PENGEMBANGAN SISTEM PEMBENTUKAN *WORD GRAPH* UNTUK TEKS BERBAHASA INDONESIA

Sri Nurdiati, Deni Romadoni

Department Ilmu Komputer, Institut Pertanian Bogor

Jl. Meranti, Kampus IPB Darmaga, Bogor 16680, Indonesia

ABSTRACT

Knowledge graph is a new method which is used to describe and model natural language or human language. A word is a basic unit of natural language processing. Word graph is a graph which has connected structure of words. Ontology between word and token in word graph is represented by a node, 8 binary relationships, 4 frame relationships, and a focus. In this research, a system to construct a word graph from Indonesian text is developed. The software was initially developed by Mark van Koningsveld, which is called DelftCoStruct. On the previous version, knowledge graph method is not fully implemented. The first step in this research is analyzing the outline description and the requirement system. There are three principal requirements: formation and modification of word graph, analyzing text in word graph, and saving the dictionary capabilities. The second step in this research is designing specification system. The third step is developing and implementing the system designed. And the last step is validating system, to ensuring that all the requirements is fully implemented on this system.

Keywords : *natural language processing, knowledge graph, word graph, token, ontology, frame*

1. PENDAHULUAN

Berbagai penelitian dalam bidang komputer telah melahirkan metode atau teknologi baru yang dapat bermanfaat bagi kemajuan ilmu pengetahuan. Satu impian yang masih dalam angan-angan adalah penggunaan bahasa alami (bahasa manusia) untuk melakukan komunikasi dengan komputer atau mesin. Secara umum, *natural language* atau bahasa alami adalah metode dan sistem simbol yang paling banyak digunakan untuk mengekspresikan pikiran manuasia dan pertukaran informasi, merujuk pada Zhang [4].

Penelitian dalam bahasa alami melahirkan bidang ilmu *Natural Language Processing* (NLP). Tantangan terbesar dalam NLP adalah adanya *ambiguitas* dalam bahasa alami. Teks merupakan bahasa alami berupa tulisan. *Ambiguitas* di dalam teks pun banyak dijumpai, sehingga pemahaman terhadap teks tersebut dapat bersifat subjektif. Metode yang digunakan untuk memecahkan masalah *ambiguitas* dalam teks salah satunya adalah dengan metode "*Knowledge Graph*" (KG).

Secara alami, menggambarkan dan memodelkan bahasa alami adalah dasar untuk perkembangan dari proses menganalisis dan memaknai bahasa alami, dan menentukan arah proses penelitian dari bahasa alami. Ada dua faktor yang diperhatikan dalam menganalisis sebuah kalimat, yaitu sintaksis dan semantik. Perbedaan sintaksis dan semantik adalah sintaksis melakukan analisis berdasarkan bentuk dari sebuah kalimat, sedangkan semantik menganalisis bagaimana mengartikan suatu kalimat, Hulliyah [2].

Metode KG merupakan satu jenis dari representasi NLP, yang mengarahkan pada cara baru dalam menjelaskan dan memodelkan NLP, dan juga sebagai langkah besar ke depan untuk pemahaman terhadap semantik, menurut Zhang[4]. Dengan cara menganalisis teks diharapkan dapat dihasilkan sebuah pengetahuan baru. Berbagai penelitian KG yang dilakukan diharapkan mampu merancang suatu sistem yang dapat melakukan pembacaan terhadap sembarang dokumen yang diinginkan dan menginterpretasikan informasi yang didapat dalam bentuk *graph*. Penelitian yang akan dilakukan merupakan sebagian dari proses untuk mewujudkan harapan tersebut.

Penelitian yang akan dilakukan adalah mengembangkan perangkat lunak yang telah dibangun oleh Mark van Koningsveld pada tahun 2003-2008. Perangkat lunak tersebut bernama "DelftConStruct". DelftConStruct adalah *tools* yang menganalisis teks bahasa Inggris dan menampilkan visualisasi dalam bentuk *graph* dengan menggunakan metode KG. DelftConStruct dibangun dengan menggunakan bahasa pemrograman MATLAB.

DelftConStruct yang telah dibuat oleh Mark van Koningsveld mengimplementasikan sistem yang dapat menganalisis suatu kata dan membentuk suatu *graph* antara satu kata dengan kata lainnya (*word graph*). Namun pembentukan *word graph* tersebut belum sesuai dengan konsep KG. Sebagai contoh salah satu ketidaksesuaian konsep tersebut adalah tidak ada arah panah sebagai penunjuk relasi.

Analisis yang dilakukan DelftConStruct ini terbatas hanya pada teks berbahasa Inggris. Sistem ini juga belum mendukung proses modifikasi *edit* dan hapus pada elemen *graph* (verteks atau *edge*). Sistem juga belum mendukung penyimpanan *graph* dan pembentukan *graph* baru. Keterbatasan sistem ini memberikan banyak peluang untuk melengkapi fitur pada perangkat lunak ini. Sistem yang baru akan disebuat BogorDelftConStruct

2. TEORI *KNOWLEDGE GRAPH*

Pengertian KG

Menurut Zhang[4], teori KG adalah jenis sudut pandang baru, yang digunakan untuk menggambarkan bahasa manusia saat lebih memfokuskan pada aspek semantik daripada aspek sintaksis. KG mempunyai kemampuan lebih kuat untuk mengekspresikan dan menggambarkan lebih dalam *semantic layers*, untuk meminimumkan penggunaan *relation set* dan untuk menirukan pengertian dari jalan pikiran manusia.

