Penguraian Dependensi Kalimat Bahasa Jepang dengan Penerapan Kendala pada Algoritma *Exhaustive Search*

Muhammad Iqbal Al Khowarizmi (13515086)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515086@std.stei.itb.ac.id; iqbalkhowarizmi@ymail.com

***Abstract*—Untuk memahami makna semantik sebuah kalimat, hubungan antarkata perlu diketahui. Hubungan antarkata dapat diketahui dengan melakukan penguraian dependensi pada kalimat. Memilih algoritma untuk menguraikan kalimat dalam sebuah bahasa perlu mempertimbangkan struktur sintaksis dan dependensi bahasa yang bersangkutan. Struktur sintaksis bahasa yang berbeda-beda membuat algoritma penguraian yang bekerja dengan baik pada suatu bahasa belum tentu dapat bekerja sebaik itu pada bahasa yang lain, termasuk untuk bahasa Jepang. Karena itu, dengan makalah ini, saya mencoba menerapkan strategi algoritma pada proses penguraian kalimat dalam bahasa Jepang. Penerapan akan dilakukan secara iteratif: dimulai dari algoritma yang paling sederhana hingga algoritma yang menerapkan kendala-kendala struktur dependesi bahasa Jepang. Makalah ini adalah kelanjutan dari makalah sebelumnya.**

Keywords—algoritma; dependensi; bahasa Jepang; penguraian kalimat

I. PENDAHULUAN

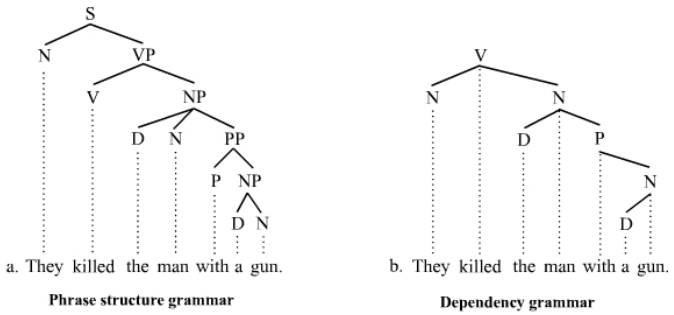
Penguraian (*parsing*) kalimat adalah salah satu masalah mendasar dalam pemrosesan bahasa alami. Penguraian kalimat diperlukan untuk memahami makna semantik kalimat tersebut. Salah satu cara untuk menguraikan kalimat adalah dengan membentuk pohon dependensi. Dari pohon dependensi tersebut, hubungan antarkata dapat terlihat sehingga makna semantik kalimat tersebut menjadi jelas.

Akan tetapi, struktur sintaksis tiap bahasa yang unik membuat algoritma penguraian untuk satu bahasa belum tentu dapat diterapkan dengan tingkat efisiensi dan akurasi yang sama pada bahasa lain. Merumuskan atau memilih algoritma yang digunakan untuk mengurai suatu bahasa harus memperhitungkan berbagai kendala struktur dependensi bahasa tersebut: proyektif atau non-proyektif, *head-initial* atau *head-final*, dan akar-tunggal atau akar-ganda. Karena itu, untuk menguraikan bahasa yang berbeda, seringkali diperlukan pendekatan yang berbeda pula. Pun begitu dengan bahasa Jepang.

Pendahuluan tentang bahasa Jepang, pohon dependensi, dan bagaimana pohon dependensi tersebut dapat menunjukkan ketaksaan sebuah kalimat telah saya bahas pada makalah saya sebelumnya [1]. Berikutnya, makalah ini akan berisi strategi algoritma yang dapat diterapkan untuk membentuk pohon sintaks berbasis dependensi dari sebuah kalimat dalam bahasa Jepang.

Pembahasan dalam makalah ini disusun sebagai berikut. Bagian 2 akan berisi penjelasan tentang bagaimana cara menggambarkan struktur kalimat dalam pemrosesan bahasa alami. Bagian 3 akan menguraikan struktur dependensi pada bahasa Jepang. Setelah itu, pada Bagian 4, penguraian kalimat dengan pendekatan dependensi akan dibahas dan algoritma paling sederhana untuk melakukan penguraian dependensi akan diperkenalkan. Pada Bagian 5, algoritma tadi akan diperbaiki dengan menerapkan kendala-kendala struktur dependensi bahasa Jepang. Terakhir, Bagian 6 akan menjelaskan kompleksitas algoritma yang telah dikembangkan.

