24607/H/06

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

APLIKASI PENERJEMAH BAHASA SATU ARAH MENGGUNAKAN ARSITEKTUR DIRECT-TRANSFORMER

STUDI KASUS PENERJEMAHAN BAHASA INGGRIS KEDALAM BAHASA INDONESIA

TUGAS AKHIR

R SIF
005.45
Yud
a-1
2005

Disusun Oleh :

TAUFAN ANANTA YUDHA

NRP. 5100 100 046

PERPUSTAKAAN
ITS

Tgl. Terima	6-8-2005
Terima Dari	H
No. Agenda Prp.	222986

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2005

APLIKASI PENERJEMAH BAHASA SATU ARAH MENGGUNAKAN ARSITEKTUR DIRECT-TRANSFORMER
STUDI KASUS PENERJEMAHAN BAHASA INGGRIS KEDALAM BAHASA INDONESIA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Yudhi Purwananto S.Kom, M.Kom.	**Imam Kuswardayan S.Kom.**
NIP. 132.172.210	**NIP. 132 306.543**

SURABAYA

JULI, 2005

ABSTRAK

Sebuah sistem aplikasi penerjemahan dibutuhkan dalam berbagai macam kepentingan terutama bidang pendidikan, ekonomi, bisnis, media massa, hukum, teknologi, dan lain sebagainya. Dengan adanya sistem penerjemahan otomatis menggunakan komputer sebagai medianya, maka dapat dicapai tingkat produktifitas dan efisiensi yang tinggi di segala macam aktifitas yang berhubungan dengan penerjemahan bahasa seperti dokumentasi, korespondensi, pencarian referensi hingga alih teknologi.

Salah satu metode konservatif yang tergolong lama namun masih populer untuk diterapkan dalam bidang *Machine Translation* ini hingga saat ini adalah metode *direct-transformer*. Pada metode tersebut sebuah teks asal diubah struktur gramatikalnya secara langsung kedalam struktur gramatikal teks tujuan berdasarkan aturan-aturan pembentukan atau transformasi yang telah ditentukan terlebih dahulu. Dengan metode ini aturan gramatikal dan pengetahuan linguistik suatu bahasa yang terlibat dalam algoritma sistem bukan menjadi prioritas yang utama, melainkan aturan-aturan pembentukan dan perubahan struktur kalimat.

Kelebihan yang diperoleh dengan menerapkan metode ini adalah aturan-aturan pembentukan atau transformasi bersifat dinamis, dapat ditambahkan dan diubah secara langsung oleh user dari dalam aplikasi. Masing-masing aturan kemudian dianalisa dan diproses oleh sistem untuk diterapkan pada struktur bahasa asal sehingga membentuk sebuah struktur baru yang merupakan representasi dari struktur bahasa tujuan. Kelemahan daripada metode ini adalah aturan-aturan pembentukan boleh tidak harus sesuai dengan aturan tata bahasa sesungguhnya, sehingga dapat saja dihasilkan teks keluaran yang berisi bahasa campuran dan tidak dapat dimengerti artinya. Namun hal tersebut dapat diatasi dengan menambahkan dan atau menyesuaikan aturan-aturan tertentu hingga hasil yang diperoleh menjadi lebih mendekati aturan tata bahasa yang sesungguhnya.

Kata Kunci:
Machine Translation, *Computer Aided Translation*, *PHP MySQL*

KATA PENGANTAR

Bismillahirahmanirrahim,

Puji dan syukur penulis panjatkan kepada Allah swt karena atas nikmat, rahmat dan hidayah-Nya penulis dapat menyelesaikan Buku Tugas Akhir ini yang sekaligus menandai berakhirnya masa pendidikan kesarjanaan strata satu pada Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya yang telah dijalani oleh penulis dalam jangka waktu lima tahun terakhir ini.

Pada Tugas Akhir ini penulis berkesempatan untuk melakukan perancangan dan pembangunan suatu sistem aplikasi penerjemahan antar bahasa, menggunakan metode parsing dan analisa aturan dinamis dengan contoh kasus Bahasa Inggris sebagai bahasa asal dan Bahasa Indonesia sebagai bahasa tujuan yang, insyaAllah, telah berjalan sesuai dengan yang telah direncanakan sebelumnya meskipun banyak yang masih perlu untuk diperbaiki dan disempurnakan lebih lanjut.

Untuk itu penulis ingin menyampaikan ucapan terima kasih dan rasa hormat kepada orang-orang berikut ini, baik yang disebutkan ataupun yang tersirat didalamnya, atas partisipasi, kebaikan dan bantuannya selama penulis mengerjakan Tugas Akhir ini dan seluruh masa penulis selama di berada di kota pahlawan yang panas ini :

- Kedua orang tua dan saudara penulis, untuk doa dan segala dukungannya yang tidak akan dapat penulis balas seumur hidup serta seluruh keluarga besar yang selalu memberikan semangat dan dorongan kepada penulis selama ini.

- Prof. Dr. Ir. Arif Djunaidy, M.Sc. selaku Dekan Fakultas Teknologi Informasi, ITS.
- Bapak Yudhi Purwananto, S.Kom. M.Kom. selaku Ketua Jurusan Teknik Informatika sekaligus dosen pembimbing Tugas Akhir penulis.
- Bapak Imam Kuswardayan S.Kom. selaku dosen pengajar dan pembimbing Tugas Akhir penulis.
- Seluruh kawan-kawan angkatan 2000 yang telah bersama-sama dalam suka dan duka serta warga Teknik Informatika ITS dan teman-teman jurusan lain yang memberikan warna yang beragam di kampus ini.
- Seluruh dosen dan karyawan Fakultas Teknologi Informasi pada khususnya dan ITS pada umumnya untuk semua pelayanan dan pengabdiannya di institusi ini.
- Hendra, Haris, Domplenk, Mahendra, Satria, Aang, dewa-dewa M3: Adhie, Dhika, Doni, Farid, Rusdi, Indra, Wempy, Rona dan semua sahabat yang telah menemani penulis dalam hari-hari pengungsiannya.

Penulis menyadari masih banyak kekurangan dan kesalahan dalam penyusunan buku ini sehingga segala saran dan kritik sangat diharapkan namun besar harapan penulis bahwa buku ini akan dapat bermanfaat bagi penulis sendiri dan para pembaca sekalian pada umumnya.

Terima kasih.

Surabaya, Juli 2005

Taufan Ananta Yudha

DAFTAR ISI

ABSTRAK ..... **I**

KATA PENGANTAR ..... **II**

DAFTAR ISI ..... **IV**

DAFTAR GAMBAR ..... **VI**

DAFTAR TABEL ..... **VII**

BAB 1 PENDAHULUAN **1**

1.1. LATAR BELAKANG 1

1.2. TUJUAN DAN MANFAAT 2

1.3. PERUMUSAN MASALAH 3

1.4. BATASAN MASALAH 3

1.5. METODOLOGI 4

1.6. SISTEMATIKA PENULISAN 6

BAB 2 DASAR TEORI **8**

2.1. MACHINE TRANSLATION 8

2.1.1. Representasi dan Pemrosesan 9

2.1.2. Arsitektur Transformasi 14

2.1.3. Kamus 22

2.1.4. Permasalahan dalam Penerjemahan 26

2.2. PHP DAN MYSQL 30

2.2.1. PHP 30

2.2.2. MySQL 40

BAB 3 ANALISA DAN PERANCANGAN **45**

3.1. DESKRIPSI 45

3.2. USE CASE DIAGRAM 48

3.3. CLASS DIAGRAM 49

3.4. CONCEPTUAL DATA MODEL 56

BAB 4 IMPLEMENTASI **59**

4.1. KONFIGURASI PERANGKAT KERAS DAN SISTEM OPERASI 59

4.2. KONFIGURASI PERANGKAT LUNAK 59

4.3. IMPLEMENTASI MODUL 60

4.3.1. Aturan-aturan Analisa 60

4.3.2. Proses Strukturisasi Awal (*Genuine Structurization*) 71

4.3.3. Proses Strukturisasi Sumber (*Source Structurization*) 73

4.3.4. Proses Strukturisasi Tujuan (*Destination Structurization*) 75
4.3.5. Pembangkitan Hasil (*Result Generation*) 77

4.4. IMPLEMENTASI APLIKASI 78
4.4.1. Implementasi Dengan Halaman Web 78
4.4.2. Antarmuka Menu HOME 79
4.4.3. Antarmuka Menu DICTIONARY 81
4.4.4. Antarmuka Menu HELP 82
4.4.5. Antarmuka Setting TYPE 82
4.4.6. Antarmuka Setting SOURCES 82
4.4.7. Antarmuka Setting SEQUENCES 83
4.4.8. Antarmuka Setting DESTINATIONS 85
4.4.9. Antarmuka Setting DELIMITERS 85

BAB 5 UJI COBA DAN EVALUASI 86

5.1. PARAMETER UMUM UJI COBA 86

5.2. SKENARIO UJI COBA 86

5.3. PELAKSANAAN DAN EVALUASI HASIL UJI COBA 88

BAB 6 PENUTUP 98

6.1. KESIMPULAN 98

6.2. SARAN 98

DAFTAR PUSTAKA 100

LAMPIRAN 101

A1. SINTAKS ATURAN MODIFIKASI SUMBER 101

A2. SINTAKS ATURAN TRANSFORMASI 102

A3. SINTAKS ATURAN MODIFIKASI TUJUAN 103

B1. TABEL ATURAN TIPE UJI COBA 106

B2. TABEL ATURAN MODIFIER SUMBER UJI COBA 107

B3. TABEL ATURAN MODIFIKASI TUJUAN UJI COBA 111

B4. TABEL ATURAN TRANSFORMASI UJI COBA 112

DAFTAR GAMBAR

Gambar 2 - 1 Diagram pohon sebagai representasi struktur kalimat 10
Gambar 2 - 2 Contoh struktur aturan grammar Bahasa Inggris 11
Gambar 2 - 3 Arsitektur direct / transformer 15
Gambar 2 - 4 Arsitektur indirect / linguistic knownledge 19
Gambar 2 - 5 Konsep interlingua 21
Gambar 2 - 6 Contoh sebuah entri dalam kamus monolingual Bahasa Inggris 22
Gambar 2 - 7 Contoh sebuah entri dalam kamus bilingual Inggris-Perancis 23
Gambar 2 - 8 Contoh struktur kamus elektronik 24

Gambar 3 - 1 Use case diagram 48
Gambar 3 - 2 Class Diagram untuk kelas ConstituentsTable 50
Gambar 3 - 3 Class Diagram untuk kelas-kelas proses analisa 51
Gambar 3 - 4 Class Diagram untuk kelas-kelas aturan 52
Gambar 3 - 5 Class Diagram untuk kelas SentencesTable 54
Gambar 3 - 6 Database Conceptual Data Model 57

Gambar 4 - 1 Contoh entri dalam database Kamus 61
Gambar 4 - 2 Struktur konstituen awal kalimat sumber 63
Gambar 4 - 3 Contoh bentuk perubahan akibat aturan transformasi 65
Gambar 4 - 4 Struktur Sumber dan struktur Tujuan hasil transformasi 66
Gambar 4 - 5 Struktur Sumber dan Struktur Tujuan hasil semua aturan transformasi 68
Gambar 4 - 6 Contoh struktur tujuan hasil penerapan seluruh bentuk aturan 70
Gambar 4 - 7 Diagram alir proses analisa struktur kalimat awal 72
Gambar 4 - 8 Diagram alir proses pembentukan struktur asal 74
Gambar 4 - 9 Diagram alir proses pembentukan struktur tujuan 75
Gambar 4 - 10 Diagram alir proses pengujian Aturan Transformasi Abstrak 76
Gambar 4 - 11 Diagram alir proses pembentukan struktur hasil 77
Gambar 4 - 12 Halaman pembuka aplikasi index.php 78
Gambar 4 - 13 Halaman home.php untuk menerima inputan 79
Gambar 4 - 14 Hasil proses penerjemahan yang dilakukan 80
Gambar 4 - 15 Halaman kamus elektronik dictionary.php 81
Gambar 4 - 16 Halaman help.php 82
Gambar 4 - 17 Halaman editor aturan tipe kata 83
Gambar 4 - 18 Halaman editor aturan modifikasi kata sumber 83
Gambar 4 - 19 Halaman editor aturan transformasi 84
Gambar 4 - 20 Halaman editor aturan modifikasi kata tujuan 84
Gambar 4 - 21 Halaman editor aturan pembatas kalimat 85

DAFTAR TABEL

Tabel 5 - 1 Uji coba *to be* sebelum diberi aturan 88
Tabel 5 - 2 Hasil uji coba aturan-aturan *to be* 89
Tabel 5 - 3 Uji coba noun sebelum diberi aturan 90
Tabel 5 - 4 Hasil uji coba aturan-aturan kata benda (*Noun*) 92
Tabel 5 - 5 Uji coba pronoun sebelum diberi aturan 92
Tabel 5 - 6 Hasil uji coba aturan-aturan kata ganti orang (*pronoun*) 94
Tabel 5 - 7 Uji coba tenses sebelum diberi aturan 94
Tabel 5 - 8 Hasil uji coba aturan-aturan kata waktu (*tenses*) 96
Tabel 5 - 9 Hasil uji coba tenses yang mengalami kegagalan 97

BAB 1

PENDAHULUAN

BAB 1
PENDAHULUAN

1.1. LATAR BELAKANG

Bahasa adalah sebuah cara yang dilakukan oleh manusia untuk menyampaikan keinginannya agar dapat dipahami oleh manusia lainnya. Bahasa telah menjadi kebudayaan yang paling menonjol dari setiap bangsa di dunia ini yang memiliki ciri khas serta tingkat kemajuan yang tidak sama antara yang satu dengan yang lain yang membedakan antara bangsa yang satu dengan bangsa yang lainnya. Kebutuhan untuk saling mengerti dan memahami berbagai macam bahasa pada saat ini semakin meningkat seiring bertambahnya kepentingan manusia dan kemudahan yang berikan dalam hal teknologi informasi dan komunikasi global antar bangsa.

Bahasa Inggris adalah bahasa yang saat ini telah diakui sebagai bahasa internasional dan banyak digunakan untuk menyampaikan, mendokumentasikan, ataupun mempublikasikan apa yang telah menjadi pekerjaan mereka kepada orang-orang di negara lain yang memiliki bahasa yang berbeda dengan mereka. Bahasa Indonesia adalah bahasa yang digunakan oleh Bangsa Indonesia sebagai bahasa resmi negara, bahasa nasional, dan bahasa iptek. Bahasa Indonesia memiliki bentuk baku tata bahasanya sendiri yang memiliki pola-pola dasar dan struktur kalimat yang berbeda dengan bahasa lain.

Untuk menjembatani antara kebutuhan material dan informasi-informasi tekstual yang harus diadaptasi dari Bahasa Inggris dengan keterbatasan waktu,

kemampuan, dan biaya dari si pencari informasi tersebut, maka dibutuhkan sebuah aplikasi penerjemah bahasa yang dapat menerjemahkan sekaligus sebuah dokumen dengan hasil seoptimal mungkin. Perkembangan teknologi komputer dan metode-metode pengembangan dan *tool* yang lebih maju dalam bidang *Machine Translation* dapat menjadi komponen penting bagi perkembangan dan kemajuan suatu bangsa dalam hal memperlancar proses asimilasi dan interaksi antar budaya dan bangsa menjadi lebih baik.

1.2. TUJUAN DAN MANFAAT

Tujuan dari tugas akhir ini adalah untuk membuat sebuah sistem aplikasi penerjemahan dwi-bahasa satu arah berdasarkan metode *direct-transformer* dengan contoh kasus penerjemahan kalimat dari Bahasa Inggris ke Bahasa Indonesia. Sistem dibuat untuk menganalisa aturan-aturan dengan melakukan perubahan-perubahan pada struktur gramatikal sebuah kalimat sumber secara dinamis sehingga dapat menghasilkan suatu bentuk struktur kalimat tujuan yang benar.

Dengan adanya sistem aplikasi tersebut maka kegiatan penerjemahan, pencarian dan penyampaian informasi materi-materi berbahasa Inggris kedalam bahasa Indonesia dapat berlangsung dengan lebih cepat, efektif, efisien. Serta pula turut memberikan alternatif metode dan pendekatan dalam hal pengembangan sistem aplikasi penerjemahan antar bahasa berbasis teknologi komputer terutama yang berkaitan dengan Bahasa Indonesia.

1.3. PERUMUSAN MASALAH

Permasalahan yang diangkat dalam penyusunan tugas akhir ini adalah :

1. Bagaimana aplikasi dapat menerjemahkan input dalam Bahasa Inggris secara harfiah dan struktural dan menghasilkan output yang sesuai dalam Bahasa Indonesia.
2. Bagaimana aplikasi dapat memilih padanan kata dan menyusunnya sesuai dengan kaidah-kaidah Bahasa Indonesia yang tersimpan didalam database aturan-aturan analisa.
3. Bagaimana membuat *parser* pengelompokkan kata yang dapat memproses aturan-aturan analisa yang bersifat *customizeable*.
4. Bagaimana aplikasi mampu memberikan alternatif-alternatif solusi untuk sebuah kasus yang memiliki percabangan makna atau lebih dari satu arti.
5. Bagaimana aplikasi dapat mentolerir terjadinya kesalahan inputan atau keterbatasan aturan dan perbendaharaan kata dan istilah sehingga output yang dihasilkan menjadi seoptimal mungkin.

1.4. BATASAN MASALAH

Dari permasalahan-permasalahan diatas, maka batasan masalah untuk Tugas Akhir ini adalah :

1. Bahasa Inggris yang digunakan berupa kata-kata Bahasa Inggris yang terdapat dalam database Kamus.
2. Klasifikasi kata untuk aturan Bahasa Indonesia yang digunakan sesuai dengan database aturan grammar.

PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

3. Aturan yang dimaksud dalam sistem yang dibuat adalah aturan-aturan analisa yang terdapat dalam database yang telah didefinisikan dahulu sebelumnya.
4. Kata yang diterjemahkan oleh aplikasi adalah kata yang terdapat dalam database kamus.
5. Proses analisa dioptimalisasi untuk menerjemahkan teks dalam level kalimat, yaitu bagian teks yang dibatasi oleh delimiter kalimat yang telah didefinisikan sebelumnya.

1.5. METODOLOGI

Pengerjaan Tugas Akhir ini terbagi menjadi beberapa tahapan yaitu :

1. Studi Literatur

Tahapan ini merupakan tahap pencarian informasi yang diperlukan untuk pengumpulan data dan desain sistem yang akan dibuat. Informasi didapatkan dari paper, buku pelajaran Bahasa Inggris, software-software translator yang sudah ada, dan materi-materi lain yang berhubungan di internet.

2. Definisi Kebutuhan User dan Pengumpulan Data

Tahapan ini merupakan tahap melakukan identifikasi yang lebih terperinci terhadap masalah yang dihadapi oleh seorang penerjemah, kalangan pembaca bacaan berbahasa asing biasa ataupun pembaca yang spesifik pada suatu bidang tertentu.

Pengumpulan data akan didapat dari kamus Bahasa Indonesia, kamus Bahasa Inggris, kamus Bahasa Inggris-Indonesia, kamus istilah-istilah, dan sebagainya.

3. Perancangan Perangkat Lunak dan Desain Sistem

Tahapan ini merupakan tahap perancangan, analisa, dan desain sistem perangkat lunak berdasarkan hasil yang telah diperoleh dari tahap 1 dan 2.

4. Pembuatan Aplikasi

Tahapan ini merupakan tahap implementasi dan pengembangan perangkat lunak berdasarkan rancangan yang telah dibuat sebelumnya.

5. Uji Coba dan Evaluasi

Tahapan ini merupakan tahap dimana digunakan bermacam inputan yang bervariasi untuk mencoba jalannya aplikasi apakah telah sesuai dengan rancangan dan disain implementasi yang dibuat juga untuk mencari kesalahan-kesalahan program yang mungkin terjadi untuk selanjutnya dapat dilakukan penyempurnaan.

6. Penyusunan Buku Tugas Akhir

Pada tahap terakhir ini dilakukan penyusunan buku sebagai dokumentasi dari pelaksanaan Tugas Akhir ini. Buku Tugas Akhir juga diharapkan dapat berguna bagi orang lain yang memiliki keinginan untuk mengembangkan sistem tersebut lebih lanjut.

1.6. SISTEMATIKA PENULISAN

Sistematika penulisan yang digunakan dalam buku tugas akhir ini adalah sebagai berikut :

BAB I PENDAHULUAN

Berisi pendahuluan yang terdiri dari latar belakang, tujuan dan manfaat, perumusan masalah, batasan masalah, metodologi, dan sistematika penulisan buku ini.

BAB II DASAR TEORI

Berisi tentang dasar-dasar teori dan konsep mengenai hal-hal yang menjadi landasan pembuatan tugas akhir ini yaitu mengenai *Machine Translation* serta PHP dan MySQL.

BAB III ANALISA DAN PERANCANGAN

Berisi mengenai deskripsi mengenai sistem aplikasi yang dibuat dalam tugas akhir ini serta perancangan berupa diagram use case, diagram alir, diagram model data konseptual, dan diagram kelas dari sistem aplikasi tersebut.

BAB IV IMPLEMENTASI

Berisi spesifikasi kebutuhan perangkat keras dan perangkat lunak yang dibutuhkan dalam pembuatan sistem aplikasi serta penjabaran dan penerapan modul-modul, aturan, dan proses-proses yang telah diterangkan dalam bab sebelumnya hingga akhirnya membentuk sebuah sistem akhir yang utuh.

BAB V UJI COBA DAN EVALUASI

Berisi pengujian dan pengetesan sistem aplikasi untuk menemukan permasalahan yang tidak diketahui sebelumnya dan juga memberi evaluasi dan penilaian terhadap kinerja sistem secara keseluruhan.

BAB VI PENUTUP

Berisi ringkasan berupa kesimpulan dari hasil implementasi dan uji coba sistem yang telah dibahas dari bab-bab sebelumnya beserta saran-saran yang dapat disampaikan selama mengerjakan tugas akhir ini.

BAB 2

DASAR TEORI

BAB 2
DASAR TEORI

2.1. *MACHINE TRANSLATION*

Machine Translation (MT) adalah sebuah istilah dalam bidang teknologi informasi dalam rangka melakukan upaya penerjemahan sebuah bahasa kedalam bahasa lainnya secara otomatis baik secara keseluruhan maupun sebagian. Untuk mengerti suatu bahasa maka sebuah sistem MT memerlukan pengetahuan-pengetahuan yang berupa informasi yang dibutuhkan untuk melakukan proses penerjemahan.

Seorang penerjemah manusia dalam melakukan penerjemahan, secara manual, setidaknya perlu memiliki lima macam pengetahuan yang berbeda yaitu :

- Pengetahuan mengenai bahasa asal.
- Pengetahuan mengenai bahasa tujuan. Hal ini berguna agar penerjemah menghasilkan teks yang dapat diterima dalam bahasa tujuan.
- Pengetahuan mengenai keberagaman makna yang saling berkesesuaian antara bahasa asal dan bahasa tujuan, pada tingkat paling sederhana yaitu arti dari masing-masing kata yang akan diterjemahkan.
- Pengetahuan mengenai pokok bahasan sebuah kalimat, termasuk pengetahuan umum dan akal sehat, sehingga penerjemah dapat mengerti bagaimana cara menerjemahkan bahasa asal dalam bahasa tujuan.
- Pengetahuan tentang budaya, adat, kebiasaan, harapan, dan sebagainya dari kedua pembicara dari bahasa asal dan bahasa tujuan.

Pengetahuan tentang kebahasaan terbagi dalam empat hal sebagai berikut :

- Pengetahuan Fonologis, yaitu bagaimana sebuah kata diucapkan / disuarakan.
- Pengetahuan Morfologis, yaitu bagaimana sebuah kata dibentuk atau mengalami sebuah bentukan.
- Pengetahuan Sintaksis, yaitu bagaimana sebuah kalimat atau frase dibentuk dari kata-kata pembentuknya.
- Pengetahuan Semantik, yaitu bagaimana memilih arti dari sebuah kata, atau bagaimana arti sebuah frase dapat diketahui dari arti kata-kata pembentuknya.

2.1.1. Representasi dan Pemrosesan

Dalam melakukan analisa terhadap sebuah kalimat secara umum dilakukan dua buah analisa yaitu analisa konstituen atau struktur frase atau pengkategorian dan analisa gramatikal. Analisa konstituen termasuk menentukan jenis kata pembentuk kalimat apakah berupa kata benda, kata kerja, kata sifat, keterangan, dan lain-lain. Sedangkan analisa gramatikal adalah menentukan tata letak sebuah kata atau frase dalam kalimat yang memenuhi kaidah suatu bahasa yaitu penentuan subjek, objek, predikat, dan lain-lain.

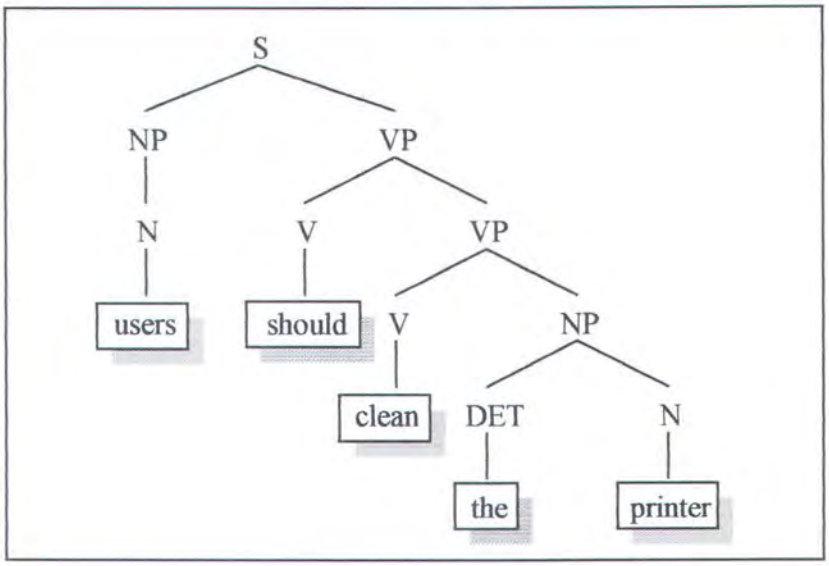
Struktur sebuah kalimat dapat direpresentasikan menjadi sebuah diagram pohon dari unsur-unsur kalimat mulai dari yang terendah (kata atau frase yang dapat berdiri sendiri) hingga bentukan-bentukannya sampai akhirnya menjadi sebuah kalimat utuh. Seperti dalam contoh kalimat berikut :

<u>Users should print the printer</u>

Dapat direpresentasikan dalam bentuk lainnya sebagai berikut :

$[_{S}[_{NP}[_{N}$ users$]]][_{VP}[_{AUX}$ should$][_{V}$ clean$][_{NP}[_{DET}$ the$][_{N}$ printer$]]]]$

Secara logika struktur semacam itu dapat digambarkan sebagai bentuk diagram pohon yang ditunjukkan pada gambar 2-1.



Gambar 2 - 1
Diagram pohon sebagai representasi struktur kalimat

Dalam MT nama unsur-unsur kata yang digunakan sama dengan yang digunakan dalam bidang bahasa sehari-hari namun dengan adanya beberapa tambahan nama untuk beberapa ekspresi yang menjelaskan adanya perubahan antar unsur bahasa satu dengan bahasa lain. Karena Bahasa Sumber yang dibahas adalah Bahasa Inggris maka unsur-unsur yang sering dibahas adalah unsur-unsur dalam Bahasa Inggris, antara lain S untuk *Sentence* (kalimat), N untuk *Noun* (kata benda), V untuk *Verb* (kata kerja), VP untuk *Verb Phrases* (frase kerja), NP untuk *Noun Phrases* (frase benda), DET untuk *Determiner*, dan AUX untuk *Auxiliaries*.

Struktur daripada diagram pohon tersebut mengikuti sebuah aturan yang tetap yang dapat direpresentasikan pula dalam suatu notasi aturan lain seperti contoh struktur aturan kalimat Bahasa Inggris pada gambar 2-2.