KG sebagai bagian dari metode baru yang merepresentasikan pengetahuan, tergolong pada kategori *semantic network*. Dalam prinsipnya, KG tersusun dari *concept* dan *relationship*, merujuk pada Zhang[4]. *Concept* terdiri atas *token*, *type*, dan *name*. *relationship* terdiri atas *binary relationship* dan *multivariate relationship*.

Concept

Token adalah sebuah *node* dalam KG yang disimbolkan dengan persegi " □ ". *Token* mengekspresikan sesuatu dalam dunia nyata atau sebuah konsep dari dalam persepsi manusia, lihat Zhang[4]. *Token* bersifat subjektif, karena merupakan konsep yang dipahami oleh bahasa manusia menurut persepsi masing-masing. Contoh sebuah *token* menurut, lihat Rusiyamti[3], misalkan seseorang menemukan kata "apel", orang tersebut dapat menghubungkan hal ini dengan informasi bentuk, warna, rasa, dan sebagainya. Demikian juga orang lain akan menghubungkan dengan hal berbeda. Seseorang dalam mengamati sesuatu, pada kenyataannya akan dibandingkan dan dihubungkan dengan dunia nyata.

Type adalah suatu konsep yang masih bersifat umum dan merupakan hasil dari kesepakatan yang dibuat sebelumnya, lihat Rusiyamti[3]. Contoh *type* misalnya buah, binatang, dan sebagainya. Suatu konsep yang bersifat individual dikategorikan ke dalam sebuah *name*. Sebagai contoh fuji adalah sebuah *name* yaitu nama dari sebuah apel. *Type* dan *name* dibedakan oleh jenis relasi yang menghubungkannya dengan *token*.

2.1.1 *Relationship*

Untuk menggambarkan dunia nyata, dibutuhkan hubungan yang membedakan antar-*token*. Sebuah *relationship* antara 2 *concepts* "a" dan "b" adalah sebuah *graph* yang di dalamnya terdapat kedua *concepts* "a" dan "b" tersebut. Dalam teori KG, sangat diperlukan prinsip penggunaan kumpulan *relationship* yang sangat terbatas, lihat Zhang[4]. Dengan demikian sangat diperlukan *basic relationship* atau disebut juga "ontologi", untuk menghindari pertumbuhan *relationship* yang tidak terbatas dari *semantic network*. Menurut Zhang[4], hubungan (*relationship*) yang dibentuk pada *word graph* tersebut direpresentasikan dengan sebuah *node*, 8 *binary relationships*, dan 4 *frame relationships*. Hoede dan Nurdiati[1], menambahkan sebuah ontologi F, sebagai fokus token.

Word Graph

Kata adalah unit dasar dari NLP untuk mengekspresikan pikiran-pikiran manusia dan pertukaran informasi. Setiap *natural language* telah disusun ke dalam sebuah jenis sistem simbol tradisional, yang masing-masing mempunyai jenis-jenis kata, dan struktur kata yang di dalamnya terdapat perbedaan komponen *relationship* dan relasi komponen-komponen tersebut memainkan peran sebagai sebuah kesatuan yang mempunyai suatu makna.

Untuk dapat mengerti makna dari sebuah kalimat terlebih dahulu harus dimengerti makna dari setiap kata yang terdapat dalam kalimat tersebut dengan demikian dapat memperoleh makna dari keseluruhan kalimat. *Word graph* adalah *graph* dari sebuah kata atau frase kata. Pada akhirnya dengan menggabungkan *word graph* dapat diperoleh makna dari kesatuan teks.

Aspek Ontologi

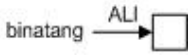
Ontologi adalah keterangan untuk mengambarkan beberapa konsep dan relasi-relasi di antaranya, dengan maksud untuk memberikan definisi yang cukup terhadap ide-ide yang dituangkan dengan komputer untuk merepresentasikan ide-ide tersebut dan logikanya, lihat Hulliyah[2]. Berdasarkan ontologi yang dimilki inilah maka KG dapat membangun sebuah model yang dapat digunakan untuk memahami bahasa alami. Hal ini diperlukan agar arti dari suatu kalimat dapat diekspresikan. Arti dari kata harus terlebih dahulu diketahui untuk dapat mengartikan sebuah kalimat.

Ontologi *word graph* sampai saat ini direpresentasikan dengan sebuah *node*, 8 *binary*

relationships, sebuah ontologi F, dan 4 *frame relationships*. Berikut ini adalah gambaran dari 8 *types relationship* menurut Zhang[4]:

1. ALI (*alikeness*)

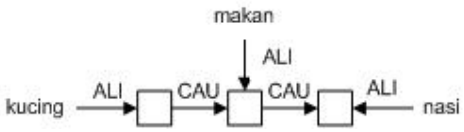
Relasi ALI digunakan di antara *type* dan *token* atau antara 2 *token* yang memiliki unsur-unsur yang sama. Contoh : "binatang adalah *type*", maka dinyatakan seperti pada Gambar 1.