II. STRUKTUR KALIMAT



Gambar 1: Tata bahasa konstituen dan dependesi. (Sumber: https://upload.wikimedia.org/wikipedia/commons/7/74/Theykilledthemanwithagun-1b.jpg)

A. Tata Bahasa Konstituen dan Dependensi

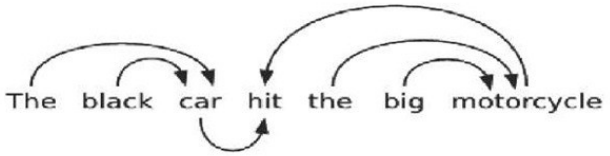
Ada dua cara untuk menggambarkan struktur sintaksis suatu kalimat dalam bahasa alami: dengan tata bahasa konstituen (*constituency grammar*) dan dengan tata bahasa dependensi (*dependency grammar*). Secara sederhana, tata bahasa konstituen memecah kalimat menjadi beberapa konstituen atau frasa, yang kemudian dipecah lagi menjadi konstituen yang lebih kecil, sedangkan tata bahasa dependensi

menggambarkan hubungan antarkata dalam sebuah kalimat. Gambar 1 menunjukkan perbedaan antara tata bahasa konstituen (a) dan tata bahasa dependensi (b).

B. Hubungan Dependensi Antarkata

Untuk setiap pasang kata yang memiliki hubungan dependensi, kita akan menyebut salah satu kata tersebut sebagai penguasa (*head*) dan kata lainnya sebagai unsur bergantung (*dependent*), dan bahwa ada keterkaitan di antara keduanya. Kata penguasa biasanya akan menentukan perilaku pasangan kata yang bersangkutan.

Pada pohon dependensi, penguasa digambarkan sebagai simpul orang tua dan unsur bergantung digambarkan sebagai simpul anak. Selain dengan pohon, struktur dependensi dapat juga digambarkan sebagai graf berarah dengan sisi yang ditarik dari unsur bergantung menuju penguasa.



Gambar 2: Struktur dependensi yang digambarkan sebagai graf berarah.
(Sumber: http://lh3.ggpht.com/_1wtadqGaaPs/TH_fk1UMZpI/AAAAAAAAAXSU/x31eftuvvxA/tmp7E73_thumb_thumb.jpg?imgmax=800)

III. Struktur dependensi pada Bahasa Jepang

Pada dasarnya, bahasa Jepang adalah bahasa dengan struktur kalimat SOP (subjek-objek-predikat) dengan urutan kata yang cenderung bebas. Tak seperti bahasa Indonesia yang fungsi sintaksis setiap katanya ditentukan oleh posisi kata tersebut, bahasa Jepang menentukan fungsi sintaksis setiap kata dengan posposisi yang disebut partikel. Mudahnya, partikel menentukan fungsi kata yang mendahuluinya. Sebagai contoh, perhatikan kalimat berikut.

(a) *Dia membaca buku.* (b) *Buku membaca dia.*

Ketika posisi kata ditukar, terjadi perubahan makna pada kalimat dalam bahasa Indonesia karena fungsi sintaksis kata ditentukan oleh letak kata tersebut di dalam sebuah kalimat. Tidak demikian dengan bahasa Jepang.

(a) 彼は本を読む。	(b) 本を彼は読む。
kare-wa hon-wo yomu	*hon-wo kare-wa yomu*
Dia membaca buku.	Dia membaca buku.

Kedua kalimat di atas masih memiliki makna yang sama karena letak kata dalam kalimat bahasa Jepang tidak menentukan fungsi sintaksis kata tersebut.

