DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

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 salah satu informasi penting yang dibutuhkan jika ingin mengetahui hubungan antar kata. Informasi tersebut sering digunakan untuk menunjang penelitian populer lainnya yaitu *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 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*.

WordNet mengandung beberapa kelas kata seperti 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 *synonymy*, *antonymy*, *hyponymy*, *hypernym*, *meronymy*, *holonymy*, *troponymy*, dan *entailment*.

2.1.1 Indonesian WordNet

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.

2.1.2 WordNet Bahasa

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.

2.1.3 Kelemahan WordNet Bahasa Indonesia

Hingga saat ini, relasi semantik antar kata yang dimiliki oleh WordNet Bahasa Indonesia merupakan hasil turunan dari relasi semantik WordNet Princeton. Sayangnya, pemanfaatan PWN untuk mendapatkan relasi semantik kata Bahasa Indonesia menemui beberapa hambatan. WordNet Bahasa Indonesia menjadi sangat tergantung dengan struktur PWN untuk mendapatkan relas semantik suatu *synset*. Selain itu, beberapa *synset* Bahasa Indonesia juga tidak dapat dipetakan secara tepat ke *synset* PWN, beberapa kata kehilangan arti atau mendapat arti yang kurang tepat. Hal ini menyebabkan 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 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 semantik antar kata. Domainnya adalah kata-kata yang tergabung dalam kelas kata benda (*noun*).

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. Banyak pula relasi yang hanya menghubungkan antar dua entitas (relasi biner), seperti *terletak-di(Universitas Indonesia, Depok)* atau *ditulis-oleh(Habis Gelap Terbitlah Terang, RA Kartini)*.

2.2.1 Relasi Semantik

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

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

- *Antonymy* 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'.

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

- *Meronymy* dan *holonym* adalah relasi yang menyatakan hubungan bagian satu dengan yang lain, dimana *meronym* menyatakan sub-bagian dan *holonym* menyatakan bagian yang lebih besar. Seperti relasi hyponym-hypernym, relasi *meronym-holonym* 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 (*holonym*) memiliki nukleus, ribosom, mitokondria (*meronym*).

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

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).

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 *synonymy*. 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 Korpus

Korpus adalah data yang dengan format yang dapat dibaca oleh mesin dan memiliki fungsi spesifik (Atkins et al., 1992).

2.4.1 Korpus Pasangan Kata Relasi Semantik

Saat ini, belum ada korpus independen yang menggambarkan relasi semantik antar kata dalam Bahasa Indonesia. Relasi semantik Bahasa Indonesia yang ada saat ini didapatkan dari hasil relasi turunan pemetaan *synset* Bahasa Indonesia dengan PWN.

Korpus relasi semantik yang ingin dibuat menyimpan informasi dalam bentuk pasangan kata yang berelasi serta relasi yang menghubungkannya. Selanjutnya, pasangan kata berelasi akan disebut sebuah *pair*. Untuk relasi yang bersifat transitif seperti *hypernym-hyponym*, ada identifikasi kata mana yang merupakan *hypernym* dan mana yang merupakan *hyponym*. Relasi *hypernym* dan *hyponym* adalah relasi bersifat transitif yang dapat direpresentasikan dalam bentuk hirarki. Pada penelitian

ini, korpus *pair* yang diperoleh merepresentasikan hubungan yang berjarak tepat satu dari yang lain.

2.5 Wikipedia

Wikipedia[1] adalah ensiklopedia terbuka yang memuat berbagai bahasa dan merupakan hasil kolaborasi berbagai penulis (Denoyer dan Gallinari, 2006). Wikipedia adalah salah satu korpus teks dokumen terbesar yang disimpan secara *online* yang dapat diakses dan 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).

2.5.1 Korpus Wikipedia

Wikipedia Bahasa Indonesia saat ini telah memuat lebih dari 400.000 artikel dari berbagai domain. Artikel-artikel yang disimpan dalam Wikipedia dapat diunduh secara gratis dalam bentuk *dumps*[2] dengan format XML. Beberapa tipe *dump* 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*.

2.5.2 Wikipedia sebagai Lexical Semantic Resource

Artikel dalam Wikipedia terdiri dari berbagai domain kategori dan memuat berbagai *noun* umum. 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. Selain itu, sudah banyak penelitian ekstraksi relasi semantik dengan metode pattern yang menggunakan artikel Wikipedia seperti penelitian Ruiz-Casado et al. (2005) dan Arnold dan Rahm (2014).

[1] www.wikipedia.org
[2] dumps.wikimedia.org

2.6 Pattern

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.

2.6.1 Lexico-syntactic Pattern

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 *hyponymy* 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.

2.7 Pattern Extraction dan Matching

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*.

2.8 Sequence Analyisis

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*.