Gambar 1 Contoh penggunaan relasi ALI.

2. CAU (*causality*)

Relasi CAU mengekspresikan hubungan antara sebab dan akibat, atau sesuatu hal mempengaruhi sesuatu yang lain. Contoh : "kucing makan nasi". Representasi *word graph* untuk kalimat tersebut dapat dilihat pada Gambar 2.



Gambar 2 Contoh penggunaan relasi CAU.

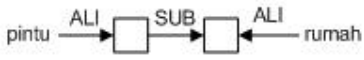
3. EQU (*equality*)

Relasi ini digunakan di antara *name* dan *token* atau antara 2 *token* yang mengekspresikan keduanya adalah sama dan sederajat. Relasi EQU juga digunakan untuk menyatakan kata hubung seperti "adalah" dan "merupakan". Contoh : "Fuji adalah *name* dari apel". Gambar 3 adalah contoh dari penggunaan relasi EQU.

Gambar 3 Contoh penggunaan relasi EQU.

4. SUB (*subset*)

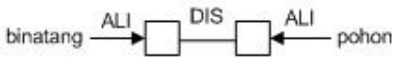
Bila terdapat 2 *token* yang mengekspresikan dua rangkaian secara berurutan, dan 1 *token* adalah bagian dari *token* yang lainnya, maka di antara kedua *token* tersebut terdapat relasi SUB. Contoh : "pintu adalah bagian dari rumah". Gambar 4 merupakan representasi kalimat tersebut.



Gambar 4 Contoh penggunaan relasi SUB.

5. DIS (*dissparatness*)

Relasi DIS digunakan untuk mengekspresikan bahwa 2 *token* tidak memiliki hubungan satu dengan yang lainnya. Contoh relasi DIS digunakan juga untuk menunjukkan kata "berbeda", misalnya "binatang berbeda dengan pohon", dinyatakan seperti pada Gambar 5.



Gambar 5 Contoh penggunaan relasi DIS.

6. ORD (*ordering*)

Relasi ORD mengekspresikan bahwa 2 hal mempunyai urutan satu sama lain. Contoh penggunaan relasi ORD untuk menyatakan "pagi sebelum sore", ditunjukkan oleh Gambar 6.

Gambar 6 Contoh penggunaan relasi ORD.

7. PAR (*attribute*)

Relasi PAR mengekspresikan bahwa sesuatu adalah sebuah atribut dari sesuatu yang lain. Contohnya untuk menyatakan frase "buku baru", dinyatakan seperti pada Gambar 7.

Gambar 7 Contoh penggunaan relasi PAR.

8. SKO (*informational dependency*)

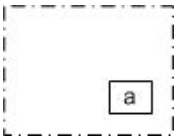
Relasi SKO digunakan, jika suatu *token* informasinya bergantung pada *token* yang lainnya. Contohnya digunakan untuk menyatakan "harga bergantung kualitas", dapat dinyatakan seperti pada Gambar 8.

Gambar 8 Contoh penggunaan relasi SKO.

Sebuah *frame* adalah sebuah *node* yang diberikan label lihat Zhang[4]. Adapun untuk 4 *frame relationships* dapat dijelaskan sebagai berikut:

1. FPAR : *Focusing on a situation*

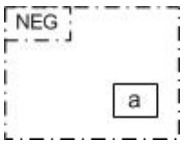
FPAR mengekspresikan bahwa sekumpulan *subgraph* dari *graph* adalah bagian dari seluruh *graph* yang telah dibentuk. Misalkan *a* adalah preposisi yang menyatakan "adik bahagia", maka bisa direpresentasikan dengan *frame* FPAR seperti yang ditunjukkan oleh Gambar 9.



Gambar 9 Contoh penggunaan *frame* FPAR.

2. NEGPAR : *Negation of a situation*

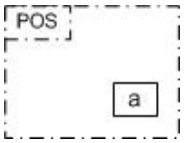
NEGPAR mengekspresikan peniadaan atau pengingkaran terjadinya isi dari *frame*. Misalkan untuk menyatakan pengingkaran terhadap preposisi *a*, maka dapat dilihat pada Gambar 10. Gambar tersebut menunjukkan bahwa “tidak benar adik bahagia”.



Gambar 10 Contoh penggunaan *frame* NEGPAR.

3. POSPAR : *Possibility of a situation*

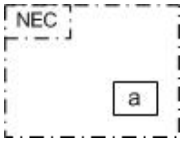
POSPAR mengekspresikan kemungkinan terjadi dari isi *frame*. Misalkan untuk menyatakan kemungkinan terjadinya peristiwa pada preposisi *a*, digunakan sebuah *frame relationship* POSPAR, seperti ditunjukkan oleh Gambar 11. Gambar tersebut berarti menyatakan “mungkin saja adik bahagia”.



Gambar 11 Contoh penggunaan *frame* POSPAR.

