ANTARMUKA BAHASA ALAMI UNTUK MELAKUKAN QUERY TERHADAP TERJEMAHAN AL-QURAN

Suwanto Raharjo[1], Sri Hartati[2]
[1)] Teknik Informatika Fakultas Teknologi Industri Institut Teknologi AKPRIND Yogyakarta,
[2)]Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Gadjah Mada
email: [1)]wa2n@nrar.net, [2)]shartati@ugm.ac.id

ABSTRACT

Natural Language Processing (NLP) can be used to get data and information from relational database management system. NLP approach can retrieve Qur'an translation data saved in relational database. Python language is used to develop NLP interface that translate user question into database query. Development of production rules are important to translate natural language into database query. The research only limit 5 questions to develop the production rules that create 7 production rules. The proposed questions in the application are adjusted according the production rules. The correctness of given questions are checked using syntax analysis. SQL as a query result is produced if the question pass the parser. The final evaluation of the result is needed to meet the guidance is done by evaluator.
Keywords : NLP, Terjemahan Al-Quran, SQL, Python

INTISARI

Pengambilan data dan informasi dalam sebuah dari sebuah sistem basis data relasional dapat dilakukan dengan menggunakan *Natural Language Processing* (NLP). Terjemahan Al-Quran yang tersimpan dalam sistem basis data relasional dimungkinkan untuk diambil datanya dengan menggunakan pendekatan NLP. Antarmuka NLP yang menerjemahkan pertanyaan ke dalam *query* basis data dapat dibangun dengan bahasa pemorograman Python. Pembangunan aturan produksi yang diperlukan untuk menerjemahkan bahasa alami ke dalam *query* terhadap basis data merupakan hal yang penting. Pembatasan 5 pertanyaan untuk membangun aturan produksi menghasilkan 7 aturan produksi dalam penelitian ini. Pertanyaan yang diajukan dalam aplikasi ini masih terbatas dan disesuaikan dengan aturan produksi yang dibangun. Kebenaran pertanyaan yang diberikan akan diperiksa dengan melakukan analisis sintaks (*parser*). Pertanyaan yang mampu melewati tahapan *parser* akan diterjemahkan ke dalam bahasa *query* yakni bahasa SQL. Evaluasi hasil dari bahasa SQL oleh evaluator diperlukan untuk melakukan pengecekan apakah hasil sudah sesuai dengan kaidah yang dibenarkan.
Kata Kunci : NLP, Terjemahan Al-Quran, SQL, Python

PENDAHULUAN

Penggunaan bahasa alami, bahasa keseharian seperti bahasa Indonesia dalam era internet sekarang bukan lagi merupakan domain ilmu sosial saja namun juga ilmu eksak seperti komputasi dengan tujuan interoperabilitas (Wicaksana,dkk, 2005). Semakin banyak dan beragamnya asal pengguna komputer dari berbagai negara yang terhubung dalam internet dan semakin besarnya data yang ada menjadikan kemudahan untuk mendapatkan data sesuai yang diharapkan oleh pengguna adalah sebuah keniscayaan. Sebagai contoh misalkan seorang pengguna internet dari Indonesia akan mengharapkan dapat menggunakan bahasa ibunya untuk mendapatkan informasi. *Natural Language Processing* (NLP) sebagai salah satu bidang ilmu komputer yang mempelajari interaksi komputer dengan bahasa yang digunakan secara umum dalam kehidupan sehari-hari. Pengembangan teknik yang bertujuan bagaimana komputer memahami bahasa alami manusia dipelajari dalam NLP. Bahasa alami yang digunakan oleh manusia dari berbagai negara akan memiliki perbedaan dalam bentuk penulisan dan pengucapan. NLP dapat digunakan untuk melakukan pengambilan kembali informasi (*information retrieval*) (Lewis dan Jones, 1996), baik dengan menggunakan penulisan ataupun diucapkan. Penelitian ini melakukan implementasi NLP untuk mendapatkan informasi dari data yang tersimpan dalam sistem basis data relasional.

