IMPLEMENTASI COREFERENCE PADA KONVERSI BAHASA INDONESIA KE BAHASA QUERY TERSTRUKTUR (SQL) UNTUK PERUBAHAN STRUKTUR TABEL

Fauzan Abdulwahid[1], Alif Finandhita[2]

[1,2] Teknik Informatika –Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung 40132
E-mail : dragonknightfarawell@email.unikom.ac.id[1], alif.finandhita@email.unikom.ac.id[2]

ABSTRAK

NLP (*Natural Language Processing*) merupakan keilmuan yang memproses bahasa alami atau bahasa yang sering digunakan oleh manusia agar dapat dimengerti komputer.Sudah ada beberapa penelitian translasi bahasa alami ke SQL (*Structured Query Language*) pada kalimat tunggal dengan berbagai metode, kemudian penelitian oleh Iqram Anwar yang dapat menangani kalimat majemuk menggunakan metode *rule based* dengan hasil akurasi akhir sebesar 72.04%. Perolehan nilai akhir tersebut dikarenakan pada penelitian tidak dapat mentranslasikan perubahan struktur tabel jika terdapat kata rujukan pada kalimatnya. Implementasi Coreference Pada Konversi Bahasa Indonesia Ke Bahasa Query Terstruktur (Sql) Untuk Perubahan Struktur Tabel ini bertujuan untuk membangun suatu proses yang dapat menerjemahkan kalimat baku Bahasa Indonesia. Menggunakan metode *rule based* yang digabung dengan logika *coreference* dengan tahapan melakukan ekstraksi kalimat masukan terlebih dahulu lalu mengambil *template* SQL dari *database* berdasarkan *keyword – keyword* yang sudah diperoleh. Hasil akurasi dari pengujian akhir adalah 89,47% yang menunjukan metode ini cukup efektif namun belum optimal karena tidak dapat menangani perintah yang merujuk pada perintah sebelumnya. Pada penelitian selanjutnya, penyempurnaan *coreference* atau metode pengambilan perintah rujukan dapat mengatasi masalah tersebut..

Kata kunci : *Natural Language Processing*, *Structured Query Language*, *Converter*, Bahasa Indonesia, *Coreference*.

1. PENDAHULUAN

NLP (*Natural Language Processing*) atau pemrosesan bahasa alami memungkinkan manusia berinteraksi dengan komputer menggunakan bahasa sehari-hari. Salah satu terapan NLP adalah translasi, yaitu menerjemahkan bahasa alami manusia ke bahasa yang dapat diproses oleh komputer[1].

Sudah ada beberapa penelitian yang membahas tentang pemrosesan bahasa alami ke bahasa SQL(*Structured Query Language*), baik menggunakan metode translasi yang melihat pada struktur kalimat masukan[2][3][4] maupun metode yang hanya mentranslasi berdasarkan keyword saja[5][6][7].

Penelitian translasi Bahasa Indonesia ke Bahasa SQL dengan metode translasi yang hanya melihat keywordnya saja diantaranya ada penelitian oleh Kuspriyanto yang memproses *query select* pada basis data akademik dengan perolehan akurasi sebesar 86%[5]. Penelitian oleh Defy M. A yang memproses DDL (*Data Definition Language*) dengan perolehan akurasi sebesar 92.08%[6]. Penelitian oleh Iqram A. yang memproses DDL dan dapat menangani kalimat majemuk dengan perolehan akurasi sebesar 72.04%[7]. Perolehan akurasi pada penelitian Iqram turun dikarenakan proses tidak dapat mentranslasi kalimat yang terdapat kata rujukan didalamnya.

Berdasarkan pemaparan di atas, dapat disimpulkan bahwa dibutuhkannya suatu pengembangan metode translasi yang dapat mengani kalimat majemuk yang terdapat kata rujukan di dalamnya. Metode yang digunakan adalah metode *rule based serta* tanpa melihat struktur kalimat masukan. Metode translasi kemudian akan diuji dengan aplikasi web sederhana yang dibangun menggunakan HTML(*Hypertext Markup Language*) serta CSS(*Cascading Style Sheets*) pada *frontend*[8] dan PHP(*Hypertext Preprocessor*) pada *backend* nya[9].

