2.3 Hasil POS Tagging Kalimat

Kalimat-kalimat yang telah dibentuk, dimasukan ke dalam program POS Tagger, sehingga dihasilkan kalimat yang setiap tokennya memiliki *tag* berdasarkan kelas kata. Total ukuran berkas adalah 623 MB. Berkas ini digunakan sebagai masukan proses *pattern matching*.

5.2.4 Pemodelan Word Embedding

Pembuatan model *word embedding* menggunakan korpus Wikipedia yang sudah berbentuk kalimat. Proses ini menghasilkan tiga berkas model (tabel 5.1). Berkas hasil pemodelan berukuran besar karena menggunakan seluruh kalimat yang ada dalam korpus Wikipedia.

Tabel 5.1: Berkas model *word embedding*

Nama Berkas	Jenis Berkas	Ukuran Berkas
model-word2vec	File	306.1 MB
model-word2vec.syn0.npy	NPY File	1.8 GB
model-word2vec.syn1neg.npy	NPY File	1.8 GB

5.3 Pengumpulan Seed

Pasangan kata relasi *hypernym-hyponym* awal diambil dari korpus yang dimiliki oleh WordNet Bahasa. Setelah melalui proses pembentukan *seed*, berikut adalah jumlah *seed* yang dihasilkan berdasarkan jenis filterisasi. Kedua tipe *seed* tersebut digunakan proses pembentukan pattern pada iterasi pertama.

Tabel 5.2: Jumlah *seed* hasil ekstraksi

No	Jenis Filterisasi	Jumlah Seed
1.	lema sama	8.985
2.	*strict*	8.602

Walau sudah melakukan proses filterisasi sebagai upaya mengurangi *error* dan ambiguitas yang terjadi pada proses pemenentukan *seed* serta untuk meningkatkan

kulatias *seed*, masih ada hambatan yang belum dapat diatasi dalam penelitian ini. Beberapa kelemahan dari *seed* awal yang dihasilkan adalah sebagai berikut.

- *Seed* yang mengandung kata bukan Bahasa Indonesia. Korpus yang ingin dibuat berdomain Bahasa Indonesia, namun *seed* yang dihasilkan mengandung Bahasa Melayu atau Bahasa Indonesia lama. Beberapa kata bukan Bahasa Indonesia yang dihasilkan adalah 'had', 'bonjol', dan 'cecok'.

- Kesalahan semantik *synset* dan lema Bahasa Indonesia. Beberapa *synset* nltk memiliki lema Bahasa Indonesia yang kurang sesuai jika dilihat secara semantik. Sebagai contoh *Synset('scholar.n.01')* dengan lemma Bahasa Indonesia *'buku_harian', 'pelajar'*. Dalam Bahasa Indonesia, 'buku_harian' memiliki makna yang berbeda dengan 'pelajar'.

- Kesalahan lema Bahasa Indonesia untuk suatu *synset* menyebabkan dihasilkannya *seed* yang jika dievaluasi kualitatif oleh manusia dirasa kurang tepat. Contoh *seed* yang tidak baik adalah dari pemetaan *('sejarawan', Synset('historian.n.01')) => (['buku_harian', 'pelajar'], [Synset('scholar.n.01')])* dihasilkan *seed (sejarawan,buku harian)* dan *(sejarawan,pelajar). Seed* (sejarawan,buku harian) adalah salah.

5.4 Pembentukan Pattern Pertama

Iterasi pertama untuk seluruh percobaan selalu menggunakan *seed* diatas, sehingga hasil untuk proses pembentukan *pattern* sama. Berikut adalah detil hasil untuk *sentence tagging* dan *pattern extraction* pada iterasi pertama.

5.4.1 Sentence Tagging dengan Seed

Dari kedua tipe *seed*, masing-masing digunakan untuk membentuk korpus kalimat yang memiliki *tag* relasi *hypernym-hyponym*. Berikut adalah hasil proses sentence tagging menggunakan kedua tipe *seed*.