Natural Language Processing dan Basis Data

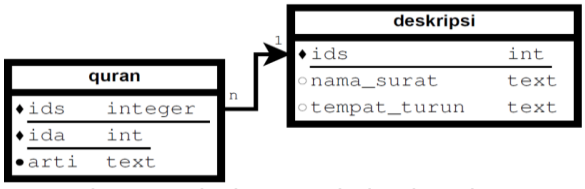
Pengambilan kembali informasi yang tersimpan dalam suatu basis data dapat dilakukan dengan menggunakan NLP. *Natural language interface to a database* (Nlidb) merupakan sebuah sistem yang

digunakan untuk mendapatkan informasi yang tersimpan dalam basis data dengan menggunakan masukan yang menggunakan bahasa alami (Androutsopoulos, dkk, 1995). Sistem untuk mendapatkan informasi dalam basis data telah lama diteliti yakni di tahun 1961 dengan sistem yang bernama Green's BASEBALL dan sistem LUNAR di tahun 1972 (Copestake dan Jones, 1989). NLP menawarkan banyak manfaat jika digunakan untuk mengambil informasi dalam sistem basis data namun diperlukan usaha untuk menterjemahkan bahasa query (Lewis dan Jones, 1996).

Penggunaan NLP untuk mendapatkan informasi dalam basis data kini semakin banyak diharapkan karena semakin banyaknya pengguna yang ingin mendapatkan informasi dengan berbagai macam perangkat dari laptop sampai dengan telepon genggam. Dengan demikian memiliki sistem antarmuka NLP untuk sistem basis data yang dapat diandalkan merupakan hal yang pokok. Antarmuka NLP ke sistem basis data harus mampu mengerti apa yang diharapkan oleh pengguna (Popescu, dkk, 2005).

Konsep Bahasa Alami untuk Terjemahan Al-Quran

Penelitian ini melakukan perancangan aplikasi pengolah bahasa alami dengan bahasa Indonesia untuk mendapatkan data dan informasi dari sistem basis data. Informasi yang didapatkan adalah data-data yang berkaitan dengan terjemahan Al-Quran dalam bahasa Indonesia. Data terjemahan Al-Quran disimpan dalam 2 tabel yang berelasi seperti tertampil dalam gambar 1 dengan sistem manajemen basis data relasional yang digunakan adalah PostgreSQL. Antarmuka dibangun menggunakan bahasa pemograman Python dalam sistem operasi Debian Linux.



Gambar 1. Relasi antar tabel terjemahan Al-Quran

Data yang ada dalam basis data diambil dengan menggunakan pertanyaan-pertanyaan berbahasa Indonesia seperti :

a. Deskripsikan Surat Yunus
b. Tampilkan ayat ke 2 dari Surat An-Nashr
c. Dimanakah Surat ke 4 turun ?
d. Sebutkan surat yang jumlah ayatnya sama dengan 7
e. Berapa jumlah ayat dari surat Ibrahim ?

Jawaban atas pertanyaan-pertanyaan tersebut dapat berupa hasil sebuah *query* atau beberapa query dari basis data pada gambar 1. Sebagai contoh pertanyaan "*Tampilkan ayat ke 2 dari surat An-Nashr*" akan memberikan jawaban dari hasil sebuah *query* "*SELECT arti FROM quran JOIN deskripsi USING (ids) WHERE nama surat = 'An-Nashr'* ", sedangkan pertanyaan dari "*Deskripsikan Surat Yunus*" akan menampilkan jawaban dari hasil beberapa *query* yakni no surat, nama surat, jumlah ayat dan turunnya surat. Jadi dalam penelitian ini secara garis umum adalah melakukan perancangan dan implementasi aplikasi yang menterjemahkan bahasa alami dengan bahasa Indonesia ke dalam bahasa SQL yang kemudian ditampilkan hasil dari *query*nya.