2. ISI PENELITIAN

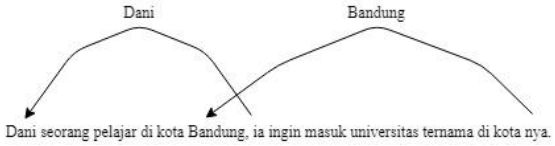
2.1 *Natural Language Processing*

NLP merupakan cabang dari keilmuan AI (*Artificial Intelligence*) yang membahas tentang hubungan manusia dengan komputer lewat bahasa sehari – hari manusia. Pemrosesan bahasa alami tidaklah mudah karena perbedaan pada cara kerja otak manusia dan komputer saat mengolah bahasa.

Manusia dapat membedakan makna suatu kata dengan melihat keseluruhan kalimat, tanda baca, letak kata, dan dengan mudah mempelajari jika menemukan kata baru. Sedangkan pada komputer ambiguitas sering terjadi karena pada pemrosesan kata dibatasi oleh aturan yang sudah dibuat[1].

2.2 *Coreference*

Coreference merupakan solusi untuk menyelesaikan masalah penentuan suatu ekspresi rujukan atau acuan kepada entitas, orang, atau benda pada bahasa alami[10].



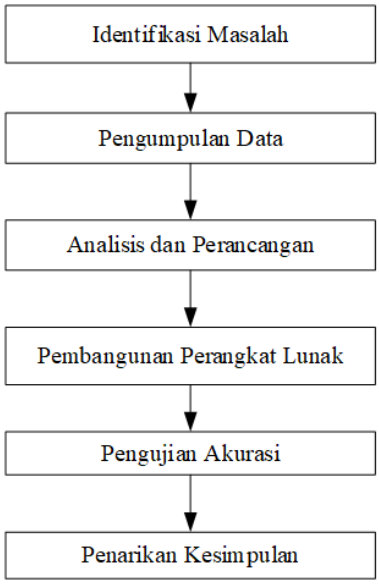
Gambar 1. Contoh *Coreference*

Pada Gambar 1, terdapat kalimat yang di dalamnya ada kata orang yaitu 'Dani' dan kata entitas kota 'Bandung', kemudian terdapat kata yang merujuk pada kata sebelumnya yaitu 'ia' dan 'nya'. Dengan penggunaan coreference kata merujuk tersebut dapat memiliki makna orang dan entitas kota yaitu 'Dani' dan 'Bandung'.

2.3 Metode Penelitian

Penelitian ini menggunakan metode penelitian deskriptif, yakni metode yang menggambarkan situasi dan kejadian berdasarkan fakta dan interpretasi yang tepat[11].

Berikut adalah tahapan yang dilakukan pada penelitian ini.



Gambar 2. Tahap Penelitian

1. Identifikasi Masalah

Berangkat dari penelitian sebelumnya, yang dilakukan pada tahap ini adalah mencari dan mengidentifikasi masalah – masalah pada penelitian sebelumnya agar dapat fokus pada penyelesaian masalah tersebut.

2. Pengumpulan Data

Metode pengumpulan data yang digunakan adalah studi literatur, yakni mengumpulkan data dari literatur, paper, jurnal, dan buku yang berhubungan dan membantu terhadap penelitian.

3. Analisis dan Perancangan

Pada tahap ini dilakukan analisis masalah yang sudah ditemukan sehingga dapat dilakukan perancangan solusi yang akan menyelesaikan masalah tersebut.

4. Pembangunan Perangkat Lunak

Pada tahap ini dilakukan pembangunan perangkat lunak untuk menguji perancagan solusi yang sudah didapatkan.

5. Pengujian Akurasi

Pengujian akurasi menggunakan metode uji kasus dilakukan untuk melihat seberapa besar nilai akurasi yang diperoleh dari solusi yang sudah ditentukan.

6. Penarikan Kesimpulan

Pada tahap ini yang dilakukan adalah menarik kesimpulan dari penelitian yang sudah dilakukan.