Tabel 5.3: Hasil *sentence tagging* dengan *seed*

No.	Jenis Seed	Jumlah kalimat di-tag	Ukuran berkas
1.	Seed dengan filterisasi lema sama	69.574	14 MB
2.	Seed dengan filterisasi strict	64.718	13 MB

5.4.2 Hasil Pattern Extraction

Kedua korpus kalimat yang masing-masing menghasilkan daftar *pattern*. Kedua daftar *pattern* digabung dan diurutkan sesuai bobot. Dari 106 *pattern* yang dihasilkan pada iterasi pertama, tabel 5.4 memaparkan lima *pattern* terbaik yang digunakan untuk proses *pattern matching*. Kelima *pattern* tersebut dibentuk dari total 104 *seed* yang langsung masuk ke korpus pasangan kata relasi.

Tabel 5.4: Pattern terbaik iterasi pertama

start <hyponym> adalah <hypernym>
<hyponym> merupakan <hypernym>
<hyponym> adalah <hypernym> yang
<hypernym> seperti <hyponym> dan
<hypernym> termasuk <hyponym>

Kelima *pattern* tersebut dibangun dengan total 140 *seed*. Dari 140 *seed* yang merepresentasikan kelima *pattern*, dicari nilai akurasinya. Diambil 20 *pair* secara acak sehingga satu sampel merepresentasikan 7 data asli, kemudian *pair* tersebut dianotasi secara manual oleh tiga anotator. Total sampel yang benar adalah 18 data, sehingga dapat dikatakan bahwa akurasi *pair* pertama yang masuk ke dalam korpus pasangan kata relasi adalah 0.9. Jika dilihat dari keseluruhan data, masih terdapat 14 *pair* yang salah namun masuk ke dalam pasangan kata relasi.

5.5 Hasil Eksperimen

Bagian ini akan memaparkan penjelasan parameter untuk setiap eksperimen yang telah dilakukan dan hasil dari eksperimen tersebut.

5.5.1 Eksperimen 1

Eksperimen pertama dilakukan dengan nilai *threshold* untuk filterisasi *pair* baru sebesar 0.6. Percobaan pertama menerima seluruh kata, baik yang merupakan *multi token* maupun *single token*. Tabel 5.5 menampilkan total *pattern* yang dihasilkan dan digunakan untuk setiap iterasi. Sementara tabel 5.6 menampilkan rincian jumlah *pair* yang dihasilkan dan diterima untuk setiap iterasi.

Tabel 5.5: Pattern Hasil Eksperimen 1

Iterasi ke-	Total pattern	Pattern untuk ekstraksi
1	106	Pattern utama
2	347	<start> <hyponym> merupakan <hypernym>
3	395	<hyponym> merupakan <hypernym> yang
4	425	<hypernym> atau <hyponym>

Tabel 5.6: Pair Hasil Eksperimen 1

Iterasi ke-	Total Pair	Pair baru	Korpus Pasangan kata relasi
1	82409	799	939
2	104389	316	1255
3	108397	175	1430
4	108542	2	1432

Pair yang dihasilkan dicari nilai akurasinya dengan mengambil sampel secara *random*. Sampel tersebut kemudian dianotasi secara manual oleh anotator. Tabel memaparkan nilai akurasi *pair* untuk setiap iterasi.

Tabel 5.7: Akurasi Eksperimen 1

Iterasi ke-	Sampel baru	Sampel benar	Akurasi sampel baru	Akurasi total
1	168	133	0.79	0.8
2	59	45	0.76	0.79
3	40	36	0.9	0.81
4	2	1	0.5	0.81

Secara total, korpus pasangan kata relasi untuk eksperimen pertama menghasilkan *pair* dengan akurasi kebenaran 0.81. Walau nilai akurasi cukup tinggi, namun jika dilihat berdasarkan jumlah data masih terdapat sekitar 277 *pair* salah yang masuk ke dalam korpus pasangan kata relasi. Kesalahan *pair* disebabkan oleh berbagai hal. Beberapa *pair* tidak memiliki hubungan sama sekali dimana hal tersebut bisa terjadi karena ada kalimat yang memuat pattern leksikal namun kata tidak merepresentasikan relasi hipernim-hiponim. Kesalahan lain ketika *pair* yang terbentuk saling berelasi namun bukan relasi hipernim-hiponim atau posisi kata yang tertukar.