Karena karakteristik tersebut, kalimat dalam bahasa Jepang tidak diurai berdasarkan satuan kata tetapi berdasarkan satuan yang disebut *bunsetsu*[1]. Pohon dependensi akan dibangun berdasarkan hubungan antar-*bunsetsu*. Oleh sebab itu, sebenarnya lebih tepat mengatakan bahwa dalam sebuah kalimat, bukan urutan kata yang bebas melainkan urutan *bunsetsu* kecuali untuk *bunsetsu* yang mengandung verba utama kalimat tersebut. *Bunsetsu* yang mengandung verba utama sebuah kalimat harus ditempatkan di akhir kalimat itu. Sebagai contoh, perhatikan kembali kalimat bahasa Jepang di atas.

「 / 彼は / 本を / 読む」

Garis miring menunjukkan pemenggalan *bunsetsu*. Selain *bunsetsu* terakhir yang mengandung verba utama kalimat tersebut, posisi *bunsetsu* dapat dipertukarkan tanpa mengubah makna semantik kalimat.

Mengenai struktur dependensi, bahasa Jepang memiliki beberapa karakteristik:

1. *Head-final*:
   Head-final berarti ketergantungan selalu berarah ke kanan atau ke akhir kalimat.
2. *Single-head*:
   Single-head berarti setiap unsur bergantung hanya memiliki satu penguasa.
3. Berakar-tunggal:
   Berakar-tunggal berarti pohon dependensi yang dihasilkan hanya akan memiliki satu akar.
4. Terhubung:
   Terhubung berarti setiap *bunsetsu* memiliki hubungan dependensi.
5. Asiklik:
   Asiklik berarti tidak ada hubungan dependensi yang membentuk siklus atau dengan kata lain, jika A bergantung pada B, B dan anak-anaknya tidak akan bergantung pada A.
6. Proyektif:
   Proyektif berarti tidak ada sisi pohon dependensi yang saling tumpang tindih.

IV. Penguraian Dependensi

A. Keuntungan Penguraian Dependensi

Karakteristik dan struktur dependensinya menjadi alasan penguraian dependensi (*dependency parsing*) lazim digunakan untuk mengurai kalimat bahasa Jepang. Selain itu, ada beberapa hal yang menjadi keuntungan penggunaan penguraian dependensi[2]:

- Keterkaitan dependensi dekat dengan hubungan semantik yang dibutuhkan untuk interpretasi pada tahap selanjutnya.
- Pohon dependensi mengandung satu simpul untuk setiap kata. Karena tugas pengurai hanya untuk menghubungkan simpul-simpul yang ada, bukan

1 Pembahasan mengenai *bunsetsu* telah dilakukan pada makalah sebelumnya[1].

untuk mengusulkan simpul baru, proses penguraian agaknya bisa dikatakan lebih sederhana.

- Penguraian dependensi mengurai kalimat dengan menerima dan memasangkan kata satu persatu, alih-alih dengan menunggu frasa yang telah lengkap.

B. Permasalahan dalam Proses Penguraian Dependensi

Secara umum, proses penguraian dependensi dilakukan dengan cara memeriksa hubungan ketergantungan antara dua buah kata. Jika ada hubungan ketergantungan antara keduanya, maka akan dibentuk sisi berarah yang menghubungkan kedua simpul kata tersebut dari unsur bergantung menuju penguasanya. Semua kata dalam kalimat yang diurai akan diperiksa hingga akhirnya terbentuk sebuah pohon dependensi.

Ada dua permasalahan dalam proses penguraian dependensi: 1) bagaimana menentukan hubungan ketergantungan antarkata dan 2) bagaimana cara setiap kata dalam kalimat diperiksa. Permasalahan yang pertama berkaitan erat dengan tata bahasa bahasa target. Apakah tata bahasa mengizinkan kedua kata memiliki hubungan ketergantungan, kata mana yang seharusnya menjadi penguasa dan kata mana yang seharusnya menjadi unsur bergantung menjadi pertanyaa-pertanyaan utama dalam permasalahan ini. Bagaimana permasalahan ini ditangani akan menentukan akurasi penguraian yang dilakukan. Dibutuhkan pembelajaran mesin dan kecerdasan buatan agar komputer dapat menjawab pertanyaan-pertanyaan tersebut. Permasalahan ini berada di luar lingkup makalah.