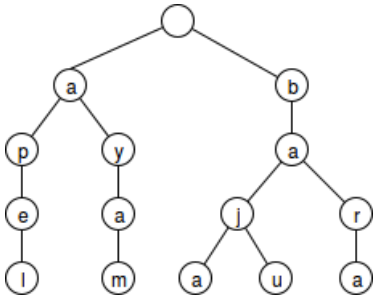
2.8.1 Standard Trie

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.



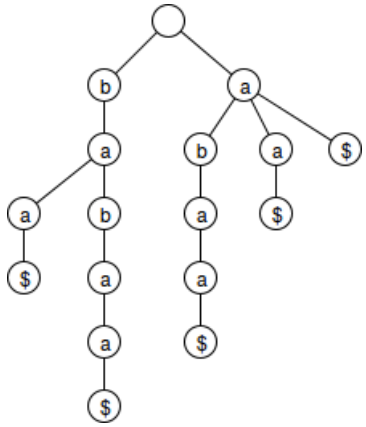
Gambar 2.1: Contoh Standard Trie

2.8.2 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

2.9 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*.

2.9.1 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 berulang, 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.

2.9.2 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.

2.9.3 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*.

2.10 Word Embedding

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

2.11 Pointwise Mutual Information

Pointwise Mutual Information (PMI) adalah pengukuran nilai asosiasi antar variabel. Dalam bidang *information theory*, PMI dapat dimanfaatkan untuk menghitung asosiasi kemunculan dua buah kata. Jika diberikan dua buah kata x dan y,

maka nilai PMI kata tersebut dalam suatu dokumen dapat dihitung menggunakan rumus berikut.

$$pmi(x;y) = log\frac{p(x,y)}{(x)p(y)} = log\frac{p(x|y)}{p(x)} = log\frac{p(y|x)}{p(y)}$$

dengan $p(x) = \frac{f(x)}{N}$

$$pmi(x;y) = log\frac{f(x)N}{f(x)f(y)}$$

- $p(x)$ adalah probabilitas kemunculan kata x dalam korpus
- $f(x)$ adalah frekuensi kemunculan kata x dalam korpus
- N adalah toatal seluruh kata dalam korpus

Pengukuran ini bersifat simetris, sehingga $p(x;y) = p(y;x)$. Nilai PMI dapat merupakan bilangan positif maupun negatif. Jika nilai PMI adalah nol (0), berarti kedua variabel saling *independent*.

2.11.1 Skip PMI

PMI umumnya hanya menggunakan model *bigram* atau *trigram*. Model ini hanya melihat hubungan kata yang berdampingan. Sebagai contoh ingin diketahui PMI untuk *bigram* 'hong kong', 'sepak bola', dan 'amerika serikat'. Pada penelitian ini dilakukan modifikasi yaitu membuat model *skip-gram* PMI. Kita menghitung nilai PMI antar dua kata yang dipisahkan dengan *n* diantaranya.

2.12 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.

2.12.1 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.

2.12.2 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.

2.12.3 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.