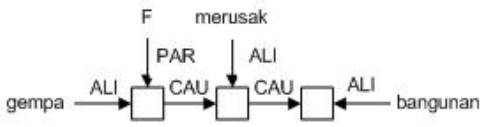
4. NECPAR : *Necessity of a situation*

NECPAR mengekspresikan perlu, butuh atau keharusan terjadi dari isi *frame*. Misalkan kalimat “seharusnya adik bahagia” direpresentasikan oleh *frame* NECPAR terhadap preposisi *a*, maka dapat dinyatakan seperti pada Gambar 12.



Gambar 12 Contoh penggunaan *frame* NECPAR.

Ontologi F digunakan untuk menunjukkan fokus dari suatu *graph*. Penggunaan ontologi ini, misalnya untuk menyatakan *word graph* “gempa merusak bangunan”, dapat dinyatakan pada Gambar 13. Gambar tersebut menunjukkan bahwa *token* gempa menjadi fokus dari *graph* tersebut.



Gambar 13 Contoh penggunaan ontologi F.

3. HASIL DAN PEMBAHASAN

Deskripsi Umum Sistem

BogorDelftConStruct merupakan perangkat lunak berbasis *desktop* yang dikembangkan dengan menggunakan bahasa pemograman MATLAB. BogorDelftConStruct adalah sebuah *tools* yang digunakan untuk pembentukan *word graph* dan melakukan analisis terhadap teks menggunakan metode KG. Pada awalnya, DelftConStruct dikembangkan untuk teks berbahasa Inggris. Pada penelitian kali ini BogorDelftConStruct diimplementasikan dalam teks berbahasa Indonesia.

Pengguna BogorDelftConStruct adalah mereka yang memahami konsep KG. Hal tersebut dikarenakan BogorDelftConStruct merupakan sistem pembentukan *word graph*, sehingga pengguna hendaknya mempunyai pengetahuan untuk membentuk *word graph* sesuai konsep KG.

Deskripsi Batasan Sistem

BogorDelftConStruct dikembangkan dengan berbagai fitur tambahan yang mendukung terbentuknya *word graph* yang sesuai dengan konsep KG. Beberapa perintah mungkin saja tidak sesuai dengan konsep KG, sehingga sistem akan menolak proses untuk perintah tersebut. Berikut ini dijelaskan batasan-batasan sistem dan beberapa kondisi yang akan ditolak sistem :

a. Sistem tidak membedakan huruf kapital atau bukan.

b. Pembuatan relasi antar teks tidak dapat dilakukan pada BogorDelftConStruct.

c. Pembentukan relasi ontologi F hanya diperkenankan untuk sebuah *token* saja dalam sebuah *word graph.*

d. Sistem akan menolak penyimpanan sebuah *word graph* menjadi kamus *word graph*, apabila masih terdapat sebuah *token* atau lebih yang belum terhubung dalam *graph*.

e. Sistem akan menolak penyimpanan sebuah *word graph* apabila ada minimal sebuah kata yang yang belum terhubung dengan minimal sebuah *token*.

Deskripsi Proses Sistem

Keadaan *default* BogorDelftConStruct hasil pengembangan pada saat dibuka adalah sebuah lembar kerja (*workspace*) dengan sebuah *token* telah tergambar dan siap untuk dibentuk menjadi sebuah *word graph*. Operasi modifikasi terhadap *token* tersebut dapat dilakukan melalui *form add relation* yang tersedia, maupun dengan cara klik kanan *mouse* pada *token* tersebut, maka menu untuk modifikasi *token* dapat dipilih. Proses penambahan, hapus *token* dan lain-lain dapat dilakukan dengan klik kanan, maupun melalui *menubar*. Operasi modifikasi terhadap teks maupun relasi dapat pula dilakukan dengan cara yang sama.

Analisis teks dilakukan dengan membaca setiap kata yang terbentuk pada *word graph*, sehingga makna dari kalimat yang dimaksud dapat tersampaikan melalui *word graph* tersebut. Hasil analisis terhadap *word graph* dapat diketahui ketika salah satu *token* diklik, maka keterangan makna *relationship token* tersebut dengan *token* yang lainnya dapat dilihat pada panel *relationship* yang berada di sebelah kiri jendela aplikasi. Pengguna dapat mengatur kedalaman level *graph* yang akan dianalisis pada panel tersebut. Hasil analisis terhadap *word graph* yang ditampilkan merupakan makna dari *inward relationship* dan *outward relationship*.

Setelah *word graph* berhasil dibentuk, maka dapat disimpan dan ditambahkan ke dalam kamus *word graph*. *File-file* yang telah tersimpan dalam kamus dapat dilihat di panel *dictionary*. Dalam panel tersebut dapat dilakukan modifikasi langsung terhadap topik-topik pada kamus *word graph*. Modifikasi tersebut dapat pula dilakukan melalui *menubar*.

Pada saat keluar dari sistem BogorDelftConStruct, maka sistem otomatis meyimpan hasil kerja terakhir, dan menampilkannya pada saat BogorDelftConStruct dijalankan kembali suatu waktu.