2.4 Analisis Masalah

Penelitian oleh Iqram Anwar[7] dapat menangani kalimat majemuk yang belum dapat ditangani oleh penelitian – penelitian lain[2][3][4][5][6], namun terdapat kekurangan pada metode translasinya sehingga tidak dapat dengan baik memproses perubahan struktur tabel. Hal ini terjadi karena sistem tidak bisa memproses kata yang merujuk pada kata lain sehingga penambahan *coreference* dapat mengatasi masalah tersebut.

Tabel 1. Tabel contoh query yang salah

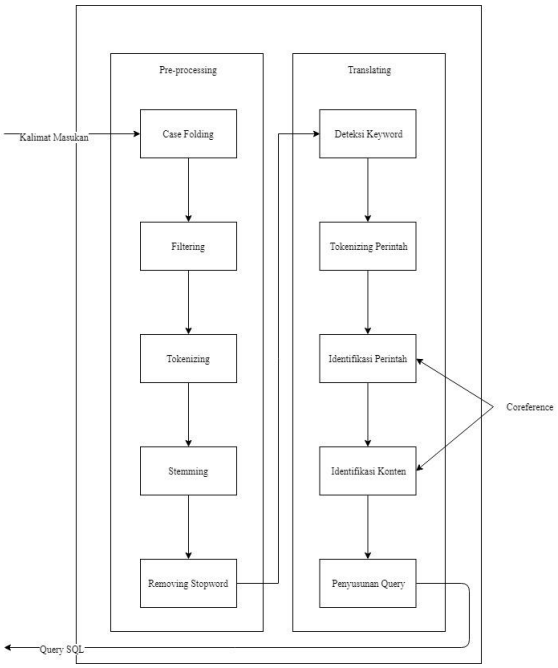
Kalimat Masukan	Harapan	Hasil
hapus index idx_nim pada tabel mahasiswa lalu tambah kolom umur dengan tipe data integer	DROP INDEX idx_nim ON mahasiswa ;	DROP INDEX idx_nim ON mahasiswa;
	ALTER TABLE mahasiswa ADD umur integer(25) ;	ALTER TABLE umur ADD integer varchar(255);

Pada Tabel 1, terlihat perintah kedua tidak sesuai harapan karena tidak disebutkan nama tabelnya. Jika diasumsikan nama tabel adalah sebuah entitas, maka masalah tersebut dapat diatasi jika ditambahkan *coreference* sehingga proses akan mengambil nama tabel dari perintah pertama[10].

2.5 Gambaran Umum Sistem

Pada dasarnya, penelitian ini merupakan pengembangan dari dua penelitian sebelumnya[6][7],

sehingga terdapat dua tahapan utama yaitu tahap *Preprocessing* dan Translasi.



Gambar 2. Gambaran Umum Sistem

Proses – proses yang dilakukan pada tahap *Preprocessing* meliputi:

1. *Case Folding*

Penyeragaman karakter dilakukan di tahap ini, oleh karena itu kalimat masukan dijadikan huruf kecil.

Tabel 2. Tabel *Case Folding*

Sebelum	Sesudah
Hapuslah index idx_nim pada tabel mahasiswa lalu tambah kolom umur dengan tipe data integer!	**h**apuslah index idx_nim pada tabel mahasiswa lalu tambah kolom umur dengan tipe data integer!

2. *Filtering*

Pada tahap ini dilakukan penyaringan terhadap karakter yang tidak diperlukan hingga menyisakan karakter tertentu saja.

Tabel 3. Tabel *Filtering*

Sebelum	Sesudah
hapuslah index idx_nim pada tabel mahasiswa lalu tambah kolom umur dengan tipe data integer**!**	hapuslah index idx_nim pada tabel mahasiswa lalu tambah kolom umur dengan tipe data integer

3. *Tokenizing*

Pada tahap ini kalimat masukan yang sudah diproses akan dipisah kata per kata.

Tabel 4. Tabel *Tokenizing*

Sebelum	Sesudah	
	Indeks	**Token**
hapuslah index idx_nim pada tabel mahasiswa lalu tambah kolom umur dengan tipe data integer	1	hapuslah
	2	index
	3	idx_nim
	4	pada
	5	tabel
	6	mahasiswa
	7	lalu
	8	tambah
	9	kolom
	10	umur
	11	dengan
	12	tipe
	13	data
	14	integer

4. *Stemming*

Pada tahap ini tiap token kata dikembalikan ke bentuk dasarnya dengan menghilangkan imbuhannya.