Data di atas juga memperlihatkan bahwa jumlah *pair* yang diyakini benar atau nilai bobotnya melebihi *threshold* dapat dikatakan sedikit jika dibandingkan dengan

total *pair* yang terekstraksi. Perbandingan total *pair* yang masuk ke dalam korpus pasangan kata relasi dengan total *pair* yang dihasilkan oleh sistem hanya 0.013. Untuk meningkatkan jumlah *pair* yang masuk ke dalam korpus pasangan kata relasi, dilakukanlah eksperimen dimana nilai *threshold* diturunkan.

Selanjutnya, dari seluruh data sampel, *pair* yang kata hipernim dan hiponimnya merupakan *single token* ada sejumlah 73 *pair*, dengan 63 diantaranya adalah *pair* yang berlabel benar yang berarti akurasinya adalah 0.86. Kemudian dari data tersebut, 44 *pair* termasuk dalam kategori *class-class*. Hal tersebut memperlihatkan bahwa *pair* yang kedua katanya adalah *single token* memiliki kualitas yang jauh lebih baik dibanding *multi token*. Untuk membuktikan hal tersebut, dilakukan eksperimen yang hanya memperhatikan data yang merupakan *single token*.

5.5.2 Eksperimen 2

Melihat sedikitnya jumlah pattern yang dihasilkan dari eksperimen pertama, maka diputuskan untuk menurunkan sedikit nilai *threshold*. Nilai *threshold* untuk filterisasi *pair* baru menjadi sebesar 0.55. Tabel 5.8 dan 5.9 memaparkan hasil dari eksperimen kedua.

Tabel 5.8: Pattern Hasil Eksperimen 2

Iterasi ke-	Total pattern	Pattern baru terpilih
1	106	Pattern utama
2	699	<start> <hyponym> merupakan <hypernym>
3	791	<hyponym> merupakan <hypernym> yang
4	842	<hyponym> menjadi <hypernym>

Tabel 5.9: Pair Hasil Eksperimen 2

Iterasi ke-	Total Pair	Pair baru	Korpus Pasangan kata relasi
1	82409	2053	2193
2	104389	878	3071
3	108397	413	3484
4	108737	9	3493

Dari *pair* yang dihasilkan, tabel 5.10 menunjukan nilai akurasinya.

Tabel 5.10: Akurasi Eksperimen 2

Iterasi ke-	Sampel baru	Sampel benar	Akurasi sampel baru	Akurasi total
1	308	257	0.83	0.84
2	129	106	0.82	0.83
3	45	37	0.82	0.83
4	9	9	1	0.84

Jika dibandingkan dengan eksperiment pertama, jumlah *pair* yang nilainya melebihi *threshold* bertambah hingga dua kali dari sebelumnya walaupun selisih nilai *threshold* hanya 0.05. Ini berarti banyak *pair* yang nilai bobotnya dekat dengan nilai batas tersebut. Jika dilihat dari total *pair* yang diekstrak, hingga iterasi ke-3 jumlahnya sama karena *pattern* yang digunakan juga sama. Pada iterasi ke-4 dimana *pattern* eksperimen ke-1 dan ke-2 diekstrak total *pair* berbeda.

5.5.3 Eksperimen 3

Setelah menganalisis secara kualitatif *pair* yang dihasilkan, banyak *pair* yang kata *hypernym*-nya adalah *class* sementara kata *hyponym*-nya adalah *instance*. Untuk relasi semantik *hypernym-hyponym* yang lebih umum, hubungan antar kata yang lebih diinginkan adalah *class-class*. Untuk pasangan kata yang bertipe *class-class* lebih banyak dijumpai jika kedua kata merupakan *single word*. Selain it, pembentukan *multi word* yang dilakukan belum sepenuhnya baik karena masih banyak kata yang kelebihan token kata. Untuk itu, dicoba eksperimen yang hanya mengekstrak *pair* yang kedua katanya adalah *single word*. Untuk eksperimen ini, menggunakan *threshold* sebesar 0.55.

Tabel 5.11: Pattern Hasil Eksperimen 3

1	106	Pattern utama
2	437	<hyponym> adalah sebuah <hypernym>

Tabel 5.12: Pair Hasil Eksperimen 3

Iterasi ke-	Total Pair	Pair baru	Korpus Pasangan kata relasi
1	10267	298	438
2	11262	21	459

Hasil perhitungan akurasi untuk eksperimen ini dapat dilihat pada tabel 5.13.