Aturan Produksi

Bahasa Indonesia telah memiliki grammar dan aturan produksi namun dalam penggunaannya untuk melakukan *query* basis data perlu diberikan aturan produksi secara khusus (Hartati dan Zuliarso, 2008). Pola pertanyaan yang diberikan dalam bahasa alami untuk mendapatkan hasil dari *query* sistem basis data perlu disusun ke dalam suatu aturan produksi. Aturan produksi dibangun dengan melakukan pemetaan keteraturan pola dari suatu pertanyaan. Pada penelitian ini dilakukan pembatasan pertanyaan yang diijinkan untuk diinputkan yakni terbagi dalam 5 pertanyaan utama yakni:

a. Deskripsikan
   Contoh:
   1) *Deskripsikan Surat Ibrahim*
   2) *Deskripsikan Surat Al-Baqarah*
   3) *Deskripsikan Surat Ke-2*

b. Tampilkan
   Contoh:
   1) *Tampilkan terjemahan ayat ke-2 dari surat Quraisy*
   2) *Tampilkan terjemahan ayat ke-2 dari surat Al-Baqarah*
   3) *Tampilkan terjemahan ayat ke-2 dari surat ke-2*

c. Berapa
   Contoh:
   1) *Berapa jumlah ayat surat Maryam*
   2) *Berapa jumlah ayat dari surat Al-Baqarah ?*

3) *Berapa jumlah ayat dari surat ke-2 ?*
4) *Berapa jumlah surat dari Juz ke-1 ?*
5) *Jumlah ayat dari surat ke-2 ada berapa ?*

d. Dimana
Contoh:
1) *Dimana Surat Al-Baqarah turun ?*
2) *Dimana Surat ke-2 turun ?*
3) *Surat ke 2 turun dimana ?*

e. Sebutkan
Contoh:
1) *Sebutkan Surat yang jumlah ayatnya lebih dari 7 ?*
2) *Sebutkan Surat yang jumlah ayatnya kurang dari 7 ?*
3) *Sebutkan Surat yang jumlah ayatnya sama dengan 7 ?*

Pertanyaan-pertanyaan di atas memiliki pola keteraturan yang dapat disusun dengan notasi dari aturan *Backus Naur Form* (Naur, dkk, 1963) sebagai berikut:

a. S → <Perintah/Tanya> <atribut> <obyek> <kata sambung> <keterangan obyek> <atribut> <obyek> <Operator> <keterangan obyek> <keterangan> <Perintah/Tanya>
b. <Perintah/Tanya> → <Deskripsikan> |<Tampilkan> | <Berapa> | <Dimana> |<Sebutkan> | <kosong>
c. <Atribut> → <Jumlah> | <Terjemahan> |<kosong>
d. <Obyek> → <Surat> | <Ayat> | <Juz> |<kosong>
e. <Kata Sambung> → <dari> | <yang> |<dengan> | <ke> | <al> | <an> | <ali> | <az>| <at> | <ath> | <ad> | <asy> | <yaa> | <ash>|<ar> <kosong>
f. <keterangan obyek> → <Nama surat> | <angka surat> | <angka ayat> | <angka juz> | <kosong>
g. <Operator> → <lebih dari> | <kurang dari> |<sama dengan> | <kosong>
h. <Angka Surat> → <1 s/d 114 >
i. <Angka Ayat> → <1 s/d 286 >
j. <Angka Juz> → <1 s/d 30 >
k. <Keterangan> → <turun> | <ada> | <kosong>

Dari aturan produksi di atas dapat diidentifikasi pola keteraturan pertanyaan yang dapat terbagi dalam tipe pertanyaan :

a. Tipe 1 <Perintah/Tanya> <Obyek> [kata sambung] <keterangan obyek>
Pola ini memiliki 2 tipe yakni
1) Pola tanpa kata sambung
contoh : *Deskripsikan Surat Maryam*
2) Pola dengan kata sambung
contoh : *Deskripsikan Surat Ke-2*

b. Tipe 2 <Perintah/Tanya> <Obyek> [kata sambung] <keterangan obyek> <keterangan>
Pola ini memiliki 2 tipe yakni
1) Pola tanpa kata sambung
contoh : *Dimana Surat Yusuf turun ?*
2) Pola dengan kata sambung
contoh : *Dimana Surat An-Nashr turun ?*