Tabel 5. Tabel *Stemming*

Indeks	Sebelum	Sesudah
1	hapus**lah**	hapus
2	index	index
3	idx_nim	idx_nim
4	pada	pada
5	tabel	tabel
6	mahasiswa	mahasiswa
7	lalu	lalu
8	tambah	tambah
9	kolom	kolom
10	umur	umur
11	dengan	dengan
12	tipe	tipe
13	data	data
14	integer	integer

5. *Removing Stopword*

Pada tahap ini dilakukan penyaringan terhadap kata yang tidak diperlukan sehingga menyisakan karakter tertentu saja.

Tabel 6. Tabel *Removing Stopword*

Indeks	Sebelum	Sesudah	Hasil	
			Indeks	**Token**
1	hapus	hapus	1	hapus
2	index	index	2	index
3	idx_nim	idx_nim	3	idx_nim
4	pada		4	tabel
5	tabel	tabel	5	mahasiswa
6	mahasiswa	mahasiswa	6	tambah
7	lalu		7	kolom
8	tambah	tambah	8	umur
9	kolom	kolom	9	tipe

10	umur	umur	10	data
11	dengan		11	integer
12	tipe	tipe		
13	data	data		
14	integer	integer		

Setelah tahap *Preprocessing* dilakukan, maka tahap selanjutnya adalah Translasi yang meliputi:

1. Deteksi *Keyword*

Pada tahap ini setiap token akan dibandingkan dengan database *keyword*. Tahap ini untuk menentukan perintah tunggal atau majemuk.

Tabel 7. Tabel Deteksi *Keyword*

Indeks	Token	Keyword?
1	**hapus**	Ya
2	index	Bukan
3	idx_nim	Bukan
4	tabel	Bukan
5	mahasiswa	Bukan
6	**tambah**	Ya
7	kolom	Bukan
8	umur	Bukan
9	tipe	Bukan
10	data	Bukan
11	integer	Bukan

2. *Tokenizing* Perintah

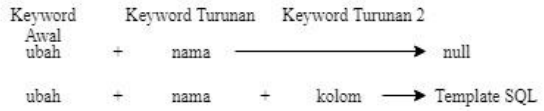
Pada tahap ini token yang sudah teridentifikasi sebagai keyword akan dipisahkan agar kalimat perintah dapat terpisah.

Tabel 8. Tabel *Tokenizing* Perintah

Sebelum		Sesudah		
Indeks	**Token**	**Perintah**	**Indeks**	**Token**
1	**hapus**	1	1	hapus
2	index		2	index
3	idx_nim		3	idx_nim
4	tabel		4	tabel
5	mahasiswa		5	mahasiswa
6	**tambah**	2	1	tambah
7	kolom		2	kolom
8	umur		3	umur
9	tipe		4	tipe
10	data		5	data
11	integer		6	integer

3. Identifikasi Perintah

Pada tahap ini setiap token di identifikasi dengan cara membandingkan dengan kamus *keyword* awal dan *keyword* turunan sehingga mendapatkan struktur kalimat SQL, bila *keyword* awal dan *keyword* turunan tidak menghasilkan template SQL, maka akan dibandingkan dengan *keyword* turunan kedua untuk mendapatkan struktur kalimat SQL nya.



Gambar 3. Pengambilan *Template* SQL

Tabel 9. Tabel *Identifikasi* Perintah

Perintah	Indeks	Token	Keyword	Template
1	1	hapus	hapus + index	ALTER TABLE {{nama_tabel}} DROP {{nama_index}};
	2	index		
	3	idx_nim		
	4	tabel		
	5	mahasiswa		
2	1	tambah	tambah + kolom	ALTER TABLE {{nama_tabel}} ADD {{kolom_tipedata}};
	2	kolom		
	3	umur		
	4	tipe		
	5	data		
	6	integer		

Setelah didapatkan template SQL nya, token keyword dihapus sehingga menyisakan token yang belum terpakai.