Tabel 5.13: Akurasi Eksperimen 3

Iterasi ke-	Sampel baru	Sampel benar	Akurasi sampel baru	Akurasi total
1	78	67	0.86	0.87
2	5	4	0.8	0.86

Untuk *pair* yang hanya *single word* sudah pasti jumlahnya jauh lebih sedikit dibanding jika dibanding keseluruhan. Jika dibandingkan dengan dua eksperimen sebelumnya, total *pattern* yang dihasilkan juga jauh lebih sedikit. Dapat dikatakan bahwa jumlah pasangan kata yang digunakan mempengaruhi jumlah *pattern* yang dihasilkan, dimana semakin banyak jumlah pasangan kata relasi maka semakin banyak pula *pattern* yang dihasilkan.

5.5.4 Evaluasi Eksperimen

Dari ketiga eksperimen yang telah dilakukan, berikut adalah beberapa hal yang dapat dilihat.

- *Pair* yang dianggap baik tergolong sedikit jika dibandingkan dengan total *pair* yang terkestraksi.
- *Pattern* yang sama akan menghasilkan total *pair* yang terkestraksi sama. Pada eksperimen 1 dan 2, hingga iterasi 3, total *pair* yang dihasilkan adalah sama akibat ektraksi menggunakan *pattern* yang sama.
- Semakin banyak pasangan kata yang digunakan untuk membentuk *pattern*, maka semakin banyak pula *pattern* yang dihasilkan.
- Menurunkan sedikit nilai *threshold* dapat menambah banyak *pair* yang lolos filterisasi. Pada eksperimen 2, menurunkan *threshold* sebesar 0.05 meningkatkan jumlah *pair* hinga dua kali lipat. Hal tersebut juga menunjukan bahwa banyak *pair* yang nilai *threshold*-nya berada di ambang batas.
- Kenaikan atau penurunan akurasi setiap iterasi hanya tergantung dari data yang digunakan sebagai sampel.
- *Pair* yang hanya terdiri dari *single token* memiliki nilai akurasi yang relatif lebih tinggi dibanding *multi token*.
- *Stopping condition* dengan hanya menghitung jumlah *pair* baru tidak menjamin bahwa sudah tidak ada *pair* benar.

5.6 Hasil Evaluasi Pair

Anotasi *pair* dilakukan oleh tiga anotator berbeda dengan daftar *pair* yang sama. Nilai yang diberikan antar anotator tidak selalu sama, untuk itu ingin diketahui tingkat persetujuan antar anotator. Tingkat persetujuan ketiga anotator dihitung menggunakan Fleiss' Kappa karena anotasi dilakukan oleh lebih dari dua anotator.

5.6.1 Tingkat Persetujuan Anotator Pair

Perhitungan Fleiss' Kappa menggunakan data pada eksperimen pertama. Tingkat persetujuan diukur berdasarkan nilai anotasi data berlabel benar atau salah. Tiga anotator ($n = 3$) memberi penilaian terhadap 289 data sampel ($N = 289$) ke dalam dua label ($k = 2$) yaitu benar atau salah. Tabel 5.14 memaparkan langkah-langkah hingga didapatkan nilai kappa. Sementara, tabel 5.15 adalah perhitungan nilai kappa berdasarkan lima kategori ($k = 5$).

Tabel 5.14: Perhitungan tingkat persetujuan anotator *pair*

$sum\ P_i$	241.67
p_{True}	0.8
p_{False}	0.2
P_o	0.84
P_e	0.68
$Kappa$	0.49

Tabel 5.15: Perhitungan tingkat persetujuan anotator *pair*

$sum\ P_i$	189.33
p_{CC}	0.31
p_{IC}	0.49
p_{FF}	0.06
p_{FR}	0.12
p_{FP}	0.02
P_o	0.66
P_e	0.35
$Kappa$	0.47

Nilai kappa yang dihasilkan adalah 0.49 dan 0.47, hal ini menunjukan bahwa tingkat persetujuan antar anotator tergolong *moderate agreement*.