c. Tipe 3 <Obyek> [kata sambung] <Keterangan obyek> <Keterangan> <Perintah/Tanya>
Pola ini memiliki 2 tipe yakni
1) Pola tanpa kata sambung
contoh : *Surat Ibrahim turun dimana ?*
2) Pola dengan kata sambung
contoh : *Surat ke- 3 turun dimana ?*

d. Tipe 4 <Perintah/Tanya> <atribut> <Obyek> [kata sambung] <obyek> [kata sambung] <keterangan obyek>
Pola ini memiliki 4 tipe yakni
1) Pola dengan kata sambung
contoh : *Berapa Jumlah Ayat dari Surat ke 2 ?*
2) Pola dengan kata sambung di depan
contoh : *Berapa Jumlah Ayat dari Surat Yunus ?*
3) Pola dengan kata sambung di belakang
contoh : *Berapa Jumlah Ayat Surat At-Taubah ?*
4) Pola tanpa kata sambung
contoh : *Berapa Jumlah Ayat Surat Shaad ?*

e. Tipe 5 <Atribut> <Obyek> [kata sambung] <obyek> [kata sambung] <keterangan obyek> <keterangan> <Perintah/Tanya>
Pola ini memiliki 4 tipe yakni
1) Pola dengan kata sambung
contoh : *Jumlah ayat dari surat ke-2 ada berapa ?*
2) Pola dengan kata sambung di depan
contoh : *Jumlah ayat dari surat Huud ada berapa ?*
3) Pola dengan kata sambung di belakang
contoh : *Jumlah ayat surat ke-3 ada berapa ?*
4) Pola tanpa kata sambung
contoh : *Jumlah ayat surat Yusuf ada berapa ?*

f. Tipe 6 <Perintah/Tanya> <atribut> <Obyek> <kata sambung> <keterangan obyek> [kata sambung] <obyek> [kata sambung] <keterangan obyek>
Pola ini memiliki 4 tipe yakni
1) Pola dengan kata sambung

contoh : *Tampilkan terjemahan Ayat ke 3 dari Surat ke 2?*

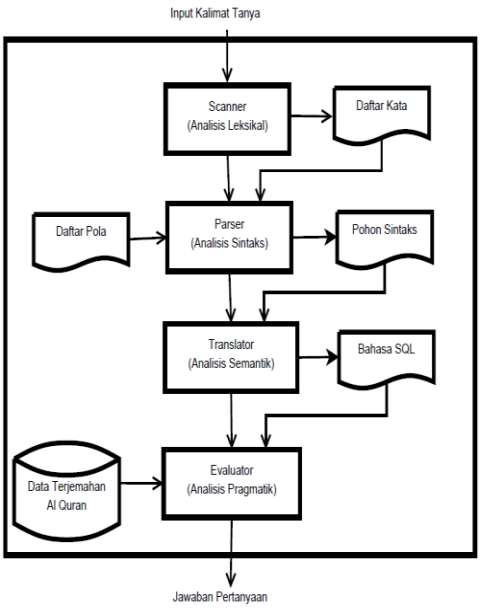
2) Pola dengan kata sambung di depan
contoh : *Tampilkan terjemahan Ayat ke 3 dari Surat Yunus*
3) Pola dengan kata sambung di belakang
contoh : *Tampilkan terjemahan Ayat ke 3 Surat Al-Nashr*
4) Pola tanpa kata sambung
contoh : *Tampilkan terjemahan Ayat ke 3 Surat Ibrahim*

g. Tipe 7 *<Perintah/Tanya> <Obyek> <kata sambung> <atribut> <obyek> <operator> <keterangan obyek>*
Contoh: *Sebutkan Surat yang jumlah ayatnya lebih dari 7*

Sistem yang dirancang hanya dapat menerima pertanyaan sesuai dengan aturan produksi yang ditetapkan di atas.