Tabel 10. Tabel Token Yang Tersisa

Perintah	Indeks	Token
1	1	idx_nim
	2	tabel
	3	mahasiswa
2	1	umur
	2	tipe
	3	data
	4	integer

4. Identifikasi Konten

Pada tahap ini token-token yang tersisa di cek apakah merupakan konten dari tabel yang akan diolah seperti nama tabel dan nama kolom dari token yang tersisa, pengecekan ini menggunakan *template handlebar* dari template yang sudah diperoleh. Pada tahap inilah kata rujukan atau *coreference* diproses.

Pada kasus query SQL, sebagian besar kata rujukan digunakaan saat memilih tabel yang akan digunakan, sehingga pemrosesan *coreference* dapat dilakukan di tahap identifikasi konten, dimana token kata ‘tabel’ yang dianggap sebagai entitas[10] akan dicari pada kumpulan token perintah, dan bila tidak terdapat kata ‘tabel’ maka pencarian kata tersebut akan dilakukan terhadap perintah yang sebelumnya.

Gambar 4. Pengambilan *Template* SQL

Pada contoh di atas, tidak ada kata tabel pada kalimat perintah kedua sehingga proses akan mengambil nama tabel dari kalimat pertama dengan cara memecah kembali kalimat perintah pertama yang sudah tersusun lalu mencari kata ‘tabel’ di dalamnya.

Terdapat tiga jenis *template handle bar* yang teridentifikasi yakni {{nama_tabel}}, {{kolom_tipedata}} dan {{nama_index}}.

- Template Handle Bar {{nama_tabel}}

Pada proses *template handle bar* ini yang harus dilakukan adalah mencari kata ‘tabel’, jika kata ‘tabel’ ditemukan maka nama table berada pada indeks kata ‘tabel’ + 1. Jika kata ‘tabel’ tidak ditemukan maka nama tabel diambil dari kata tabel yang berada pada kalimat atau template sebelumnya. Jika kata ‘tabel’ sama sekali tidak ditemukan maka nama tabel berada pada indeks ke-1. Setelah nama tabel ditentukan, token ‘tabel’ dan token nama tabel dihapus.

Pada contoh diatas, token kata ‘tabel’ ditemukan pada indeks ke-1 pada perintah pertama namun tidak ditemukan pada perintah ke-2, maka nama tabel berada pada indeks ke-2 untuk perintah pertama dan nama tabel untuk perintah kedua akan mengikuti nama tabel di perintah pertama. Setelah itu hapus token-token kata tersebut.

Tabel 11. Tabel *Handle Bar* {{nama_tabel}}

Perintah	Token	Template Handle Bar	Konten
1	idx_nim	{{nama_tabel}}	mahasiswa
		{{nama_index}}	
2	umur	{{nama_tabel}}	mahasiswa
	tipe		
	data	{{kolom_tipedata}}	
	integer		

- Template Handle Bar {{nama_index}}

Pada proses *template handle bar* ini yang harus dilakukan adalah mencari kata ‘index’, jika kata tersebut ditemukan maka nama kolom berada pada indeks kata ‘index’ + 1. Jika kata tersebut tidak ditemukan maka nama index berada pada indeks ke-1.

Pada contoh diatas, handle bar hanya ada pada perintah pertama dan token kata ‘index’ tidak ditemukan pada perintah tersebut, maka nama index berada pada indeks ke-1 untuk perintah kedua. Setelah itu hapus token-token kata tersebut.

Tabel 12. Tabel *Handle Bar* {{nama_index}}

Perintah	Token	Template Handle Bar	Konten
1	kosong	{{nama_tabel}}	mahasiswa
		{{nama_index}}	idx_nim
2	umur	{{nama_tabel}}	mahasiswa
	tipe		
	data	{{kolom_tipedata}}	
	integer		