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

2.12.4 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

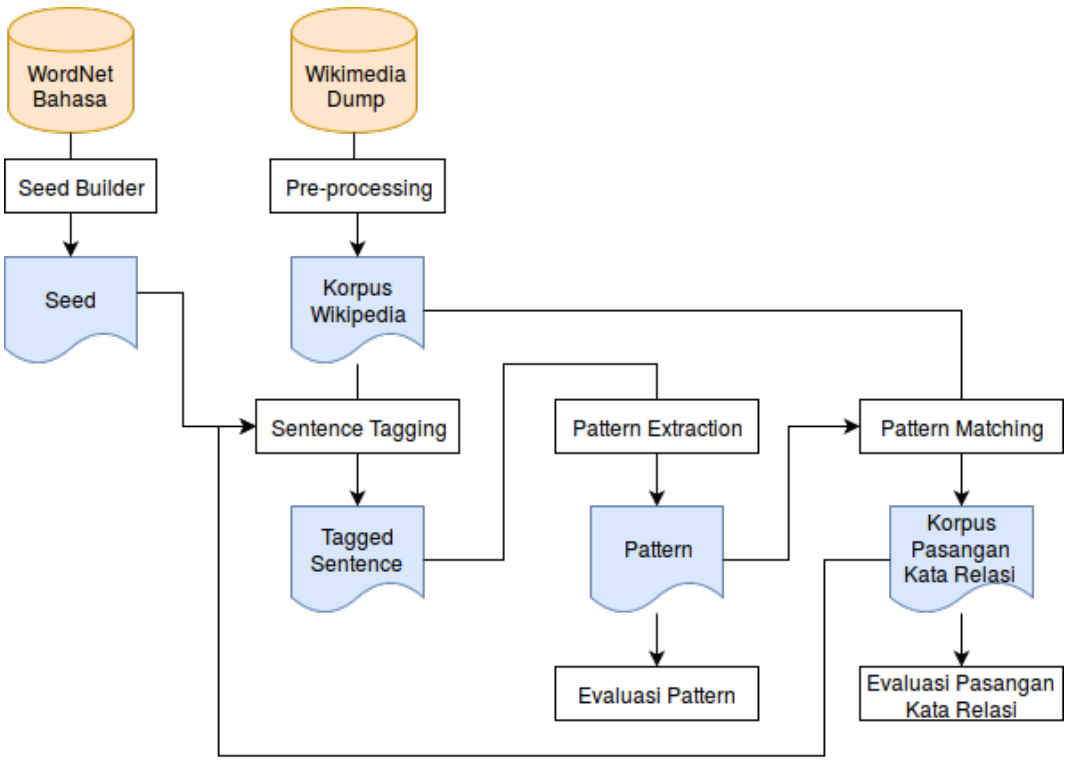
BAB 3
RANCANGAN METODOLOGI

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 Spesifikasi Penelitian

Berikut adalah spesifikasi penelitian yang dilaksanakan dan batas-batasanya.

- Relasi yang diperhatikan adalah *hypernym-hyponym*.
- Kelas kata yang diperhatikan adalah *noun* dan *proper noun*.
- Kata yang diekstrak termasuk *multi word*. Contoh: sepak bola, Amerika Serikat, dan lainnya.



Gambar 3.1: Arsitektur Penelititan

3.2 Rancangan Pengembangan Korpus

Pengembangan korpus pasangan kata berelasi yang digunakan pada penelitian ini 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*. Berikut adalah penjelasan singkat setiap tahapan.

1. Pembentukan *seed* dilakukan untuk mendapatkan pasangan kata berelasi yang digunakan sebagai dasar penelitian. Proses ini memanfaatkan *resource* yang dimiliki WordNet Bahasa.

2. Artikel Wikipedia diperoleh dalam bentuk *Wikipedia dump* dan perlu diolah sehingga memiliki representasi sesuai dengan format yang diharapkan. Informasi yang diperlukan hanya bagian isi artikel dan kemudian ditulis berdasarkan kalimat untuk setiap baris.

3. Menggunakan *seed* dan korpus 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.

4. 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.

5. Proses berikutnya adalah *pattern matching* dimana *pattern* hasil ekstraksi dan korpus Wikipedia kembali digunakan untuk membentuk pasangan kata relasi baru (*pair*).

6. Korpus pasangan relasi kata yang terbentuk digunakan untuk iterasi selanjutnya sesuai dengan metode *bootstrapping*. Proses iterasi dilakukan hingga pasangan kata relasi baru yang dihasilkan jenuh. Terakhir dilakukan evaluasi untuk mengetahui akurasi data yang dihasilkan.

Proses ini diharapkan dapat menghasilkan korpus pasangan kata relasi *hyponym-hypernym* yang berkualitas baik dan berukuran besar. *Pair* yang di-

hasilkan ditulis dalam bentuk *tuple* $(kata_hyponym; kata_hypernym)$ dengan kedua kata berada dalam kelas kata benda.

3.3 Pre-processing Data

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

3.3.1 Pre-processing Data 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. Setelah data Wikipedia dibersihkan dari simbol-simbol tersebut, langkah selanjutnya adalah merepresentasikan korpus dalam bentuk kalimat.

Artikel-artikel Wikipedia dibentuk ke dalam format yang telah didefinisikan dengan satu kalimat dipisahkan untuk setiap barisnya. Data tersebut juga digunakan untuk proses *part-of-speech tagging*. Hasil dari *pre-processing* data Wikipedia adalah dua korpus besar yaitu korpus tanpa *pos tag* yang digunakan sebagai masukan *pattern extraction* dan korpus dengan *pos tag* yang digunakan sebagai masukan *pattern matching*.

3.3.2 Pengumpulan Seed

Pasangan kata relasi *hypernym-hyponym* 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 dimiliki oleh WordNet tersebut akan dipasangkan dengan lema *hypernym*-nya, sehingga terbentuk relasi biner antara kata yang merupakan *hyponym* dan kata yang merupakan *hypernym*. Format untuk korpus ini mengikut format korpus pasangan kata relasi yang akan digunakan.

3.4 Pembentukan Pattern

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

3.4.1 Sentence Tagging

Sentence tagging adalah proses *intermediate* sebelum sistem dapat membentuk sebuah *pattern* leksikal. Proses ini akan memberi *tag hypernym* dan *hyponym* 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 hypernym* dan *hyponym*. 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 hypernym* atau *hyponym*.

3.4.2 Pattern Extraction

Setelah didapatkan kumpulan kalimat yang sudah diberi *tag hypernym* dan *hyponym*, 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 hypernym* dan *hyponym*. 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.

3.5 Ekstraksi Pair

Tujuan utama dari penelitian ini adalah membentuk korpus pasangan kata relasi, sehingga proses ekstraksi *pair* adalah tahapan utama dalam 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.

3.5.1 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 *hypernym-hyponym*. 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.