ALGORITMA PEMROSESAN

Sistem yang dirancang dalam penelitian ini hanya dapat menerima pertanyaan sesuai dengan aturan produksi yang ditetapkan di atas. Pertanyaan yang tidak sesuai dengan pola aturan di atas akan diberikan pesan kesalahan atau diabaikan. Komponen bahasa alami dalam penelitian ini untuk mendapatkan hasil terjemahan Al-Quran yang tersimpan dalam basis data tertampil dalam gambar 2. Menurut Utami dan Hartati, 2008, sebelum mendapatkan hasil maka setiap teks input akan melewati proses: analisis leksikal, analisis sintaks, analisis semantik, dan analisis pragmatik.



Gambar 2. Komponen bahasa alami

Analisis Leksikal (*Scanner*)

Pemberian batasan pertanyaan dilakukan oleh subsistem yang bernama *scanner* dan *parser*. Pemrosesan awal pertanyaan dilakukan oleh *scanner* yang akan membuat pertanyaan menjadi sebuah daftar token. Pada penelitian ini *scanner* melakukan 3 aktivitas yakni: menghilangkan tanda baca, memecah pertanyaan dan mengubah menjadi huruf kecil. Potongan program dalam *scanner* yang digunakan penelitian ini adalah :

a. Membersihkan kalimat tanya dari tanda baca, menggunakan script Python dilakukan dengan:
```
if char in " ,?.!/;:-":
kalimat=kalimat.replace(char,' ')
```

b. Memecah kalimat menjadi daftar kata dan mencacah kata, menggunakan script Python dilakukan dengan:
```
tokens=nltk.word_tokenize(kalimat)
cacah=len(tokens)
```

c. Mengubah ke dalam huruf kecil, menggunakan script Python dilakukan dengan:
```
tokens=[t.lower() for t in tokens]
```

Analisis Sintaks (*Parser*)

Daftar token yang terbentuk akan dianalisa oleh *parser* apakah sesuai dengan dengan pembentukan pola kalimat yang telah ditetapkan. Penentuan struktur kalimat bisa jadi merupakan pekerjaan yang sulit bergantung dari bahasa yang digunakan. Parsing morfologi merupakan kegiatan yang dilakukan untuk menentukan struktur kata (Utami dan Hartati, 2007). Pada penelitian ini secara umum *parser* akan melakukan:

a. Membaca daftar token
b. Membaca cacah token
c. Membaca isi token
d. Membandingkan dengan aturan produksi

Analisis sintaks melakukan pelacakan terhadap token-token yang dihasilkan oleh *scanner* kemudian dibandingkan dengan daftar token yang tersedia. Jika sesuai dengan daftar token yang ada maka dilihat apakah terdapat kecocokan dengan aturan produksi yang ada. Sebagai contoh adalah membaca isi token dalam terminal kata sambung apakah sesuai dengan daftar token yang diijinkan, menggunakan script Python berikut:

```
ksambungid=['ke', 'dari', 'yang', 'dengan']
if kata3 not in ksambungid :
```

Potongan program di atas membandingkan apakah token yang diberikan termasuk dalam daftar token kata sambung dalam bahasa Indonesia, jika tidak termasuk maka akan ditolak atau dicocokan dengan terminal yang lain.

Analisis Semantik (*Translator*)

Translator berfungsi untuk memetakan hasil *parser* yang sesuai dengan aturan produksi ke dalam bahasa hasil yakni bahasa *query*, sehingga yang dilakukan pada analisis semantik meliputi:

a. Membaca hasil scanner
b. Memetakan kedalam bahasa Query

Pada penelitian ini token yang ada dalam terminal atribut, **obyek** dan **keterangan obyek** memegang peranan penting dalam *translator*. Sebagai contoh pertanyaan "*Tampilkan terjemahan ayat ke 1 dari surat Yunus*", maka dalam translator dapat diambil kata *Terjemahan ayat 1 Surat Yunus*, kata yang lain dapat diabaikan, sehingga dari hasil tersebut didapatkan :

a. **Atribut** nya adalah *terjemahan*,
b. **Obyek** nya adalah *Ayat* dan *Surat*
c. **Keterangan obyek** nya adalah *Yunus* dan *1*.