Spesifikasi dan Perancangan Fungsional

Tiga modul yang dibutuhkan sebagai sebuah perancangan fungsional sistem memiliki fungsi-fungsi sebagai berikut:

1. Modul pembentukan dan modifikasi *word graph*

Berdasarkan konsep KG, *word graph* yang dibentuk mempunyai sebuah *edge* sebagai relasi yang menghubungkan antar-*token*, dan juga menghubungkan antara *token* dengan kata. Berikut ini adalah fungsi-fungsi yang digunakan untuk melakukan penambahan 8 *binary relationships* dan 4 *frame relationships* pada *graph*:

a. Penambahan dan modifikasi relasi pada *token*
b. Penambahan dan modifikasi relasi pada teks
c. Penambahan dan modifikasi relasi pada *frame*

Terdapat satu fungsi yang berlaku untuk semua relasi yang menghubungkan setiap elemen pada *word graph*, baik *token*, teks, maupun *frame* yaitu fungsi untuk mengubah ontologi dari setiap relasi yang berhasil dibentuk pada *word graph*.

2. Modul analisis *graph*

Analisis *word graph* dilakukan dengan membaca elemen-elemen pada *word graph* yang mempunyai relasi ontologi. Analisis *word graph* meliputi :

a. Analisis suatu *token* yang mempunyai relasi ontologi terhadap setiap kata pada *token* tersebut.
b. Analisis antar-*token* yang mempunyai relasi ontologi tertentu.
c. Analisis tehadap *frame* dengan *relationship frame* tertentu, yang mempunyai sejumlah anggota *token* di dalamnya.
d. Analisis suatu label dengan relasi ontologi tertentu terhadap *frame*.
e. Analisis antara *frame* dengan *token* yang terhubung padanya.
f. Analisis antar-*frame* yang mempunyai relasi ontologi tertentu.
g. Kedalaman level *graph* yang dianalisis dapat ditentukan oleh pengguna melalui *form input* pada panel *relationship*.

Hasil analisis *graph* memberikan makna dari *word graph* yang terbentuk. Hasil analisis tersebut ditampilkan dalam panel *relationship* yang berada di samping kiri. Selain itu dapat pula dilakukan *double* klik pada *token* maupun pada *frame* yang dimaksud untuk ditampilkan hasil analisis pada jendela dialog.

3. Modul pembentukan dan modifikasi kamus *word graph*

Modul pembentukan kamus *word graph* membutuhkan fungsi-fungsi pendukung sebagai berikut :

a. Mengganti *workspace* baru.
b. Menyimpan *word graph*.
c. Menampilkan dan mengubah *word graph* pada kamus *word graph*.
d. Mengubah nama *file* pada kamus *word graph*.
e. Menyalin *file* pada kamus *word graph.*
f. Menghapus *file* pada kamus *word graph*.

Spesifikasi dan Perancangan Struktur Data

Token, kata, *frame*, dan semua elemen pada *graph* yang terbentuk dalam sebuah *word graph* menyimpan titik-titik koordinat masing-masing elemen. Melalui koordinat tersebut, maka semua *token*, kata, dan *frame* dapat dihubungkan dengan sebuah relasi yang menghubungkan antar-koordinat tersebut. Relasi yang digunakan pun meyimpan data sebuah *id* tipe ontologi untuk setiap koordinat yang dihubungkan. *Graph* dapat dianalisis dengan membaca tiap kata yang terhubung, sehingga data pada sistem ini juga berisi tentang kata yang terdapat pada *word graph*. Stuktur data pada BogorDelftConStruct yang dikembangkan ini dapat dijelaskan sebagai berikut:

a. Data mengenai *token*
   - Info *token* bertipe *cell*

- *Line* dan *arrow* bertipe *double*
- Ontologi antar-*token* dan label ontologi bertipe *double*

b. Data mengenai teks
- Info teks bertipe *cell*
- *Line* dan *arrow* bertipe *double*
- Ontologi antara teks dan *token* dan label ontologi bertipe *double*

c. Data *frame*
- Info *frame* bertipe *cell*
- *Line* dan *arrow* bertipe *double*
- Ontologi antar *frame* dan label ontologi bertipe *double*
- Ontologi antara *frame* dan token, serta label ontologi bertipe *double*

d. Data Relasi Ontologi

Data mengenai semua hal menyangkut ontologi, baik itu tipe ontologi, maupun makna dari ontologi itu sendiri disimpan dalam dua buah *file* dan fungsi *getOntology* dan *getFrameOntology*. Fungsi *getOntology* menyimpan data 8 *binary relationships*, dan fungsi *getFrameOntology* menyimpan data 4 *frame relationships*. Fungsi-fungsi inilah yang kemudian dipakai sebagai penanda tipe ontologi pada tiap relasi *token*, teks, dan *frame*. Analisis teks juga diambil dari makna tiap ontologi yang tersimpan dalam fungsi-fungsi ini.

Semua variabel yang menyimpan data *token*, data teks, data garis dan data panah disimpan dalam variabel bertipe *struct*. Variabel tersebut diberi nama "*session*". Variabel *session* inilah yang setiap saat digunakan untuk menggambar *word graph*. Begitu pula pada saat penyimpanan *word graph*, variabel *session* inilah yang disimpan ke dalam sebuah *file* sebagai data dari *word graph*. Namun khusus pada saat penyimpanan *word graph*, variabel yang menyimpan data garis dan panah dihapus, dan dibangkitkan kembali ketika *word graph* ditampilkan atau dilakukan proses *edit*.