3.6 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 adalah *Basilisk* dengan beberapa modifikasi. Pemilihan *Basilisk* sebagai metode *Bootrapping* didasari proses ekstraksi *seed* baru yang memanfaatkan *pattern*. Pada penelitian ini, dilakukan modifikasi sesuai kebutuhan. Secara umum, proses *bootstrapping* dikelompokan ke dalam dua tahap yaitu iterasi ke-1 dan iterasik ke-2 hingga n.

Penggunaan *seed* yang berasal dari WordNet Bahasa hanya dilakukan pada iterasi pertama. Pada iterasi ini, *pattern* yang merupakan hasil dari *seed* dengan lema

sama dan *seed strict* digabung dan diurutkan berdasarkan bobot *pattern*. Lima *pattern* terbaik diambil untuk selanjutnya digunakan dalam proses *pattern matching*. *Seed* yang membentuk kelima *pattern* ini langsung dimasukkan ke dalam korpus pasangan relasi kata (*pair*). Hal tersebut dapat memfilter *pair* yang dihasilkan akibat *error* dari *resource* WordNet Bahasa tidak ikut terambil. *Pair* yang bobotnya melebihi nilai *threshold* digabung ke dalam korpus pasangan kata relasi.

Pada iterasi ke-2 hingga n, *pair* dalam korpus pasangan relasi kata yang telah terbentuk digunakan sebagai *seed* untuk proses *sentence tagging* dan *pattern extraction*. *Pattern* yang dihasilkan akan digabung dengan seluruh *pattern* lama kemudian di *ranking* kembali. Seperti pada metode *Basilisk*, jumlah *pattern* yang digunakan pada iterasi berikutnya akan terus bertambah. Satu *pattern* terbaik dari hasil pengurutan bergabung dengan *pattern* terpilih lama untuk digunakan dalam proses *pattern matching*. Hal ini membuka kemungkinan *pair* baru terekstraksi. Sama seperti proses sebelumnya, *pair* bobtnya melebihi nilai *threshold* digabung ke dalam korpus.

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 di akhir pengembangan. Anotator melakukan evaluasi secara manual untuk mengetahui kualitas *pattern* dan *pair* yang dihasilkan.

3.7 Metode Evaluasi

Penelitian ini tidak hanya menghasilkan korpus pasangan kata relasi *hypernym-hyponym*, namun juga *pattern* leksikal Bahasa Indonesia yang merepresentasikan relasi tersebut. Setelah proses iterasi selesai, evaluasi dilakukan terhadap *pattern* dan *seed* yang dihasilkan. Proses evaluasi dilakukan dengan bantuan anotator.

3.7.1 Evaluasi Pattern

Evaluasi *pattern* dilakukan dengan bantuan 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*.

3.7.2 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.

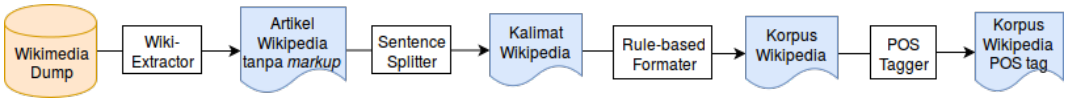
Dari setiap nilai anotasi, pair dimasukan ke dalam kategori yang lebih spesifik. Untuk data yang dianotasi benar, pair dapat dimasukan ke dalam dua kategori yaitu

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

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



Gambar 4.1: Pre-processing Data Wikipedia

Data hasi Wikipedia Dump diformat secara otomatis menggunakan *tools* WikiExtractor. WikiExtractor adalah program berbasis Python yang dibuat oleh Giuseppe Attardi dan Antonio Fuschetto. Menggunakan *tools* ini, didapatkan data yang sudah tidak mengikuti format MediaWiki Markup, Language. Program ini dapat diunduh dari Github dan dijalankan 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.

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 *hypernym-hyponym* antar dua kata pada kalimat yang sama. Untuk itu, perlu dilakukan proses *sentence splitting* yang dapat mengidentifikasi suatu kalimat dalam paragraf. Hasil dari proses tersebut adalah dokumen yang terdiri dari baris-baris yang merepresentasikan satu kalimat.

Proses ini dilakukan dengan menggunakan *script* yang telah dibuat sebelumnya oleh Ken Nabila Setya dari Fasilkom UI, Indonesia. Ditambah pula satu *script* yang

dapat secara otomatis melakukan *splitting* untuk seluruh dokumen. 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.

Setelah melalui proses *sentence splitter*, berikut adalah hasilnya.