- Template Handle Bar {{kolom_tipedata}}

Pada proses *template handle bar* ini yang harus dilakukan adalah mencari kata ‘kolom’ atau ‘field’, jika kata tersebut ditemukan maka nama kolom berada pada indeks kata ‘kolom’ + 1. Jika kata tersebut tidak ditemukan maka nama kolom berada pada indeks ke-1. Token setelah nama kolom merupakan token tipedata, bila terdapat token ‘tipe’ atau ‘data’ maka token tersebut dihapus. Setelah nama kolom dan tipe data ditentukan, token selanjutnya adalah token ukuran, setelah token ukuran ada beberapa token lain seperti status null, dan posisi kolom. Khusus untuk token nama kolom, tipe data, dan ukuran, token-token tersebut harus memiliki nilai yang valid, jika tidak maka output akan menghasilkan detail kesalahan kalimat masukan. Setelah nama kolom dan konstrain sudah ditentukan kata ‘kolom’ dan token-token setelahnya dihapus.

Pada contoh diatas, handle bar hanya ada pada perintah kedua dan token kata ‘kolom’ tidak ditemukan pada perintah tersebut, maka nama kolom berada pada indeks ke-1 untuk perintah pertama serta konstrain yang lain pada indeks setelahnya. Setelah itu hapus token-token kata tersebut.

Tabel 13. Tabel *Handle Bar* {{kolom_tipedata}}

Perintah	Token	Template Handle Bar	Konten
1	kosong	{{nama_tabel}}	mahasiswa
		{{nama_index}}	idx_nim
2	kosong	{{nama_tabel}}	mahasiswa
		{{kolom_tipedata}}	umur, integer

5. Penyusunan Query

Pada tahap ini penyusunan query dilakukan berdasarkan token perintah dan token konten yang sudah didapatkan dari proses-proses sebelumnya. Proses ini dilakukan dengan mengganti *Template Handle Bar* dengan konten yang telah diperoleh.

Tabel 14. Tabel Penyusunan Query

Perintah	Sebelum	Sesudah
1	ALTER TABLE {{nama_tabel}} DROP {{nama_index}};	ALTER TABLE mahasiswa DROP idx_nim;

2	ALTER TABLE {{nama_tabel}} ADD {{kolom_tipedata}};	ALTER TABLE mahasiswa ADD umur integer (Size invalid!);

Pada output perintah kedua terdapat kata ‘Size invalid!’ karena aturan pada *handlebar* {{kolom_tipedata}} yang membutuhkan minimal nama kolom, tipe data, dan size nya.

2.6 Hasil Pengujian

Pada bab ini dilakukan pengujian akurasi pada metode translasi yang sudah dikembangkan. Kalimat masukan merupakan kalimat majemuk dengan atau tanpa kata rujukan. Kalimat perintah akan dianggap benar jika query yang dihasilkan sesuai dengan harapan. Pengujian akurasi dilakukan pada 45 jenis kalimat ALTER yang tidak berulang, dan setiap jenis kalimat diberikan minimal 2 jenis kalimat masukan yaitu kalimat benar dan kalimat salah.

No	Jenis Perintah	Jumlah Kalimat	Hasil Klasifikasi	
			Benar	Salah
1	ADD – ADD	3	2	1
2	ADD – DROP	2	2	0
3	ADD – CHANGE	2	1	1
4	ADD – ADD PRIMARY	3	1	2
5	ADD – DROP PRIMARY	3	2	1
6	ADD – ADD FOREIGN	3	2	1
7	ADD – DROP FOREIGN	2	2	0
8	ADD – MODIFY	2	2	0
9	ADD – RENAME	4	2	2
10	DROP – DROP	2	2	0
11	DROP – CHANGE	2	2	0
12	DROP – ADD PRIMARY	2	2	0
13	DROP – DROP PRIMARY	2	2	0
14	DROP – ADD FOREIGN	2	2	0
15	DROP – DROP FOREIGN	2	2	0
16	DROP – MODIFY	2	2	0
17	DROP – RENAME	2	2	0
18	CHANGE – CHANGE	2	2	0
19	CHANGE – ADD PRIMARY	2	1	1
20	CHANGE – DROP PRIMARY	2	2	0
21	CHANGE – ADD FOREIGN	2	1	1
22	CHANGE – DROP FOREIGN	2	2	0
23	CHANGE – MODIFY	2	2	0
24	CHANGE – RENAME	2	2	0
25	ADD PRIMARY – ADD PRIMARY	2	1	1
26	ADD PRIMARY – DROP PRIMARY	2	2	0
27	ADD PRIMARY – ADD FOREIGN	2	2	0
28	ADD PRIMARY – DROP FOREIGN	2	2	0
29	ADD PRIMARY – MODIFY	2	2	0
30	ADD PRIMARY – RENAME	2	2	0
31	DROP PRIMARY – DROP PRIMARY	2	2	0
32	DROP PRIMARY – ADD FOREIGN	2	2	0
33	DROP PRIMARY – DROP FOREIGN	2	2	0
34	DROP PRIMARY – MODIFY	2	2	0
35	DROP PRIMARY – RENAME	2	2	0
36	ADD FOREIGN – ADD FOREIGN	2	2	0
37	ADD FOREIGN – DROP FOREIGN	2	2	0
38	ADD FOREIGN – MODIFY	2	2	0
39	ADD FOREIGN – RENAME	2	2	0
40	DROP FOREIGN – DROP FOREIGN	2	2	0