Permasalahan yang kedua berkaitan erat dengan karakteristik struktur dependensi bahasa target. Bagaimana permasalahan ini ditangani akan menentukan efisiensi penguraian yang dilakukan. Pada dasarnya, permasalahan ini termasuk permasalahan pencarian. Menentukan strategi algoritma untuk memecahkan permasalahan ini dalam penerapannya pada proses penguraian kalimat bahasa Jepang akan menjadi bahasan dalam makalah ini.

C. Strategi Brute-force

Karena pada dasarnya pembentukan pohon dependensi adalah masalah pencarian, strategi yang paling langsung dan sederhana adalah pencarian dengan *brute-force*.

Strategi 1 (Pencarian dengan *brute-force*) *Untuk setiap pasangan kata dalam sebuah kalimat, periksa dan hubungkan pasangan tersebut sebagai penguasa-ke-unsur-bergantung atau unsur-bergantung-ke-penguasa jika tata bahasanya mengizinkan.*

Dengan kata lain, untuk kalimat dengan jumlah kata n, akan ada $n(n-1)$ pasangan yang diperiksa. Dari jumlah pasangan tersebut, terlihat kompleksitas algoritma terburuk untuk strategi ini adalah $O(n^2)$.

V. Perbaikan Algoritma Penguraian Dependensi untuk Bahasa Jepang

Strategi pencarian naif di atas jelas tidak efisien. Banyak perbaikan yang dapat dilakukan untuk mendapatkan algoritma yang lebih efisien dengan mempertimbangkan karakteristik struktur dependensi bahasa Jepang. Namun, sebelum berbicara tentang perbaikan algoritma, perlu dilakukan penyesuaian dahulu terhadap definisi Strategi 1 pada bagian sebelumnya mengingat bahasa Jepang memiliki karakteristik yang berbeda dari bahasa Indonesia dan bahasa Inggris.

Pada bahasa Indonesia dan bahasa Inggris, penguraian dilakukan dengan memecah kalimat menjadi kata-kata. Hal itu mudah dilakukan karena dalam bahasa-bahasa tersebut, kata-kata dipisahkan dengan spasi. Tidak demikian dengan bahasa Jepang. Pada bahasa Jepang, pencarian lebih natural jika dilakukan dengan memecah kalimat menjadi *bunsetsu*. Dengan demikian, Strategi 1 pada bagian sebelumnya didefinisikan ulang sebagai berikut.

Strategi 1 (Pencarian dengan *brute-force*) *Untuk setiap pasangan* bunsetsu *dalam sebuah kalimat, periksa dan hubungkan pasangan tersebut sebagai penguasa-ke-unsur-bergantung atau unsur-bergantung-ke-penguasa jika tata bahasa mengizinkan.*

Selanjutnya, Strategi 1 di atas akan dikembangkan dengan menerapkan karakteristik-karakteristik struktur dependensi bahasa Jepang ke dalam algoritma.

A. Menerapkan Kendala Head-final *ke dalam Algoritma*

Masalah pertama dalam Strategi 1 adalah tentang bagian pemeriksaan: apakah lebih baik *bunsetsu* diperiksa terhadap *bunsetsu-bunsetsu* setelahnya atau sebelumnya? Contohnya, ketika menerima *bunsetsu* ke-i, apakah pemeriksaan dilakukan terhadap *bunsetsu* ke-$(i+1)$ hingga ke-n, ataukah terhadap *bunsetsu* ke-1 hingga ke-$(i-1)$? Untuk alasan kemudahan menjawab pertanyaan-pertanyaan berikutnya, pemeriksaan akan dilakukan terhadap *bunsetsu-bunsetsu* sebelumnya.

Masalah kedua adalah tentang urutan pemeriksaan: apakah lebih baik pemeriksaan bergerak mundur atau maju. Contohnya, ketika menerima *bunsetsu* ke-i, apakah *bunsetsu* diperiksa dari *bunsetsu* ke-$(i-1)$ hingga *bunsetsu* ke-1 atau sebaliknya?

Untuk bahasa Jepang yang memiliki karakteristik proyektif, *bunsetsu* cenderung akan memiliki hubungan dependensi dengan *bunsetsu* di dekatnya daripada *bunsetsu* yang lebih jauh. Oleh karena itu, hubungan dependensi akan ditemukan lebih cepat jika pemeriksaan dilakukan dengan bergerak mundur. Dengan kata lain, ketika menerima *bunsetsu* ke-i, *bunsetsu* akan diperiksa dari *bunsetsu* ke-$(i-1)$ hingga *bunsetsu* ke-1.