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

Dari korpus yang berisi kalimat Wikipedia, diberikan beberapa aturan tambahan untuk membuat korpus sesuai format yang diinginkan. Penambahan aturan juga untuk mengurangi ambiguitas dan bentuk usaha generalisasi *pattern*. Berikut adalah beberapa aturan tambahan untuk *pre-processing* 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>

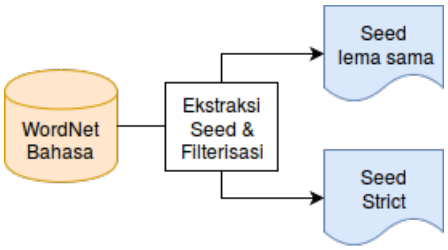
4.2 POS Tagging Kalimat Wikipedia

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* digunakan untuk identifikasi kata-kata tersebut. Proses *pos tagging* menggunakan *tools* 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

4.3 Seed Builder



Gambar 4.2: Proses Pembentukan *Seed*

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 *hypernym*-nya. Dari setiap *synset hypernym*, 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* 1-1 $(kata\_hyponym, kata\_hypernym)$.

Kode 4.2: Algoritme pembentukan *seed*

```
buildSeed:
  ambil seluruh lemma bahasa indonesia noun
  untuk setiap lema:
    ambil seluruh synset dari lema tersebut
    untuk setiap synset:
      ambil seluruh synset hypernym
      untuk setiap synset hypernym
        ambil seluruh lemma hypernym
        untuk setiap lemma hypernym:
          filterisasi
          bentuk pasangan lemma-lemma hypernym
```

4.3.1 Filterisasi Seed

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* adalah 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'.

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 *hypernym*-nya, 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')]) ('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')])

4.4 Sentence Tagging

Setelah memperoleh pasangan kata relasi Bahasa Indonesia, perlu dilakukan *tagging* pasangan kata tersebut ke kalimat-kalimat dalam korpus Wikipedia. Data yang digunakan untuk proses ini adalah korpus Wikipedia tanpa *pos tag*. Beberapa tahapan dilakukan pada proses *tagging sentence* dengan pasangan kata relasi adalah sebagai berikut.

1. Dibaca seluruh pasangan kata relasi *hyponym-hypernym*.
2. Untuk setiap kalimat pada korpus Wikipedia, di cek apakah kalimat tersebut mengandung 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 di-*tag* sesuai relasinya dan disimpan ke dalam korpus berisi kalimat dengan *tag hyponym* dan *hypernym*.

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 satu kata yang merupakan sub kata pasangannya, maka pasangan kata tersebut tidak diikutsertakan untuk menghindari ambiguitas. Contoh pasangan kata yang

tidak diikutsertakan adalah $(ikan\ gurame; ikan)$, kata 'ikan' terkandung dalam kedua kata relasi.

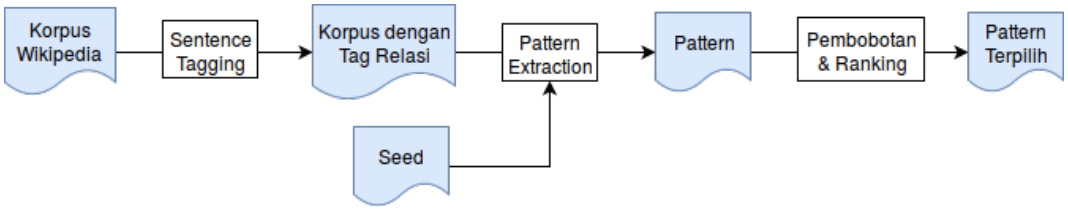
Kode 4.3: Algoritme *sentence tagging*

```
tagAll:
  seeds = load_all_seeds()
  foreach sentence in sentences:
    tagSentence(sentence, seeds)

tagSentece(sentence, seeds):
  foreach seed in seeds:
    if (sentence.contains(seed.hypernym) && sentence.contains(
        seed.hyponym)):
      put_tag_hyponym()
      put_tag_hypernym()
      save_sentence()
```

Berikut adalah contoh kalimat yang terbentuk dari proses *sentence tagging*. Diberikan pasangan kata relasi *hyponym-hypernym* $(fermion; partikel)$ 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>



Gambar 4.3: Proses Pembentukan *Pattern*

4.5 Pattern Extraction

Setelah mendapatkan kalimat-kalimat yang telah di-*tag* dengan kata relasi, ingin dicari *pattern* yang dapat digunakan untuk menambah jumlah relasi kata. Pembuatan *pattern* menggunakan algoritma dasar *standard trie* dengan beberapa modifikasi. Proses ini diimplementasi secara mandiri menggunakan program Java dengan mengikuti algoritma pembuatan *Trie* sederhana.

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.
Sebagai contoh dua kalimat sebagai berikut.

<start> <hyponym>singa<hyponym> adalah <hypernym>kucing<hypernym> yang berukuran besar <end>
<start> <hyponym>serigala<hyponym> adalah <hypernym>anjing<hypernym> yang tinggal di hutan <end>

Akan menghasilkan *Pattern* Tree sebagai berikut. Angka setelah kata adalah bobot suatu *node*.