5.6.2 Analisis Pair

Dari hasil anotasi terebut, beberapa hal yang diketahui adalah sebagai berikut.

- Jika dibandingkan dengan pair yang berhasil di ekstrak dengan pattern yang diberikan, pair yang diyakini benar dapat dikatakan sedikit.
- Relasi semantik lain yang paling banyak dijumpai pada pair yang salah dengan kategori False Relation adalah relasi synonym.
- Pair dengan kategori benar instance-class banyak yang merupakan multi word, dengan kata hyponym merupakan Proper Noun.
- Beberapa pair masih dapat dianggap benar walau dihilangkan token katanya, seperti (ballantine's, merek wiski) lebih tepat ditulis sebagai (ballantine's,wiski)
- Beberapa pair merupakan Noun Phrase seperti (rikard nordraak,komposer norwegia) dimana kata 'komposer' saja sebenarnya sudah cukup. Tambahan kata 'norwegia' dapat menambah informasi relasi antar kata seperti Rikard Nordraak berasal dari Norwegia.
- Beberapa pair memiliki token berlebih yang tidak dibutuhkan, seperti 'pesepak bola asal' yang kelebihan kata 'asal' atau 'aslinya paralegal' yang kelebihan kata 'aslinya'. Hal ini terjadi akibat upaya pembentukan multi word.
- Beberapa pair merupakan kata fiksi seperti nama karakter suatu cerita.

5.7 Hasil Evaluasi Pattern

Evaluasi *pattern* dilakukan secara manual oleh dua anotator yang ahli dibidang linguistik. Seperti sudah dijelaskan sebelumnya, selain memberi penilaian kepada *pattern* yang dihasilkan oleh sistem, anotator juga membuat *pattern* manual untuk dibandingkan. Berikut adalah hasil *pattern* yang dibuat manual dan evaluasi *pattern* yang dibentuk sistem.

5.7.1 Pattern Buatan Manual

Lampiran 1 menampilkan daftar *pattern* leksikal yang dibentuk secara manual oleh anotator. Anotator mengamati kalimat-kalimat di dalam teks dokumen yang mengandung pasangan kata hipernim-hiponim kemudian membuat *pattern* yang dapat

merepresentasikan relasi. Terdapat total 67 *pattern* leksikal manual yang dibentuk oleh anotator. Tidak seperti *pattern* yang dihasilkan oleh sistem dimana satu *pattern* hanya dapat mengandung tepat satu pasang *tag* hipernim-hiponim, beberapa *pattern* manual mengandung lebih dari satu *tag* hiponim.

Selanjutnya, *pattern* hasil sistem dibandingkan dengan *pattern* manual tersebut. Satu *pattern* dapat dikategorikan sebagai *exact match*, *partial match*, atau *no match*. Tabel 5.16 memperlihatkan jumlah *pattern* yang masuk ke dalam setiap kategori untuk setiap iterasi.

Tabel 5.16: Kategori *pattern* sistem dibandingkan dengan *pattern* manual

Percobaan	Exact Match	Partial Match	No Match
1	10	301	114
2	10	367	465
3	8	132	297

Jika dibandingkan dengan total *pattern* yang dihasilkan sistem, jumlahnya memang jauh lebih sedikit. Tabel diatas memperlihatkan bahwa *pattern* yang tergolong *exact match* sebenarnya masih sangat sedikit. Pada eksperimen 2 dan 3, walau total *pattern* yang dihasilkan berbeda, jumlah *pattern* yang merupakan *exact match* adalah sama. Hal ini berarti semakin banyak *pattern* yang dihasilkan belum tentu semakin baik. Kemudian, dari keseluruhan *pattern*, yang tergolong ke dalam *no match* selalu mendominasi. Hal tersebut memperlihatkan bahwa *pattern* oleh sistem masih terlalu banyak yang hanya kebetulan terbentuk.

5.7.2 Hasil Anotasi Pattern

Dua anotator memberi penilaian terhadap 200 *pattern* terbaik yang dihasilkan pada percobaan ke-1. Anotasi hanya dilakukan pada percobaan ke-1 karena banyak variasi *pattern* yang dihasilkan antar percobaan mirip. Selain itu, keterbatasan waktu dan tenaga juga menjadi salah satu alasan. Tabel 5.17 menampilkan perbandingan hasil anotasi antar anotator jika dilihat dari dimensi jumlah *pair* yang dihasilkan benar, sementara tabel 5.18 memaparkan perbandingan jika dilihat dari dimensi jumlah *pair* yang dihasilkan salah.