Kalimat tersebut akan ditranslasikan menjadi "SELECT **arti** FROM quran JOIN Deskripsi USING (ids) WHERE nama surat LIKE **%yunus** and ayat**=1**".

Jadi dalam sistem ini translator hanya akan mengambil token-token yang berguna, sedangkan token yang tidak berguna akan diabaikan. Token yang termasuk dalam token yang diabaikan dalam penelitian ini adalah token dalam <kata sambung>, <kata tanya/perintah > dan <keterangan>.

Analisis Pragmatik (*Evaluator*)

Proses paling akhir dalam tahapan ini adalah memeriksa hasil translasi apakah sudah sesuai dengan kaidah yang dibenarkan. Pada penelitian ini akan memeriksa token dari terminal **obyek** dan **keterangan obyek** yang dihasilkan oleh *translator*. *Evaluator* akan memeriksa apakah **keterangan obyek** yang dimiliki sesuai dengan **obyek** yang digunakan, sebagai contoh misalkan **obyek** yang dimiliki adalah *Al-Fatihah* dan **keterangan obyek** adalah 8, maka secara grammar diijinkan namun secara *evaluator* tidak diperkenankan karena maksimum dari **keterangan obyek** adalah 7. Potongan program bagian dalam *evaluator* pada penelitian ini adalah sebagai berikut :

```
if bentuk =="4a" :
js=cur.fetchone()
if int(kata5) <= int(js[0]) :
cur.execute("SELECT arti FROM alquran
JOIN deskripsi using (ids) where nama_surat
iLIKE '%%'|| %(id)s||'%%' and ida=%(ida)s
",{'id': kata7, 'ida':kata5 })
ns=cur.fetchone()
print ns[0]
```

Pengujian Sistem

Program dibangun dengan menggunakan bahasa pemrograman Python 2.7.3 dan PostgreSQL 9.1. Program antarmuka berbasis teks yang dijalankan dari terminal Linux. Untuk menjalankan program cukup dengan memanggil program diikuti dengan memberikan pertanyaan serta opsi jika diinginkan. Gambar 3 menunjukkan opsi yang dapat diberikan ke program.

```
wa2n@debian:~/program$ python nlpdb.py --help
usage: nlpdb.py [-h] -t TANYA [-p PARSING]

NLP untuk Terjemahan Quran.

optional arguments:
 -h, --help  show this help message and exit
 -t TANYA, --tanya TANYA
  Masukan kalimat tanya dengan tanda kutip
  -p PARSING, --parsing PARSING
   +n Menampilkan hasil a token,
   +r Menampilkan hasil parser,
   +t Menampilkan hasil translator
```

Gambar 3. Opsi program

Seperti terlihat pada gambar 3 program memiliki 4 opsi yakni :

a. *-t* dengan *-pn* menampilkan daftar token pertanyaan
b. *-t* dengan *-pr* menampilkan nomor pola pertanyaan
c. *-t* dengan *-pt* menampilkan hasil translator
d. *-t* untuk menampilkan jawaban dari pertanyaan

Pertanyaan diberikan dengan harus menyertakan opsi *–t* di belakang nama program yang dipanggil diikuti dengan pertanyaan yang diapit dengan tanda kutip dan diberikan tambahan opsi jika diinginkan. Jika tanpa diberikan opsi tambahan maka program akan langsung memberikan hasil pertanyaan. Sebagai contoh untuk menampilkan daftar token dari pertanyaan yang diberikan dapat diberikan tambahan *-pn*

di belakang pertanyaan seperti terlihat dalam gambar 4.

```
wa2n@debian:~/program$ python nlpdb.py -t
"Jumlah Ayat Surat ke-7 ada berapa ?" -pn
Cacah token adalah => 7
['jumlah', 'ayat', 'surat', 'ke',
'7', 'ada', 'berapa']
kata 1 ===> jumlah
kata 2 ===> ayat
kata 3 ===> surat
kata 4 ===> ke
kata 5 ===> 7
kata 6 ===> ada
kata 7 ===> berapa
```