Sintak dalam aturan-aturan pada gambar 2-2 dapat dijelaskan sebagai berikut, aturan di sisi kiri tanda panah disebut aturan sisi kiri (LHS / *Left Hand Side*) sedangkan aturan di sisi kanannya disebut aturan sisi kanan (RHS / *Right Hand Side*). Aturan dilihat sesuai dengan urutan penulisannya, contoh aturan kedua, *Verb Phrases* dapat berupa *Verb* yang diikuti, namun tidak mesti (disimbolkan dengan tanda kurung), dengan *Noun Phrases*, diikuti dengan *Prepositional Phrase* sebanyak mungkin setelahnya (disimbolkan dengan tanda bintang).

S	→ NP (AUX) VP
VP	→ V (NP) PP*
NP	→ (DET) (ADJ) N PP*
PP	→ P NP
N	→ user
N	→ users
N	→ printer
N	→ printers
V	→ clean
V	→ cleans
AUX	→ should
DET	→ the
DET	→ a
P	→ with

Gambar 2 - 2
Contoh struktur aturan grammar Bahasa Inggris

Selain pengetahuan mengenai struktur pohon dari konstituten atau pembentuk kalimat, dalam sebuah kalimat juga terdapat informasi lainnya yaitu relasi grammar atau fungsi grammar seperti *head*, *subject*, dan *object*.

Struktur dengan unsur-unsur relasi grammar sebenarnya tidak terlalu berbeda dengan struktur pohon konstituen. Namun dalam struktur relasi grammar informasi yang dapat diperoleh sebenarnya lebih banyak lagi yaitu :

- Informasi urutan susunan kata-kata pembentuk kalimat
- Informasi mengenai peran masing-masing kata dalam kalimat

Dalam struktur relasi grammar kita dapat menambahkan attribut informasi lainnya yang disebut *feature*, yang terdiri dari *attribute* dan *value*. *Feature* dapat berupa informasi mengenai bentuk waktu, jenis dasar, serta aspek lain yang berhubungan dengan sebuah kata. Antara struktur relasi grammar dengan struktur konstituen dapat dan seharusnya diproses secara bersamaan untuk mendapatkan hasil yang benar. Untuk itu dapat dilakukan penggabungan antara kedua struktur itu yang mana unsur-unsur dari keduanya tetap tidak hilang dan dapat dilacak ulang. Menyangkutkan informasi dalam sebuah kata tidak lepas dari menyertakan pula arti atau padanan katanya dalam bahasa tujuan. Pemilihan padanan kata yang tepat sangat mempengaruhi hasil yang akan dicapai dalam sebuah pemrosesan MT.

Dalam kehidupan nyata, ada tiga buah pengetahuan yang terlibat ketika kita menerjemahkan suatu kalimat :

1. Pengetahuan Semantik, yaitu pengetahuan mengenai padanan kata umum dari sebuah kata sebagai kata-kata yang berdiri sendiri.

2. Pengetahuan Pragmatik, yaitu pengetahuan mengenai arti atau makna yang terkandung di dalam sebuah kata dalam konteksnya sebagai bagian dari keseluruhan kalimat yang dapat saja berbeda dari arti semantiknya.
3. Pengetahuan Dunia Nyata (*Real World*), yaitu hal-hal yang berhubungan dengan lingkungan fisik atau nyata di sekitar pembicara yang dimaksud pada saat kalimat tersebut dinyatakan.

Proses yang dilakukan oleh sebuah sistem MT selanjutnya adalah proses transformasi yang terbagi menjadi dua bagian utama yaitu proses analisa atau *parsing* dan proses sintesa atau *generating*.

Proses analisa adalah suatu proses membentuk sebuah representasi, baik itu struktur konstituen, struktur relasi grammar, dan sebagainya, dari sebuah *string* kalimat *input* dan informasi yang berhubungan dengannya dari database kamus yang dimiliki oleh sistem MT tersebut. Proses analisa ini melibatkan algoritma-algoritma parsing struktur pohon yang mungkin berbeda-beda antara sebuah sistem MT dan yang lainnya, dimana hasil akhirnya adalah mendapatkan bentuk struktur representasi yang dapat digunakan untuk proses selanjutnya. Sementara proses sintesa adalah kebalikan dari proses analisa, proses sintesa melakukan pembangkitan kalimat tujuan berdasarkan struktur representasi yang telah dibuat oleh proses analisa.

Dalam setiap proses tersebut terlibat aturan-aturan yang telah dan harus didefinisikan sebelumnya agar sistem MT dapat berjalan dengan semestinya. Aturan-aturan tersebut merepresentasikan bentuk grammar atau tata bahasa dari

masing-masing bahasa yang diproses dan hubungannya dalam penerjemahan makna antar bahasa.

2.1.2. Arsitektur Transformasi

Keseluruhan proses dan aplikasi penerjemahan bahasa oleh sistem MT yang dikerjakan sepenuhnya oleh komputer disebut sebagai MT engine, yang mana setiap sistem MT dibedakan berdasarkan algoritma dan teknik penerjemahan yang diterapkan pada engine sistem tersebut.

Arsitektur dari sebuah MT engine tersebut dapat dipisahkan menjadi dua jenis yang akan dibahas satu persatu di bagian ini, yaitu:

- Arsitektur Direct atau Transformer, dan
- Arsitektur Indirect atau Linguistic-Knowledge

Arsitektur Direct atau Transformer

Arsitektur *Direct* (langsung), selanjutnya disebut Transformer, adalah jenis konsep MT yang paling dahulu diterapkan dan bahkan tetap digunakan hingga saat ini oleh berbagai sistem MT komersial. Cara kerja arsitektur jenis ini secara sederhana digambarkan sebagai berikut:

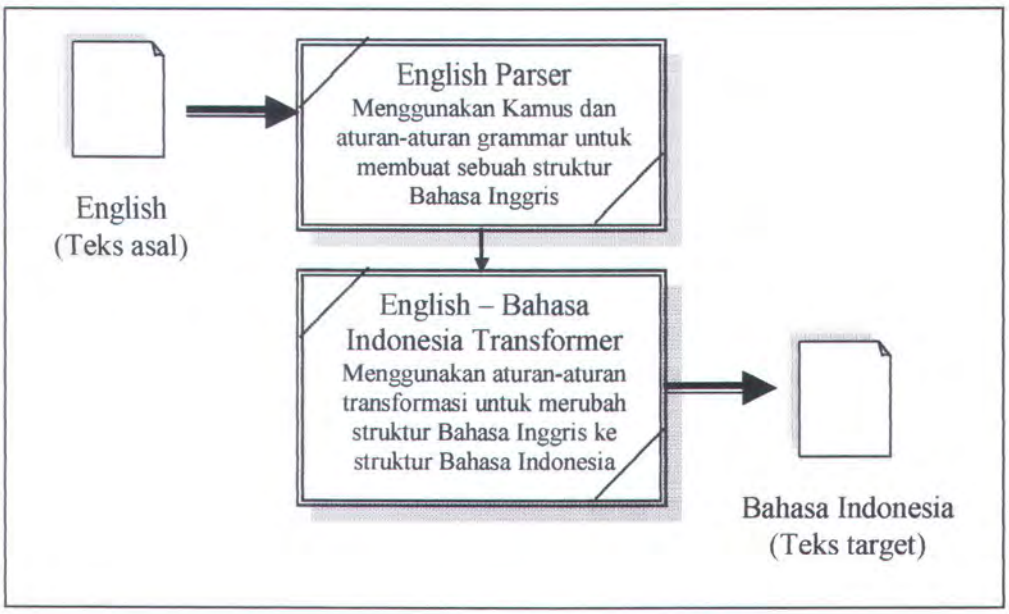
- Menemukan bentuk sederhana struktur representasi dari kalimat asal (*source language*).
- Menemukan padanan kata dari bahasa tujuan dalam kamus.
- Mengatur urutan susunan kata dalam bahasa tujuan sesuai dengan aturan yang ada.
- Transformer disebut langsung (*direct*) karena secara langsung menerjemahkan struktur permukaan (*surface-structure*) dari pembentuk

PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

kalimat asal, menyesuaikan urutannya sesuai dengan aturan bahasa tujuan, dan membentuk sebuah string akhir yang 'tampak' sesuai dengan tata bahasa tujuan yang ditentukan dalam rule yang ada. Perbedaannya dengan proses tidak langsung dijelaskan di bagian arsitektur Indirect.

Transformer memerlukan unsur-unsur berikut untuk mengerjakan tugasnya :

- Aturan kamus bilingual
- Aturan urutan kata pembentuk kalimat
- Aturan pengolah kata bahasa tujuan, sebagai tambahan



Gambar 2 - 3
Arsitektur direct / transformer

Langkah-langkah dalam menerjemahkan sebuah kalimat asal menggunakan Transformer dijabarkan sebagai berikut :

Langkah 1: Menemukan kata dalam kamus bahasa asal untuk memperoleh informasi yang diperlukan mengenai kata tersebut.

Langkah 2: Melakukan proses parsing awal apabila ditemukan aturan yang menyebutkan demikian pada informasi grammar pada kamus.

Langkah 3: Menemukan padanan kata yang sesuai dalam bahasa tujuan dan juga melakukan parsing apabila ditemukan atuan yang menyebutkan demikian pada langkah ini.

Langkah 4: Melakukan parsing keseluruhan sehingga membentuk susunan tata bahasa tujuan yang sesuai dengan aturan.

Proses arsitektur Transformer, yang diilustrasikan pada gambar 2-3, menunjukkan bahwa arsitektur Transformer memiliki beberapa kelebihan dan kekurangan sebagai berikut :

- Proses parsing dapat berhenti sebelum mencapai aturan grammar yang benar yang sesuai aturan, karena Transformer biasanya memiliki aturan grammar yang tidak terlalu sempurna sehingga ada pola-pola yang kadang tidak tertampung oleh aturan yang ada.
- Proses analisa atau penyusunan struktur representasi tidak terlalu sempurna dari sudut pandang tata bahasa yang baik

- Transformer memiliki beberapa pengetahuan tentang grammar yang diperbandingkan (*comparative grammar*) dari kedua bahasa yang mendefinisikan perbedaan-perbedaan yang ada di antara keduanya.
- Aturan-aturan Tranformer bersifat berpasangan antara bahasa asal dan aturan yang bersesuaian dalam bahasa tujuan, sehingga Transformer tidak memiliki pengetahuan khusus tentang suatu bahasa terutama bahasa tujuan.
- Karena proses parsing pada Transformer tidak mengikat, dapat berhenti pada kondisi apapun yang terjadi, maka Tranformer bersifat sangat tahan kesalahan (*highly robust*).
- Aturan-aturan yang terlalu banyak pada Transformer membuat arsitektur ini sulit untuk dipahami dan akibatnya sulit untuk dikembangkan atau dimodifikasi.
- Pendekatan Transformer di-desain memang untuk penerjemahan satu-arah (*one-direction*). Pada sebuah sistem MT dengan MT *engine* Transformer dapat saja dibuat multi-bahasa yaitu sekumpulan penerjemahan satu-arah dan bukan sebuah sistem penerjemahan multi-bahasa murni.
- Karena berdasarkan aturan yang terpenuhi, maka hasil akhir Transformer bervariasi mulai dari sangat baik dan memenuhi kaidah tata bahasa ketika seluruh aturan terpenuhi dengan sempurna, hasil campuran antara bahasa asal dan bahasa sumber ketika beberapa aturan tidak terpenuhi dengan

sempurna, dan pada contoh kasus terburuk kalimat bahasa asal bahkan tidak diterjemahkan sama sekali karena tidak memenuhi satu aturan pun.

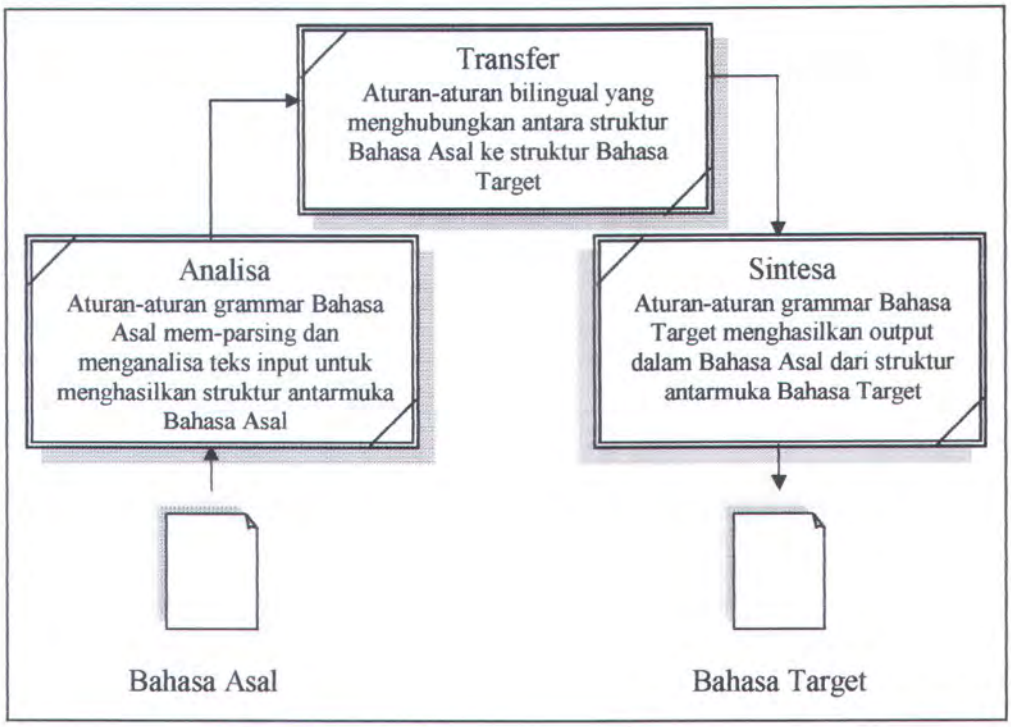
Arsitektur Indirect / linguistic Knowledge

Arsitektur *Indirect* (tidak langsung), selanjutnya disebut LK, adalah jenis konsep MT yang tergolong baru dan secara sederhana digambarkan sebagai berikut :

- LK memiliki pengetahuan mengenai tata bahasa dan analisa mengenai kedua bahasa asal dan tujuan
- LK memiliki pengetahuan mengenai perbedaan antara kedua bahasa dan memiliki pemecahannya
- LK adalah jenis arsitektur yang dioptimalisasi untuk membentuk bahasa tujuan yang tepat dan sesuai dengan tata bahasanya. Sebuah *engine* LK memiliki aturan grammar yang mandiri pada setiap bahasa yang didukungnya. Konsepnya adalah bahwa setiap bahasa dapat dilakukan studi masing-masing tentang bagaimana bahasa tersebut dibentuk dan diekstrak, diubah menjadi struktur-struktur representasinya.
- LK memenuhi syarat sebagai konsep penerjemahan multi-bahasa murni, walau dalam praktiknya disebutkan masih ditemukan banyak kendala sehingga perlu banyak tambahan aturan penyesuaian yang membuatnya 'tidak terlalu' murni.

Untuk menerapkan metode arsitektur LK setidaknya diperlukan pengetahuan mengenai hal-hal berikut :

- Grammar pokok (*substantial grammar*) dari kedua bahasa, yang digunakan untuk proses *parsing* serta proses *generating*.
- Grammar perbandingan (*comparative grammar*) sebagai tambahan, untuk menghubungkan struktur representasi dari bahasa asal dan bahasa tujuan.



Gambar 2 - 4
Arsitektur indirect / linguistic knownledge

Arsitektur LK yang memiliki diagram seperti yang ditunjukkan pada gambar 2-4 memiliki langkah-langkah sebagai berikut :

Langkah 1: Lakukan analisa grammar terhadap kalimat bahasa asal dengan melakukan parsing string inputan untuk membentuk sebuah struktur representasi antarmuka dari kalimat asal. Proses ini disebut proses Analisa.

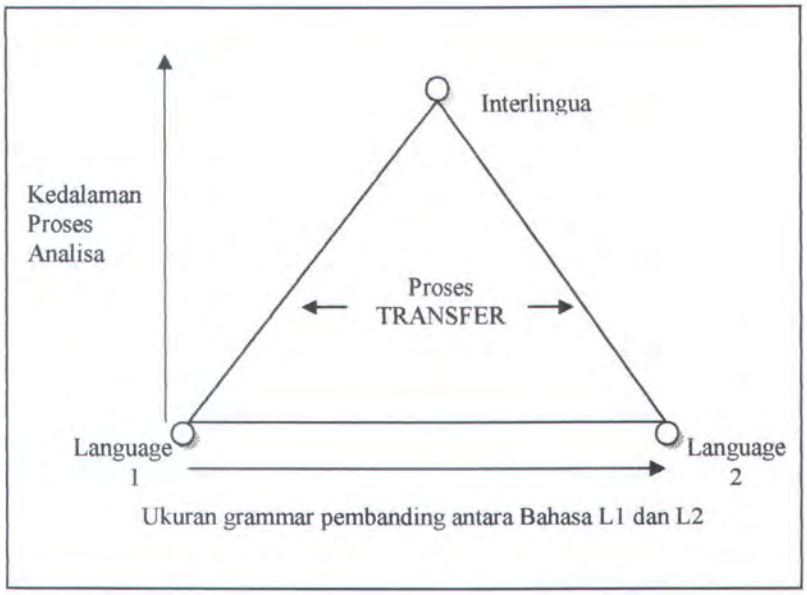
Langkah 2: Lakukan pengecekan terhadap aturan pembanding yang menghubungkan antara struktur bahasa asal dan struktur bahasa tujuan, dengan kata lain membuat struktur representasi bahasa tujuan yang sesuai dengan struktur representasi bahasa asal. Proses ini disebut proses Transfer.

Langkah 3: Lakukan pembangkitan hasil bahasa tujuan dari struktur representasi grammar bahasa tujuan yang dihasilkan pada proses Transfer. Proses ini disebut proses sintesa

Dari penjelasan diatas, dapat disebutkan hal-hal yang berhubungan dengan LK sebagai berikut :

- LK menitikberatkan pada pembentukan kalimat tujuan yang sesuai dengan aturan bahasa yang sudah disusun dengan baik sehingga hasil dari *engine* LK adalah sebuah kalimat yang memenuhi aturan-aturan grammar bahasa tujuan sekalipun hasil penerjemahannya tidak sesuai dengan yang diharapkan.
- Dalam ilustrasi LK pada gambar 2-4 tersebut dapat dilihat bahwa proses berjalan dua-arah sehingga untuk menerjemahkan bahasa tujuan ke bahasa asal tinggal membalik proses yang ada.
- LK didesain untuk dapat mengakomodir penerjemahan multi-bahasa sehingga penambahan sebuah bahasa dalam sistem yang telah memiliki beberapa bahasa sebelumnya akan lebih mudah, karena hanya memerlukan pengetahuan mengenai bahasa yang akan dimasukkan tersebut.

- Berbeda dengan Transformer yang hanya menganalisa struktur permukaan dari struktur representasi, LK didesain untuk dapat menganalisa struktur yang lebih dalam hingga bentuk kata yang paling primitif sekalipun.



Gambar 2 - 5
Konsep interlingua

Dalam praktiknya banyak hal yang menjadi kendala dalam LK antara lain pada proses Transfer yang sebenarnya masih menggunakan pemetaan antara struktur bahasa asal dan bahasa tujuan yang bersifat langsung (*direct*), hal ini menjadikan LK tidak murni bersifat *indirect*, sehingga memperbesar kemungkinan adanya aturan-aturan baru yang disesuaikan untuk kebutuhan penerjemahan antar bahasa tertentu.

Pada konsepnya disebutkan bahwa proses Transfer akan berjalan lebih baik dan efisien apabila dilakukan analisa struktur yang semakin dalam dari

grammar kedua bahasa. Proses ini disebut Interlingual seperti tampak pada ilustrasi pada gambar 2-5.

2.1.3. Kamus

Kamus (*Dictionary*) adalah bagian yang terpenting dan utama dalam sebuah sistem MT. Dalam jumlah informasi yang dimilikinya dapat dikatakan bahwa Kamus adalah komponen MT yang paling besar mulai dari sekedar daftar kata-kata hingga berbagai informasi yang dapat disertakan pada masing-masing kata tersebut baik itu untuk mendukung proses penerjemahan ataupun untuk keperluan lain seperti dokumentasi. Sebagai sebuah bagian yang terpenting, ukuran dan kualitas dari sebuah Kamus menjadi ukuran dan batasan yang dapat diharapkan dari sebuah sistem MT yang menggunakannya.

but.ton /'bʌtn *n* **1** knob or disc made of wood, metal, etc sewn onto a garment as a fastener or as an ornament: *a coat, jacket, shirt, trouser button* ° *lose a button* ° *sew on a new button* ° *do one's button up* ⇒ illus at JACKET. **2** small knob that is pressed to operate a doorbell, a switch on a machine, etc: *Which button do I press to turn the radio on?* **3**(idm) **bright as button** ⇒ BRIGHT. **on the 'button** (*US infml*) precisely: *You've got it on the button!*
▷ **but.ton** *v* **1(a)**[Tn,Tn.p] **~sth(up)** fasten with buttons: *button (up) one's coat, jacket, shirt, etc.* **(b)**[I,Ip] **~(up)** be fastened with buttons: *This dress buttons at the back.* **2**(idm) **button (up) one's lip** (*US sl*) be silent. **3**(phr v) **button sth up** (*infml*) complete sth successfully: *The deal should be buttoned up by tomorrow.*
□ **,buttoned 'up** silent and reserved; shy: *I've never met anyone so buttoned up.*
,button-down 'collar collar with ends that are fastened to the shirt with buttons.
'buttonhole *n* **1** slit through which a button is passed to fasten clothing. ⇒ illus at JACKET. **2** flower worn in the buttonhole of the lapel of a coat or jacket. – *v*[Tn] make (sb) stop and listen, often reluctanly, to what one wants to say.
'buttonhook *n* hook for pulling a button into place through a buttonhole.
,button 'mushroom small unopened mushroom.

Gambar 2 - 6
Contoh sebuah entri dalam kamus monolingual Bahasa Inggris

Dalam mendesain sebuah struktur Kamus dalam MT, kita dapat membandingkannya dengan kamus yang sudah umum beredar di khalayak yaitu Kamus Buku. Kamus yang digunakan dalam MT biasanya disebut sebagai Kamus Elektronik, yang selanjutnya disebut Kamus (dengan K besar) saja, untuk membedakannya dengan Kamus Buku.

Dalam kenyataannya tidak ada struktur dan aturan baku dalam penyusunan sebuah Kamus Elektronik. Berbagai sistem MT dapat memiliki bentuk susunan Kamus yang berbeda karena masing-masing memiliki cara dan pendekatan yang berbeda dalam jenis, teknik dan metode dalam hal memanfaatkan informasi yang ada dalam Kamus untuk digunakan dalam aplikasinya.

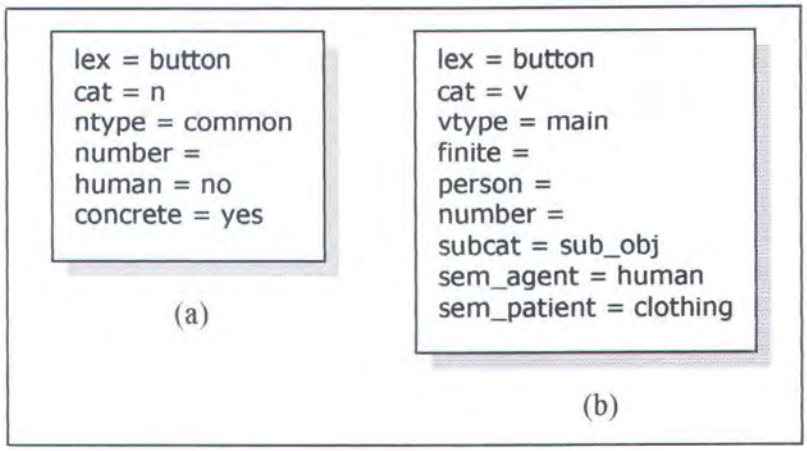
Contoh sebuah entri dalam sebuah Kamus Buku ditunjukkan pada gambar 2-6 untuk kamus monolingual dan gambar 2-7 untuk kamus bilingual, sedangkan contoh struktur Kamus Elektronik ditunjukkan pada gambar 2-8.

but.ton [/'b∧tn] **1** *n* **(a)** (*garment, door, bell, lamp, fencing foil*) bouton *m*. **chocolate ~s** pastilles *fpl* de chocolate. **2** *vt* (*also* **~ up**) *garment* boutonner. **3** *vi* (*garment*) se boutonner. **4** *cpd* **buttonhook** tirebouton *m*; **button mushroom** (petit) champignon *m* de couche *or* de Paris.

printer ['prIntə] *n* **(a)** imprimeur *m*; (*typographer*) typographe *mf*, imprimeur. **The text has gone to the ~** le texte est chez l'imprimeur; **~'s devil** apprenti imprimeur; **~'s error** faute *f* d'impression, coquille *f*; **~'s ink** encre *f* d'imprimerie; **~'s reader** correcteur *m*, -trice *f* (d'épreuves). **(b)**(*Comput*) imprimante *f*. **(c)**(*Phot*) tifeuse f.

Gambar 2 - 7
Contoh sebuah entri dalam kamus bilingual Inggris-Perancis

Informasi-informasi yang melekat pada sebuah entri dalam Kamus dapat disimpan sebagai sebuah record dalam database Kamus. Masing-masing informasi dapat dianggap sebagai atribut dan nilai dari sebuah entri.



Gambar 2 - 8
Contoh struktur kamus elektronik

Pada gambar 2-8 ditunjukkan sebuah struktur Kamus Elektronik dimana sebuah entri yang sama (*button*) dapat memiliki atribut yang berbeda untuk jenis kata yang berbeda yaitu 8.a (*noun*) dan 8.b (*verb*)

Salah satu masalah yang terdapat dalam struktur Kamus semacam ini adalah ketika menemui kata yang bersifat homografi, yaitu dua kata yang sama penulisannya namun dengan konteks yang tidak ada hubungannya sama sekali antara keduanya.

Salah satu hal yang menjadi kekurangan dalam sebuah Kamus Buku (karena Kamus Buku dibatasi juga oleh jumlah halaman) yang dapat diterapkan dalam Kamus Elektronik adalah dapat disertakannya informasi yang berupa dokumentasi dari sebuah entry yang berupa sejenis komentar-komentar dari

lexicographer (ahli kata) tentang siapa dan kapan entry tersebut dimasukan atau direvisi, contoh-contoh penggunaannya, masalah-masalah yang ada dalam penggunaan kata tersebut, dan mungkin saran-saran penggunaan dan perbaikan kata tersebut.

Masalah lain yang menjadi kendala dalam penerjemahan secara umum yang berhubungan dengan entry kamus adalah morfologi, yaitu mengenai struktur internal sebuah kata dan bagaimana sebuah kata dapat dibentuk dari kata-kata yang lain, yang dibagi menjadi tiga kelompok:

Inflectional, yaitu ketika sebuah kata merupakan bawaan dari kata lain tetapi tetap merupakan bagian kalimat atau kategori yang sama. Contoh *walk* (bentuk infinitive) dan *walks* (bentuk perubahan verb orang ketiga). Perubahan inflectional juga dapat disebabkan ditambahkannya prefiks dan sufiks, konversi, pengulangan dan lain-lain.

Derivational, mirip kelompok inflektional, yaitu kata merupakan bawaan dari kata lainnya namun memiliki katogori dan menempati bagian kalimat yang berbeda. Contoh *grammar* (bentuk noun) dan *grammatical* (bentuk perubahan noun menjadi adjective).

Compounding, yaitu ketika sebuah kata merupakan gabungan dari dua atau lebih kata lain seperti kata *button-mushroom* dan *button-hook* pada contoh kamus diatas.