Tabel 5.17: Perbandingan hasil anotasi berdasarkan jumlah *pair* benar

Jumlah Benar	anotator 1		
anotator 2	kategori 1	kategori 2	kategori 3
kategori 1	83	5	3
kategori 2	17	4	27
kategori 3	10	1	50

Tabel 5.18: Perbandingan hasil anotasi berdasarkan jumlah *pair* salah

Jumlah Salah	anotator 1		
anotator 2	kategori 1	kategori 2	kategori 3
kategori 1	57	1	2
kategori 2	33	1	8
kategori 3	74	2	22

5.7.3 Tingkat Persetujuan Anotator Pattern

Dari data pada tabel 5.17 dan 5.18, dapat dihitung tingkat persetujuan antar anotator. Perhitungan nilai persetujuan berdasarkan Cohen's Kappa karena jumlah anotator tepat dua. Tabel menampilkan nilai Kappa berdasarkan jumlah *pair* terbentuk benar maupun salah.

Tabel 5.19: Cohen's Kappa untuk setiap dimensi

	Jumlah benar	Jumlah Salah
p_o	0.685	0.4
$p_{kategori1}$	0.25	0.25
$p_{kategori2}$	0.012	0.0042
$p_{kategori3}$	0.12	0.078
p_e	0.38	0.33
$kappa$	0.49	0.11

Jika dilihat dari jumlah *pair* yang terbentuk benar, maka tingkat persetujuan antar anotator tergolong moderate agreemnet. Sementara jika dilihat dari jumlah *pair* yang terbentuk salah, tingkat persetujuan adalah *slight agreement*. Hal tersebut menunjukan bahwa sangat sulit bagi anotator untuk membayangkan apakah satu *pattern* dapat menghasilkan *pair* salah.

5.7.4 Analisis Pattern

Ekstraksi *pattern* dibatasi dengan beberapa aturan yang diharapkan dapat menghasilkan *pattern* dengan kualitasi baik. Sistem sudah dapat secara otomatis menghasilkan *pattern* yang tergolong baik untuk merepresentasikan relasi semantik *hypernym-hyponym* seperti <*hyponym*> *adalah* <*hypernym*>, <*hyponym*> *adalah* <*hypernym*> *yang*, <*hyponym*> *merupakan* <*hypernym*> dan lainnya. Namun, masih banyak kekurangan dari *pattern* yang dihasilkan. Berikut adalah beberapa kekurangan *pattern* yang terbentuk serta analisa mengapa hal tersebut dapat terjadi.

- Belum dapat membentuk *pattern* yang mengandung lebih dari satu *tag hyponym*.
  Banyak kalimat yang mengandung lebih dari satu kata *hyponym*, namun saat ini *pattern* yang terbentuk belum dapat membuatnya. Hal ini karena pada masa awal implementasi, ditemukan beberapa kalimat yang mengandung dua pasangan kata relasi berbeda dalam satu kalimat. Untuk mencegah ambiguitas dibatasi satu kalimat hanya akan di-*tag* dengan satu pasang kata relasi. Kedepannya, hal ini dapat diatasi dengan mengubah batasan sehingga dalam satu kalimat boleh di-*tag* dengan lebih dari satu pasang kata relasi asal kata *hypernym*-nya sama.

- Beberapa *pattern* yang kurang baik secara semantik.
  Beberapa *pattern* yang dihasilkan sistem seperti <*hypernym*> *dan* <*hyponym*> dan <*hypernym*> *atau* <*hyponym*> kurang cocok jika digunakan dalam *pattern matching*. Hal ini disebabkan kata 'dan' dan 'atau' memuat relasi yang bersifat simetris. Sementara *hypernym-hyponym* merupakan relasi yang hanya memiliki sifat transitif.