Gambar 4. Menampilkan daftar token

Program dengan opsi *-pn* seperti terlihat dalam gambar 4 menunjukkan bahwa progam telah melakukan proses di bagian *scanner*. Proses tersebut melakukan tokenisasi yakni merubah kalimat menjadi kata. Proses *scanner* melakukan 3 kegiatan yakni :

a. menghilangkan tanda baca dalam kalimat pertanyaan "*Jumlah Ayat Surat ke-7 ada berapa* ?".
b. memecah kalimat pertanyaan tersebut menjadi per kata
c. terakhir masing-masing token yang berupa kata dijadikan huruf kecil

Hasil akhir dari proses *scanner* dengan pertanyaan "*Jumlah Ayat Surat ke-7 ada berapa* ?" adalah berupa susunan token : [*'jumlah', 'ayat', 'surat', 'ke', '7', 'ada', 'berapa'*]. Pada tahapan *scanner* semua pertanyaan yang diberikan belum mengalami proses validasi apakah sesuai dengan pola aturan prosuksi yang diijinkan ataupun tidak. Opsi *-pn* pada program ini juga berlaku demikian, yakni hanya melakukan proses *scanner* yakni menghilangkan tanda baca dalam kalimat, memecah kalimat menjadi per kata dan mengubah ke dalam huruf kecil. Sebagai contoh seperti terlihat dalam gambar 5 bahwa dengan opsi ini semua pertanyaan akan diterima walaupun tidak sesuai dengan pola yang ditentukan.

```
wa2n@debian:~/program$ python nlpdb.py
-t "Jumlah Surat ayat ke-115 berapa ada?"
Cacah token adalah => 7
['jumlah', 'surat', 'ayat', 'ke', '115',
'berapa', 'ada']
kata 1 ===> jumlah
kata 2 ===> surat
kata 3 ===> ayat
kata 4 ===> ke
kata 5 ===> 115
kata 6 ===> berapa
kata 7 ===> ada
```

Gambar 5. Proses tokenisasi mengabaikan validitas pertanyaan

Terlihat dalam gambar 5 bahwa input pertanyaan tidak memenuhi pola aturan produksi, bahkan tidak sesuai dengan kaidah bahasa Indonesia ("*Jumlah Surat ayat ke-115 berapa ada?*"), namun karena masih dalam tahapan tokenisasi maka pertanyaan tersebut akan diproses dan dijadikan daftar token. Tahapan selanjutnya adalah *parser*, tahapan ini merupakan sebuah proses dimana pertanyaan yang diberikan akan dicocokan dengan pola yang telah ditentukan. Pada penelitian ini opsi *-pr* di belakang pertanyaan digunakan untuk mewakili tahapan *parser*. Tahapan *parser* melakukan perbandingan hasil tokenisasi dengan daftar token yang diijinkan. Jika daftar token yang dihasilkan sesuai dengan daftar token yang ada dan memiliki kesamaan pola aturan produksi yang ditetapkan maka akan ditampilkan pola yang sesuai dengan pertanyaan tersebut, seperti tertampil dalam gambar 6.

```
wa2n@debian:~/program$ python nlpdb.py -t
"Tampilkan terjemahan ayat ke 3 dari surat
ke-1?" -pr
Cacah token adalah => 9
['tampilkan', 'terjemahan', 'ayat', 'ke',
'3', 'dari','surat', 'ke', '1']
Memenuhi bentuk ke 6a
<Perintah Tanya><Atribut><Obyek>
<kata sambung><keterangan obyek>
<kata sambung><obyek><kata sambung>
<keterangan obyek>
```