Selain kosakata (*vocabulary*) dalam sebuah bahasa dikenal jenis kata-kata lain yang berupa pengistilahan (*terminology*). Term adalah satu atau lebih kata yang merupakan ekspresi yang sifatnya tertentu dan khas dipakai oleh masyarakat

yang menggunakan suatu bahasa tertentu yang dapat berbeda atau sama dengan bahasa lainnya. *Term* umumnya berupa kata-kata yang menyangkut hal-hal yang beruhubungan dengan bidang-bidang tertentu dalam kehidupan sehari-hari yang dapat saja merupakan bentukan dari kosakata sehari-hari namun mengalami perubahan makna dan penggunaan, contoh yang umum adalah istilah-istilah baru dalam bidang kedokteran, telekomunikasi, teknologi, nama organisasi internasional dan lain-lain yang memiliki padanan istilah yang berbeda antar bahasa.

2.1.4. Permasalahan dalam Penerjemahan

Secara umum permasalahan-permasalahan yang terjadi dalam proses penerjemahan dari sebuah bahasa asal ke bahasa tujuan dapat dibagi sebagai berikut :

- Ambiguitas
- Perbedaan Leksikal dan Struktural
- Perubahan Makna (Idiom dan Collocation)
- Ukuran dan Kualitas Aturan Grammar

Ambiguitas

Ambigu berarti suatu hal yang memiliki lebih dari satu maksud. Dalam MT dikenal dua macam ambigu yaitu ambigu kata (*lexically ambigous*) dan ambigu struktur (*structurally ambigous*). Dalam prosesnya, penganalisaan suatu unsur yang ambigu bersifat kuadratik, artinya analisa berkembang berkelipatan sebanyak ambiguitas yang ditemui. Hal ini membuat proses analisa menjadi lama dan berat karena MT engine harus tetap menganalisa semua kemungkinan kecuali

ada aturan tertentu yang membuatnya dapat menghentikan suatu kondisi tertentu dari ambiguitas yang ditemui. Pengetahuan yang tidak lengkap yang diberikan kepada sistem juga dapat menyebabkan suatu kondisi yang berbahaya karena kemungkinan sistem akan menemui ambiguitas yang sangat banyak.

Contoh ambiguitas leksikal:

- I bought a computer with 500 USD

Saya membeli sebuah komputer seharga 500 USD

- I bought a computer with a wireless link adapter

Saya membeli sebuah komputer yang memiliki adapter sambungan nirkabel

Contoh ambiguitas struktural

- *Cleaning fluids can be dangerous*

Kalimat diatas bersifat ambigu karena dapat diartikan sebagai berikut

- Cleaning fluids is dangerous

Membersihkan cairan (itu) berbahaya

(yang berbahaya adalah aktifitas membersihkan)

- Cleaning fluids are dangerous

Cairan pembersih (itu) berbahaya

(yang berbahaya adalah cairannya)

Ambiguitas terutama nampak jelas apabila sistem menerjemahkan sebuah teks dengan konteks sebuah percakapan, karena banyak melibatkan hal-hal yang berhubungan dengan pengetahuan mengenai dunia nyata.

Namun ada juga beberapa jenis ambiguitas yang tidak perlu diperhatikan karena makna dalam penerjemahannya juga perlu bersifat ambigu, contoh:

- I saw her with glasses

Saya melihatnya dengan kacamata

(entah saya atau dia yang berkacamata tidak penting karena kedua kalimat bersifat ambigu)

Perbedaan Leksikal dan Struktural

Setiap bahasa di dunia memiliki bentuk dan interpretasinya masing-masing mengenai bentuk dan susunan kata-kata yang digunakannya dalam kalimat sehari-hari. Perbedaan secara leksikal adalah mengenai bagaimana sebuah bahasa mengkategorikan kata-katanya, konsep apa yang digunakan dalam setiap ekspresi sebuah kata, dan kata-kata apa saja yang perlu diperlukan seperti apa. Perbedaan secara struktural timbul karena perbedaan bahasa-bahasa menyusun strukturnya untuk maksud yang sama dan atau perbedaan maksud yang dihasilkan untuk struktur-struktur yang sama.

Lubang leksikal adalah sebuah permasalahan lain dimana sebuah frase atau ekspresi yang terdiri dari beberapa kata memiliki padanan kata yang hanya terdiri dari kata tunggal saja dalam bahasa lainnya.

Contohnya adalah perbedaan bentuk unsur DM (diterangkan menerangkan) dalam Bahasa Indonesia ke bentuk unsur MD (menerangkan diterangkan) dalam Bahasa Inggris.

- a <u>blue</u> book *Sebuah buku <u>biru</u>*

(biru sebagai kata yang menerangkan sifat dari buku)

Perubahan makna dalam idiom dan collocation

Idiom adalah sebuah ekspresi atau pernyataan yang maknanya tidak dapat diartikan dan dimengerti secara langsung dari kata-kata pembentuknya.

- When he kicks the bucket, his wife will be very sad

Jika ia menendang ember, istrinya akan sangat sedih (bukan arti yang sebenarnya)

Jika ia meninggal, istrinya akan sangat sedih (arti sebenarnya)

Collocation adalah penggunaan kata ganti atau sinonim yang hanya digunakan ketika diikuti atau menjelaskan suatu hal tertentu dan tidak dapat digunakan ketika menjelaskan hal lain meskipun bentuknya identik.

- They took a walk
- They made an attempt
- They had a talk

Ketiga kata yang bergaris bawah dalam kalimatnya masing-masing memiliki arti yang sama namun tidak dapat saling menggantikan meskipun struktur bahasanya adalah sama.

Ukuran dan kualitas aturan grammar

Ketiga masalah sebelumnya adalah mengenai pengetahuan mengenai dunia nyata yang perlu disertakan sebagai informasi yang perlu disertakan dalam sebuah sistem MT. Informasi-informasi dan MT engine yang lengkap tidak akan sempurna tanpa disertai aturan-aturan yang benar dan lengkap.

Ukuran atau jumlah aturan yang banyak bukan berarti sistem MT tersebut lebih baik dari sistem lainnya, karena selain waktu pemrosesan menjadi lebih

besar juga karena struktur pembentukan aturan yang efektif juga mempengaruhi kualitas sebuah sistem MT.

2.2. PHP DAN MYSQL

2.2.1. PHP

PHP adalah sebuah akronim rekursif dari "PHP: Hypertext Preprocessor", yaitu sebuah bahasa script sisi-server yang terintegrasi dengan HTML dan bersifat open-source. Sebagian besar sintak-sintak yang digunakan adalah bawaan dari bahasa C, Java dan Perl yang ditambahkan dengan beberapa kekhususan yang unik. Tujuan utama dari diciptakannya bahasa ini adalah agar para pengembang web dapat menulis dengan cepat halaman-halaman web yang dibangkitkan secara dinamis.

PHP adalah bahasa script, seperti halnya bahasa pemrograman web lainnya, artinya kode PHP bersifat *script interpreter* dimana kode PHP dianalisa, dieksekusi dan hasilnya dikeluarkan oleh server secara run-time. Berbeda halnya apabila sebuah bahasa bersifat desktop programming dimana kode akan di-*compile* terlebih dahulu ke dalam bentuk *binary* sebelum akhirnya dieksekusi oleh sistem.

PHP bersifat sisi-server (*server-side*) artinya kode PHP hanya akan dieksekusi oleh server saja dan selanjutnya server akan mengembalikan hasil yang diperolehnya kepada client sehingga dari sisi client tidak akan menerima maupun melihat kode dan proses yang dijalankan dalam algoritma PHP. Perbandingan untuk hal ini contohnya adalah bahasa script sisi-client (*client-side*) seperti

javascript dan vbscript dimana pemrosesannya dilakukan pada komputer client (browser).

PHP juga bersifat *HTML-embedded* artinya *script* atau *code* PHP diletakkan (tetapi tidak selalu) didalam sebuah script HTML. Namun karena PHP bersifat sisi-server dan browser (terutama browser-browser modern saat ini) akan menerima hasil apapun dari server dan menampilkannya sebagai sebuah halaman web maka hal ini tidak terlalu nampak berpengaruh.

Contoh sebuah kode PHP dalam sebuah *script* HTML:

```
<html>
<head>
<title>Example</title>
</head>

<body>

<?php
echo "Hi, I'm a PHP script!";
?>

</body>
</html>
```

File-file PHP biasanya diakhiri dengan ekstensi .php atau .php3 untuk membedakannya dengan file-file lainnya terutama dengan file –file HTML biasa yang berekstensi .htm atau .html dan juga agar server bisa mengenalnya sebagai sebuah file PHP karena jika tidak diberi ekstensi PHP tersebut maka server akan memprosesnya sebagai halaman HTML biasa dan tidak mengeksekusi kode PHP apapun didalamnya, malah akan mengembalikannya lagi ke client sehingga kode PHP akan terlihat di source code halaman tersebut.

PHP pada dasarnya digunakan untuk pemrograman berbasis web, namun mengingat kompleksitas dan fungsionalitasnya maka PHP juga cocok digunakan

untuk pemrosesan algoritma pemrograman yang lebih besar daripada sekedar pemrosesan komponen web walaupun hal itu dikompensasi dengan waktu eksekusi yang jauh lebih lambat dibandingkan apabila program tersebut di-compile terlebih dahulu dan dijalankan dalam bentuk binary, namun hasilnya tetap dapat dijadikan perbandingan dan referensi untuk pengembangan sistem selanjutnya yang lebih baik.

Dalam menyusun dan membuat penjelasan mengenai PHP ini, penulis menggunakan PHP dan manual untuk PHP versi 4. Untuk spesifikasi lengkap PHP yang digunakan oleh penulis dapat dilihat pada halaman lampiran.

Sintaks Dasar

Seperti yang sudah dicontohkan dalam meletakan kode PHP dalam script HTML diatas, kode PHP dipisahkan dengan sebuah simbol tag khusus atau disebut juga delimiter sebagai pengenal apakah kode setelah tag tersebut akan diperlakukan sebagai kode HTML, kode PHP atau kode lainnya.

Ada beberapa macam cara untuk mendefinisikan kode PHP dalam sebuah halaman web (HTML) :

```
<? echo ("this is the simplest, an SGML processing
instruction\n"); ?>
<?= expression ?>  This is a shortcut for "<? echo expression ?>"
<?php echo("if you want to serve XHTML or XML documents, do like
this\n"); ?>
<script language="php">
echo ("some editors (like FrontPage) don't
like processing instructions");
</script>
<% echo ("You may optionally use ASP-style tags"); %>
<%= $variable; # This is a shortcut for "<%echo .." %>
```

Cara pertama adalah cara standar untuk menulis kode PHP, sedangkan cara kedua disarankan oleh pengembang PHP jika melakukan pemrograman XML

dan kompatibilitas XHTML di masa depan. Cara ketiga adalah yang paling tidak umum diterapkan, sementara cara keempat dapat digunakan setelah meng-enable-kan option ***asp_tags*** pada file konfigurasi PHP.

Contoh lain yang dapat dilakukan menyangkut penggunaan delimiter PHP dalam memproses struktur kontrol :

```
<?php
if (boolean-expression) {
?>
<strong>This is true.</strong>
<?php
} else {
?>
<strong>This is false.</strong>
<?php
}
?>
```

PERPUSTAKAAN INSTITUT TEKNOLOGI SEPULUH - NOPEMBER

Variabel dan Tipe Data

Dalam PHP sebuah variabel diawali dengan karakter '$' (dollar, tanpa tanda petik) dan diikuti dengan nama variabelnya. Variabel dalam PHP bersifat case-sensitive artinya huruf besar dan huruf kecil dalam sebuah nama variabel tidak sama. Sebuah nama variabel akan valid jika diawali dengan sebuah huruf atau garis bawah, diikuti dengan sembarang angka, huruf, dan atau garis bawah, dimana huruf berarti karakter dengan nilai antara 0x7f – 0xFF yaitu a-z,A-Z ditambah karakter ASCII 127-255.

Contoh penggunaan nama variabel :

```
<?php
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";        // hasil output: "Bob, Joe"
$4site = 'not yet';       // invalid; dimulai dengan sebuah angka
$_4site = 'not yet';      // valid; dimulai dengan garis bawah
$täyte = 'mansikka';      // valid; 'ä' memiliki ASCII 228.
?>
```

Melewatkan nilai antar variabel dengan operator '=' (samadengan) dapat berupa *passing by value* atau *passing by reference*. *Passing by value* artinya kita meng-copy nilai dari sebuah variabel ke variabel lain, sehingga nilai variabel-variabel tersebut saling berdiri sendiri satu sama lain. Sementara *passing by reference* artinya kita menunjuk ke sebuah alamat memori variabel yang sama sehingga variabel-variabel tersebut adalah identik satu sama lain, diterapkan dengan menambahkan karakter '&' (dan, tanpa tanda petik) pada variabel yang dijadikan referensi.

Contoh penggunaannya *passing by value*:

```
<?php
$a = 5;
$a = $b;
$b = 3;
echo $a;    // hasil output: 5
echo $b;    // hasil output: 3
?>
```

Contoh penggunaan *passing by reference*:

```
<?php
$a = 5;
$b = &$a;
$b = 3;
echo $a;        // hasil output: 3
echo $b;        // hasil output: 3
                // $a adalah $b dan $b adalah $a
?>
```

Tipe-tipe data yang didukung oleh PHP adalah :

4 buah tipe skalar, yaitu :

Boolean

Hanya memiliki nilai TRUE dan FALSE.

Integer

Nilai bilangan integer yang didukung oleh PHP adalah sebanyak 32bit signed, yang dapat dituliskan dalam notasi desimal (basis-10), oktal (basis-8), atau heksadesimal (basis-16) dengan nilai – (negatif) dan + (positif).

Floating-point number (float)

Ukuran tipe data float bersifat platform-depended, sehingga bergantung pada sistem tempatnya dijalankan, namun PHP mampu mendukung bilangan pecahan dengan batas presisi hingga 14 digit di belakang koma (64 bit).

String

String adalah sebuah kumpulan berurut karakter-karakter. Sebuah karakter dalam string berukuran 1 byte sehingga terdapat 256 jenis karakter dalam PHP, sementara itu tidak ada batasan panjang sebuah string dalam PHP karena prosesnya dilakukan secara dinamis.

(2 buah tipe kumpulan)

Array

Sebuah array dalam PHP sebenarnya adalah sebuah *map* (peta) yang berurutan (*ordered-map*), dimana map adalah sebuah tipe yang memetakan sebuah *values* (nilai) ke sebuah *keys* (kunci/index). Pada tipe map tersebut telah dilakukan optimalisasi sehingga dapat dianggap dan digunakan, sebagaimana halnya dalam konsep sesungguhnya dari, sebuah array, list (vektor), hastable, dictionary, collection, stack, queue, dan lain sebagainya dimana sebuah *value* dari PHP-array dapat berupa sebuah PHP-array lain demikian seterusnya.

Object

Tipe data object adalah sebuah bentuk inplementasi dan dukungan PHP terhadap konsep pemrograman berorientasi objek, dimana dalam konsep OOP sendiri sebuah object adalah penjelmaan dari sebuah class, yaitu sebuah struktur variabel yang memiliki atribut dan metode sendiri dengan sifat-sifat tertentu yang berhubungan dengan class-class lainnya.

2 buah tipe spesial, yaitu :

Resource

Adalah sebuah variabel khusus yang menunjuk dan mengarah ke sebuah fungsi-fungsi luar tambahan (external resource). Contoh Resource yang sering digunakan adalah resource untuk koneksi ke database tertentu seperti MySQL, Oracle dan komponen-komponen lain yang didukung seperti PDF dan XML.

NULL

Null adalah simbol dimana sebuah variabel tidak memiliki suatu nilai apapun didalamnya.

Struktur Kontrol

Sebuah kode sumber (*source-code*) bahasa pemrograman terdiri dari kumpulan pernyataan-pernyataan (*statements*) yang dapat berupa pemberian nilai, pemanggilan fungsi, sebuah *loop*, pengecekan kondisi, ataupun sebuah pernyataan kosong. Pernyataan adalah baris kode yang umumnya diakhiri dengan tanda titik koma.

Pernyataan-pernyataan tersebut dapat juga dilakukan pengelompokan menjadi sebuah grup-pernyataan (*statements-group*) yang berisi banyak

Fungsi

Fungsi adalah sebuah struktur yang mengelompokkan pernyataan-pernyataan kode sumber yang memiliki nama fungsi dan dapat menerima argumen-argumen sebagai nilai inputan fungsi dan mengembalikan sebuah nilai sebagai hasil dari fungsi tersebut. Fungsi dapat dipanggil dari kode di luar fungsi tersebut sehingga suatu operasi yang dikerjakan oleh fungsi tersebut dapat diringkas hanya dengan memanggil fungsi tersebut.

Fungsi dapat dibuat dengan sintak sebagai berikut

```
<?php
function nama_fungsi($argumen1,$argumen2,...){
// pernyataan-pernyataan
return $nilai_kembalian;
}
?>
```

Sementara itu PHP memiliki bermacam-macam fungsi-fungsi *built-in* yang dapat langsung digunakan untuk mempermudah pengkodean. Beberapa fungsi dalam PHP yang digunakan dalam aplikasi yang sedang dibahas ini adalah:

Fungsi-fungsi Array

array_merge, digunakan untuk menggabungkan dua atau lebih array, sintaksnya:

```
array array_merge (array array1, array array2 [, array ...])
```

array_slice, digunakan untuk membagi array menjadi array-array yang merupakan nodenya sesuai posisi yang kita tentukan posisi dan ukurannya, sintaksnya:

```
array array_slice (array array, int offset [, int length])
```

in_array, digunakan untuk pengecekan suatu nilai dalam sebuah array, menghasilkan nilai TRUE jika benar, sintaksnya:

```
bool in_array (mixed needle, array haystack [, bool strict])
```

sizeof, digunakan untuk mendapatkan ukuran (jumlah node) sebuah array , sintaksnya:

```
int sizeof (mixed var)
```

Fungsi-fungsi String

explode, digunakan untuk membagi sebuah string menjadi array string hasil dengan menyebutkan string pemisah yang terdapat didalamnya, sintaksnya:

```
array explode (string separator, string string [, int limit])
```

strlen, digunakan untuk mendapatkan panjang sebuah string, sintaksnya:

```
int strlen (string str)
```

strpos, digunakan untuk mendapatkan posisi awal sebuah string yang dicari dalam sebuah string, sintaksnya:

```
int strpos (string haystack, string needle [, int offset])
```

substr, digunakan untuk mengembalikan sebuah string dalam string yang telah kita tentukan posisi dan ukurannya, sintaksnya:

```
string substr (string string, int start [, int length])
```

Fungsi-fungsi MySQL

mysql_connect, digunakan untuk membuat koneksi ke sebuah server MySQL, sintaksnya:

```
resource mysql_connect ([string hostname [:port]
[:/path/to/socket] [, string username [, string password]]])
```

mysql_select_db, digunakan untuk memilih sebuah database SQL, sintaksnya:

```
bool mysql_select_db (string database_name [, resource
link_identifier])
```

mysql_query, digunakan untuk mengirim sebuah query SQL ke database, sintaksnya:

```
resource mysql_query (string query [, resource link_identifier])
```

mysql_fetch_array, digunakan untuk menyandangkan baris hasil dari query SQL menjadi sebuah array, sintaknya:

```
array mysql_fetch_array (resource result [, int result_type])
```

2.2.2. MySQL

MySQL adalah sebuah aplikasi database yang dibuat dan dikembangkan oleh MySQL AB. MySQL berasal dari kalangan dan komunitas open source sehingga lisensinya adalah berbasis GPL (GNU General Public License.

MySQL adalah sebuah Relational Database Management System, yaitu sebuah konsep manajemen basis data menggunakan tabel-tabel yang dapat saling berhubungan satu sama lainnya sehingga membentuk sebuah sistem yang kompleks dan terpadu.

MySQL adalah sebuah database SQL dimana data-data didalam MySQL diakses dengan sintaks-sintaks standar SQL (Structured Query Language).

Sintaks Dasar

MySQL menggunakan bahasa SQL untuk pengaksesan datanya sehingga sintaks-sintaks yang didukung adalah sintaks-sintaks standar SQL. Contoh penggunaan sintaks-sintaks dasar SQL untuk mengakses tabel berikut.

Contoh tabel: mhs

```
+--------+----------+-------------+
|   id   | name     |    city     |
+--------+----------+-------------+
| 5101   | Anton    | Jakarta     |
| 5102   | Iwan     | Bandung     |
| 5104   | Joko     | Surabaya    |
| 5105   | Abdi     | Malang      |
| 5106   | Tika     | Jogjakarta  |
+--------+----------+-------------+
```

Sintaks Select

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy
```

Perintah mengambil semua data dari semua field dari tabel mhs

```
mysql> Select * From mhs;
```

Hasil:

```
+--------+----------+-------------+
|   id   | name     |    city     |
+--------+----------+-------------+
| 5101   | Anton    | Jakarta     |
| 5102   | Iwan     | Bandung     |
| 5104   | Joko     | Surabaya    |
| 5105   | Abdi     | Malang      |
| 5106   | Tika     | Jogjakarta  |
+--------+----------+-------------+
```

Perintah mengambil nilai field id dari tabel mhs

mysql> Select id From mhs;

Hasil:

```
+--------+
|   id   |
+--------+
| 5101   |
| 5102   |
| 5104   |
| 5105   |
| 5106   |
+--------+
```

Perintah mengambil record tertentu

mysql> Select * from mhs where id=5101;

Hasil:

```
+--------+----------+-------------+
|   id   | name     |    city     |
+--------+----------+-------------+
| 5101   | Anton    | Jakarta     |
+--------+----------+-------------+
```

Sintaks lengkap Select:

```
SELECT [STRAIGHT_JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT]
[SQL_BUFFER_RESULT]
       [HIGH_PRIORITY]
       [DISTINCT | DISTINCTROW | ALL]
    select_expression,...
    [INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
    [FROM table_references
        [WHERE where_definition]
        [GROUP BY {unsigned_integer | col_name | formula} [ASC |
DESC], ...]
        [HAVING where_definition]
        [ORDER BY {unsigned_integer | col_name | formula} [ASC |
DESC] ,...]
        [LIMIT [offset,] rows]
        [PROCEDURE procedure_name]
        [FOR UPDATE | LOCK IN SHARE MODE]]
```

Sintaks Insert

INSERT INTO which_table
[(define_fields_to_inserted)]
VALUES
(values_to_inserted_to_each_fields)

Perintah mengisi suatu record baru pada tabel mhs

mysql> Insert Into mhs (id,name,city) Values(5107,'Sinta','Semarang');

Isi tabel saat ini menjadi:

id	name	city
5101	Anton	Jakarta
5102	Iwan	Bandung
5104	Joko	Surabaya
5105	Abdi	Malang
5106	Tika	Jogjakarta
5107	Sinta	Semarang

Sintaks lengkap Insert:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name [(col_name,...)]
        VALUES (expression,...),(...),...
or  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name [(col_name,...)]
        SELECT ...
or  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name
        SET col_name=expression, col_name=expression, ...
```

Sintaks Update

UPDATE which_table
SET field_name=new_value
WHERE conditions_to_satisfy

Perintah meng-update tabel mhs pada field id=5104

mysql> Update name='Reva',city='Balikpapan' Where id=5104

Isi tabel saat ini menjadi:

id	name	city
5101	Anton	Jakarta
5102	Iwan	Bandung
5104	Reva	Balikpapan
5105	Abdi	Malang
5106	Tika	Jogjakarta

Sintaks lengkap Update:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1, [col_name2=expr2, ...]
    [WHERE where_definition]
    [LIMIT #]
```

Sintaks Delete

DELETE FROM which_table
WHERE conditions_to_satisfy

Perintah menghapus record pada tabel mhs

mysql> Delete From mhs Where id=5101

Isi tabel saat ini menjadi:

id	name	city
5102	Iwan	Bandung
5104	Reva	Balikpapan
5105	Abdi	Malang
5106	Tika	Jogjakarta

Sintaks lengkap Delete:

```
DELETE [LOW_PRIORITY] FROM tbl_name
    [WHERE where_definition]
    [LIMIT rows]
```

BAB 3

ANALISA DAN PERANCANGAN

BAB 3
ANALISA DAN PERANCANGAN

3.1. DESKRIPSI

Pada penyusunan tugas akhir ini akan dibangun sebuah sistem MT yang sesuai dengan teori-teori yang telah dibahas pada bab sebelumnya dengan berbagai tambahan dan pengembangan yang disesuaikan dengan kebutuhan-kebutuhan dan permasalahan yang dihadapi. Pada bab ini akan dibahas gambaran dan deskripsi kebutuhan sistem aplikasi secara lengkap yang akan dibangun berikut hal-hal yang nantinya akan dibahas pada bab-bab selanjutnya.

Adapun sistem aplikasi yang akan dibahas ini diberi sebutan Transovanay (selanjutnya disingkat dengan ToV). ToV secara umum dideskripsikan sebagai sebuah aplikasi yang digunakan untuk menerjemahkan teks dari suatu Bahasa Sumber (BS) ke dalam teks padanannya dari suatu Bahasa Tujuan (BT) dengan mengambil studi kasus Bahasa Inggris sebagai Bahasa Sumber dan Bahasa Indonesia sebagai Bahasa Tujuan. ToV memiliki engine yang menggunakan pendekatan dan konsep dasar dari arsitektur MT DIRECT/TRANSFORMER yang telah disesuaikan.

Sistem ToV didisain sejak awal dengan asumsi-asumsi yang terjadi dalam dunia nyata sebagai berikut :

- User dapat dengan bebas menentukan aturan-aturan penerjemahan yang diinginkannya dan tidak selalu bergantung pada aturan *default* dari sistem aplikasi tersebut.

- Proses *post-editing* oleh user masih sangat dibutuhkan dan masih memiliki prosentase yang cukup besar dari keseluruhan waktu pekerjaan penerjemahan, oleh karena itu sistem juga perlu menampilkan semua hasil ambiguitas yang terjadi untuk mempermudah user melakukan editing hasil.
- Proses parsing aturan dinamis dapat dicapai dengan menerapkan arsitektur Direct/Transformer.

Meskipun ToV pada akhirnya dapat digunakan untuk memproses lebih dari satu kalimat, *engine* ToV dioptimasi hanya untuk penerjemahan satu kalimat Bahasa Sumber menjadi satu kalimat Bahasa Tujuan, bukan paragraf atau dokumen keseluruhan, sehingga untuk selanjutnya seluruh proses penerjemahan yang akan dibahas diawali dengan asumsi bahwa aplikasi telah memisahkan kalimat-kalimat tersebut menjadi sebuah kalimat tunggal dan proses-proses ToV dijalankan hanya pada satu kalimat pada satu waktu.

Alur yang dijalankan oleh ToV dapat dibagi menjadi empat proses utama, yang masing-masing melakukan pengetesan terhadap aturan-aturan yang berbeda yaitu :

Proses Strukturisasi Awal BS, yaitu pemecahan kalimat menjadi kata-kata pembentuknya, baik yang terdaftar maupun yang tidak dalam kamus untuk selanjutnya disusun menjadi struktur awal (*start structure*) yang disebut genuineStructures.

Proses Strukturisasi BS, yaitu proses analisa struktur awal menjadi sebuah atau beberapa struktur Bahasa Sumber yang sesuai dengan aturan grammar yang telah

didefinisikan sebelumnya. Struktur Bahasa Sumber ini disebut srcStructures (singkatan dari *source structures*).

Proses Strukturisasi BT, yaitu proses perubahan atau transformasi struktur Bahasa Sumber menjadi struktur Bahasa Tujuan yang juga sesuai dengan aturan perubahan yang telah didefinisikan sebelumnya. Struktur Bahasa Tujuan ini disebut destStructures (singkatan dari *destination structures*).

Proses Filterisasi BT, yaitu proses perubahan bentuk Bahasa Tujuan dari aturan-aturan modifikasi Bahasa Tujuan yang memenuhi. Struktur yang dihasilkan dari proses ini disebut destResults.

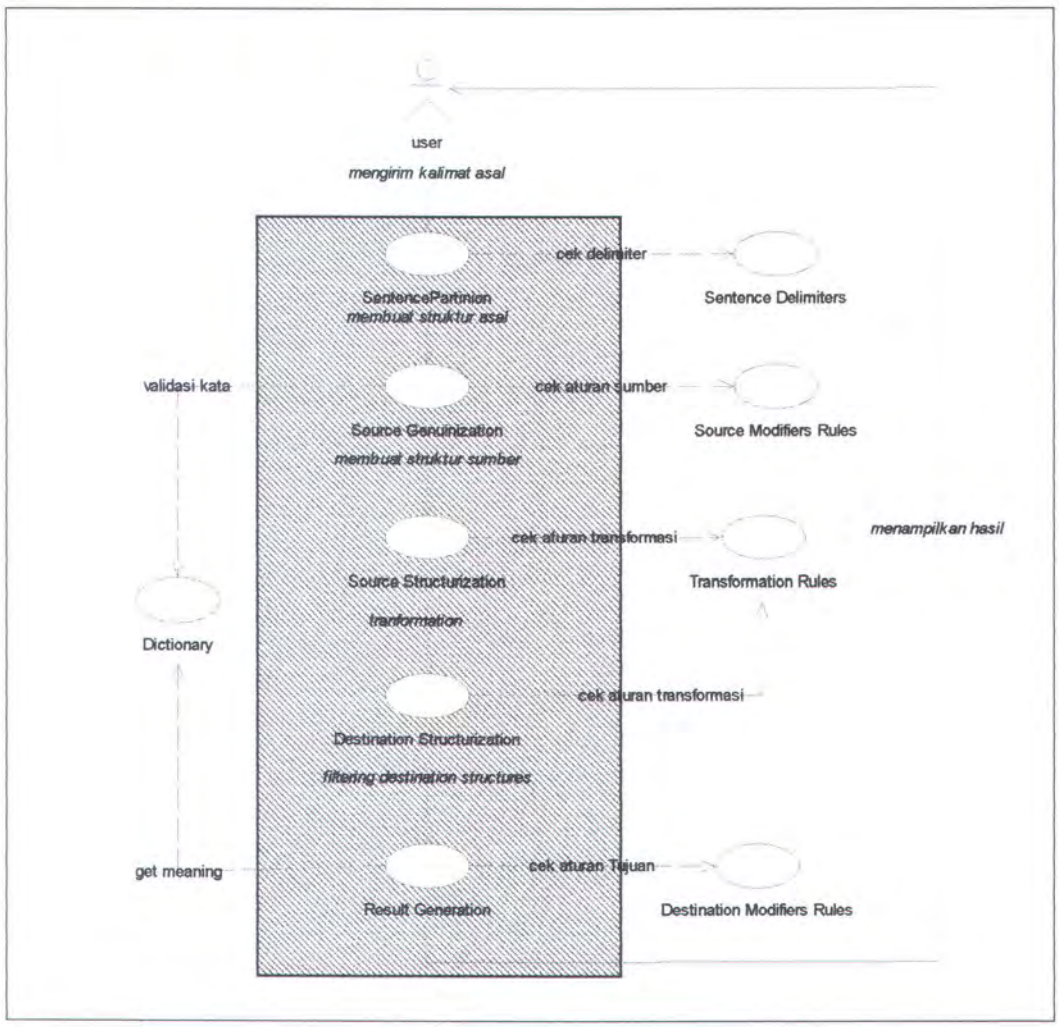
Adapun proses-proses yang dapat melewatkan ambiguitas, sehingga membuat proses analisa menjadi lebih panjang, dapat disebutkan disini yaitu dua proses awal untuk ambiguitas struktural dan proses terakhir melewatkan ambiguitas leksikal. Sebagai proses tambahan lain yang dilakukan dalam aplikasi ini adalah :

Proses pemecahan kalimat, sebagaimana telah disebutkan bahwa ToV hanya memproses satu kalimat pada satu waktu maka ToV harus memecah teks inputan menjadi kalimat-kalimat yang akan diproses secara terpisah dalam proses ini. Proses ini dikerjakan sekali saja paling awal ketika aplikasi menerima inputan teks Bahasa Sumber.

Proses pembentukan tampilan, pada proses ini dilakukan perubahan tampilan akhir yang bertujuan membuat program menampilkan berbagai alternatif hasil sesuai dengan prioritas yang didapatnya dan juga antarmuka yang dapat dimanfaatkan user untuk melakukan pengeditan dengan lebih nyaman.

Kedua proses tambahan diatas bukan merupakan komponen yang penting dalam engine yang akan dibahas sehingga untuk selanjutnya tidak akan dijelaskan secara mendetail.

3.2. USE CASE DIAGRAM



Gambar 3 - 1
Use case diagram

Diagram Use Case menjelaskan aktifitas-aktifitas yang terjadi dalam proses penerjemahan pada lingkungan sistem aplikasi ToV ini. Diagram Use Case

dimulai ketika seorang user memasukkan sebuah inputan string kalimat Bahasa Sumber ke dalam textbox input untuk diproses oleh sistem. Seperti yang nampak pada gambar 3-1, sistem kemudian menjalankan empat proses utama yaitu proses Pembagian Kalimat (Sentence Partition), strukturisasi awal (genuine structurization), strukturisasi sumber (source structurization), strukturisasi tujuan (destination structurization), dan pembentukan hasil (result generation).

Aktifitas lain yang menyertai proses-proses tersebut adalah pengambilan data untuk mendapatkan aturan-aturan dan nilai-nilai atribut sebuah token kata dalam database di luar sistem tersebut. Hasil yang diperoleh kemudian akan ditampilkan kembali kepada user.

3.3. CLASS DIAGRAM

Kelas digunakan dalam teknik pemrograman untuk mengelompokkan struktur data yang sama dan menerapkannya sebagai objek yang dapat memiliki nilai atribut dan menjalankan prosedur atau method yang khusus untuk masing-masing objek.

- Kelas SrcToken

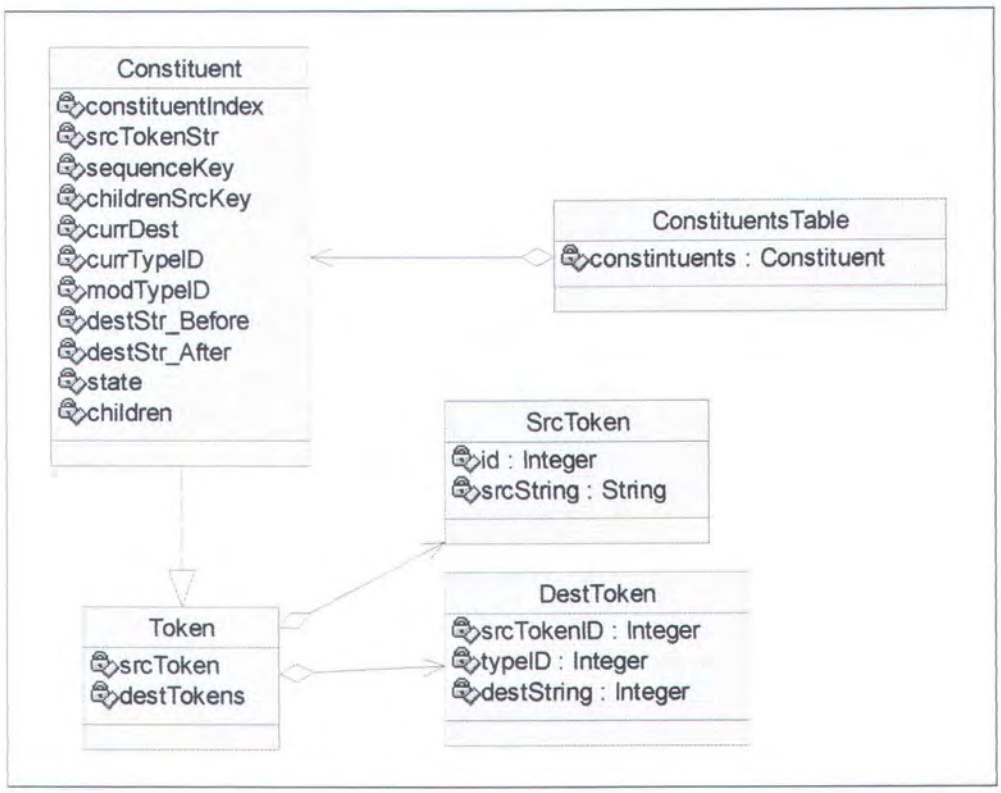
  Adalah kelas yang menyimpan informasi token BS. Implementasinya hanya pada kelas Token.

- Kelas DestToken

  Adalah kelas yang menyimpan informasi token BT. Implementasinya hanya pada kelas Token.

- Kelas Token

Adalah kelas yang menyimpan dan mengembalikan informasi mengenai sebuah token.



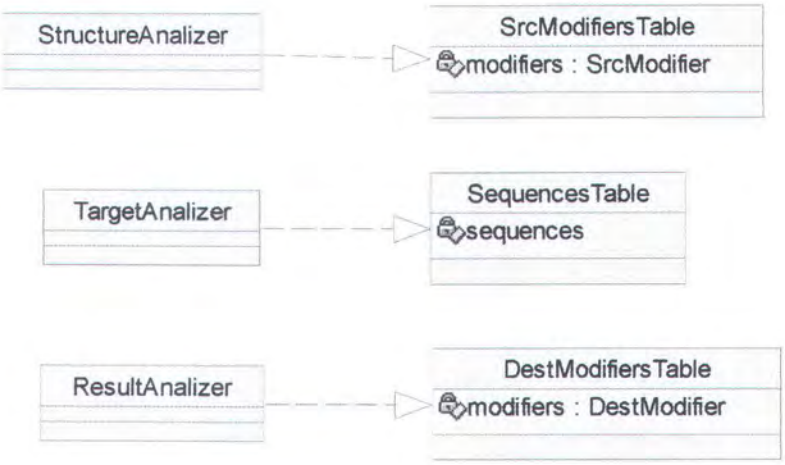
Gambar 3 - 2
Class Diagram untuk kelas ConstituentsTable

- Kelas Constituent

Adalah kelas yang digunakan untuk merepresentasikan sebuah unit terkecil dari kalimat. Obyek kelas ini menyimpan dan mengembalikan berbagai atribut informasi mengenai sebuah unit seperti string token BS, tipe kata, dan sebagainya.

- Kelas ConstituentsTable

  Adalah kelas yang digunakan untuk menyusun struktur kalimat yang terdiri dari obyek-obyek constituent menjadi suatu struktur kalimat yang utuh. Implementasi obyek kelas ini ada dalam kelas Sentence. Hubungan kelima kelas ini ditunjukkan pada gambar 3-2.



Gambar 3 - 3
Class Diagram untuk kelas-kelas proses analisa

- Kelas StructureAnalizer

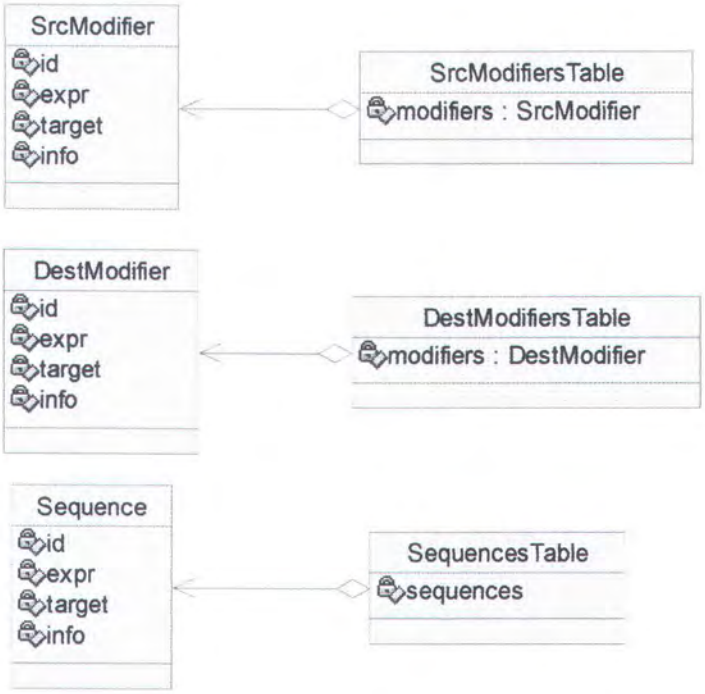
  Adalah kelas yang digunakan untuk melakukan pembuatan peta aturan transformasi dan melakukan proses analisa atau parsing terhadap struktur kalimat Bahasa Sumber.

- Kelas TargetAnalizer

  Adalah kelas yang digunakan untuk melakukan proses transformasi abstrak dan perubahan struktur Bahasa Sumber menjadi struktur Bahasa Tujuan.

- Kelas ResultAnalizer

Adalah kelas yang digunakan untuk melakukan proses pembentukan kalimat Bahasa Tujuan dengan mengecek aturan modifikasi Bahasa Tujuan sehingga membentuk kalimat Bahasa Tujuan yang lebih sesuai dengan aturan grammar Bahasa Tujuan yang sesungguhnya.



Gambar 3 - 4
Class Diagram untuk kelas-kelas aturan

- Kelas Sequence

Adalah kelas yang digunakan untuk menyimpan dan mengembalikan sebuah nilai aturan transformasi.

- Kelas SequencesTable

  Adalah kelas yang digunakan untuk memuat, menyimpan dan mengembalikan nilai-nilai aturan transformasi dari obyek kelas SrcSequence.

- Kelas SrcModifier

  Adalah kelas yang digunakan untuk menyimpan nilai sebuah aturan modifikasi Bahasa Sumber.

- Kelas SrcModifiersTable

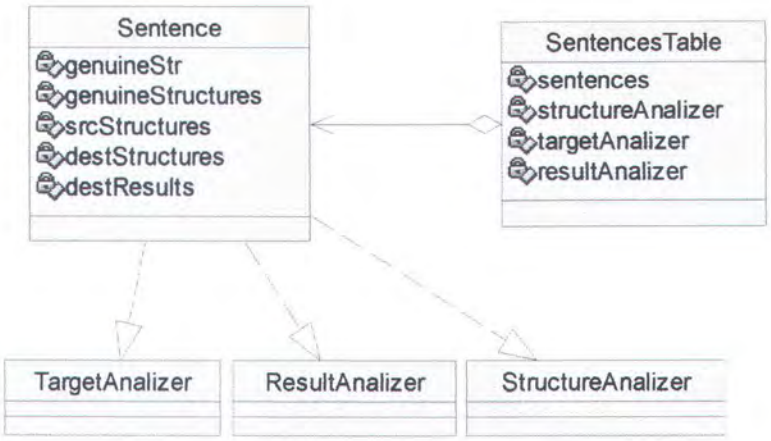
  Adalah kelas yang digunakan untuk memuat, menyimpan dan mengembalikan nilai-nilai aturan modifikasi Bahasa Sumber dari obyek kelas ScrModifier.

- Kelas DestModifier

  Adalah kelas yang digunakan untuk menyimpan nilai sebuah aturan modifikasi Bahasa Tujuan.

- Kelas DestModifiersTable

  Adalah kelas yang digunakan untuk memuat, menyimpan, dan mengembalikan nilai-nilai aturan modifikasi Bahasa Tujuan dari obyek kelas DestModifier.



Gambar 3 - 5
Class Diagram untuk kelas SentencesTable

- Kelas Sentence

  Adalah kelas yang digunakan untuk membuat dan menyimpan informasi struktur-strukur genuine, source, destination, dan result dari sebuah kalimat Bahasa Sumber.

- Kelas SentencesTable

  Adalah kelas yang digunakan untuk menyimpan daftar obyek-obyek kelas Sentence, memberi perintah kepada masing-masing obyek sentence, dan menyimpan informasi mengenai teks input secara keseluruhan. Hubungan kelas SentencesTable dengan kelas lainnya ditunjukkan pada gambar 3-5.

Aturan-aturan yang digunakan dalam sistem aplikasi ini dibagi menjadi lima buah aturan sebagai berikut:

- Karakter pembatas kalimat (*Sentence Delimiter*)

  Yaitu aturan yang berupa karakter-karakter yang digunakan untuk memisahkan suatu kalimat dengan kalimat lain pada Bahasa Sumber dimana sebuah kalimat diasumsikan sebagai suatu bentuk yang memiliki unsur-unsur pembentuk kalimat secara lengkap meskipun bukan sebagai sebuah kalimat sempurna. Contoh karakter pemisah kalimat dalam Bahasa Inggris adalah "." (titik) dan "," (koma).

- Tipe token (*Token Type*)

  Yaitu sebuah aturan yang mendaftar seluruh unsur-unsur yang mewakili sebuah kata atau token dalam tata bahasa Bahasa Sumber dan Bahasa Tujuan.

- Aturan modifikasi bahasa sumber (*Source Modification Rule*)

  Yaitu sebuah aturan yang merinci bentuk-bentuk perubahan morfologis yang kemungkinan terjadi pada sebuah kata Bahasa Sumber. Contoh aturan-aturan ini adalah aturan untuk kata-kata yang disingkat dan perubahan bentuk tense.

- Aturan transformasi (*Applied Transformation Rules*)

  Yaitu sebuah aturan pembentuk struktur pohon kalimat Bahasa Sumber dan padanannya dalam struktur kalimat Bahasa Tujuan.

- Aturan transformasi yang ditolak (*Rejected Transformation Rules*)

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

Yaitu sebuah aturan yang ditolak dalam struktur kalimat Bahasa Sumber karena tidak mungkin aturan demikian diterima oleh tata bahasa Bahasa Sumber sesungguhnya.

- Aturan transformasi abstrak (*Abstract Transformation Rules*)

  Yaitu sebuah aturan yang merangkai struktur kalimat Bahasa Sumber menjadi struktur kalimat Bahasa Tujuan dengan penambahan dan perubahan kata-kata yang biasa dilakukan pada proses penerjemahan dengan tidak memandang bentuk struktur transformasinya.

- Aturan modifikasi bahasa tujuan (*Destination Modification Rules*)

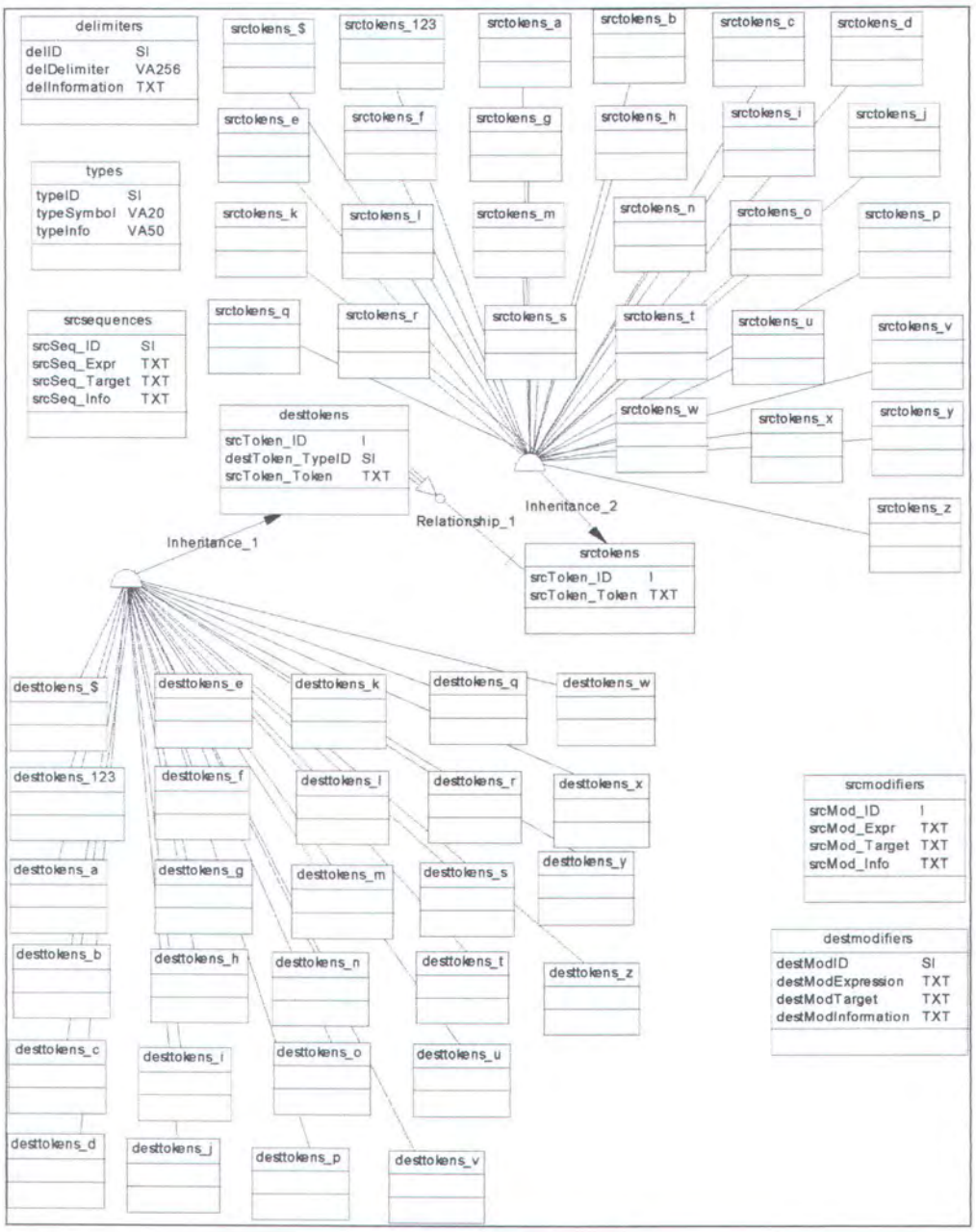
  Yaitu sebuah aturan yang merubah bentuk kata-kata Bahasa Tujuan sehingga sedapat mungkin memiliki makna yang dimaksudkan dalam kalimat Bahasa Sumber.

3.4. CONCEPTUAL DATA MODEL

Conceptual Data Model menerangkan tentang struktur dan relasi antar tabel pada suatu sistem manajemen basis data yang digunakan oleh sistem aplikasi. Adapun tabel-tabel yang digunakan pada sistem ini mencakup hal-hal sebagai berikut, sebagaimana ditunjukkan pada gambar 3-6.

Tabel delimiter, digunakan untuk menyimpan data karakter-karakter pemisah kalimat.

Tabel types, untuk menyimpan data tipe-tipe kata dan konstituen yang akan dijadikan dasar analisa dari seluruh proses-proses yang dilakukan.



Gambar 3 - 6
Database Conceptual Data Model

Tabel srctokens, untuk menyimpan data string token atau kata dalam Bahasa Sumber, tabel ini nantinya akan diturunkan menjadi tabel-tabel lain dengan struktur yang sama untuk menampung data yang diurutkan berdasarkan abjad awal dari string token.

Tabel desttokens, untuk menyimpan data string token atau kata dalam Bahasa Tujuan, tabel ini nantinya akan diturunkan menjadi tabel-tabel lain dengan struktur yang sama untuk menampung data yang diurutkan berdasarkan abjad awal dari string token.

Tabel srcmodifiers, untuk menyimpan data aturan modifikasi sumber.

Tabel destmodifiers, untuk menyimpan data aturan modifikasi tujuan.

Tabel sequences, untuk menyimpan data aturan transformasi.

BAB 4

IMPLEMENTASI

BAB 4
IMPLEMENTASI

4.1. KONFIGURASI PERANGKAT KERAS

Adapun spesifikasi perangkat keras dan sistem operasi komputer yang digunakan untuk pengembangan dan pelaksanaan uji coba adalah sebagai berikut:

Prosessor	Intel Pentium 4 3.06 MHz (mobile)
Kapasitas total memori fisik	512 MB DDR SD-RAM
Kapasitas efektif memori fisik	480 MB (video shared)
Kapasitas memori video adapter	32 MB (shared)
Kapasitas total fisik hard disk	80 GB
Kapasitas space efektif sistem	31.3 GB
File sistem OS	NTFS
Sistem operasi	Microsoft Windows XP Professional
Version	2002 Service Pack 1

4.2. KONFIGURASI PERANGKAT LUNAK

Spesifikasi perangkat lunak yang dibutuhkan pada sistem komputer tersebut untuk membuat sistem ToV tersebut adalah :

- Apache Web Server (versi PHPTriad [*http://www.phpgeek.com*])(Windows)
- PHP (versi 4.3.2)(Windows)
- MySQL (versi 3.23.47)(Windows)
- PHPMyAdmin (versi 2.2.3)(Windows)

- Text / HTML editor (contoh PHPCoder, Dreamweaver)
- Aplikasi Software Modelling (contoh Visio, Rational Rose, Power Designer)
- Browser (contoh Mozilla, IE)

Sementara spesifikasi perangkat lunak yang dibutuhkan pada sistem komputer yang hanya digunakan untuk implementasi sistem ToV minimal adalah:

- PHP Web Server
- MySQL Database Server
- Browser

4.3. IMPLEMENTASI MODUL

4.3.1. Aturan-aturan Analisa

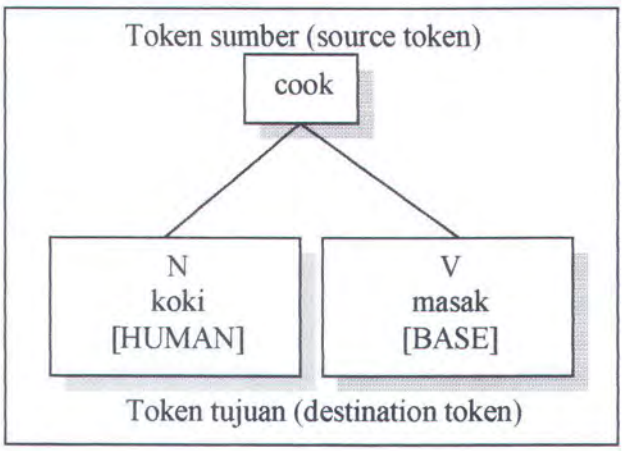
Aturan yang digunakan dalam sistem ini terbagi menjadi 4 macam aturan utama, yaitu Aturan tipe, Aturan Modifikasi Sumber, Aturan Transformasi, dan Aturan Modifikasi Tujuan.

Aturan Tipe

Aturan tipe adalah representasi dari jenis-jenis kata dalam bahasa sumber seperti noun, verb, article, dan be dalam Bahasa Inggris. Aturan tipe digunakan di dalam Kamus untuk membagi kata ke dalam tipenya masing-masing dimana aturan ini disimpan di dalam tabel desttokens (destination tokens / token tujuan), yaitu tabel yang menyimpan data padanan kata dari sebuah kata sumber pada tabel srctokens (source tokens / token sumber), sehingga sebuah token sumber dapat memiliki beberapa padanan kata dengan tipe katanya masing-masing.

Selain aturan tipe pokok, terdapat sebuah aturan bebas yang disebut dengan atribut,yang berupa string yang menyatu di dalam token tujuan yang merupakan representasi dari informasi tambahan dari sebuah kata. Atribut digunakan sebagai parameter pengujian konstituen pada aturan-aturan yang lain sehingga sifatnya adalah *optional* dan bukan merupakan sebuah *reserved word* dari aplikasi.