- *Pattern* yang tergolong dalam lebih dari satu relasi.
  Dalam beberapa penelitian lain, satu *pattern* bisa merepresentasikan beberapa relasi semantik. Terutama *pattern* yang berukuran pendek seperti yang dihasilkan sistem. Penelitian ini hanya fokus pada relasi semantik *hypernym-hyponym* sehingga tidak adanya pembanding antar *pattern* yang merepresentasikan relasi semantik lain. Hal ini menjadi hambatan dalam menentukan apakah *pattern* dapat dikatakan baik.

- *Pattern* yang tidak menghasilkan *pair* apapun.
  Penggunaan korpus yang sedikit berbeda antara proses pembentukan *pattern* dan proses ekstraksi *pair* serta upaya pembentukan *multiword* membuat sedikit kejanggalan dari hasil *pattern* yang terbentuk. Setiap *pattern*

yang terbentuk seharusnya dapat mengekstrak minimal *pair* yang membentuk *pattern* tersebut. Namun, beberapa *pattern* hasil sistem seperti <*hypernym*> *yunani* <*hyponym*> tidak mengekstrak *pair* setelah dijalankan dalam proses *pattern matching*. Hal ini disebabkan oleh upaya pembentukan *multi word* yang memanfaatkan korpus Wikipedia dengan *pos tag* membuat beberapa kata bergabung membentuk suatu *multi word*, *proper noun*, atau *noun phrase*. Sementara pada pembentukan *pattern*, batasan tersebut tidak diterapkan.

Cara membentuk *pattern* leksikal yang baik masih perlu diteliti lebih lanjut. Proses lain seperti generalisasi *pattern* perlu diimplementasi agar satu *pattern* mengandung informasi yang lebih kaya.

DAFTAR REFERENSI

Arnold, P. dan Rahm, E. (2014). Extracting semantic concept relations from wikipedia. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, page 26. ACM.

Bach, N. dan Badaskar, S. (2007). A review of relation extraction. *Literature review for Language and Statistics II*.

Bikel, D. M., Schwartz, R., dan Weischedel, R. M. (1999). An algorithm that learns what's in a name. *Machine learning*, 34(1):211–231.

Denoyer, L. dan Gallinari, P. (2006). The wikipedia xml corpus. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 12–19. Springer.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Jurafsky, D. dan James, H. (2000). Speech and language processing an introduction to natural language processing, computational linguistics, and speech.

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

Margaretha, E. dan Manurung, R. (2008). Comparing the value of latent semantic analysis on two english-to-indonesian lexical mapping tasks. In *Australasian Language Technology Association Workshop 2008*, volume 6, pages 88–96.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Noor, N. H. B. M., Sapuan, S., Bond, F., et al. (2011). Creating the open wordnet bahasa. In *PACLIC*, pages 255–264.

Prakash, V. J. dan Nithya, D. L. (2014). A survey on semi-supervised learning techniques. *arXiv preprint arXiv:1402.4645*.

Putra, D. D., Arfan, A., dan Manurung, R. (2008). Building an indonesian wordnet. In *Proceedings of the 2nd International MALINDO Workshop*, pages 12–13.

Riloff, E., Jones, R., et al. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479.

Riloff, E., Wiebe, J., dan Wilson, T. (2003). Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25–32. Association for Computational Linguistics.

Ruiz-Casado, M., Alfonseca, E., dan Castells, P. (2005). Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia. In *International Conference on Application of Natural Language to Information Systems*, pages 67–79. Springer.

Sumida, A. dan Torisawa, K. (2008). Hacking wikipedia for hyponymy relation acquisition. In *IJCNLP*, volume 8, pages 883–888. Citeseer.

Thelen, M. dan Riloff, E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 214–221. Association for Computational Linguistics.

Zesch, T., Gurevych, I., dan Mühlhäuser, M. (2007). Analyzing and accessing wikipedia as a lexical semantic resource.

LAMPIRAN

LAMPIRAN 1 : PATTERN BUATAN MANUAL

Berikut adalah daftar *pattern* yang dibentuk secara manual oleh anotator.