Kode 4.4: Contoh *pattern tree* yang terbentuk

```
`-^ (1)
 `-<hyponym> (2)
   `-adalah (2)
     `-<hypernym> (2)
       `-yang (2)
        |-berukuran (1)
        | `-besar (1)
        `-tinggal (1)
          `-di (1)
            `-hutan (1)
```

4.5.1 Informasi dalam Pattern

Beberapa informasi yang disimpan dalam suatu *pattern* adalah *sequence* kata yang merepresentasikan *pattern* tersebut, 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.

4.5.2 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 *hypernym-hyponym* atau *hyponym-hypernym*. Kata-kata diantara dua relasi dapat dianggap paling dekat jika ingin mencari *pattern* 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 sebelum kata relasi, dapat memberikan informasi untuk yang dapat meningkatkan kualitas *pattern* yang dihasilkan. Pada contoh diatas dengan $(n = 1)$, *sequence* kata yang dihasilkan adalah '<start> <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'

Kode 4.5: Algoritme pembentukan *pattern*

```
getPattern(type):
  createPatternTree(type):
  for leaf in leaves:
    patterns.add(leaf.getPattern())
  return patterns

createPatternTree(type):
  foreach sentence in sentences:
    index = getIndexToken(sentence, type)
    sequence = get_sequence(sentence, index)
    patterntree.add(sequence)

getIndexToken(sentence, type):
  tokens = tokenize_sentence(sentence)
  index = (start, end)
  if (type = inbetween):
    for(i = 0..tokens.size()):
```

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

4.5.3 Vektor Pattern

Suatu *pattern* dapat direpresentasikan menjadi vektor berdasarkan nilai-nilai yang dimilikinya. Vektor ini dapat dimanfaatkan untuk menentukan *pattern* yang baik (pembobotan). Fitur utama pada vektor *pattern* diambil dari informasi yang disimpan oleh suatu *pattern*, yaitu total kemunculan *pattern*, jumlah *seed* unik, dan jumlah kalimat unik yang membentuk *pattern* tersebut. Fitur lain adalah hasil kombinasi perbandingan antar nilai utama. Fitur tambahannya adalah 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.

4.5.4 Validasi dan Filterisasi Pattern

Setelah terbentuk *pattern tree*, perlu di-*list* seluruh *pattern* yang dihasilkan. Proses pembentukan *pattern* cukup dengan menelusuri *path* dari *node leaf* hingga *root*. Untuk mengurangi *pattern* yang kurang baik, dilakukan proses validasi. Beberapa aturan yang harus dipenuhi agar suatu *pattern* dianggap *valid* 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 <hyper-

nym>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 sendiri berdasarkan pengamatan nilai pada vektor *pattern*.

4.5.5 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 penempatan *hypernym-hyponym*-nya 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>. Padahal, bagian *hypernym-hyponym* tersebut nantinya digantikan dengan kata yang kemunculannya cocok dengan *pattern* yang diberikan. Jika *pattern* tersebut dibiarkan begitu saja, akan banyak dihasilkan *pair* yang salah.

Strategi yang dilakukan untuk masalah ambiguitas antar *pattern* adalah membangun suatu arsitektur yang dapat mengidentifikasi dan menyelesaikannya. Tahapan pembangunan *pattern* unik adalah sebagai berikut.

1. Dicari *pattern* dengan pendekatan hanya memperhatikan kata-kata diantara pasangan kata 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, memperhatikan *n* kata sebelum atau sesudah kemunculan relasi.

4. Hasil kedua pendekatan digabung dan dicek apakah *pattern* tersebut dibutuhkan. Suatu *pattern* hasil evaluasi ulang dinyatakan dibutuhkan jika *substring* dari *pattern* tersebut tergolong dalam *list pattern* yang membutuhkan evaluasi ulang.

5. Proses evaluasi dilakukan secara berulang dari dengan *n* yang terus bertambah dari 1 (satu). Pada penelitian ini nilai *n* dibatasi hingga 2 (dua), sehingga maksimum ada 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.

Kode 4.6: Algoritme pembentukan *pattern unik*

```
patternUnik():
  patternunik; patterntmp;
  patterns = getPattern(inbetween)
  unikfy(patterns)
  i = 0;
  while (patterntmp is not empty && i < 3):
    before-patterns = getPattern(i-before)
    after-patterns = getPattern(i-after)
    hasil = cekNeeded(before-patterns, after-patterns)
    unikfy(hasil)
  return patternunik

unikfy(patterns):
  mappatterns;
  foreach pattern in patterns:
    pk = pattern.getKey()
    if (mappatterns.containsKey(pk)):
      mappatterns.get(pk).add(pattern)
    else:
      mappatterns.put(pk, List.add(pattern))
  foreach pk in mappatterns:
    if mappatterns.get(pk).size() > 0:
      patterntmp.add(pk)
    else
      patternunik.add(mappatterns.get(pk))
```

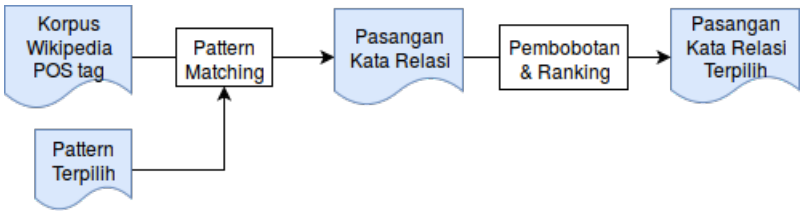
4.5.6 Pengurutan Pattern

Setelah didapatkan *pattern* yang sesuai, dilakukan proses pengurutan (*sorting*) untuk mengetahui *pattern* mana yang terbaik berdasarkan bobot 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 kecil jumlah token dalam *pattern* tersebut jika di-*parse* dengan spasi.