Implementasi Fungsional

Implementasi fungsional BogorDelftConStruct adalah sebagai berikut:

1. Implementasi modul pembentukan dan modifikasi *word graph*

Fungsi untuk pembentukan dan modifikasi *graph* disediakan pada *toolbar*, *menubar* dan pada panel *add relation* yang ada pada jendela aplikasi sebelah kiri. Panel *add relation* dapat dilihat pada Gambar 14.

Fungsi penambahan relasi antar-*token* maupun relasi antara *token* dengan teks dapat dilakukan dengan mengakses menu dari *form add relation* atau dengan melakukan klik kanan pada *token* atau kata yang diinginkan. Bila fungsi penambahan relasi tersebut dilakukan dari *form add relation*, maka pengguna mengisi *form popup* menu, dengan memilih *token* atau teks mana yang akan ditambahkan relasi, dan *token* mana yang akan menjadi tujuan dari relasi tersebut, kemudian memilih tipe relasi ontologi yang akan ditambahkan. Setelah tombol "*add relation*" ditekan, maka akan muncul jendela dialog untuk memasukkan data dari penambahan relasi tersebut.

Bila fungsi penambahan relasi diakses melalui klik kanan, maka pengguna dapat langsung memilih menu yang dimaksud, setelah itu memasukkan data pada jendela dialog yang muncul untuk penambahan relasi. Fungsi yang menangani penambahan semua kemungkinana relasi ontologi antara *token* dan teks dilakukan oleh fungsi-fungsi pendukung seperti fungsi *addRelationToken* dan *addRelationTeks*.

Pembuatan *frame relationship* dapat dilakukan melalui *menubar* maupun klik kanan pada *token*. Fungsi yang digunakan adalah fungsi *makeFrameOntology*.

Fungsi modifikasi *word graph*, baik proses *edit* maupun hapus dan fungsi-fungsi modifikasi lainnya dapat dilakukan dari menu klik kanan pada *token*, kata, *frame* maupun relasi yang akan dimodifikasi. Fungsi modifikasi tersebut juga disediakan di dalam *menubar*. Setelah menu dari fungsi ini dipilih, maka akan muncul jendela dialog untuk memasukkan informasi tentang *token*, teks, *frame* maupun relasi yang akan dimodifikasi tersebut.

2. Implementasi modul analisis *word graph*

Dengan melakukan klik terhadap *token* atau *frame* pada *graph*, maka sistem secara otomatis menganalisis *token* atau *frame* yang dipilih, terhadap *token* dan *frame* lainnya.

Sistem akan melakukan analisis terhadap *graph* dengan membaca relasi ontologi tiap kata yang terhubung pada sebuah *token* yang dipilih. Analisis terhadap tiap kata pada *token* lainnya pun dilakukan, sehingga akan didapatkan makna untuk setiap *token* pada *word graph* tersebut. Selanjutnya analisis relasi ontologi dilanjutkan terhadap *token* yang terhubung langsung dengan *token* yang dipilih tersebut, dan dinyatakan sebagai analisis level 1. *Token* lainnya yang tidak terhubung langsung dengan *token* yang dipilih dinyatakan sebagai analisis *token* pada level 2, level 3 dan seterusnya. Pengguna dapat menentukan kedalaman level *graph* yang akan dianalisis, kedalaman *default* pada sistem sebanyak 3 level. Dengan demikian akan didapatkan sebuah makna dari *word graph* tersebut atas analisis terhadap semua *token* pada berbagai level *token*.

Hal yang sama pun dilakukan pada analisis sebuah *frame*. Pada saat dilakukan *double* klik sebuah *frame*, maka sistem akan melakukan analisis terhadap semua anggota *token* yang terdapat dalam *frame* tersebut.

Setelah mendapatkan hasil analisis dari semua *token* maka sistem akan membaca ontologi *frame relationship* tersebut terhadap hasil analisis tersebut. Kemudian dilanjutkan dengan membaca label pada *frame* (jika terdapat label pada *frame* tersebut). Hasil analisis dapat dilihat pada panel *relationship*. Gambar 16 menunjukkan *form relationship*

3. Implementasi modul pembentukan dan modifikasi kamus *word graph*

Fungsi untuk pembentukan dan modifikasi kamus *word graph* disediakan pada *toolbar*, *menubar* dan pada panel *dictionary* yang ada pada jendela aplikasi sebelah kiri. Dengan penempatan posisi menu tersebut, harapannya dapat memudahkan pengguna mengakses dan menggunakannya. Gambar 16 menunjukkan gambar dari panel *dictionary*.

Pada *toolbar* disediakan jalan pintas untuk menu pembuatan *workspace* baru (fungsi *new graph*) dan menu untuk menyimpan *graph* menjadi kamus *word graph*. Pada saat pengguna menekan *icon new graph* maka muncul dialog konfirmasi untuk memastikan bahwa pengguna menginginkan *workspace* baru dan secara otomatis menghapus *graph* yang ada pada *workspace* pada saat itu.