41	DROP FOREIGN – MODIFY	2	2	0
42	DROP FOREIGN – RENAME	2	2	0
43	MODIFY – MODIFY	2	2	0
44	MODIFY – RENAME	2	2	0
45	RENAME – RENAME	2	2	0
Total		96	85	11

Berikut merupakan perhitungan untuk nilai akurasi keseluruhan data masukan.

$$\frac{\sum Score}{n} x\ 100\% = \frac{85}{96} x\ 100\% = 88.54\% \qquad (1)$$

3. PENUTUP

3.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, sistem dapat mentranslasi kalimat majemuk yang terdapat kata yang mengacu di dalamnya. Nilai akhir dari pengujian sebesar 88.54%. Hasil tersebut belum optimal karena sistem belum bisa memproses kata perintah yang merujuk ke kata lain.

3.2 Saran

Berdasarkan analisis dari pengujian akurasi, penyebab belum optimalnya hasil translasi karena logika *coreference* belum sempurna. Penanganan kata perintah rujukan dapat mengatasi masalah ini.

DAFTAR PUSTAKA

[1] D. Anita dan M. Arhami, Konsep Kecerdasan Buatan. Yogyakarta: Andi, 2006.

[2] Saravjeet Kaur, SQL *Generation And Execution From Natural Language Processing*. MMU. Mullana.

[3] Setyawan Wibisono, Aplikasi Pengolahan Bahasa Alami untuk Query Basisdata Akademik dengan Format Data XML, Skripsi S1. Teknik Informatika. Universitas Stikubank, 2013.

[4] Andri, Penerapan Bahasa Alami Sederhana Pada *Online Public Access Catalog*(OPAC) Berbasis Web Semantik. Sistem Informasi. Universitas Binadarma Palembang, 2012.

[5] Kuspriyanto, Perancangan Translator Bahasa Alami Ke Dalam Format SQL (*Structured Query Language*). Departemen Teknik Elektro. Institut Teknologi Bandung.

[6] Defy M. Aminuddin, *Data Definition Language* (DDL) Pada *Structured Query Language* (SQL) Menggunakan Bahasa Indonesia, Skripsi S1. Teknik Informatika. Universitas Komputer Indonesia.

[7] Iqram Anwar, Penanganan Teks Bahasa Indonesia Menjadi *Data Definition Language* (DDL) Dengan Penanganan Kalimat Majemuk, Skripsi S1. Teknik Informatika. Universitas Komputer Indonesia.

[8] J. Enterprice, Pengenalan HTML dan CSS, Jakarta: PT Elex Media Komputindo, 2016.

[9] Ramus Lerdorf, Kevin Tatroe, *Programing PHP*, United States of America: O'Reilly Media, 2002.

[10] Maciej Ogrodniczuk, "*Coreference*". Walter de Gruyter GmbH & Co KG.

[11] I. Afrianto, A. Heryandi, A. Finandhita, Sufa'atin, *E-Document Autentification With Digital Signature Fro Smart City : Reference Model*. *Informatic Engineering Department, Faculty of Engineering and Computer Science*. Universitas Komputer Indonesia.