4.6 Pattern Matching

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 menggunakan algoritma *Suffix Tree* dengan modifikasi. Implementasi dilakukan secara mandiri menggunakan program Java.



Gambar 4.4: Proses Pembentukan *Pair*

Sebuah *node* merepresentasikan satu kata dalam kalimat. Untuk setiap kalimat, dibentuk sebuah suffix tree yang merepresentasikan kalimat tersebut dan selanjutnya dicocokan dengan *pattern* yang ada. Dibuat pula kelas *Pair* yang merepresentasikan pasangan kata relasi *hypernym-hyponym* yang dihasilkan dari proses *pattern matching*. *Pair* menyimpan informasi seperti total kemunculan, total dokumen unik, daftar kalimat unik dan *pattern* unik yang menghasilkan *pair*.

Tahapan proses *pattern matching* jika diberikan satu kalimat dan satu *pattern* adalah sebagai berikut.

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 dengan *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, setelah dijalankan proses *pattern matching* terhadap korpus Wikipedia dengan *pattern* hasil ekstraksi, 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.

Masalah lain yang muncul adalah jika kata yang ingin diekstrak merupakan *multi word*. *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 word*. *Multi word* disimpan dalam satu *node* pada pembentukan *suffix tree*.

Kode 4.7: Algoritme ekstraksi *pair*

```
main:
  patterns = loadAllPatterns()
  openFiles()
  for sentence in sentences:
    matchAllPattern(sentence, patterns)

matchAllPattern(sentence, patterns):
  suffix_tree = buildTree(sentence)
  foreach pattern in patterns:
    matchTree(suffix_tree, pattern)

buildTree(sentence):
  tokens = buildSequence(sentence)
  suffix_tree;
  for(i = 0..tokens.size()):
    addSequence(suffix_tree, tokens, i)
  return suffix_tree;
```

```
addSequence(suffix_tree, tokens, i):
  node = suffix_tree.root
  for (j=i..tokens.size()):
    if (!node.childs.contains(tokens[j])):
      node = node.childs.add(tokens[j])
    node = node.childs.get(tokens[j])

matchTree(stree, pattern):
  foreach child in stree.root.childs:
    recursiveMatch(pattern, 0, child, ``'', ``'')
recursiveMatch(pattern, i, node, hype, hypo):
  if (i >= pattern.size() && hype && hypo):
    pair = createPair(hype, hypo)
    savePair(pair)
  else:
    cek = pattern[i]
    if (cek == ``hypenrym'' && cek.tag == (NN|NNP)):
      hype = cek
    else if (cek == ``hyponym'' && cek.tag == (NN|NNP)):
      hypo = cek
    else
      if (cek != node.name) return
    foreach c in node.childs:
      recursiveMatch(pattern, i+1, c, hype, hypo)
```

4.6.1 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.

4.6.2 Filterisasi dan Validasi Pair

Pair baru yang dihasilkan untuk proses ini berjumlah sangat banyak, namun tidak semua *pair* yang dihasilkan memenuhi relasi *hypernym-hyponym*. Beberapa *pair* kebetulan terekstrak akibat memenuhi *pattern* lekscal yang sama dengan salah satu *pattern* yang digunakan untuk proses *pattern matching*. Untuk mengeliminasi data yang tidak diyakini benar, dilakukan validasi sederhana untuk setiap *pair* yang dihasilkan. *Pair* dinyatakan benar jika terdapat lebih dari satu *pattern* yang mengek-

strak *pair* tersebut.

Setelah mengeliminasi *pair* yang hanya terbentuk dari satu *pattern*, dilakukan pembobotan. Tidak semua *pair* yang dihasilkan masuk ke dalam korpus kata relasi. Bobot satu *pair* dihitung menggunakan rumus berikut.

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

Jika nilai bobot melebihi threshold, maka *pair* dimasukan ke dalam korpus pasangan kata relasi.

4.6.3 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 penelitian ini, 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 bawah ('_'). 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*.

Model yang terbentuk, digunakan untuk memberi nilai *similarity* antara kata *hypernym-hyponym* dalam satu *pair*. Hal ini diharapkan dapat memberi informasi lebih mengenai 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 untuk Iterasi 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>

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. Tabel 5.5 memaparkan jumlah *pattern*, *pattern* baru, total *pair* hasil ekstraksi, dan total korpus pasangan kata relasi yang dihasilkan untuk setiap iterasi.