Contoh sebuah entri Kamus secara logika ditunjukkan pada gambar 4-1. Pada gambar tersebut sebuah token sumber (source token) hanya memiliki data string token sumber saja, sementara token tujuan (destination token) memiliki data tipe kata (baris pertama), string token tujuan (baris kedua) ditambah dengan atribut (baris ketiga) yang berupa informasi mengenai makna yang berhubungan dengan string token tujuan tersebut (HUMAN berarti koki adalah sebuah kata benda untuk menunjuk kepada entitas manusia, BASE berarti string token tujuan adalah merupakan sebuah kata dasar).



Gambar 4 - 1
Contoh entri dalam database Kamus

Aturan Modifikasi Sumber

Aturan modifikasi sumber adalah aturan yang memiliki fungsi untuk melakukan pengujian terhadap bentuk-bentuk perubahan bentuk kata pada kalimat sumber. Aturan Modifikasi Sumber dibagi menjadi Ekspresi, adalah bentuk pengujian konstituen yang memenuhi untuk pengujian selanjutnya, dan Target, yaitu bentuk pengujian untuk menghasilkan konstituen yang sesuai dengan aturan yang ditulis. Aturan ini dijalankan pada proses pertama yaitu proses Strukturisasi Awal yang dijelaskan pada subbab 4.3.2.

Contoh perubahan bentuk kata adalah apabila terjadi akronim dan perubahan bentuk waktu yang dapat membentuk suatu tipe kata tersendiri, seperti yang diberikan pada contoh kalimat berikut.

He's reading a new book.

Dengan asumsi bahwa data Kamus telah berisi daftar kata-kata dasar yaitu

he PRONOUN dia;ia[SUBJECT|SINGLE|P3|MALE]
is BE adalah[SINGLE|PRESENT]
read V (untuk verb) baca[BASE]
a ART (untuk article) sebuah[SINGLE]
new ADJ (untuk adjective) baru
book N (untuk noun) buku

Struktur kalimat diatas sebelum dikenai aturan modifikasi sumber menjadi

[* he's][* reading][ART a][ADJ new][N book]

Dalam struktur tersebut ada dua token yang tidak dikenali (disimbolkan dengan '*' (tanda bintang) yang merupakan tipe *default* dari Aturan Tipe) yaitu [* he's] dan [* reading]. Untuk mendapatkan bentuk konstituen yang benar diadakan pengujian terhadap aturan-aturan modifikasi sumber sebagai berikut.

[* he's] , karena dalam bahasa inggris bentuk kata he's adalah bentuk akronim wajar untuk kata 'he is' maka aturan modifikasi sumber dapat berbunyi 'jika menemui kata he's, buat dua konstituen baru yaitu [PRONOUN he] dan [BE is]', bentuk aturannya adalah :

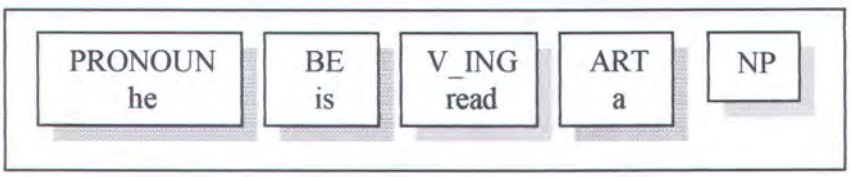
Ekspresi: he's
Target: [RETURNWORD=he][RETURNWORD=is]

[* reading], bentuk ini adalah bentuk perubahan verb ditambah prefiks –ing yang menjelaskan bahwa kata tersebut memiliki tipe kata sendiri yaitu V_ING (untuk verb_ing) (karena bentuk Verb+ing adalah bentuk umum perluasan Verb) sehingga bisa dibuat aturan modifikasi sumber yang berbunyi 'jika ditemui sebuah kata berakhiran string –ing, lihat string sebelumnya dan cek di database apakah tipe katanya adalah V, jika benar buat konstituen dengan tipe modifikasi V_ING [V_ING read]', bentuk aturannya adalah :

Ekspresi: *[SUBSTR=-3,3,ing]
Target: [REMOVE=-3,3,ing] [GETTYPE=V] [SETMODTYPE=V_ING] [RETURNWORD]

Sehingga struktur awal kalimat tersebut menjadi seperti dibawah ini dan gambar strukturnya ditujukkan pada gambar 4-2.

[PRONOUN he][BE is][V_ING read][ART a][ADJ new][N book]



Gambar 4 - 2
Struktur konstituen awal kalimat sumber

Apabila sebuah konstituen tidak dapat melewati semua bentuk aturan modifikasi sumber yang diujikan, maka konstituen tersebut akan di-set sebagai *unknown*. Sintaks lengkap Aturan Modifikasi Sumber yang dapat diterima dapat dilihat pada bagian lampiran A1.

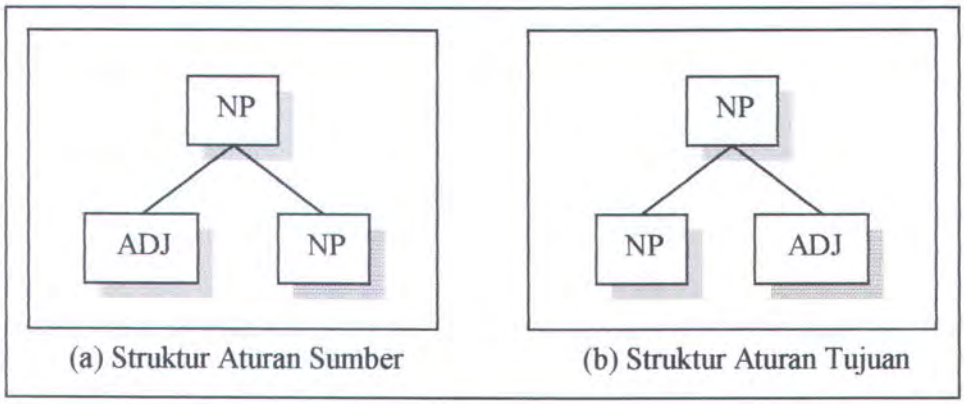
Aturan Transformasi

Aturan Transformasi adalah aturan yang berfungsi untuk membentuk sebuah struktur pohon konstituen sumber dari struktur awal pada proses Strukturisasi Sumber dan merubahnya menjadi struktur pohon konstituen tujuan pada proses Strukturisasi Tujuan. Aturan Transformasi juga terdiri dari Ekspresi, yaitu aturan untuk pembentukan struktur sumber, dan Target, yaitu aturan pembentukan stuktur tujuan. Sintaks Aturan Transformasi selengkapnya dapat dilihat pada bagian lampiran A2. Aturan Transformasi dibagi menjadi tiga macam aturan yang penerapannya berbeda namun tetap dalam rangka sebagai aturan yang melakukan perubahan Struktur Sumber menjadi Struktur Tujuan yaitu :

Aturan transformasi yang diterapkan (*applied transformation rules*), adalah aturan transformasi yang merubah struktur pohon konstituen secara fisik sehingga dapat menghasilkan sebuah struktur yang sesuai atau mendekati struktur kalimat dalam tata bahasa sumber yang sesungguhnya. Pengujian Ekspresi aturan ini diterapkan pada proses Strukturisasi Sumber, sedangkan pengujian Target aturan ini dilakukan pada proses Strukturisai Tujuan. Contoh Aturan Transformasi :

Ekspresi	**Target**
NP:(1.N)	
NP:(1.ART)(2.NP)	
NP:(1.ADJ)(2.NP)	(2)(.yang)(1)

Aturan Ekspresi merupakan bentuk representasi dari pembuatan struktur pohon yaitu *Parent Node* untuk tipe kata di sebelah kiri delimiter ':' (tanda petik). Pada contoh Ekspresi baris ketiga berarti dibaca 'jika ditemui sebuah konstituen Adjective yang berurutan langsung dengan sebuah konstituen Noun Phrase, maka gabungkan menjadi sebuah konstituen Noun Phrase baru sebagai *Parent Node*-nya', seperti yang ditunjukkan pada gambar 4-3.



(a) Struktur Aturan Sumber (b) Struktur Aturan Tujuan

Gambar 4 - 3
Contoh bentuk perubahan akibat aturan transformasi

Jika aturan Targetnya tidak diisi berarti struktur tujuan dibiarkan apa adanya, sementara jika diisi sebagaimana pada contoh Target baris ketiga dibaca 'lakukan pengurutan ulang children node pada struktur pohon Noun Phrase dan tambahkan string "yang" sebelum node Adjective'.

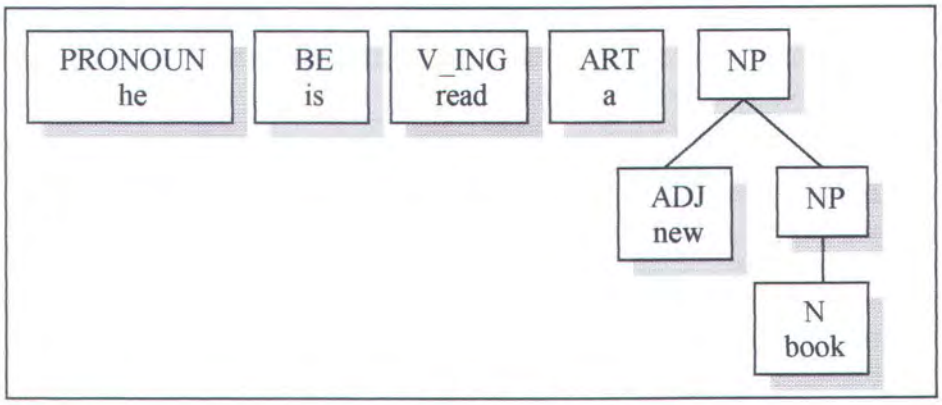
Aturan di atas memenuhi perubahan bentuk MD (menerangkan-diterangkan) dalam Bahasa Inggris menjadi bentuk DB (diterangkan-menerangkan) dalam Bahasa Indonesia untuk kasus kata sifat (adjective) yang menerangkan sebuah kata benda (Noun / Noun Phrase).

Contoh untuk sebuah kalimat sumber yang mendapat pengujian Aturan Transformasi tersebut :

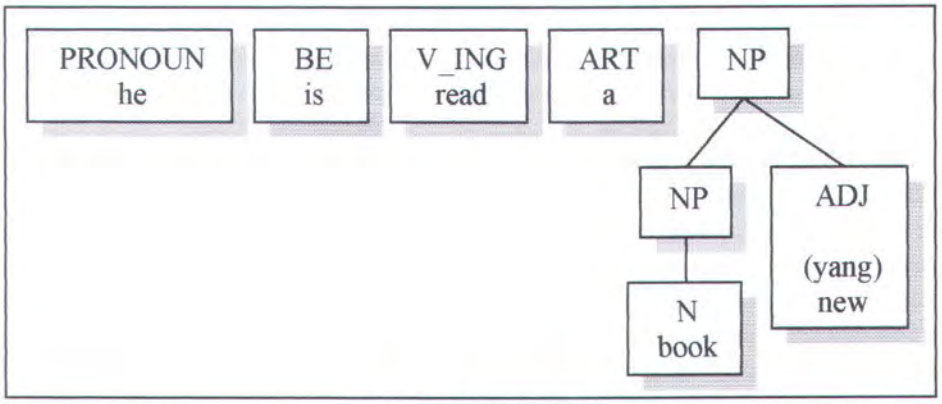
he's reading a new book

Struktur awal yang diperoleh adalah :

[PRONOUN he][BE is][V_ING read][ART a][ADJ new][N book]



(a) Struktur Aturan Sumber



(b) Struktur Aturan Tujuan

Gambar 4 - 4
Struktur Sumber dan struktur Tujuan hasil transformasi

Struktur Sumber yang dihasilkan dari aturan Ekspresi adalah :

[PRONOUN he][BE is][V_ING read] [ART a][NP ([ADJ new])([NP ([N book])])]

Struktur Sumber tersebut dapat digambarkan dalam struktur pohon seperti ditunjukkan pada gambar 4-4 (a).

Struktur Tujuan yang dihasilkan dari aturan Target adalah :

[PRONOUN he][BE is][V_ING read][ART a][NP ([NP ([N book])]) ([ADJ new])]

Struktur Tujuan tersebut dapat digambarkan dalam struktur pohon seperti ditunjukkan pada gambar 4-4 (b).

Aturan transformasi yang ditolak (*unaccepted transformation rules*), adalah aturan yang menolak sebuah struktur sesuai dengan aturan yang ditulis sehingga proses analisa tidak akan dilanjutkan untuk pembentukan struktur selanjutnya. Aturan transformasi yang ditolak memiliki tipe kata khusus secara default yaitu tipe unaccepted, disimbolkan dengan # (tanda pagar). Aturan ini diterapkan pada proses Strukturisasi Sumber. Contoh Aturan Transformasi ini adalah :

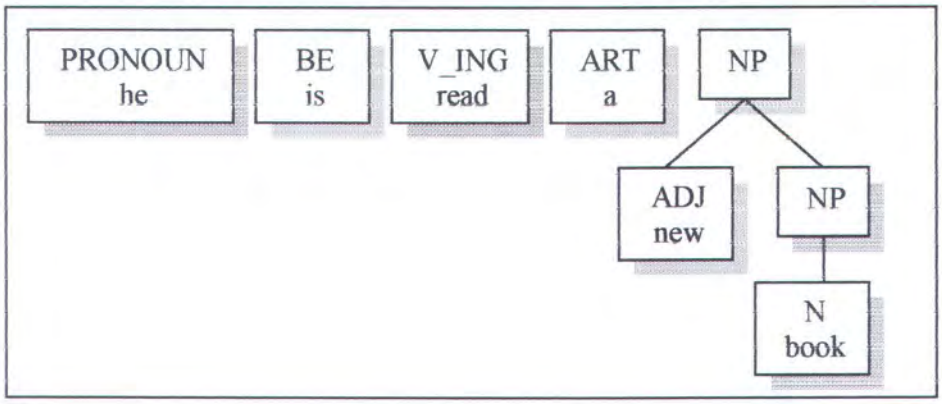
Ekspresi: # :(1.ADJ)(2.V)

Dengan adanya aturan transformasi ini maka setiap proses analisa terhadap struktur yang memiliki konstituen Adjective dan Verb yang berurutan dihentikan dan dikembalikan struktur awalnya. Aturan ini berfungsi membuat proses analisa menjadi lebih efisien dengan tidak memeriksa struktur yang pasti tidak akan memenuhi aturan bahasa yang sesungguhnya.

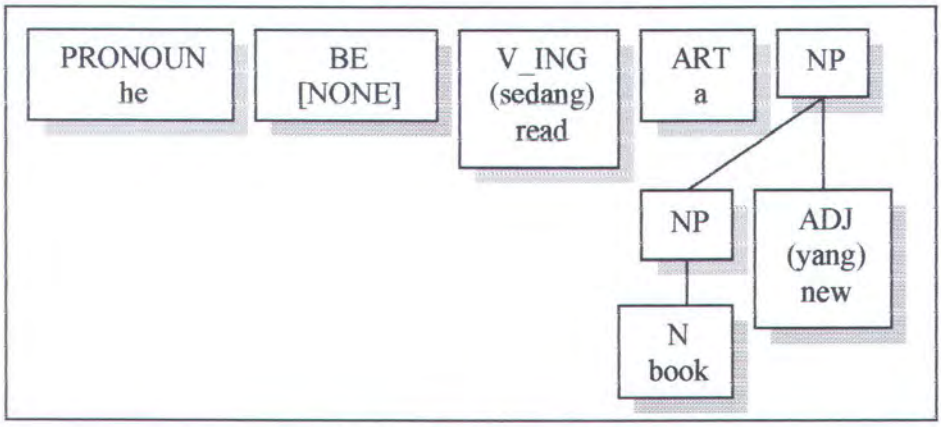
Aturan transformasi abstrak (*abstract transformation rules*), adalah aturan yang memberikan nilai-nilai pada konstituen struktur sumber yang telah jadi dan aturan ini diterapkan pada proses Strukturisasi Tujuan. Aturan ini berfungsi sebagai pemoles hasil yang akan dibentuk pada proses Pembangkitan Hasil dengan menambahkan string-string yang biasa ditemui pada suatu bahasa yang

menjadi ciri khas bahasa tersebut yang tidak dapat direpresentasikan secara grammatikal. Contoh sebuah Aturan Transformasi Abstrak :

Ekspresi: (1=BE,GETATTR=PRESENT)(2=V_ING)
Target: (1>NONE)(.sedang)(2>MAKEACTIVE)



(a) Struktur Aturan Sumber



(b) Struktur Aturan Tujuan

Gambar 4 - 5
Struktur Sumber dan Struktur Tujuan hasil semua aturan transformasi

Aturan Ekspresi diatas dapat dibaca 'jika BE berurutan dengan V_ING tanpa dibatasi oleh sebuah *leaf-node* lain, maka lakukan pengujian aturan Target'. Sedangkan Aturan Target dapat dibaca 'beri status NONE pada konstituen BE,

tambahkan string "sedang" pada konstituen V_ING, dan beri status MAKEACTIVE pada konstituen V_ING'. Dimana status yang diberikan ke konstituen adalah sebuah parameter yang akan diujikan pada Aturan Modifikasi Tujuan. Contoh untuk menunjukkan hasil yang akan diperoleh jika sebuah struktur sumber berikut dikenai aturan ini :

[PRONOUN he][BE is][V_ING read][ART a][NP ([NP ([N book])]) ([ADJ new])]

Hasil struktur tujuan yang dihasilkan adalah :

[PRONOUN he][BE=NONE is][V_ING (sedang) read][ART a][NP ([NP ([N book])]) ([ADJ new])]

Gambar struktur tujuan tersebut ditunjukkan pada gambar 4-5. Pada gambar tersebut mulai terlihat bentuk struktur kalimat tujuan yang diinginkan yang dibuat secara langsung dari struktur kalimat sumber dengan memberikan nilai-nilai tambahan tanpa banyak melakukan perubahan fisik pada node-node konstituennya.

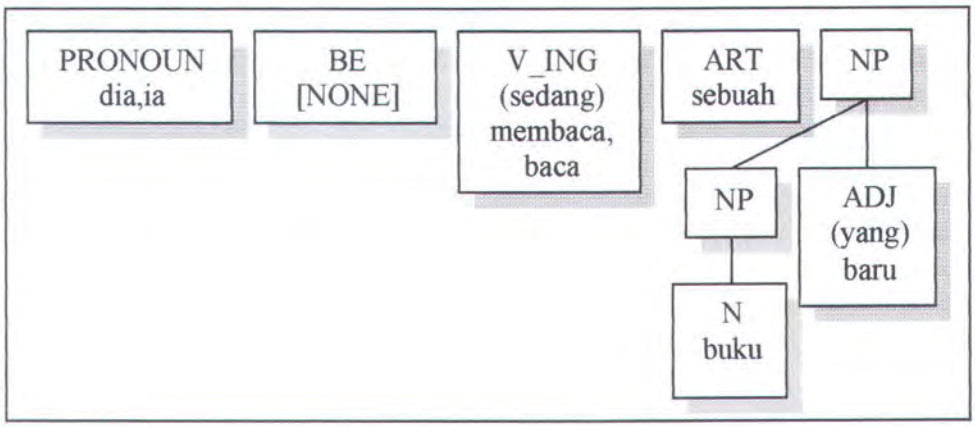
Aturan Modifikasi Tujuan (*Destination Modifiers Rules*), berfungsi untuk mengolah string token tujuan yang merupakan padanan kata dari string token sumber dari konstituen-konstituen pada struktur tujuan. Contoh Aturan Modifikasi Tujuan untuk contoh kalimat 'he is reading a new book' adalah :

Ekspresi: MAKEACTIVE=*[SUBSTR,1,1,b][GETTYPE=V]
Target: <[GETATTR=BASE][ADD=START,mem]><*>

Aturan Ekspresi diatas dapat dibaca 'jika status konstituen adalah MAKEACTIVE, cek semua string token tujuan yang memiliki huruf awal b dan bertipe dasar V, lakukan pengujian aturan Target'. Sedangkan aturan target dapat

dibaca 'cek apakah token tujuan memiliki atribut BASE, jika benar tambahkan string "mem" didepan kata, serta kembalikan juga string token tujuan sebelum dimodifikasi'. Aturan di atas dapat digunakan untuk mengubah kata tujuan dasar 'baca' pada kata sumber 'read' menjadi 'membaca' serta tetap mengembalikan kata 'baca' sebagai pilihan alternatif.

Ekspresi: NONE=*
Target: <RETURNNULL>



Gambar 4 - 6
Contoh struktur tujuan hasil penerapan seluruh bentuk aturan

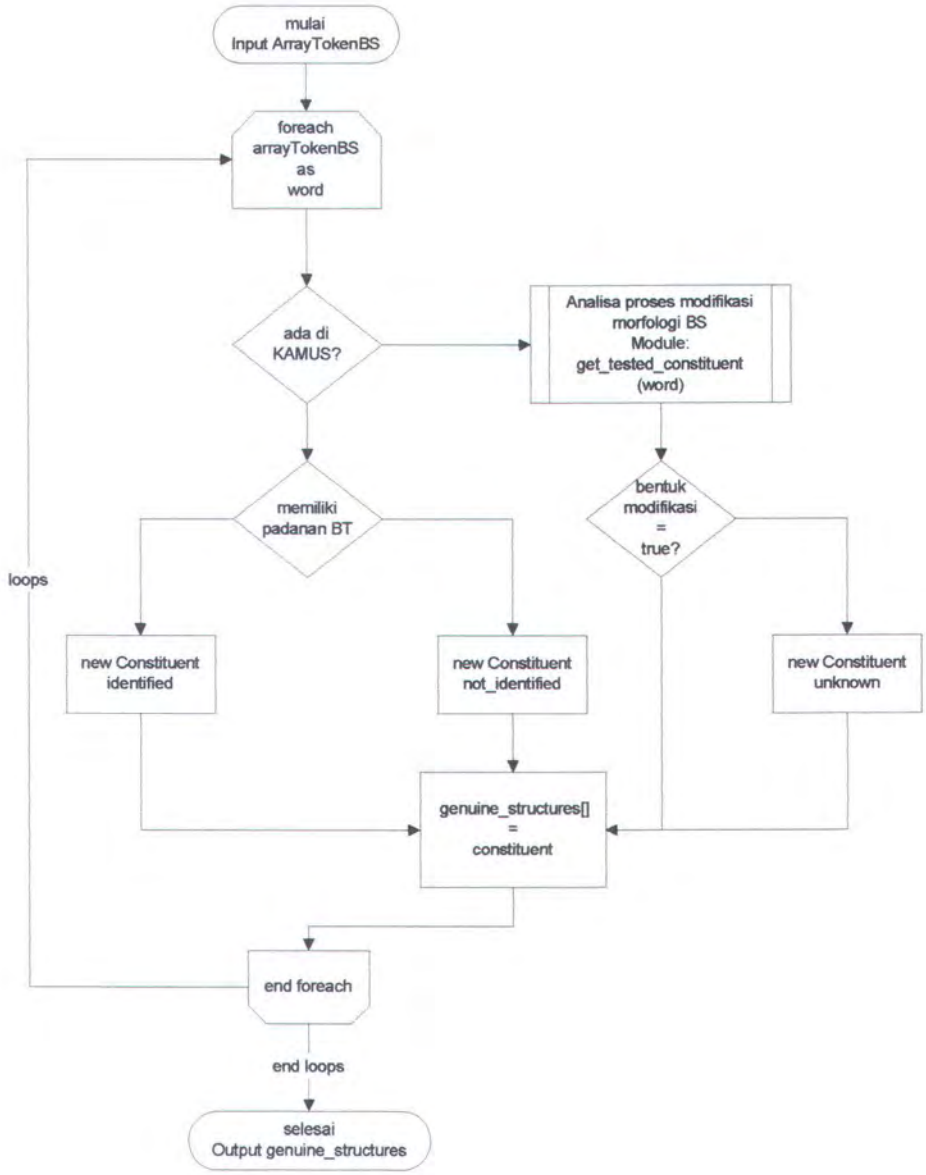
Aturan diatas dibaca 'untuk semua konstituen yang memiliki status NONE, tidak dikembalikan arti apapun'. Aturan diatas dapat digunakan untuk mengubah kata tujuan 'adalah' menjadi string kosong untuk kata sumber "is". Jika seluruh aturan-aturan diatas diterapkan pada kalimat 'he is reading a new book', bentuk struktur tujuan menjadi seperti yang nampak pada gambar 4-6. Sintaks Aturan Modifikasi Tujuan selengkapnya disertakan pada bagian lampiran A3.

4.3.2. Proses Strukturisasi Awal (*Genuine Structurization*)

Struktur awal / asli (*genuine*) adalah sebuah struktur yang menyimpan bentuk awal dari sebuah kalimat Bahasa Sumber yang dianalisa. Prosesnya ditunjukkan pada gambar 4-7. Inputan awal dari proses ini adalah sebuah array string token-token awal yang berupa kata-kata yang telah dipisah berdasarkan spasi kosong. Aplikasi kemudian mencari token-token tersebut dalam kamus, apabila ditemukan maka token tersebut dibuatkan bentuk obyek konstituen-nya yang berisi informasi awal dari sebuah token.

Pada proses ini, dilakukan pengecekan aturan modifikasi bahasa sumber (*source modification rules*) yaitu apabila ada sebuah kata yang tidak ditemukan dalam kamus maka sistem akan mencoba memprediksi (berdasarkan aturan yang telah dibuat) kata dasar yang mungkin telah mengalami perubahan sekaligus menentukan bentuk perubahan yang terjadi dan tipe katanya.

Ada tiga macam kemungkinan yang terjadi pada saat pemberian tipe token yaitu token yang ditemukan persis seperti di kamus, akan diberi tipe token seperti yang ditentukan di kamus. Kedua, token yang ditemukan setelah mengalami percobaan aturan modifikasi, pada kasus seperti ini maka akan dicatat bentuk tipe modifikasi yang terjadi dan juga tipe dasar dari kata tersebut dalam kamus. Ketiga, token tidak ditemukan sama sekali dalam kamus atau belum didefinisikan padanan katanya sama sekali dalam Bahasa Tujuan, sehingga tipe datanya akan diset menjadi 'unknown' (tipe data default bernilai -1). Masing-masing constituent yang dihasilkan kemudian disusun dalam suatu struktur kalimat yang berupa obyek constituentsTable.



Gambar 4 - 7
Diagram alir proses analisa struktur kalimat awal

Struktur genuine bersifat ambigu, artinya sebuah kalimat dapat memiliki lebih dari satu macam struktur sebab dalam proses ini dapat saja diterima satu atau lebih kata yang memiliki lebih dari satu macam tipe kata yang harus dianalisa

sehingga aplikasi melakukan proses pelipatgandaan sebanyak kemungkinan struktur yang dapat terjadi.

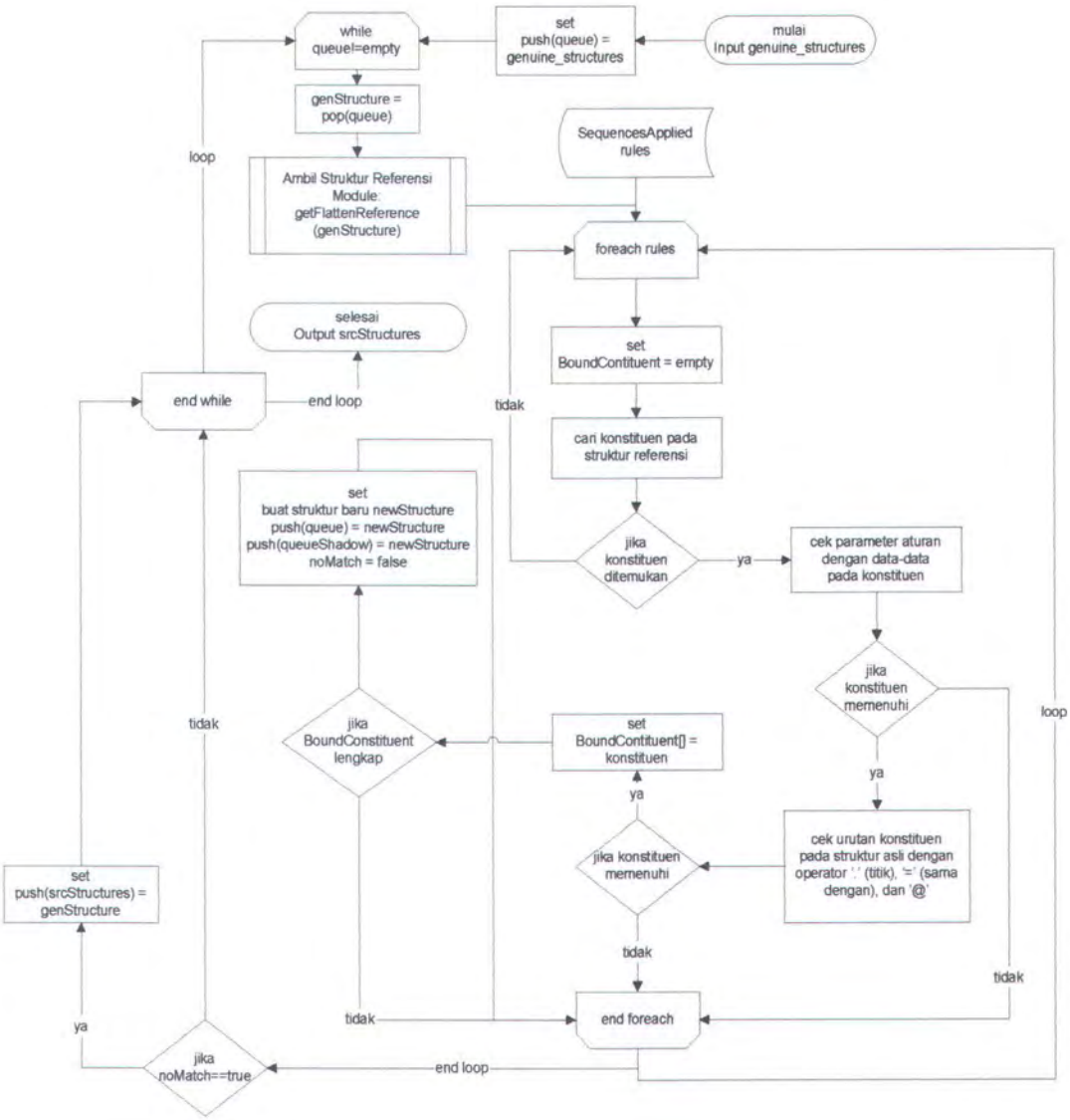
4.3.3. Proses Strukturisasi Sumber (*Source Structurization*)

Struktur Bahasa Sumber (*source*) adalah sebuah struktur kalimat yang merepresentasikan kalimat Bahasa Sumber yang telah diatur bentuk gramatikalnya sesuai dengan aturan yang ada dalam aturan transformasi. Proses pembentukan strukur Bahasa Sumber ini ditunjukkan pada gambar 4-8.

Dalam proses ini terlibat proses yang dilakukan dalam sebuah obyek structureAnalizer yang dalam deklarasinya telah dibentuk sebuah proses analisa yang dihasilkan dari aturan transformasi (sequences). Inputan proses ini adalah menerima bentuk struktur genuine dalam obyek contituentsTable.

Pada proses analisa transformasi ini besar kemungkinan ditemukan struktur yang ambigu dan membuat analisa menjadi lebih lama untuk dilakukan. Proses ini sangat penting untuk diperhatikan karena disinilah proses, yang menurut penulis, paling berat dan paling memakan waktu dan usaha, karena pada dasarnya menuntut penggunaan model dan algoritma parsing yang mampu menerima hasil yang bahkan tidak lengkap sekalipun.

Proses parsing yang dikerjakan oleh engine aplikasi ini telah dapat menyusun struktur pohon sesuai dengan keinginan dan ambiguitas sebanyak yang diinginkan. Kendala yang dihadapi adalah waktu pemrosesan berdasarkan algortima yang ditetapkan masih terlalu lama dan proses penghentian proses masih harus melalui pengujian yang lebih dalam karena masih ditemui proses yang dapat saja menjadi loop tak hingga.



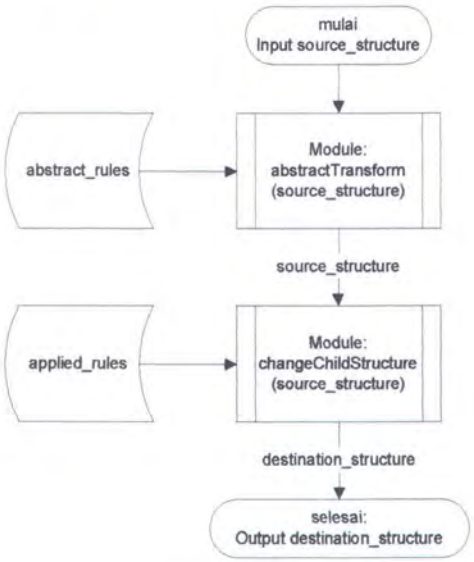
Gambar 4 - 8
Diagram alir proses pembentukan struktur asal

Hasil dari proses ini kemudian dibalikkan lagi menjadi struktur constituentTable yang merupakan representasi daripada struktur kalimat dalam Bahasa Sumber.

4.3.4. Proses Strukturisasi Tujuan (*Destination Structurization*)

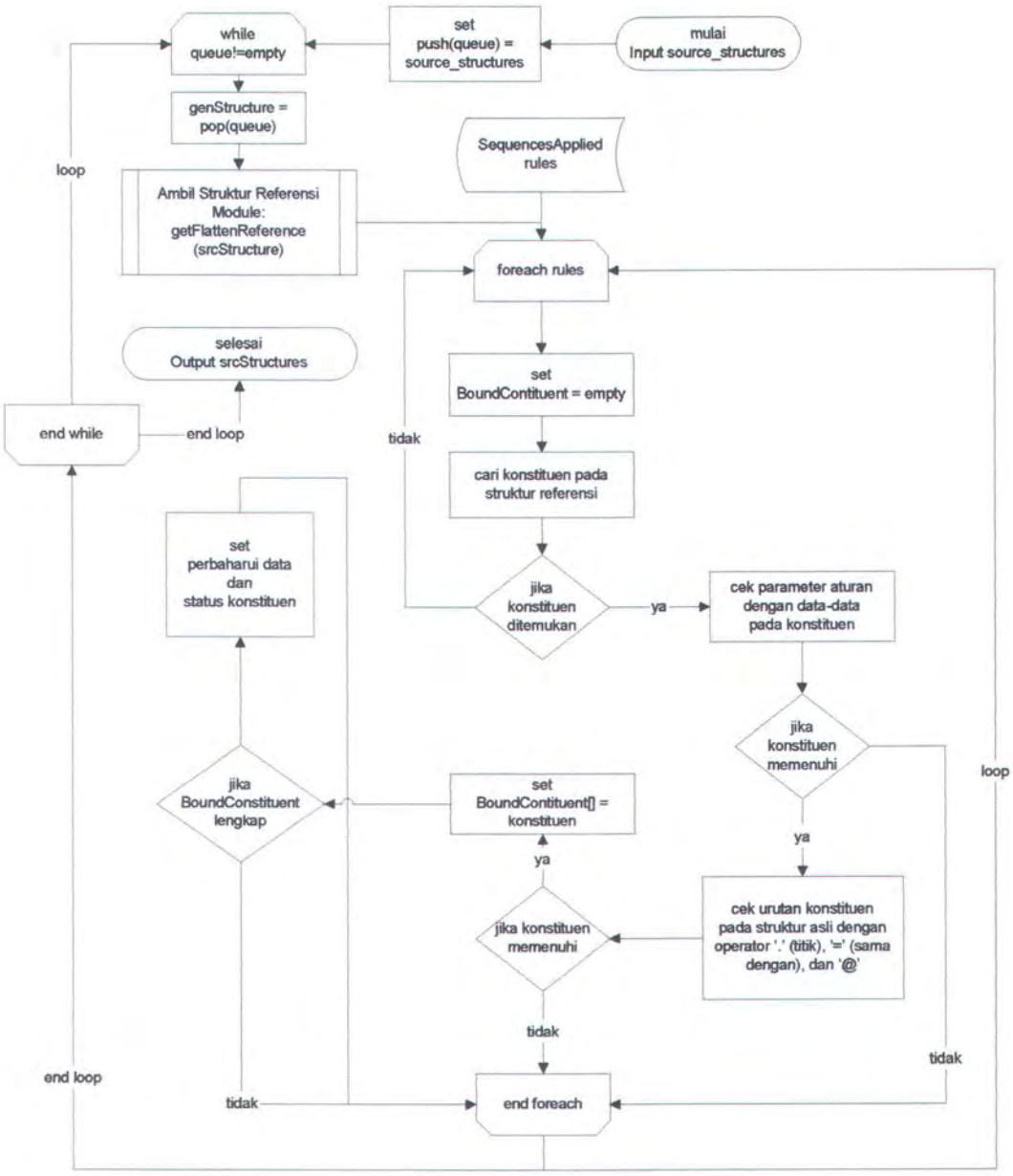
Proses ini melakukan pembentukan struktur Bahasa Tujuan (*destination*) yaitu struktur yang sedapat mungkin merepresentasikan bentuk tata bahasa Bahasa Tujuan, seperti yang ditunjukkan pada gambar 4-9.