- <hypernym> adalah kumpulan dari <hyponym>
- <hypernym> antara lain adalah <hyponym>, <hyponym>, dan <hyponym>
- <hypernym> dapat dibedakan menjadi <hyponym>
- <hypernym> lainnya, seperti <hyponym> dan <hyponym>
- <hypernym> seperti <hyponym>
- <hypernym> terdiri dari <hyponym>, <hyponym>, dan <hyponym>
- <hypernym> terdiri dari beberapa bagian, seperti <hyponym>, <hyponym>, dan <hyponym>
- <hypernym> terutama <hyponym> yang
- <hypernym>, khususnya <hyponym>, adalah
- <hypernym>, misalnya <hyponym>
- <hypernym>, misalnya <hyponym> dan <hyponym>
- <hypernym>, terutama <hyponym>, adalah
- <hyponym> adalah <hypernym>
- <hyponym> adalah <hypernym> dari
- <hyponym> adalah <hypernym> dengan
- <hyponym> adalah <hypernym> yang berhubungan dekat dengan <hyponym>
- <hyponym> adalah <hypernym> yang bersifat
- <hyponym> adalah bagian dari <hypernym> yang
- <hyponym> adalah salah satu <hypernym>

- <hyponym> adalah sebuah <hypernym> yang
- <hyponym> adalah sejenis <hypernym> dengan
- <hyponym> adalah suatu <hypernym>
- <hyponym> atau <hyponym> adalah suatu jenis <hypernym> yang
- <hyponym> berarti <hypernym>
- <hyponym> dan <hyponym> dianggap sebagai <hypernym>
- <hyponym> dan <hyponym> merupakan <hypernym> yang
- <hyponym> dan berbagai <hyponym> lainnya adalah <hypernym> yang
- <hyponym> dan sejumlah <hyponym> lainnya termasuk ke dalam kategori <hypernym>
- <hyponym> dapat digolongkan ke dalam <hypernym>
- <hyponym> dapat dimasukkan ke dalam kategori <hypernym> dan <hypernym>
- <hyponym> dianggap sebagai <hypernym> karena
- <hyponym> dikenal juga sebagai <hypernym>
- <hyponym> disebut sebagai <hypernym> yang
- <hyponym> ialah <hypernym>
- <hyponym> menjadi <hypernym> apabila
- <hyponym> menjadi <hypernym> yang
- <hyponym> menjadi salah satu bagian dari <hypernym> karena
- <hyponym> merujuk pada <hypernym>
- <hyponym> merujuk pada <hypernym> yang
- <hyponym> merupakan <hypernym>
- <hyponym> merupakan <hypernym> yang
- <hyponym> merupakan suatu <hypernym> yang

- <hyponym> secara khusus menjadi sebutan bagi <hypernym> yang
- <hyponym> termasuk <hypernym> yang
- <hyponym> termasuk ke dalam <hypernym> yang
- <hyponym> termasuk ke dalam kategori <hypernym>
- <hyponym> termasuk ke dalam salah satu <hypernym> yang
- <hyponym> tersebut merupakan <hypernym>
- Beberapa <hypernym> seperti <hyponym>
- Beberapa contoh <hypernym> lainnya adalah <hyponym>
- Beberapa jenis dari <hypernym> adalah <hyponym> dan <hyponym>
- Berbagai <hypernym> seperti <hyponym>
- Contoh dari <hypernym> adalah <hyponym>
- Contoh dari <hypernym>, yaitu <hyponym>, <hyponym>, dan <hyponym>
- Istilah umum dari <hyponym> adalah <hypernym>
- Jenis <hypernym> yang paling banyak dikenal adalah <hyponym>
- Jenis-jenis <hypernym> antara lain <hyponym> dan <hyponym>
- Salah satu <hypernym> adalah <hyponym>
- Salah satu <hypernym> yang mirip dengan <hyponym> adalah <hyponym>
- Salah satu contoh dari <hypernym> adalah <hyponym>
- Sebagai salah satu <hypernym>, <hyponym>
- Sebagai sebuah <hypernym>, <hyponym> merupakan
- Secara umum, <hyponym> merupakan <hypernym> yang dapat
- Selain <hyponym>, <hyponym> juga menjadi <hypernym> yang
- Terdapat banyak <hypernym>, seperti <hyponym>
- Terdapat beberapa contoh <hypernnym>, di antaranya <hyponym>, <hyponym>, dan <hyponym>
- Walaupun <hyponym> adalah <hypernym>, tetapi