Pada panel *dictionary* yang dapat dilakukan terhadap kamus *word graph* yang terdapat disana adalah proses modifikasi, yaitu *rename file*, *view and edit*, *copy*, dan hapus *file*. Menu-menu modifikasi pada panel tersebut dapat diakses setelah pengguna memilih (klik kiri) pada nama *file* di dalam kamus tersebut, kemudian melakukan klik kanan pada nama *file* yang telah dipilih tadi. Untuk beberapa fungsi yaitu *rename* dan hapus, jika menu ini diakses maka jendela dialog konfirmasi akan muncul untuk mengonfirmasi perintah yang diinginkan pengguna. Hal tersebut untuk memastikan bahwa pengguna akan melakukan modifikasi terhadap kamus *word graph*.

Pada *menubar*, yaitu pada menu *workspace* dan menu *dictionary*, semua fungsi dari modul pembentukan dan modifikasi kamus *word graph* dapat diakses.

Sebagai catatan, bahwa setelah melakukan salah satu fungsi dari modul ini kecuali fungsi "*view and edit graph*", maka disarankan untuk mengakses menu "*restart system*" pada *menubar workspace*. Hal tersebut dimaksudkan untuk melihat hasil dari proses yang telah dikerjakan oleh modul ini.

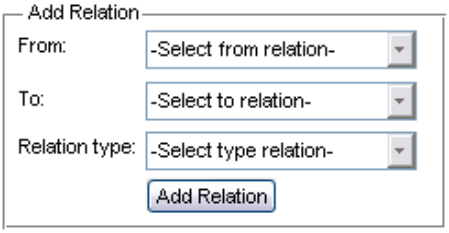
Implementasi Struktur Data

Setiap proses yang dilakukan pada BogorDelftConStruct selalu berhubungan dengan data yang disimpan dalam variabel *session* bertipe *struct*. Hal tersebut dikarenakan penggambaran *word graph* dalam *axis* selalu menggunakan titik koordinat. Pergerakan suatu elemen *word graph* sekecil apapun selalu mengubah koordinat seluruh elemen pada *word graph* tersebut. Variabel yang bersifat dinamis ini menyebabkan *token* dan teks dapat dengan bebas digerak-gerakkan atau ditarik ke sana kemari, dan relasi yang menghubungkannya bisa mengikuti pergerakan tersebut.

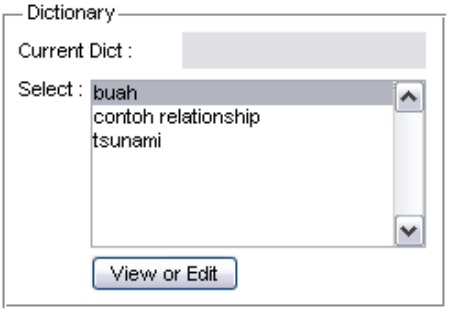
Pada saat sistem melakukan penambahan atau modifikasi terhadap *word graph* yang ada pada *workspace*, maka koordinat dari elemen *graph* tersebut berubah. Namun ada perbedaan dengan pada saat *graph* digerak-gerakkan. Sebelum dilakukan modifikasi, sistem terlebih dahulu menghapus isi dari variabel yang menangani data garis dan panah. Kemudian setelah dilakukan modifikasi, maka data teks, *token*, dan *frame* digunakan untuk menggambar dan membangkitkan ulang data garis dan panah yang menghubungkannya. Dengan demikian proses memperbaharui koordinat setiap elemen selalu dilakukan ketika ada proses menggambar *graph* (*plot* pada *axis*).

Implementasi Antarmuka

Ada beberapa perbedaan antarmuka BogorDelftConStruct dengan antarmuka DelftConStruct. Pada antarmuka BogorDelftConStruct, semua panel ditempatkan di sebelah kiri jendela aplikasi. Panel-panel yang disediakan yaitu, panel *form add relation*, panel *dictionary*, dan panel *relationship*. Gambar ketiga panel tersebut dapat dilihat pada Gambar 14, Gambar 15, dan Gambar 16. Ada penambahan panel *dictionary*, dan perbedaan *form add relation* dengan DelftConStruct. *Form add relation* DelftConStruct terdiri atas *textfield* dan *popup menu*, sedangkan *form add relation* BogorDelftConStruct semuanya merupakan *popup menu*. Gambar menubar dan *toolbar* dapat dilihat pada Gambar 17.

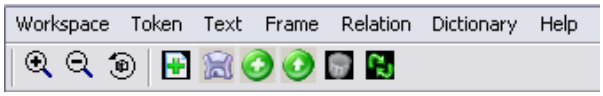


Gambar 14 Panel *add relation*.



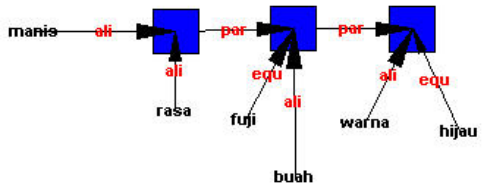
Gambar 15 Panel *dictionary*.

Gambar 16 Panel *relationship*.