Proses ini tidak membentuk struktur kalimat Bahasa Sumber menjadi padanannya dalam struktur kalimat Bahasa Tujuan, namun mencocokkan aturan tata bahasa Bahasa Sumber yang disesuaikan ke aturan tata bahasa Bahasa Tujuan. Proses inilah yang merupakan penerapan daripada model arsitektur MT langsung (*Direct-Transformer*).



Gambar 4 - 9
Diagram alir proses pembentukan struktur tujuan

Proses perubahan bentuk struktur BS menjadi struktur BT ditentukan berdasarkan aturan transformasi yang dibagi menjadi dua bagian yaitu transformasi abstrak dan transformasi aplikatif.

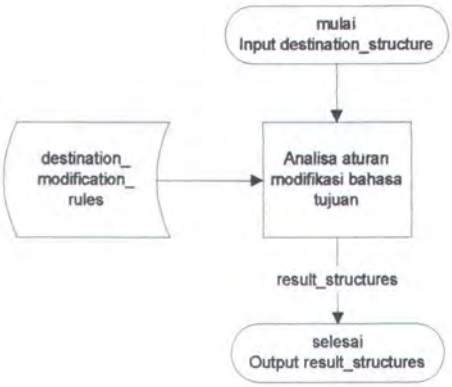


Gambar 4 - 10
Diagram alir proses pengujian Aturan Transformasi Abstrak

Transformasi abstrak adalah transformasi dimana perubahan bentuk diperoleh berdasarkan urutan dari unsur-unsur yang mungkin terjadi dalam sebuah

struktur kalimat. Sedangkan Transformasi aplikatif adalah transformasi dimana perubahan bentuk diperoleh berdasarkan struktur-struktur cabang pohon yang membentuk struktur utama.

Modul pengujian Aturan Transformasi Abstrak untuk menganalisa penambahan status dan string pelengkap ditunjukkan pada gambar 4-10. Dari analisa kedua macam transformasi ini maka diperoleh bentuk struktur BT yang tidak bersifat ambigu karena masing-masing aturan dihubungkan secara satu-satu antara aturan struktur sumber dan aturan struktur tujuan.



Gambar 4 - 11
Diagram alir proses pembentukan struktur hasil

4.3.5. Pembangkitan Hasil (*Result Generation*)

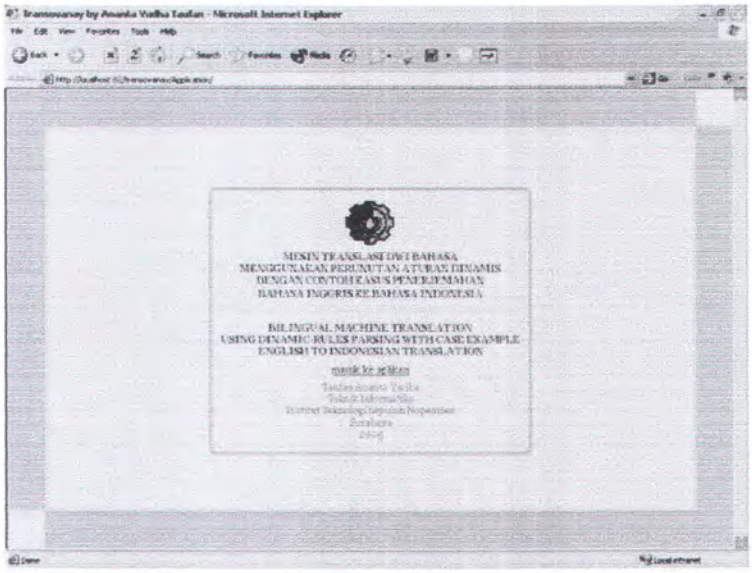
Hasil (*result*) daripada proses analisa secara keseluruhan ini adalah berupa string kalimat yang dibentuk dari struktur kalimat BT yang telah diperoleh. Sebelum diperoleh string hasil, dilakukan pengujian struktur Bahasa Tujuan dengan aturan modifikasi Bahasa Tujuan (*destination modification rules*) untuk membangkitkan bentuk-bentuk kata yang sesuai dengan tipe kata dalam Bahasa

Tujuan serta alternatif-alternatif yang mungkin diperoleh dari analisa aturan-aturan tersebut terhadap masing-masing unsur token struktur Bahasa Tujuan. Dengan demikian dapat dirangkai string kalimat Bahasa Tujuan dengan kata-kata yang m emiliki lebih dari satu arti atau alternatif dipisahkan dengan karakter "," (koma). Proses ini digambarkan pada gambar 4-11.

4.4. IMPLEMENTASI APLIKASI

4.4.1. Implementasi Dengan Halaman Web

Aplikasi yang dibangun pada Tugas Akhir ini menggunakan bahasa pemrograman PHP, sistem basis data MySQL, dan web server Apache yang mana adalah sebuah komponen standar untuk membangun sebuah aplikasi berbasis web.

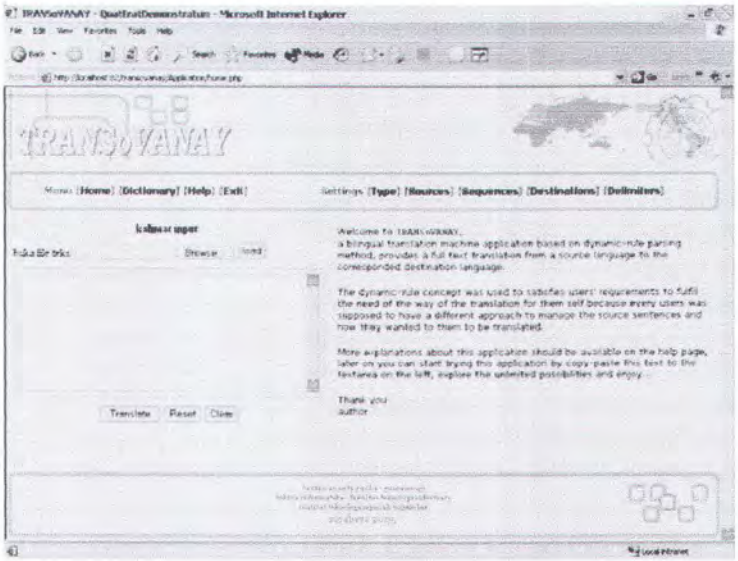


Gambar 4 - 12
Halaman pembuka aplikasi index.php

Aplikasi ini ditujukan untuk penggunaan secara lokal (alamat IP 127.0.0.1), dengan format alamat *http://localhost:PORT/APPLICATION_NAME/* dengan

PORT adalah nomor port yang dibuka untuk service Apache dan APPLICATION_NAME adalah direktori alias aplikasi pada konfigurasi Apache.

Pada implementasi sistem, alamat aplikasi, seperti yang nampak pada form *address* pada web browser, adalah *http://localhost:82/transovanay/Application/* secara default akan me-*redirect* browser ke halaman *index.php* yang akan menampilkan halaman pembuka aplikasi seperti nampak pada gambar 4-12. Link "*masuk ke aplikasi*" akan menampilkan halaman aplikasi utama.



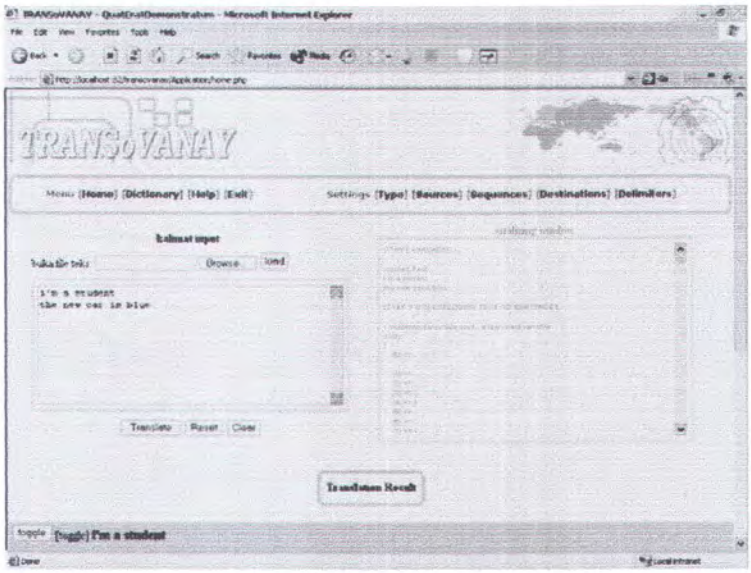
Gambar 4 - 13
Halaman home.php untuk menerima inputan

4.4.2. Antarmuka Menu HOME

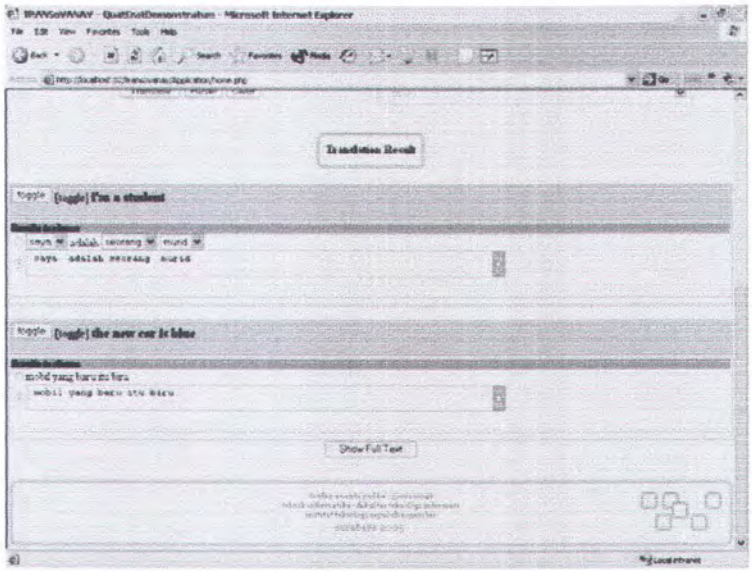
Halaman home.php digunakan untuk menerima inputan kalimat dalam Bahasa Sumber berupa file teks ataupun kalimat yang diinputkan ke dalam textbox yang sudah dipersiapkan. Tombol "*Translate*" digunakan untuk memulai proses penerjemahan. Snapshot halaman home.php ditunjukkan pada gambar 4-

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

13. Pada halaman ini link navigasi mulai dapat digunakan untuk berpindah dari satu halaman ke halaman lainnya.



(a)



(b)

Gambar 4 - 14
Hasil proses penerjemahan yang dilakukan

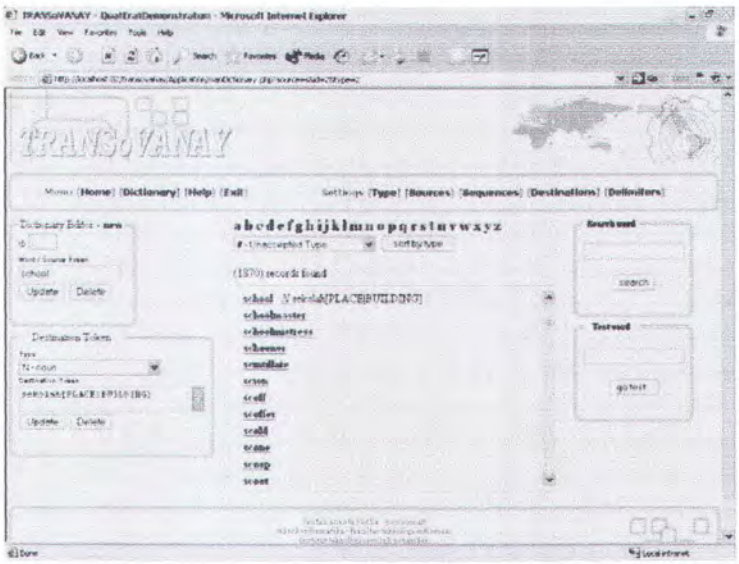
Hasil proses ini akan menampilkan 2 area baru yaitu jendela analisa (*Analizing Window*) dan daftar hasil terjemahan (*Translation Result*). Jendela

analisa menampilkan output yang dihasilkan selama proses analisa agar user ataupun tester dapat melihat secara detail proses yang terjadi dan juga dapat digunakan sebagai keluaran untuk melakukan *debugging* aplikasi seperti ditampilkan pada gambar 4-14 (a).

Sementara daftar hasil menampilkan hasil yang diperoleh dengan menampilkan form-form yang sesuai untuk menampilkan alternatif hasil dan ambiguitas yang dihasilkan serta mempermudah proses pengeditan akhir seperti yang ditampilkan pada gambar 4-14 (b).

4.4.3. Antarmuka Menu DICTIONARY

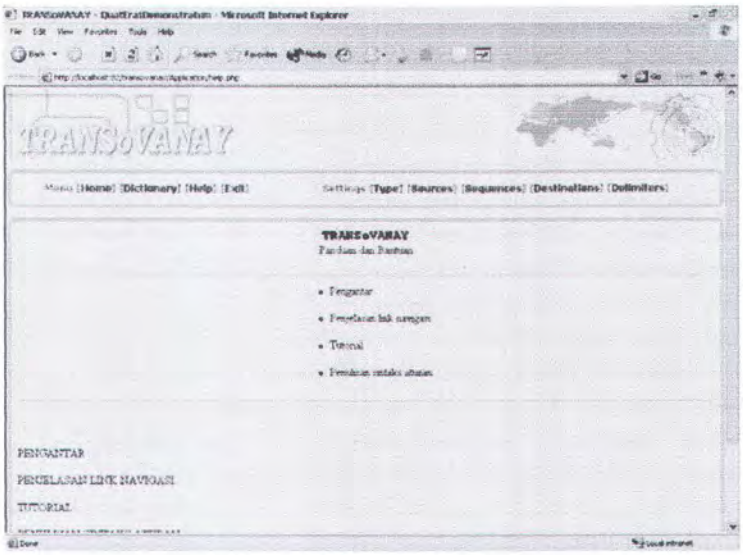
Halaman dictionary.php menampilkan daftar kata-kata yang terdapat dalam basis data kamus elektronik yang digunakan dalam aplikasi ini serta form yang memungkinkan user melakukan pengisian dan pengeditan data kamus. Halaman Dictionary ditampilkan pada gambar 4-15.



Gambar 4 - 15
Halaman kamus elektronik dictionary.php

4.4.4. Antarmuka Menu HELP

Halaman ini menampilkan petunjuk-petunjuk penggunaan dan bantuan-bantuan umum yang dapat digunakan oleh user ketika mengalami kesulitan ataupun untuk mendapatkan informasi akan suatu hal yang berhubungan dengan sistem aplikasi ToV ini. Halaman help ditampilkan pada gambar 4-16.



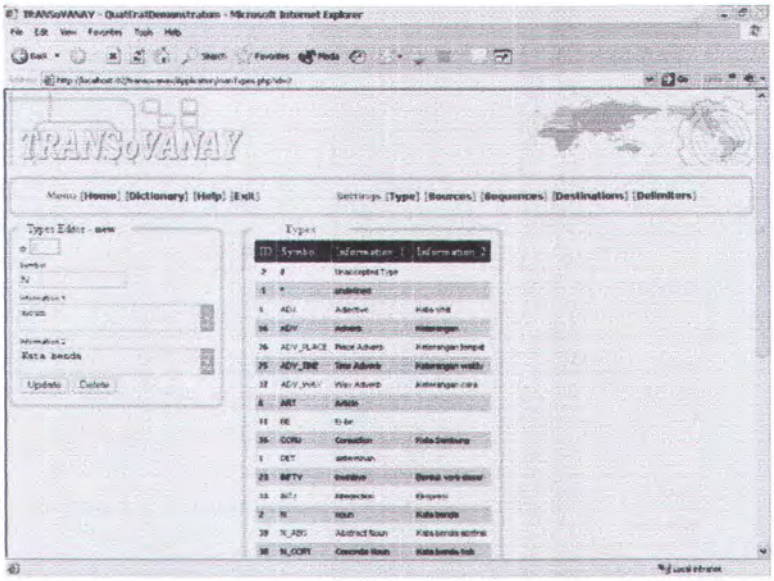
Gambar 4 - 16
Halaman help.php

4.4.5. Antarmuka Setting TYPE

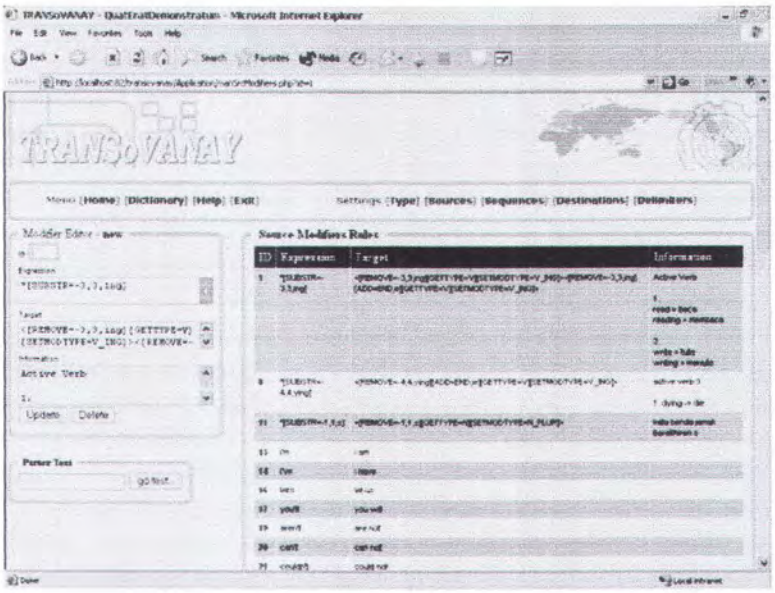
Halaman ini adalah halaman untuk menampilkan dan melakukan perubahan nilai data-data aturan tipe kata, yaitu aturan untuk representasi simbolis dari sebuah kata dan struktur kalimat. Halaman ini ditampilkan pada gambar 4-17.

4.4.6. Antarmuka Setting SOURCES

Halaman ini adalah halaman untuk menampilkan dan melakukan perubahan nilai data-data aturan modifikasi kata sumber, yaitu aturan untuk merubah bentuk kata dan atau mencari tipe kata yang sesuai dengan data kamus dan aturan tipe kata. Halaman ini ditampilkan pada gambar 4-18.



Gambar 4 - 17
Halaman editor aturan tipe kata

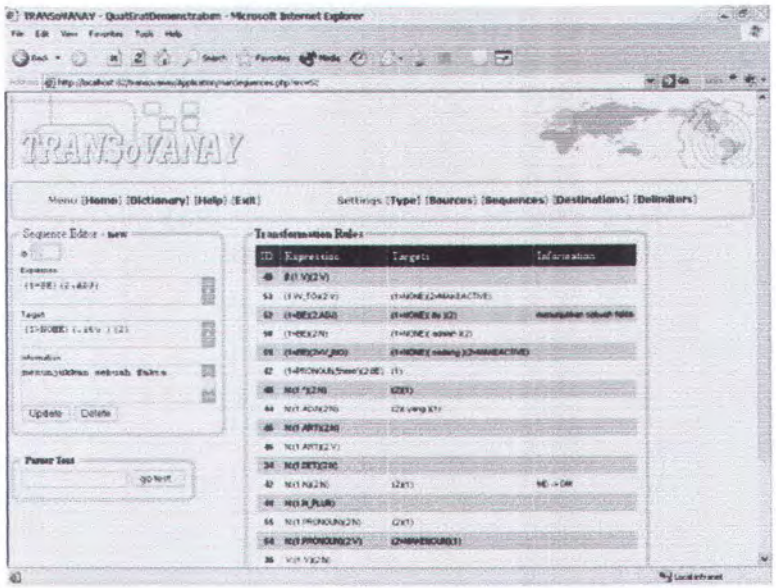


Gambar 4 - 18
Halaman editor aturan modifikasi kata sumber

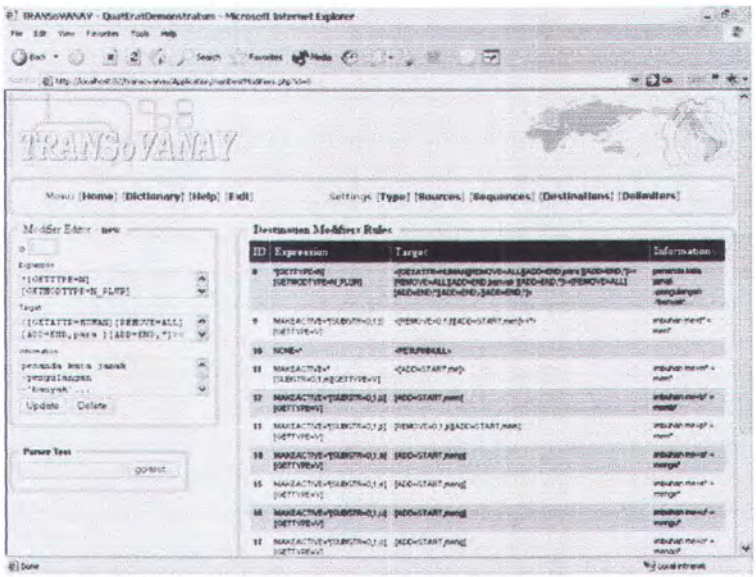
4.4.7. Antarmuka Setting SEQUENCES

Halaman ini adalah halaman untuk menampilkan dan melakukan perubahan nilai data-data aturan transformasi struktur kalimat, yaitu aturan untuk

merubah struktur kalimat Bahasa Sumber menjadi struktur kalimat Bahasa Tujuan. Halaman ini ditampilkan pada gambar 4-19.



Gambar 4 - 19
Halaman editor aturan transformasi



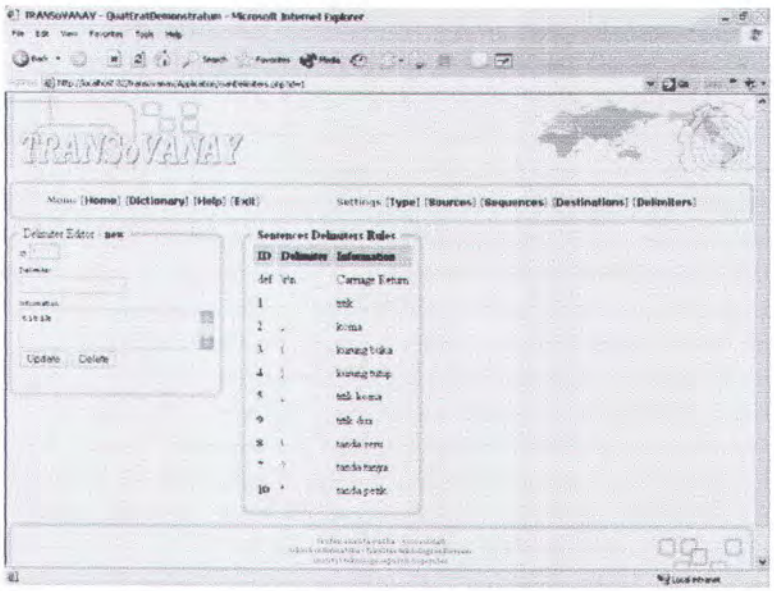
Gambar 4 - 20
Halaman editor aturan modifikasi kata tujuan

4.4.8. Antarmuka Setting DESTINATIONS

Halaman ini adalah halaman untuk menampilkan dan melakukan perubahan nilai data-data aturan modifikasi kata tujuan, yaitu aturan untuk merubah bentuk kata dan atau mencari tipe kata dan menambahkan string bantu yang sesuai dengan bentuk kalimat dalam Bahasa Tujuan. Halaman ini ditampilkan pada gambar 4-20.

4.4.9. Antarmuka Setting DELIMITERS

Halaman ini adalah halaman untuk menampilkan dan melakukan perubahan nilai data-data aturan pembatas kalimat, yaitu karakter-karakter yang digunakan untuk menunjukkan bahwa sebuah kalimat sudah selesai dan memisahkannya dengan kalimat lainnya. Halaman ini ditampilkan pada gambar 4-21.



Gambar 4 - 21
Halaman editor aturan pembatas kalimat

BAB 5

UJI COBA DAN EVALUASI

BAB 5
UJI COBA DAN EVALUASI

Sistem aplikasi ini diuji coba didalam lingkungan sistem komputer yang telah diterangkan pada Bab Implementasi.

5.1. PARAMETER UMUM UJI COBA

Tahap uji coba aplikasi dilakukan dengan menggunakan parameter-parameter sebagai berikut :

1. Keberhasilan aplikasi dalam memproses teks inputan dan mengembalikan hasil penerjemahan kepada user.
2. Keberhasilan aplikasi mengeksekusi perintah-perintah dalam aturan-aturan yang telah ada sesuai dengan yang diharapkan dengan membandingkan hasil yang diperoleh dari aplikasi dengan hasil yang ingin dicapai dan dianalisa secara manual.

5.2. SKENARIO UJI COBA

Skenario pengujian yang akan dilakukan pada sistem aplikasi untuk menguji kehandalan sistem dan memastikan tidak ada prosedur yang salah ataupun kurang pada masing-masing bagian aplikasi.

Sistem aplikasi diuji dengan memberi inputan berupa kalimat sederhana sesuai dengan aturan grammar Bahasa Inggris. Pengujian dilakukan dengan berdasarkan aturan-aturan yang ditunjukkan pada tabel aturan yang disertakan dalam lampiran bagian B1 untuk Aturan Tipe, B2 untuk Aturan Modifikasi

Sumber, B3 untuk Aturan Transformasi, dan B4 untuk Aturan Modifikasi Tujuan. Adapun bentuk-bentuk kalimat dasar yang diuji adalah:

1. Uji coba aturan-aturan *to be*

   Uji coba ini untuk menunjukkan hasil yang diperoleh ketika aplikasi menemui tipe kata *to be* dalam Bahasa Inggris dan bagaimana mengolahnya menjadi sebuah bentuk yang benar dalam Bahasa Indonesia.

2. Uji coba aturan-aturan kata benda (*Noun*)

   Uji coba ini untuk menunjukkan perubahan-perubahan yang terjadi pada kalimat ketika menemui kata benda terutama ketika menemui kata benda bentuk tunggal dan jamak.

3. Uji coba aturan-aturan kata ganti (*Pronoun*)

   Uji coba ini untuk menunjukkan pengaruh pronoun baik itu berupa *personal pronoun*, *demontrative pronoun*, dan *interrogative pronoun* ketika diterapkan dalam kalimat.

4. Uji coba aturan-aturan waktu (*Tenses*)

   Uji coba ini untuk menunjukkan pengaruh *to be* yang menyertai bentuk-bentuk tenses yang beragam dalam kalimat-kalimat Bahasa Inggris dan bagaimana aplikasi menerjemahkannya ke dalam Bahasa Indonesia berdasarkan aturan yang ada.

5.3. PELAKSANAAN DAN EVALUASI HASIL UJI COBA

1. Uji coba aturan-aturan *to be*

Inputan dan output yang dihasilkan oleh aplikasi sebelum diberi bentuk aturan yang sesuai ditunjukkan pada tabel 5-1. Angka didalam tanda kurung menunjukkan urutan ambiguitas hasil yang diperoleh (dimulai dari 0, yaitu hasil yang langsung ditampilkan kepada user).

Tabel 5 - 1
Uji coba *to be* sebelum diberi aturan

Input	Output
I am a student	saya adalah seorang murid (0)
You are happy	anda adalah gembira (0)
She is pretty	dia adalah cantik (0)
she is not pretty	dia adalah tidak cantik (0)
Is she pretty	dia cantik? (0)
Isn't she pretty?	isn't dia cantik? (0)
It is a bag	itu adalah sebuah tas (0)
he does not go	dia tidak pergi (0)
we did not come	kami tidak datang (0)
do we study?	kami belajar? (0)
did he go?	dia pergi? (0)

Setelah diberikan aturan-aturan yang sesuai sesuai dengan analisa manual terhadap hasil-hasil yang diperoleh, kemudian ditambahkan aturan-aturan yang sesuai yang dapat menerima bentuk-bentuk kasus pada tabel 5-1. Aturan-aturan yang ditambahkan tersebut adalah :

Aturan Modifikasi Sumber yang ditambahkan

Ekspresi	Target	Keterangan
isn't	[RETURNWORD=is][RETURNWORD=not]	Mendapatkan singkatan 'is not'

Aturan Transformasi yang ditambahkan

Ekspresi	Target	Keterangan
(1.BE)(2.W_NOT)(3=ADJ)	(1>NONE)(.tidak)(2>NONE)(3)	sifat negative
(1.BE,FIRSTINDEX=0)(2.W_NOT)(3@PRONOUN,GETATTR=SUBJECT)(4@ADJ)	(1>NONE)(.tidakkah)(2>NONE)(3)(.itu)(4)	question tag
(1.BE,FIRSTINDEX=0)(2.W_NOT)(3=N)(4=ADJ)	(1>NONE)(.bukankah)(2)(.itu)(3)(4)	kata tanya sifat negative
(1.BE,FIRSTINDEX=0,LASTINDEX=0)(2=PRONOUN,GETATTR=SUBJECT)	(.benarkah,apakah)(1>NONE)(2)	be tanya
(1.N)(2.BE)(3.W_NOT)(4=V)	(1)(2>NONE)(.tidak)(3)(4)	negative verb

Inputan dan output yang dihasilkan oleh aplikasi setelah diberikan tambahan aturan diatas ditunjukkan pada tabel 5-2.

Tabel 5 - 2
Hasil uji coba aturan-aturan *to be*

Input	Output
I am a student	saya adalah seorang murid (0)
You are happy	anda itu gembira (0)
She is pretty	dia itu cantik (0)
she is not pretty	dia tidak cantik (0)
Is she pretty	benarkah dia cantik? (0)
Isn't she pretty?	tidakkah dia itu cantik? (0)
It is a bag	itu adalah sebuah tas (0)
he does not go	dia tidak pergi (0)
we did not come	kami tidak datang (0)
do we study?	benarkah kami belajar? (0)
did he go?	benarkah dia pergi? (0)

Dengan aturan-aturan *to be* yang telah didefinisikan tersebut nampak bahwa hasil yang diperoleh sudah sesuai dengan yang diharapkan.

2. Uji coba aturan-aturan kata benda (*Noun*)

Inputan dan output yang dihasilkan aplikasi sebelum diberi aturan yang sesuai untuk aturan kata benda ditunjukkan pada tabel 5-3.

Tabel 5 - 3
Uji coba noun sebelum diberi aturan

Input	Output
has alice many books?	alice banyak buku? (0) banyak buku alice (1)
There aren't many students in the class-room	disana aren't banyak murid di ruang kelas (0) disana banyak murid aren't di ruang kelas (1) ada aren't banyak murid di ruang kelas (2) ada banyak murid aren't di ruang kelas (3)
I haven't much sugar	saya haven't banyak gula (0) saya banyak gula haven't (1)
gold is yellow	emas itu kuning (0)
He has a lot of shirts	dia memiliki banyak baju (0)
There is a lot of water in tin	ada adalah banyak air di kaleng (0)
how much water is there in this glass?	berapa banyak air ada di gelas ini? (0)
How many students are there in that school?	berapa banyak murid ada di sekolah itu? (0)
A dog is an animal	sebuah anjing adalah sebuah binatang (0)
Dogs are animals	anjing adalah binatang (0)
students learn english	murid belajar bahasa inggris (0)

Dari hasil output pada tabel 5-3, ditambahkan aturan-aturan yang sesuai untuk menerima bentuk-bentuk kasus kalimat yang melibatkan kata benda yaitu :

Aturan Modifikasi Sumber yang ditambahkan

Ekspresi	Target	Keterangan
aren't	[RETURNWORD=are][RETURNWORD=not]	Mendapatkan singkatan 'are not'
haven't	[RETURNWORD=have][RETURNWORD=not]	Mendapatkan singkatan 'have not'

Atururan Transformasi yang ditambahkan

Ekspresi	Target	Keterangan
(1.W_HAVE,FIRSTINDEX=0)(2.NP)(3.NP)	(.apakah)(1>NONE)(2)(.memiliki)(3)	have=memiliki (tanya)
(1=PRONOUN,STRING=th	(1>NONE)(.tidak ada	there not=tidak ada

ere)(2=BE)(3.W_NOT)	)(2>NONE)(3>NONE)	
(1=PRONOUN)(2.W_HAVE)(3.W_NOT)(4.NP)	(1)(2>NONE)(.tidak memiliki)(3>NONE)(4)	have not=tidak memiliki
(1=BE)(2@ADJ,GETATTR= COLOR)	(1>NONE)(.berwarna)(2)	tambahan untuk warna
(1=PRONOUN,STRING=th ere)(2.BE)(3.NP)	(.ada)(1>NONE)(2>NONE)(3)	there=ada
(1.PREP,STRING=in)(2=N, GETATTR=PLACE)	(.di dalam)(1>NONE)(2)	in=di dalam
(1.ART,GETATTR=SINGLE)(2@N,GETATTR=ANIMAL)	(1>NONE)(.seekor)(2>SETDEFAULTMEAN)	a=seekor
(1=N_PLUR)(2.BE)(3=N_P LUR)	(1>SETDEFAULTMEAN)(2)(3>S ETDEFAULTMEAN)	jamak+jamak=tunggal saja

Aturan Modifikasi Tujuan yang ditambahkan

Ekspresi	Target	Keterangan
*[GETTYPE=N][GETM ODTYPE=N_PLUR]	<[GETATTR=HUMAN][REMOVE=ALL][ADD=END,para][ADD=END,*]><[REMOVE=ALL][ADD =END,banyak][ADD=END,*]><[REMOVE=ALL][ADD =END,*][ADD=END,-][ADD=END,*]>	penanda kata jamak -pengulangan -'banyak'...