Masalah ketiga adalah apakah lebih baik mencari penguasa terlebih dahulu atau unsur-bergantung terlebih dahulu? Contohnya, ketika menerima *bunsetsu* ke-i dan memeriksanya terhadap *bunsetsu* ke-j, apakah *bunsetsu* ke-j lebih cenderung menjadi penguasa atau unsur-bergantung dari *bunsetsu* ke-i? (j berada dalam rentang 1 hingga $i-1$).

Dalam bahasa Jepang, arah dependensi *bunsetsu* selalu mengarah ke kanan atau, dengan kata lain, ke akhir kalimat. Maksudnya adalah penguasa selalu berada di sebelah kanan unsur-bergantung yang dikuasainya. Oleh karena itu, ketika menerima *bunsetsu* ke-i dan memeriksanya terhadap *bunsetsu* ke-j, *bunsetsu* ke-j lebih cenderung menjadi unsur-bergantung dari *bunsetsu* ke-i.(j berada dalam rentang 1 hingga $i-1$).

Setelah menjawab ketiga pertanyaan di atas, dapat dirumuskan strategi baru untuk memperbaiki strategi sebelumnya.

Strategi 2 (*Exhaustive search* kiri-ke-kanan, unsur-bergantung dahulu) *Terima* bunsetsu *satu persatu dari awal kalimat dan periksa hubungan dependensi setiap* bunsetsu *terhadap* bunsetsu-bunsetsu *sebelumnya.*

Strategi di atas dapat diimplementasikan sebagai algoritma berikut:

Algoritma ES (*Exhaustive search* kiri-ke-kanan, unsur-bergantung dahulu)

Diberikan sebuah kalimat dengan n-bunsetsu.

```
for i ← 1 to n do
begin
    for j ← (i – 1) down to 1 do
    begin
        jika tata bahasa mengizinkan
            hubungkan j sebagai
        unsur-bergantung i;
        jika tata bahasa mengizinkan
            hubungkan j sebagai
            penguasa i;
    end
end
```

Algoritma di atas merepresentasikan kalimat sebagai larik *bunsetsu*. Meskipun begitu, kalimat dapat juga direpresentasikan sebagai senarai berkait atau struktur data lainnya.

B. Menerapkan Kendala Single-head *ke dalam Algoritma*

Struktur dependensi bahasa Jepang memiliki karakteristik *single-head* yang berarti bahwa setiap *bunsetsu* hanya memiliki satu penguasa (kecuali *bunsetsu* yang menjadi akar). Strategi 2 pada subbagian sebelumnya dapat diperbaiki dengan memanfaatkan karakteristik tersebut. Berikut adalah Strategi 3 yang merupakan perpanjangan dari Strategi 2.

Strategi 3 (Penanganan s*ingle-head*)

- *Prinsip: Jika sebuah* bunsetsu *telah memiliki penguasa, maka* bunsetsu *tersebut tidak akan memiliki penguasa lain.*
- *Implementasi:*
    - *Ketika sedang mencari unsur-bergantung kata saat ini, kata yang telah memiliki penguasa tidak perlu diperiksa.*
    - *Ketika sedang mencari penguasa kata saat ini, pencarian dapat langsung dihentikan ketika satu penguasa telah ditemukan.*

Strategi ini dapat diimplementasikan menjadi algoritma berikut:

Algoritma LS (Pencarian berbasis senarai)

Diberikan sebuah kalimat dalam bentuk senarai kata dan dua senarai lain: tanpaPenguasa dan senaraiBunsetsu

```
{Inisialisasi}
tanpaPenguasa ← []; {bunsetsu yang belum
                     punya penguasa}
senaraiBunsetsu ← [];      {bunsetsu yang
                           telah diterima}

repeat
    {Terima sebuah bunsetsu dan tambahkan ke
    senaraiBunsetsu}
    W ← bunsetsu selanjutnya;
    senaraiBunsetsu ← W + senaraiBunsetsu;

    {Pencarian unsur-bergantung W}
    for D ← setiap elemen dalam tanpaPenguasa,
        mulai dari awal;
    begin
        if D dapat bergantung pada W then
        begin
            hubungkan D sebagai unsur-
            bergantung W;
            hapus D dari tanpaPenguasa;
        end
    end

    {Pencarian penguasa W}
    for H ← setiap elemen dalam
senaraiBunsetsu,
        mulai dari awal;
    begin
        if W dapat bergantung pada H then
        begin
            hubungkan W sebagai unsur-
            bergantung H;
            breakfor;
        end
    end

    if tidak ditemukan penguasa untuk W then
    begin
        tanpaPenguasa ← W + tanpaPenguasa;
    end

until semua bunsetsu telah diterima;
```