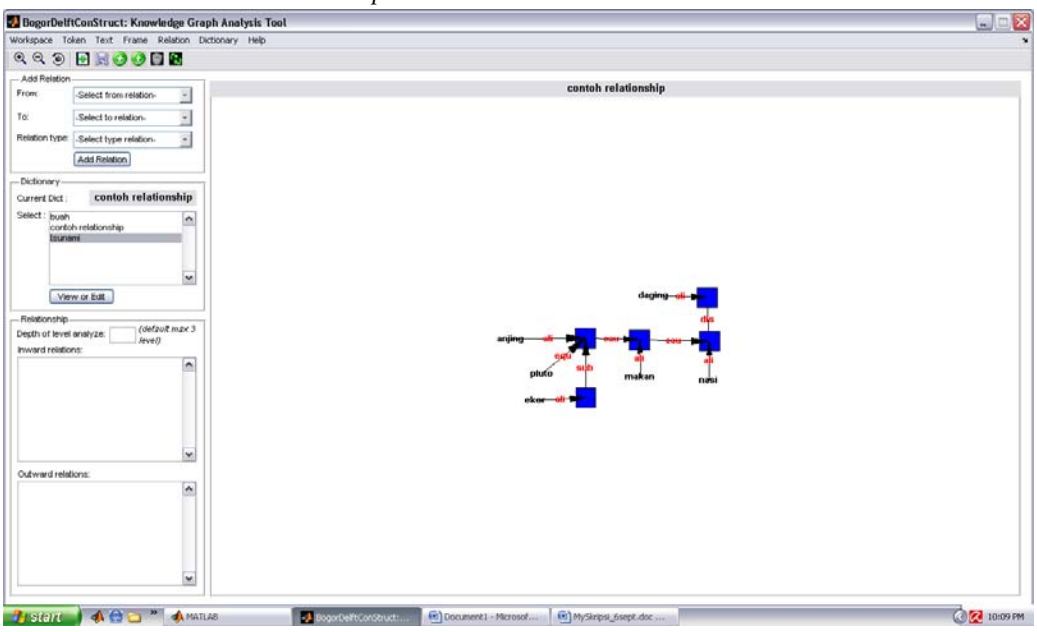
Gambar 17 *Menubar* dan *toolbar* BogorDelftConStruct

Workspace untuk membuat *graph* berada di sebagian besar jendela aplikasi BogorDeldtConStruct. Gambar 18 memperlihatkan contoh sebuah *word graph* yang berhasil dibentuk pada *workspace*.



Gambar 18 Contoh *word graph* yang dibentuk dalam *workspace*.

Gambar seluruh antarmuka BogorDelftConStruct dapat dilihat pada Gambar 19.



Gambar 19 Antarmuka BogorDelftConStruct

Validasi

Pada proses validasi ini, dilakukan proses integrasi unit fungsi dan subfungsi yang telah dibuat ke dalam sistem BogorDelftConStruct, dengan mendefinisikan fungsi ”*callback*” yang tepat untuk setiap menu dari fungsi-fungsi tersebut. Fungsi-fungsi tersebut merupakan fungsi yang sesuai dengan spesifikasi sistem itu sendiri.

Pengujian yang dilakukan menggunakan metode *black box*. Hasil pengujian menunjukkan kesesuaian antara hasil yang seharusnya dengan hasil pengujian.

Kelebihan dari sistem ini yaitu dapat membentuk dan melakukan modifikasi *word graph* sesuai dengan konsep KG, dan dapat melakukan penyimpanan terhadap *word graph* yang telah dibentuk. Sistem ini belum dilengkapi dengan beberapa modul, sehingga menjadi suatu kekurangan sistem, yaitu modul untuk membuka dan menutup sebuah *frame*, modul pembangkitan *word graph* dari masukan berupa sebuah frase atau kalimat, dan modul untuk penggabungan *word graph* dari *file* yang ada pada kamus *word graph*.

4. KESIMPULAN

BogorDelftConStruct yang dikembangkan dalam penelitian ini merupakan aplikasi yang telah sesuai dengan konsep KG. Proses penambahan dan modifikasi pada setiap elemen *word graph* telah dapat dilakukan dan sesuai konsep KG. Analisis *word graph* terhadap teks berbahasa Indonesia yang dilakukan oleh sistem ini telah cukup relevan dengan makna teks yang dimaksud. *Word graph* yang telah dibentuk dapat disimpan, sehingga dapat terbentuk kumpulan *word graph* yang disebut kamus *word graph*.

REFERENSI

[1] Hoede C, Nurdiati S. 2008. A Graph Theoritical Analysis of Certain Aspects of Bahasa Indonesia. *Memorandum No. 1870*, Departement of Applied Mathematics, Universuty of Twente, Enschede, The Netherlands, ISSN 1874-4850.

[2] Hulliyah K. 2007. *Rekayasa Memahami Teks Menggunakan Metode Knowledge Graph*. [tesis]. Bogor : Sekolah Pascasarjana, Institut Pertanian Bogor.

[3] Rusiyamti. 2008. *Analisis Teks Berbahasa Indonesia Mengunakan Teori Knowledge Graph*. [tesis]. Bogor: Sekolah Pascasarjana, Institut Pertanian Bogor.

[4] Zhang L. 2002. *Knowledge Graph Theory And Structural Parsing*. [Disertasi]. ISBN 9036518350. Netherlands : Twente University.