Inputan dan output yang dihasilkan oleh aplikasi setelah diberi aturan-aturan tambahan diatas ditunjukkan pada tabel 5-4. Dari hasil uji coba aturan-aturan untuk kata benda juga berhasil diperoleh hasil yang sesuai dengan yang diharapkan. Meskipun sebagian besar hasilnya baik, namun pada baris pertama untuk kalimat "has alice many books?" diperoleh dua buah hasil ambigu dengan hasil kedua (1) menunjukkan hasil yang tidak sesuai dengan yang diharapkan, namun tetap ditampilkan karena hasil tersebut dianggap masih memenuhi aturan yang ada. Sedangkan pada kalimat "how many students are there in that school?" nampak bahwa hasil yang juga sesuai adalah hasil kedua (1), namun aplikasi masih belum

dapat menentukan prioritas tampilan hasil sehingga hasil pertamalah yang akan ditampilkan terlebih dahulu.

Tabel 5 - 4
Hasil uji coba aturan-aturan kata benda (*Noun*)

Input	Output
has alice many books?	apakah alice memiliki banyak buku? (0) banyak buku alice (1)
There aren't many students in the class-room	tidak ada banyak murid di dalam ruang kelas (0)
I haven't much sugar	saya tidak memiliki banyak gula (0)
gold is yellow	emas itu berwarna kuning (0)
He has a lot of shirts	dia memiliki banyak baju (0)
There is a lot of water in tin	ada banyak air di kaleng (0)
how much water is there in this glass?	berapa banyak air ada di gelas ini? (0)
How many students are there in that school?	berapa banyak murid ada di sekolah itu? (0) berapa banyak murid ada di dalam sekolah itu? (1)
A dog is an animal	seekor anjing adalah seekor binatang (0)
Dogs are animals	anjing adalah binatang (0)
students learn english	para murid belajar bahasa inggris (0)

Tabel 5 - 5
Uji coba pronoun sebelum diberi aturan

Input	Output
we love you	kami cinta anda (0)
i gave him a book	saya memberi dia sebuah buku (0)
this is a chair	ini adalah sebuah kursi (0)
those are chairs	itu adalah banyak kursi (0)
who is that girl?	Siapa adalah itu gadis? (0)
for whom does she make the dress?	untuk siapa lakukan dia membuat pakaian? (0) untuk siapa dia membuat pakaian (1)
whose bag is that?	milik siapa tas itu? (0)
This is my book	ini -ku buku (0)
alice's father	alice's ayah (0) ayah alice's (1)
where do you live?	dimana anda tinggal? (0) dimana melakukan anda tinggal? (1)
why were you absent?	mengapa sedang anda tidak masuk? (0)

3. Uji coba aturan-aturan kata ganti (*Pronoun*)

Inputan dan output yang dihasilkan oleh aplikasi sebelum diberi aturan-aturan yang sesuai ditunjukkan pada tabel 5-5.

Dari hasil output pada tabel 5-5, diberikan tambahan aturan-aturan yang sesuai yaitu:

Aturan Modifikasi Sumber yang ditambahkan

Ekspresi	Target	Keterangan
*[SUBSTR=-2,2,'s]	<[REMOVE=-2,2,'s][SETMODTYPE=PRONOUN][SETATTR=POSSESIVE][RETURNWORD]>	Mendapatkan konstituent kepemilikan

Aturan Transformasi yang ditambahkan

Ekspresi	Target	Keterangan
(1.PRONOUN,GETATTR=INTERROGATIVE)(2.BE)(3.NP)	(1)(2>NONE)(3)	be=
#:(1.V)(2.PRONOUN,GETATTR=SUBJECT)		ditolak
PRONOUN:(1.PRONOUN,GETATTR=POSSESIVE)(2.PRONOUN,GETATTR=POSSESIVE)	(2)(1)	possesive+possesive dibalik
(1.BE)(2.PRONOUN)	(1>NONE)(2)	be+plural
(1.PRONOUN,GETATTR=RELATIVE,FIRSTINDEX=0)(2.BE)	(1)(2>NONE)	be=
NP:(1.PRONOUN,GETATTR=DEMONSTRATIVE)(2=N)	(2)(1)	demonstrative+N = N+demonstrative
NP:(1.PRONOUN,GETATTR=DEMONSTRATIVE)(2=NP)	(2)(1)	demonstrative+NP = NP+demonstrative
NP:(1.PRONOUN,GETATTR=POSSESIVE)(2=N)	(2)(1)	possesive+noun=noun+possesive

Inputan dan output yang dihasilkan oleh aplikasi setalah diberi tambahan aturan-aturan diatas ditunjukkan pada tabel 5-6. Pada tabel hasil nampak

bahwa lebih banyak terjadi ambiguitas yang mengganggu meskipun beberapa masih menunjukkan hasil ambiguitas yang benar.

Tabel 5 - 6
Hasil uji coba aturan-aturan kata ganti orang (*pronoun*)

Input	Output
we love you	kami cinta anda (0)
i gave him a book	saya memberi dia sebuah buku (0)
this is a chair	ini adalah sebuah kursi (0)
those are chairs	itu adalah banyak kursi (0)
who is that girl?	siapa gadis itu? (0)
for whom does she make the dress?	untuk siapa dia membuat pakaian? (0)
whose bag is that?	milik siapa tas itu? (0)
This is my book	ini -ku buku (0)
	ini adalah buku –ku (1)
alice's father	milik alice ayah (0)
	ayah alice (1)
where do you live?	dimana anda tinggal? (0)
why were you absent?	mengapa anda tidak masuk? (0)

Tabel 5 - 7
Uji coba tenses sebelum diberi aturan

Input	Output
i go to school yesterday	saya pergi ke sekolah kemarin (0)
they are studying now	mereka adalah studying sekarang (0)
he has been working in this office for ten years	dia telah working di kantor ini untuk sepuluh tahun (0)
i went yesterday	saya pergi kemarin (0)
he was studying all day yesterday	dia sedang studying sepanjang hari kemarin (0)
he had fallen before i helped him	dia jatuh sebelum saya helped dia (0)
he will post the letter	dia akan mengirim surat (0)
he'll give you a new bicycle if you go with him	he'll akan memberi anda sebuah baru sepeda jika anda pergi dengan dia (0)
i should go to malang the next day	saya harus pergi ke malang selanjutnya hari (0)
	saya harus pergi ke malang hari selanjutnya (1)
he would have graduated if he studied hard	dia akan graduated jika dia studied keras (0)
will you help me to get the book	akan anda bantuan saya mengambil buku (0)
	akan anda membantu saya mengambil buku (1)

4. Uji coba aturan-aturan waktu (*Tenses*)

Inputan dan output yang dihasilkan oleh aplikasi sebelum diberikan aturan-aturan yang sesuai ditunjukkan pada tabel 5-7. Dari hasil output pada tabel 5-7, dapat ditambahkan aturan-aturan yang sesuai untuk menerima perubahan-perubahan bentuk tenses sebagai berikut :

Aturan Modifikasi Sumber yang ditambahkan

Ekspresi	Target	Keterangan
*[SUBSTR=-3,3,ing]	<[REMOVE=-3,3,ing] [GETTYPE=V] [SETMODTYPE=V_ING] [RETURNWORD]> <[REMOVE=-3,3,ing] [ADD=END,e] [GETTYPE=V] [SETMODTYPE=V_ING] [RETURNWORD]>	Mendapatkan konstituen Verb+ing
he'll	<[RETURNWORD=he] [RETURNWORD=will]>	Mendapatkan singkatan 'he will'
*[SUBSTR=-3,3,ied]	<[REMOVE=-3,3,ied][ADD=END,y] [GETTYPE=V] [SETATTR=PAST] [RETURNWORD]>	Mendapatkan konstituen Verb lampau
*[SUBSTR=-2,2,ed]	<[REMOVE=-1,1,d] [GETTYPE=ADV] [RETURNWORD]> <[REMOVE=-1,1,d] [GETTYPE=V] [RETURNWORD]> <[REMOVE=-2,2,ed] [GETTYPE=V] [RETURNWORD]>	Mendapatkan konstituen Adjective dan Verb pasif

Aturan Transformasi yang ditambahkan

Ekspresi	Target	Keterangan
(1=BE,GETATTR=PRESENT) (2=V_ING)	(1>NONE)(.sedang)(2>MAKEACTIVE)	kalimat present continuous tense
(1.POSB,FIRSTINDEX=0)	(1)(.kah)	kata tanya kemungkinan
(1.W_HAVE)(2.BE,STRING=been)	(1>NONE)(2)	has been=telah

Input dan output yang dihasilkan oleh aplikasi untuk uji coba ini ditunjukkan pada tabel 5-8. Dari tabel hasil tersebut dapat dilihat lebih banyak lagi hasil yang kurang memuaskan yang dihasilkan. Pada kalimat input "he would have graduated if he studied hard" masih menghasilkan

kalimat output yang tidak cocok dengan konteks waktu yang diinginkan dalam kalimat aslinya. Sementara masalah lainnya masih berupa penentuan prioritas aturan-aturan yang dapat diterima yang ditampilkan oleh aplikasi.

Dalam pengujian tenses juga ditemui kalimat-kalimat yang terlalu lama mengalami proses analisa dan melewati batas waktu eksekusi server sehingga hasilnya tidak dapat ditampilkan oleh aplikasi, seperti yang ditunjukkan pada tabel 5-5. Hal ini terjadi karena banyaknya konstituen dan aturan yang harus dicek sementara sistem harus mengecek seluruh kemungkinan yang bisa dihasilkan oleh aturan-aturan tersebut.

Tabel 5 - 8
Hasil uji coba aturan-aturan kata waktu (*tenses*)

Input	Output
i go to school yesterday	saya pergi ke sekolah kemarin (0)
they are studying now	mereka sedang belajar sekarang (0)
he has been working in this office for ten years	dia telah bekerja di kantor ini untuk sepuluh tahun (0)
i went yesterday	saya pergi kemarin (0)
he was studying all day yesterday	dia sedang belajar sepanjang hari kemarin (0)
he had fallen before i helped him	dia jatuh sebelum saya membantu dia (0)
he will post the letter	dia akan mengirim surat (0)
he'll give you a new bicycle if you go with him	dia akan memberi anda sebuah baru sepeda jika anda pergi dengan dia (0)
i should go to malang the next day	saya harus pergi ke malang selanjutnya hari (0) saya harus pergi ke malang hari selanjutnya (1)
he would have graduated if he studied hard	dia akan lulus jika dia belajar keras (0)
will you help me to get the book	akankah anda bantuan saya mengambil buku (0) akankah anda membantu saya mengambil buku (1)

Tabel 5 - 9
Hasil uji coba tenses yang mengalami kegagalan

Input	Output
by last christmas i should have been working in that office for ten years	(Maximum execution time of 30 seconds exceeded)
i should be having an examination at this time the following day	(Maximum execution time of 30 seconds exceeded)
i shall have read seven of erhan's plays by the end of this year	(Maximum execution time of 30 seconds exceeded)

Langkah-langkah yang dapat ditempuh untuk menghindari hal ini yang adalah membagi kalimat menjadi kalimat yang lebih pendek agar diproses lebih cepat atau dengan melakukan perubahan tipe data dan aturan-aturan yang terlibat terutama aturan transformasi agar menjadi lebih efisien, sementara perubahan algoritma aturan parsing dan proses analisa adalah pilihan paling akhir apabila ternyata masih ditemukan kendala yang berarti.

BAB 6

PENUTUP

BAB 6
PENUTUP

6.1. KESIMPULAN

Dari hasil uji coba terhadap sistem aplikasi tersebut dapat diperoleh beberapa kesimpulan berkaitan dengan sistem MT yang telah dikerjakan yaitu :

- Aplikasi telah mampu memproses aturan-aturan yang diberikan secara dinamis sesuai dengan fungsi yang diinginkan dari setiap aturan-aturan tersebut.
- Aturan-aturan yang dibuat telah dapat berinteraksi dengan data-data dalam setiap konstituen, struktur konstituen, dan database token dalam Kamus.
- Masing-masing aturan pada jenis aturan yang berbeda (modifikasi sumber, transformasi, modifikasi tujuan) dapat digunakan untuk saling melengkapi, melewatkan kondisi, dan memperbaiki hasil teks keluaran.
- Hasil yang diperoleh lebih dari satu belum dapat ditampilkan sesuai dengan urutan yang sesuai dengan kaidah Bahasa Tujuan yang sesungguhnya.

6.2. SARAN

Dari kesimpulan diatas maka dapat diberikan saran-saran berikut untuk mengembangkan sistem MT yang lebih baik berikut ini :

- Algoritma analisa struktur kalimat diberi tambahan kemampuan untuk dapat mendeteksi adanya pemberian prioritas pada struktur tujuan yang dihasilkan.
- Konstituen harus memiliki atribut yang dapat menampung data prioritas serta memiliki metode untuk dapat saling berinteraksi dengan konstituen-konstituen lain dalam struktur yang sama.
- Perintah-perintah dan parameter-parameter pada sintaks aturan-aturan analisa perlu ditambah agar dapat melakukan modifikasi-modifikasi yang lebih banyak pada konstituen.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

1) Daniel I. A. Cohen
Introduction to Computer Theory 2nd Edition, John Wiley & Sons, Inc., Canada, 1991

2) Darmawan Sukmadjaja; Masamichi Simura
An English-Indonesian Computer Aided Translation System
web *http://www.csl.sony.co.jp/jsai/cdrom/papers/abstracts/3-1-5.html*

3) Douglas Arnold; Lorna Balkan; Siety Meijer; R. Lee Humphreys; Louisa Sadler
Machine Translation: an Introductory Guide, NCC Blackwell Ltd., London, 1994
web *http://www.essex.ac.uk/linguistics/clmt/MTbook/PostScript/*

4) John Surjadi Hartanto; Erhans A; Drs Johan Tirta; Tj. Soesieadhy, M.A.
Tata Bahasa Bahasa Inggris Lengkap, Penerbit Indah, Jakarta, 1980

5) S. Wojowasito, Prof. Drs.; Drs. Tito Wasito W.
Kamus Lengkap Inggeris-Indonesia Indonesia-Inggeris dengan ejaan yang disempurnakan, Penerbit Hasta, Bandung, 1980

6) The PHP Documentation Group
PHP Manual, 2001
web *http://www.php.net/docs.php*

LAMPIRAN

LAMPIRAN

A1. SINTAKS ATURAN MODIFIKASI SUMBER

Data dicek: String

Sintaks Ekspresi
(*|WORD)([(COMMAND)=((PARAM0),(PARAM1),...])

* berarti menerima semua bentuk string kata yang akan diproses
WORD adalah string kata tertentu yang ingin diproses dalam aturan tersebut

Daftar COMMAND dan PARAMETERS Ekspresi

Command : SUBSTR
Params : (0) start offset
(1) length offset
(2) string to compare (optional)
Fungsi : Membandingkan string offset dengan string kata yang disebutkan

contoh : *[SUBSTR=3,3,tio]
akan memenuhi semua kata yang memiliki offset string tio dari posisi karakter ke 3 sepanjang 3 karakter dengan nilai 'tio' seperti kata 'dictionary'

Sintaks Target
(<([(COMMAND)]=((PARAM0),(PARAM1),...))>)(<...>)...

Daftar COMMAND dan PARAMETERS Target

Command : REMOVE
Parameter : (0) start offset
(1) length offset
(2) string to compare
Fungsi : membuang offset string dari kondisi string yang dites pada saat tersebut

Command : ADD
Parameter : (0) offset
(1) string to add
Fungsi : menambahkan string pada string yang dites

Command : SETATTR
Parameter : (0) string atribut
Fungsi : Mengisi nilai atribut konstituen

Command : GETTYPE
Parameter : (0) simbol tipe
Fungsi : menguji apakah tipe dasar konstituen sama dengan tipe yang dimasukkan

Command : SETMODTYPE
Parameter : (0) simbol tipe
Fungsi : mengisi nilai tipe modifikasi dengan tipe yang dimasukkan

Command : RETURNWORD
Parameter : (0) kata yang ingin dibuat konstituennya (optional)
Fungsi : mengembalikan konstituen kata yang diinginkan atau string yang sedang diuji pada saat itu

A2. SINTAKS ATURAN TRANSFORMASI

Data dicek: structure of Constituent

Sintaks Ekspresi

Ekspresi aturan yang ditolak :

#:([(INDEX)(OPERATOR)(TYPE_SYMBOL),(COMMAND=PARAM),...])([...])...

Ekspresi transformasi abstrak :

([(INDEX)(OPERATOR)(TYPE_SYMBOL),(COMMAND=PARAM),...])([...])...

Ekspresi transformasi struktur :

(TARGET_TYPE_SYMBOL):([(INDEX)(OPERATOR)(TYPE_SYMBOL),(COMMAND=PARAM),...])([...])...

INDEX — adalah nomor urut elemen yang akan dijadikan referensi pada ekspresi target

OPERATOR — adalah jenis operator penentu urutan konstituen yang diinginkan, yaitu:

- . (titik) — artinya konstituen tepat berada disebelah konstituen lainnya dalam satu tingkat struktur pohon
- =(sama dengan) — artinya konstituen tepat berada disebelah konstituen lainnya dalam lebih dari satu tingkat struktur pohon
- @ (at) — artinya konstituen tidak harus tepat berada disebelah konstituen lainnya namun tetap memiliki node-moyang yang sama pada semua tingkat berapapun

TYPE_SYMBOL — adalah simbol tipe dari constituent yang akan dijadikan elemen

TARGET_TYPE_SYMBOL adalah simbol tipe dari constituent baru yang akan dijadikan sebagai pangkal cabang tree

Daftar COMMAND dan PARAMETER Ekspresi

Command : STRING
Parameter : string
Fungsi : menguji apakah string token asal sama dengan string yang dimasukkan

Command : FIRSTINDEX
Parameter : nomor index
Fungsi : menguji apakah index terdepan sebuah konstituen sama dengan nomor index yang dimasukkan

Command : LASTINDEX
Parameter : nomor index
Fungsi : menguji apakah index terakhir sebuah konstituen sama dengan nomor index yang dimasukkan

Command : GETATTR
Parameter : string atribut
Fungsi : menguji apakah ada atribut yang dimasukkan pada atribut konstituen

Sintaks Target

((INDEX)(>(STATE))|((.(STRING_TO_ADD)))(...)

INDEX adalah nomor urut elemen yang akan dijadikan referensi pada ekspresi target