Perhatikan bahwa alih-alih menggunakan larik, algoritma ini menggunakan senarai. Daripada menelusuri larik dan menentukan elemen mana yang perlu dilewati, akan lebih mudah bekerja dengan senarai. Dengan senarai, elemen yang tidak perlu dapat dengan mudah dibuang.

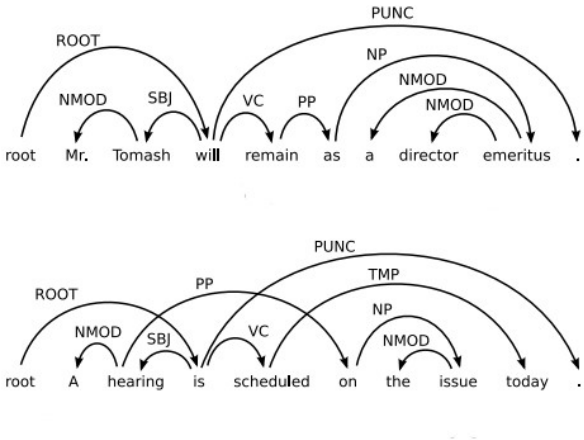
Algoritma di atas bekerja dengan dua senarai, `tanpaPenguasa` dan `senaraiBunsetsu` yang, secara berurutan, berisi *bunsetsu* yang belum memiliki penguasa dan *bunsetsu* yang telah diterima. Keduanya dibangun dengan menambahkan elemen di awal sehingga *bunsetsu* yang terbaru akan berada di bagian depan senarai. Pencarian pada kedua senarai tersebut akan menghasilkan *bunsetsu* yang terbaru terlebih dahulu.

C. Menerapkan Kendala Proyektivitas ke dalam Algoritma

Secara intuitif, proyektivitas didefinisikan sebagai "tidak ada cabang pohon yang saling tumpang tindih". Secara formal, proyektifitas didefinisikan sebagai berikut:

- Sebuah pohon proyektif jika dan hanya jika setiap kata di dalamnya membentuk sebuah substring yang kontinu.
- Sebuah kata membentuk substring kontinu jika dan hanya jika diberikan dua buah kata yang membentuknya, semua kata di antara kedua kata tersebut juga ikut membentuknya.

Syarat kedua adalah definisi dari substring kontinu. Dengan kata lain, sebuah substring kontinu adalah substring sedemikian rupa sehingga apapun yang ada di antara dua elemen yang membentuk substring tersebut juga termasuk bagian darinya[2].



Gambar 3: Struktur proyektif (atas) dan struktur non-proyektif (bawah)
(Sumber: http://languagelog.ldc.upenn.edu/myl/McDonaldSattaFig1.png)

Untuk mengaplikasikan syarat proyektivitas di atas, didefinisikanlah strategi berikut:

Strategi 4 (penanganan proyektivitas)

- *Ketika mencari unsur bergantung W, periksa setiap* bunsetsu *yang mendahului W dan belum memiliki penguasa atau hentikan pencarian.*
- *Ketika mencari penguasa W, hanya periksa* bunsetsu *yang berada di sebelah W, penguasanya, penguasa dari penguasanya, dan seterusnya hingga ke akar.*

Strategi 4 diimplementasikan ke dalam algoritma LS sehingga menghasilkan algoritma berikut:

Algoritma LSP (pencarian berbasis senarai dengan proyektivitas)