Tabel 5.5: Hasil Eksperimen 1

No.	Pattern Baru Terpilih	Total Pattern	Total Pair yang Dihasilkan	Korpus Pasangan Kata Relasi
1.		106	82409	939
2.	<start> <hyponym> merupakan <hypernym>	347	104389	1255
3.	<hyponym> merupakan <hypernym> yang	395	108397	1430
4.	<hypernym> atau <hyponym>	425	108542	1432

Jumlah *pair* yang nilainya melebihi *threshold* dapat dikatakan sedikit jika dibandingkan dengan seluruh *pair* yang terekstrak.

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 5.6 memaparkan hasil dari eksperimen kedua.

Tabel 5.6: Hasil Eksperimen 2

No.	Pattern Baru Terpilih	Total Pattern	Total Pair yang Dihasilkan	Korpus Pasangan Kata Relasi
1.		106	82409	2193
2.	<start> <hyponym> merupakan <hypernym>	699	104389	3071
3.	<hyponym> merupakan <hypernym> yang	791	108397	3484
4.	<hyponym> menjadi <hypernym>	842	108737	3493

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 5.7: Hasil Eksperimen 3

No.	Pattern Baru Terpilih	Total Pattern	Total Pair yang Dihasilkan	Korpus Pasangan Kata Relasi
1.		106	10267	438
2.	<hyponym> adalah sebuah <hypernym>	437	11262	459

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.

5.6 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.

5.6.1 Pattern Buatan Manual

Berikut adalah daftar *pattern* leksikal yang dibentuk secara manual oleh anotator. Anotator mengamati kalimat-kalimat di dalam teks dokumen yang mengandung pasangan kata *hypernym-hyponym*, kemudian membuat *patter* yang dapat merepresentasikan relasi. Daftar *pattern* manual dapat dilihat pada lampiran 1. Tidak seperti *pattern* yang dihasilkan oleh sistem dimana satu *pattern* hanya dapat mengandung satu pasang *tag hypernym-hyponym*, beberapa *pattern* manual mengandung lebih dari satu *tag hyponym*.
on-going: perbandingan pattern manual dan pattern hasil sistem

5.6.2 Hasil Anotasi Pattern

Dua anotator memberi penilaian terhadap sejumlah *pattern* yang sama.
on-going: hasil anotasi pattern, evaluasi dan analisis.

5.6.3 Analisis Pattern Hasil Sistem

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.

5.7 Hasil Anotasi Pair

Anotasi *pair* dilakukan oleh tiga anotator berbeda dengan daftar *pair* yang sama. *Pair* yang digunakan untuk proses anotasi diambil secara acak menggunakan teknik *random sampling*.
on-going: hasil anotasi pair, perbandingan anotasi antar anotator, akurasi pair, evaluasi dan analisis.

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.

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

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.

LAMPIRAN

LAMPIRAN 1 : PATTERN BUATAN MANUAL

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

<hyponym> adalah <hypernym>
<hyponym> ialah <hypernym>
<hyponym> merupakan <hypernym> yang
<hyponym> berarti <hypernym>
<hyponym> adalah salah satu <hypernym>
Salah satu <hypernym> adalah <hyponym>
<hyponym> disebut sebagai <hypernym> yang
Istilah umum dari <hyponym> adalah <hypernym>
<hypernym> terdiri dari <hyponym>, <hyponym>, dan <hyponym>
<hyponym> termasuk ke dalam kategori <hypernym>
<hyponym> adalah sebuah <hypernym> yang
Salah satu contoh dari <hypernym> adalah <hyponym>
<hyponym> adalah sejenis <hypernym> dengan
Jenis <hypernym> yang paling banyak dikenal adalah <hyponym>
<hyponym> adalah <hypernym> yang berhubungan dekat dengan <hyponym>
Beberapa jenis dari <hypernym> adalah <hyponym> dan <hyponym>
<hyponym> adalah suatu <hypernym>
<hyponym> atau <hyponym> adalah suatu jenis <hypernym> yang
<hyponym> adalah bagian dari <hypernym> yang
<hypernym> adalah kumpulan dari <hyponym>
<hyponym> secara khusus menjadi sebutan bagi <hypernym> yang
<hyponym> adalah salah satu <hypernym>
Secara umum, <hyponym> merupakan <hypernym> yang dapat
<hypernym> terdiri dari beberapa bagian, seperti <hyponym>, <hyponym>, dan <hyponym>
<hyponym> adalah <hypernym> yang bersifat
<hypernym> dapat dibedakan menjadi <hyponym>
Beberapa contoh <hypernym> lainnya adalah <hyponym>

<hypernym>, misalnya <hyponym>
<hypernym> seperti <hyponym>
<hyponym> merujuk pada <hypernym>
<hyponym> merujuk pada <hypernym> yang