STATE adalah nama kondisi yang akan disematkan pada elemen
STRING_TO_ADD adalah sebuah string yang akan ditambahkan di awal atau di akhir element kata

A3. SINTAKS ATURAN MODIFIKASI TUJUAN

Data dicek: Constituent

Sintaks Ekspresi

((STATE)=)(*|WORD)([(COMMAND)=((PARAM0),(PARAM1),...])

STATE adalah sebuah kondisi khusus yang disematkan pada sebuah constituen yang akan memeriksa constituent yang memiliki kondisi tersebut saja
* berarti menerima semua bentuk string kata yang akan diproses

WORD adalah string kata tertentu yang ingin diproses dalam aturan tersebut

Daftar COMMAND dan PARAMETERS Ekspresi

Command : SUBSTR
Parameter : (0) start offset
(1) length offset
(2) string to compare (optional)
Fungsi : Membandingkan string offset dengan string kata yang disebutkan

Command : GETTYPE
Parameter : (0) simbol tipe
Fungsi : Mengecek tipe dasar sebuah constituent

Command : GETMODTYPE
Parameter : (0) simbol tipe
Fungsi : mengecek tipe modifikasi sebuah constituent

Command : GETATTR
Parameter : (0) string atribut
Fungsi : menguji apakah ada atribut yang dimasukkan pada atribut konstituen

Command : GETCONSTTYPE
Parameter : (0) simbol tipe
Fungsi : menguji tipe modifikasi dan tipe dasar dari sebuah konstituen

contoh: MAKEACTIVE=*[SUBSTR=0,1,t]
akan memenuhi semua constituent dengan string tujuan berawalan 't' dengan status MAKEACTIVE

Sintaks Target

(<([(COMMAND)]=((PARAM0),(PARAM1),...))>)(<...>)...

Daftar COMMAND dan PARAMETERS Target

Command : REMOVE
Parameter : (0) start offset
(1) length offset
(2) string to compare (optional)
Fungsi : membuang offset string dari kondisi string yang dites pada saat tersebut

Command : ADD
Parameter : (0) offset
(1) string to add
Fungsi : menambahkan string pada string yang dites

PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

Command : GETATTR
Parameter : (0) string atribut
Fungsi : menguji apakah ada atribut yang dimasukkan pada atribut konstituen

Command : RETURNNULL
Parameter : -
Fungsi : mengembalikan nilai string token tujuan kosong

Command : *
Parameter : -
Fungsi : mengembalikan string token tujuan apa adanya

B1. TABEL ATURAN TIPE UJI COBA

(*Types Rules*)

ID	Symbol	Information_1	Information_2
-2	#	Unaccepted Type	tipe yang ditolak
-1	*	undefined	tipe tidak terdefinisi
3	ADJ	Adjective	Kata sifat
16	ADV	Adverb	Keterangan
26	ADV_PLACE	Place Adverb	Keterangan tempat
25	ADV_TIME	Time Adverb	Keterangan waktu
37	ADV_WAY	Way Adverb	Keterangan cara
8	ART	Article	Kata sandang
11	BE	to be	to be
36	CONJ	Conjuction	Kata Sambung
1	DET	determinan	Kata depan
33	INTJ	Interjection	Ekspresi
2	N	noun	Kata benda
40	NP	Noun Phrase	Frase benda
46	NUMBER	Word for number	Kata untuk angka
15	N_PLUR	Noun Plural	Kata benda jamak
34	POSB	Posibilities	Kemungkinan
14	PREP	Preposition	Kata depan
35	PRONOUN	Pronoun	Kata ganti
4	V	verb	Kata Kerja
41	VP	Verb Phrase	Frase Kerja
10	V_ING	Verb +ing	verb +ing
44	W_HAVE	special type	Tipe khusus untuk kata have, has, dan had
42	W_NOT	Special Type	Tipe khusus untuk kata not
17	W_TO	Spesial Type	Tipe khusus untuk kata to

B2. TABEL ATURAN MODIFIER SUMBER UJI COBA

(*Source Modifiers Rules*)

ID	Expressing	Target	Information
1	*[SUBSTR=-3,3,ing]	<[REMOVE=-3,3,ing][GETTYPE=V][SETMODTYPE=V_ING][RETURNWORD]><[REMOVE=-3,3,ing][ADD=END,e][GETTYPE=V][SETMODTYPE=V_ING][RETURNWORD]>	Mendapatkan konstituen Verb+ing
8	*[SUBSTR=-4,4,ying]	<[REMOVE=-4,4,ying][ADD=END,ie][GETTYPE=V][SETMODTYPE=V_ING][RETURNWORD]>	Mendapatkan konstituen Verb+ing
11	*[SUBSTR=-1,1,s]	<[REMOVE=-1,1,s][GETTYPE=N][SETMODTYPE=N_PLUR][RETURNWORD]><[REMOVE=-1,1,s][GETTYPE=ADV][SETATTR=PLURAL][RETURNWORD]>	Mendapatkan konstituen kata benda dan keterangan jamak berakhiran
13	i'm	<[RETURNWORD=i][RETURNWORD=am]>	Mendapatkan singkatan 'i am'
14	i've	[RETURNWORD=i][RETURNWORD=have]	Mendapatkan singkatan 'i have'
16	let's	[RETURNWORD=let][RETURNWORD=us]	Mendapatkan singkatan 'let us'
17	you'll	[RETURNWORD=you][RETURNWORD=will]	Mendapatkan singkatan 'you will'
19	aren't	[RETURNWORD=are][RETURNWORD=not]	Mendapatkan singkatan 'are not'
20	can't	[RETURNWORD=can][RETURNWORD=not]	Mendapatkan singkatan 'can not'
21	couldn't	[RETURNWORD=could][RETURNWORD=not]	Mendapatkan singkatan 'could not'
22	didn't	<[RETURNWORD=did][RETURNWORD=not]>	Mendapatkan singkatan 'did not'
23	doesn't	[RETURNWORD=does][RETURNWORD=not]	Mendapatkan singkatan 'does not'
24	don't	[RETURNWORD=do][RETURNWORD=not]	Mendapatkan singkatan 'do not'
25	hadn't	[RETURNWORD=had][RETURNWORD=not]	Mendapatkan singkatan 'had not'

26	hasn't	[RETURNWORD=has][RETURNWORD=not]	Mendapatkan singkatan 'has not'
27	he'd	<[RETURNWORD=he][RETURNWORD=had]><[RETURNWORD=he][RETURNWORD=would]>	Mendapatkan singkatan 'he had' & 'he would'
28	he'll	<[RETURNWORD=he][RETURNWORD=will]>	Mendapatkan singkatan 'he will'
29	haven't	[RETURNWORD=have][RETURNWORD=not]	Mendapatkan singkatan 'have not'
30	he's	<[RETURNWORD=he][RETURNWORD=is]><[RETURNWORD=he][RETURNWORD=has]>	Mendapatkan singkatan 'he is' & 'he has'
31	here's	[RETURNWORD=here][RETURNWORD=is]	Mendapatkan singkatan 'here is'
32	i'd	<[RETURNWORD=i][RETURNWORD=had]><[RETURNWORD=i][RETURNWORD=would]><[RETURNWORD=i][RETURNWORD=should]>	Mendapatkan singkatan 'i had', 'i would', 'i should'
33	i'll	<[RETURNWORD=i][RETURNWORD=will]><[RETURNWORD=i][RETURNWORD=shall]>	Mendapatkan singkatan 'i will', 'i shall'
34	isn't	[RETURNWORD=is][RETURNWORD=not]	Mendapatkan singkatan 'is not'
35	it'd	<[RETURNWORD=it][RETURNWORD=had]><[RETURNWORD=it][RETURNWORD=would]>	Mendapatkan singkatan 'it had', 'it would'
36	it'll	[RETURNWORD=it][RETURNWORD=will]	Mendapatkan singkatan 'it will'
38	mustn't	[RETURNWORD=must][RETURNWORD=not]	Mendapatkan singkatan 'must not'
39	needn't	[RETURNWORD=need][RETURNWORD=not]	Mendapatkan singkatan 'need not'
40	oughtn't	[RETURNWORD=ought][RETURNWORD=not]	Mendapatkan singkatan 'ought not'
41	shan't	<[RETURNWORD=shall][RETURNWORD=not]>	Mendapatkan singkatan 'shall not'
42	she'd	<[RETURNWORD=she][RETURNWORD=had]><[RETURNWORD=she][RETURNWORD=would]>	Mendapatkan singkatan 'she had', 'she would'

43	she'll	[RETURNWORD=she][RETURNWORD=will]	Mendapatkan singkatan 'she will'
44	she's	<[RETURNWORD=she][RETURNWORD=is]><[RETURNWORD=she][RETURNWORD=has]>	Mendapatkan singkatan 'she is', 'she has'
45	shouldn't	[RETURNWORD=should][RETURNWORD=not]	Mendapatkan singkatan 'should not'
46	that'll	[RETURNWORD=that][RETURNWORD=will]	Mendapatkan singkatan 'that will'
47	that's	<[RETURNWORD=that][RETURNWORD=is]><[RETURNWORD=that][RETURNWORD=has]>	Mendapatkan singkatan 'that is', 'that has'
48	there's	[RETURNWORD=there][RETURNWORD=is]	Mendapatkan singkatan 'there is'
49	they'd	<[RETURNWORD=they][RETURNWORD=had]><[RETURNWORD=they][RETURNWORD=would]>	Mendapatkan singkatan 'they had', 'they would'
50	they'll	[RETURNWORD=they][RETURNWORD=will]	Mendapatkan singkatan 'they will'
51	they're	[RETURNWORD=they][RETURNWORD=are]	Mendapatkan singkatan 'they are'
52	they've	[RETURNWORD=they][RETURNWORD=have]	Mendapatkan singkatan 'they have'
53	wasn't	[RETURNWORD=was][RETURNWORD=not]	Mendapatkan singkatan 'was not'
54	we'd	<[RETURNWORD=we][RETURNWORD=had]><[RETURNWORD=we][RETURNWORD=would]>	Mendapatkan singkatan 'we had', 'we would'
55	we'll	<[RETURNWORD=we][RETURNWORD=will]><[RETURNWORD=we][RETURNWORD=shall]>	Mendapatkan singkatan 'we will', 'we shall'
56	weren't	[RETURNWORD=were][RETURNWORD=not]	Mendapatkan singkatan 'were not'
57	we've	[RETURNWORD=we][RETURNWORD=have]	Mendapatkan singkatan 'we have'
58	what's	<[RETURNWORD=what][RETURNWORD=is]><[RETURNWORD=what][RETURNWORD=has]>	Mendapatkan singkatan 'what

			is', 'what has'
59	where's	[RETURNWORD=where][RETURNWORD=is]	Mendapatkan singkatan 'where is'
60	who's	<[RETURNWORD=who][RETURNWORD=is]><[RETURNWORD=who][RETURNWORD=has]>	Mendapatkan singkatan 'who is', 'who has'
61	who've	[RETURNWORD=who][RETURNWORD=have]	Mendapatkan singkatan 'who have'
62	won't	[RETURNWORD=will][RETURNWORD=not]	Mendapatkan singkatan 'will not'
63	wouldn't	[RETURNWORD=would][RETURNWORD=not]	Mendapatkan singkatan 'would not'
64	you'd	<[RETURNWORD=you][RETURNWORD=had]><[RETURNWORD=you][RETURNWORD=would]>	Mendapatkan singkatan 'you had', 'you would'
65	you're	[RETURNWORD=you][RETURNWORD=are]	Mendapatkan singkatan 'you are'
66	you've	[RETURNWORD=you][RETURNWORD=have]	Mendapatkan singkatan 'you have'
67	*[SUBSTR=-1,1,s]	<[REMOVE=-1,1,s][GETTYPE=V][RETURNWORD]>	Mendapatkan konstituen Verb present
68	*[SUBSTR=-2,2,es]	<[REMOVE=-2,2,es][GETTYPE=V][RETURNWORD]>	Mendapatkan konstituen Verb present
69	*[SUBSTR=-2,2,ed]	<[REMOVE=-1,1,d][GETTYPE=ADV][RETURNWORD]><[REMOVE=-1,1,d][GETTYPE=V][RETURNWORD]><[REMOVE=-2,2,ed][GETTYPE=V][RETURNWORD]>	Mendapatkan konstituen Adjective dan Verb pasif
70	it's	[RETURNWORD=it][RETURNWORD=is]	
71	*[SUBSTR=-2,2,'s]	<[REMOVE=-2,2,'s][SETMODTYPE=PRONOUN][SETATTR=POSSESIVE][RETURNWORD]>	Mendapatkan konstituent kepemilikan
72	*[SUBSTR=-3,3,ied]	<[REMOVE=-3,3,ied][ADD=END,y][GETTYPE=V][SETATTR=PAST][RETURNWORD]>	Mendapatkan konstituen Verb lampau

B3. TABEL ATURAN MODIFIKASI TUJUAN UJI COBA
(*Destination Modifiers*)

ID	Expression	Target	Information
8	*[GETTYPE=N][GETMODTYPE=N_PLUR]	<[GETATTR=HUMAN][REMOVE=ALL][ADD=END,para][ADD=END,*]><[REMOVE=ALL][ADD=END,banyak][ADD=END,*]><[REMOVE=ALL][ADD=END,*][ADD=END,-][ADD=END,*]>	penanda kata jamak -pengulangan -'banyak'...
9	MAKEACTIVE=*[SUBSTR=0,1,t][GETTYPE=V]	<[GETATTR=BASE][REMOVE=0,1,t][ADD=START,men]><*>	imbuhan me+t* = men*
10	NONE=*	<RETURNNULL>	kembalikan kosong
11	MAKEACTIVE=*[SUBSTR=0,1,m][GETTYPE=V]	<[GETATTR=BASE][ADD=START,me]><*>	imbuhan me+m* = mem*
12	MAKEACTIVE=*[SUBSTR=0,1,b][GETTYPE=V]	<[GETATTR=BASE][ADD=START,mem]><*>	imbuhan me+b* = memb*
13	MAKEACTIVE=*[SUBSTR=0,1,p][GETTYPE=V]	[GETATTR=BASE][REMOVE=0,1,p][ADD=START,mem]	imbuhan me+p* = mem*
14	MAKEACTIVE=*[SUBSTR=0,1,a][GETTYPE=V]	[ADD=START,meng]	imbuhan me+a* = menga*
15	MAKEACTIVE=*[SUBSTR=0,1,e][GETTYPE=V]	[ADD=START,meng]	imbuhan me+e* = menge*
16	MAKEACTIVE=*[SUBSTR=0,1,u][GETTYPE=V]	[ADD=START,meng]	imbuhan me+u* = mengu*
17	MAKEACTIVE=*[SUBSTR=0,1,o][GETTYPE=V]	[ADD=START,meng]	imbuhan me+o* = mengo*
18	MAKEACTIVE=*[SUBSTR=0,1,i][GETTYPE=V]	[ADD=START,meng]	imbuhan me+i* = mengi*
19	MAKENOUN=*[GETTYPE=V]	[ADD=END,an]	merubah V->N
20	SETDEFAULTMEAN=*	<[*]>	mengembalikan arti asli

21	*[GETMODTYPE=PRONOUN][GETATTR=POSSESIVE]	[ADD=START,milik]	menyatakan kepunyaan

B4. TABEL ATURAN TRANSFORMASI UJI COBA
(*Transformation Rules*)

ID	Expression	Target	Information
94	#:(1.*)(2.*)		ditolak
68	#:(1.ADJ)(2.V)		ditolak
88	#:(1.ADV_PLACE)(2.BE)		ditolak
89	#:(1.ADV_PLACE)(2.PREP)		ditolak
108	#:(1.V)(2.PRONOUN,GETATTR=SUBJECT)		ditolak
49	#:(1.V)(2.V)		ditolak
60	#:(1.V)(2.W_NOT)		ditolak
75	(1.ART)(2=N_PLUR)	(1)(2>SETDEFAULTMEAN)	artikel+plural
82	(1.ART,GETATTR=SINGLE)(2@N,GETATTR=ANIMAL)	(1>NONE)(.seekor)(2>SETDEFAULTMEAN)	a=seekor
67	(1.ART,GETATTR=SINGLE)(2@N,GETATTR=HUMAN)	(.seorang)(1>NONE)(2)	a=seorang
70	(1.BE)(2.PREP)	(1>NONE)(.sedang berada)(2)	posisi
91	(1.BE)(2.PRONOUN)	(1>NONE)(2)	be+plural
71	(1.BE)(2.W_NOT)(3=ADJ)	(1>NONE)(.tidak)(2>NONE)(3)	sifat negative
66	(1.BE,FIRSTINDEX=0)(2.W_NOT)(3=N)(4=ADJ)	(1>NONE)(.bukankah)(2)(.itu)(3)(4)	kata tanya sifat negative
61	(1.BE,FIRSTINDEX=0)(2.W_NOT)(3@PRONOUN,GETATTR=SUBJECT)(4@ADJ)	(1>NONE)(.tidakkah)(2>NONE)(3)(.itu)(4)	question tag
65	(1.BE,FIRSTINDEX=0)(2.W_NOT)(3@V)	(1>NONE)(.jangan)(2)(3)	negative verb

62	(1.BE,FIRSTINDEX=0,LASTINDEX=0)(2=PRONOUN,GETATTR=SUBJECT)	(.benarkah,apakah)(1>NONE)(2)	be tanya
102	(1.DET,STRING=by)(2@N,GETATTR=VEHICLE)	(.menggunakan)(1>NONE)(2)	by=menggunakan
64	(1.N)(2.BE)(3.W_NOT)(4=V)	(1)(2>NONE)(.tidak)(3)(4)	negative verb
112	(1.POSB,FIRSTINDEX=0)	(1)(.kah)	kata tanya kemungkinan
69	(1.PREP,STRING=in)(2=N,GETATTR=PLACE)	(.di dalam)(1>NONE)(2)	in=di dalam
96	(1.PRONOUN,GETATTR=INTERROGATIVE)(2.BE)(3.NP)	(1)(2>NONE)(3)	be=
83	(1.PRONOUN,GETATTR=RELATIVE,FIRSTINDEX=0)(2.BE)	(1)(2>NONE)	be=
100	(1.PRONOUN,STRING=how much)(2.BE)(3@N,STRING=cost)	(.berapa harga)(1>NONE)(2>NONE)(3>NONE)	tanya harga
105	(1.PRONOUN,STRING=it)(2@V,STRING=take)(3@ADV,GETATTR=TIME)	(1>NONE)(.membutuhkan waktu)(2>NONE)(3)	it take=membutuhkan waktu
98	(1.PRONOUN,STRING=what)(2.BE)(3@N,STRING=name)	(.siapa)(1>NONE)(2>NONE)(3)	what=siapa
107	(1.PRONOUN,STRING=where)(2.BE)(3@V,STRING=go)	(1>NONE)(2>NONE)(.kemana)(3)	where=kemana
109	(1.V,STRING=get)(2.V,STRING=up)	(.bangun)(1>NONE)(2>NONE)	get up=bangun
113	(1.W_HAVE)(2.BE,STRING=been)	(1>NONE)(2)	has been=telah
87	(1.W_HAVE,FIRSTINDEX=0)(2.NP)(3.NP)	(.apakah)(1>NONE)(2)(.memiliki)(3)	have=memiliki (tanya)
76	(1.W_HAVE,FIRSTINDEX=0)(2=PRONOUN)(3.NP)	(.apakah)(1>NONE)(2)(.memiliki)(3)	have=memiliki (tanya)
53	(1.W_TO)(2=V)	(1>NONE)(2>MAKEACTIVE)	infinitive

52	(1=BE)(2@ADJ)	(1>NONE)(.itu)(2)	menunjukkan sebuah fakta
85	(1=BE)(2@ADJ,GETATTR=COLOR)	(1>NONE)(.berwarna)(2)	tambahan untuk warna
86	(1=BE)(2@ADJ,GETATTR=TASTE)	(1>NONE)(.rasanya)(2)	tambahan untuk rasa
84	(1=BE)(2@ADJ,GETATTR=VALUE)	(1>NONE)(.harganya)(2)	tambahan untuk harga
51	(1=BE,GETATTR=PRESENT)(2=V_ING)	(1>NONE)(.sedang)(2>MAKEACTIVE)	kalimat present continuous tense
50	(1=N)(2.BE)(3.NP)	(1)(2>NONE)(.adalah)(3)	be=adalah
56	(1=N)(2=V)	(1)(2>MAKEACTIVE)	active verb
104	(1=NUMBER,STRING!one)(2=N_PLUR)	(1))(.buah)(2>SETDEFAULTMEAN)	angka+benda jamak
72	(1=N_PLUR)(2.BE)(3=N_PLUR)	(1>SETDEFAULTMEAN)(2)(3>SETDEFAULTMEAN)	jamak+jamak=tunggal saja
77	(1=PRONOUN)(2.W_HAVE)(3.NP)	(1)(.memiliki)(2)(3)	have=memiliki
74	(1=PRONOUN)(2.W_HAVE)(3.W_NOT)(4.NP)	(1)(2>NONE)(.tidak memiliki)(3>NONE)(4)	have not=tidak memiliki
92	(1=PRONOUN,GETATTR=RELATIVE,GETATTR=PLURAL)(2=N_PLUR)	(1)(2>SETDEFAULTMEAN)	relative+jamak=tunggal
93	(1=PRONOUN,STRING=it)(2.BE)(3=PRONOUN,GETATTR=OBJECT)	(1>NONE)(2)(3)	it=
47	(1=PRONOUN,STRING=there)(2.BE)(3.NP)	(.ada)(1>NONE)(2>NONE)(3)	there=ada
79	(1=PRONOUN,STRING=there)(2=BE)(3.W_NOT)	(1>NONE)(.tidak ada)(2>NONE)(3>NONE)	there not=tidak ada
111	ADV:(1.ADV,GETATTR=SERIES)(2=N)	(2)(1)	[series]+N = N+[series]
46	N:(1.ART)(2.V)		art+v = kt benda
103	N:(1.NUMBER)		angka = kt benda
54	N:(1.PRONOUN,GETATTR=POSSESIVE)(2.V)	(2>MAKENOUN)(1)	possesive+veb = noun+possesive
81	NP:(1.*)		unknown = NP
48	NP:(1.*)(2.NP)	(2)(1)	*+NP = NP+*
44	NP:(1.ADJ)(2.NP)	(2)(.yang)(1)	MD = DM

80	NP:(1.ART)(2.ADV)		art+kt keterangan = NP
45	NP:(1.ART)(2.NP)		art+NP = NP
34	NP:(1.DET)(2.NP)		det+NP = NP
58	NP:(1.N)		N = NP
63	NP:(1.N)(2.NP)		N+NP = NP
106	NP:(1.NP)(2.ADV)		NP+kt keterangan = NP
41	NP:(1.N_PLUR)		kt benda jamak = NP
97	NP:(1.PRONOUN,GETATTR=DEMONSTRATIVE)(2=N)	(2)(1)	demonstrative+N = N+demonstrative
90	NP:(1.PRONOUN,GETATTR=DEMONSTRATIVE)(2=NP)	(2)(1)	demonstrative+NP = NP+demonstrative
99	NP:(1.PRONOUN,GETATTR=POSSESIVE)(2=N)	(2)(1)	possesive+noun=noun +possesive
55	NP:(1.PRONOUN,GETATTR=SUBJECT)(2.NP)		[subject]+NP = NP
110	POSB:(1.POSB)(2.W_NOT)	(2)(1)	kemungkinan negative dibalik
101	PRONOUN:(1.PRONOUN,GETATTR=POSSESIVE)(2.PRONOUN,GETATTR=POSSESIVE)	(2)(1)	possesive+possesive dibalik
13	V:(1.V_ING)		V+ing = V
57	VP:(1.V)		V = VP
36	VP:(1.VP)(2.NP)		VP + NP = NP