Diberikan sebuah kalimat dalam bentuk senarai kata dan dua senarai lain: tanpaPenguasa dan senaraiBunsetsu

```
{Inisialisasi}
tanpaPenguasa ← []; {bunsetsu yang belum
                     punya penguasa}
senaraiBunsetsu ← [];      {bunsetsu yang
                           telah diterima}

repeat
    {Terima sebuah bunsetsu dan tambahkan ke
    senaraiBunsetsu}
    W ← bunsetsu selanjutnya;
    senaraiBunsetsu ← W + senaraiBunsetsu;

    {Pencarian unsur-bergantung W}
    for D ← setiap elemen dalam tanpaPenguasa,
        mulai dari awal;
    begin
        if D dapat bergantung pada W then
        begin
            hubungkan D sebagai unsur-
            bergantung W;
            hapus D dari tanpaPenguasa;
        end else
        begin
            breakfor;
        end
    end

    {Pencarian penguasa W}
    H ← bunsetsu yang berada tepat di sebelah
        W dalam kalimat;
    loop
        if W dapat bergantung pada H then
        begin
            hubungkan W sebagai unsur-
            bergantung H;
            breakfor;
        end
        if H tidak punya penguasa then
        begin
            breakloop;
        end
        H ← penguasa H;
    endloop;

    if tidak ditemukan penguasa untuk W then
    begin
        tanpaPenguasa ← W + tanpaPenguasa;
    end

until semua bunsetsu telah diterima;
```

Algoritma LSP di atas telah menerapkan semua kendala struktur dependensi bahasa Jepang yang dapat diterapkan.

VI. KOMPLEKSITAS

Strategi *brute-force* yang telah dibahas pada bagian sebelumnya memiliki kompleksitas algoritma terburuk $O(n^2)$ karena, untuk kalimat dengan *n bunsetsu*, pencarian dilakukan terhadap $n(n-1)$ pasang *bunsetsu.* Semakin besar *n,* $n(n-1)$ akan mendekati n^2 . Jika diperhatikan, Algoritma LSP di atas

juga memiliki kompleksitas algoritma terburuk $O(n^2)$ dengan alasan serupa.

Penerapan kendala-kendala struktur dependensi bahasa Jepang pada Algoritma LSP memang tidak bertujuan untuk mengurangi kompleksitas algoritma terburuk. Akan tetapi, penerapan kendala-kendala tersebut bertujuan untuk menekan probabilitas terjadinya kasus terburuk. Penerapan kendala-kendala tersebut memastikan algoritma tidak akan melakukan pemerikasaan yang tidak diperlukan sehingga kemungkinan algoritma menemukan kasus terburuk sangatlah kecil.

UCAPAN TERIMA KASIH

Penulis mengucapkan rasa syukur kepada Allah SWT. yang atas rahmat dan kuasanya makalah ini bisa selesai tepat pada waktunya. Penulis mengucapkan terima kasih kepada Ibu Ulfa Maulidevi, Bapak Rinaldi Munir dan Ibu Masayu sebagai dosen pengampu mata kuliah IF2211 Strategi Algoritma atas bimbingannya. Penulis juga berterima kasih kepada orangtua, keluarga dan teman-teman yang selalu mendukung penulis melalui doa.

REFERENSI

[1] Al Khowarizmi, M. Iqbal, "Mengatasi kalimat ambigu dalam bahasa Jepang menggunakan pohon sintaks," tidak dipublikasikan.

[2] Covington, Michael A. "A fundamental algorithm for dependency parsing." Proceedings of the 39th annual ACM southeast conference. 2001.

[3] Covington, Michael A. "Parsing discontinuous constituents in dependency grammar." Computational linguistics 16.4 (1990): 234-236.

[4] Sassano, Manabu. "Linear-time dependency analysis for Japanese." Proceedings of the 20th international conference on Computational Linguistics. Association for Computational Linguistics, 2004.

[5] Iwatate, Masakazu, Masayuki Asahara, and Yuji Matsumoto. "Japanese dependency parsing using a tournament model." Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1. Association for Computational Linguistics, 2008.

[6] Kudo, Taku, and Yuji Matsumoto. "Japanese dependency analysis using cascaded chunking." proceedings of the 6th conference on Natural language learning-Volume 20. Association for Computational Linguistics, 2002.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Mei 2017

Muhammad Iqbal Al Khowarizmi
13515086