Gambar 6. Menampilkan kesesuaian pola

Jika hasil token yang dihasilkan tidak terdapat dalam daftar token atau tidak sesuai dengan pola aturan produksi yang ada maka pertanyaan tersebut tidak akan diproses dan diberikan pesan kesalahan kepada pengguna. Sebagai contoh misalkan terdapat pertanyaan "*Tampilkan terjemahan ayat ke 3 dari surat Yusup*?", dalam pertanyaan tersebut terdapat kesalahan nama surat yakni *Yusup* yang tidak terdapat dalam daftar token, sehingga pertanyaan tersebut akan ditolak seperti tertampil dalam gambar 7.

```
wa2n@debian:~/program$ python nlpdb.py -t
"Tampilkan terjemahan ayat ke 3 dari surat
Yusup?" -pr
Cacah token adalah => 8
['tampilkan', 'terjemahan', 'ayat', 'ke',
'3', 'dari', 'surat', 'yusup']
Mohon cek kata nama surat--> yusup
```

Gambar 7. Menampilkan pesan kesalahan

Apabila daftar token yang dihasilkan sesuai dengan salah satu pola aturan produksi maka pertanyaan tersebut akan diterima walaupun terdapat kesalahan makna dalam pertanyaan tersebut. Gambar 8 menunjukkan bahwa tahapan *parser* hanya mencocokan pola hasil *scanner* atau melakukan analisa semantik dan tidak melakukan pelacakan terhadap makna kalimat (analisa pragmatik).

```
wa2n@debian:~/program$ python nlpdb.py -t
"Tampilkan terjemahan ayat ke-8 dari
surat ke-1?" -pr
Cacah token adalah => 9
['tampilkan', 'terjemahan', 'ayat', 'ke', '8',
'dari', 'surat', 'ke', '1']
Memenuhi bentuk ke 6a
<Perintah Tanya><Atribut><Obyek><kata sambung>
<keterangan obyek><kata sambung><obyek>
<kata sambung><keterangan obyek>
```

Gambar 8. Menampilkan hasil parser

Dari pertanyaan dalam gambar 8 yakni "*Tampilkan terjemahan ayat ke-8 dari surat ke-1?*" memiliki kesamaan dalam salah satu pola aturan produksi yang ada, namun terdapat kesalahan bahwa surat ke-1 hanya sampai dengan ayat ke-7 sehingga terdapat kesalahan makna dalam pertanyaan tersebut. Jika pertanyaan yang diberikan mampu melewati tahapan *parser* maka pola pertanyaan tersebut akan diterjemahkan ke dalam bahasa *query* yakni bahasa SQL. Pada penelitian ini dapat digunakan opsi *-pt* untuk menampilkan hasil proses *translator* yakni berupa bahasa *query* SQL sesuai dengan pola yang didapatkan. Sebagai contoh seperti tertampil dalam gambar 9.

```
wa2n@debian:~/program$ python nlpdb.py -t
"Deskripsikan Surat Ibrahim?" -pt
Cacah token adalah => 3
['deskripsikan', 'surat', 'ibrahim']
Memenuhi bentuk ke 1a
SELECT nama_surat, tempat,jumlah from deskripsi
where nama_surat iLIKE '%'|| 'ibrahim'||'%'
```

Gambar 9. Menampilkan hasil translator

Jika diberikan pertanyaan yang sesuai dengan pola aturan produksi yang diberikan namun memiliki kesalahan makna seperti misalkan pertanyaan "*Tampilkan terjemahan ayat ke-8 dari surat ke-1*?" maka *translator* akan memberikan hasil terjemahan bahasa SQLnya seperti tergambar dalam gambar 10.

```
wa2n@debian:~/program$ python nlpdb.py -t
"Tampilkan terjemahan ayat ke 8 dari
surat ke-1?" -pt
Cacah token adalah => 9
['tampilkan', 'terjemahan', 'ayat', 'ke', '8',
'dari', 'surat', 'ke', '1']
Memenuhi bentuk ke 6a
SELECT arti FROM alquran where
ids='1' and ida='8'
```

Gambar 10. Menampilkan hasil translator yang lain

Jika program dijalankan program tanpa opsi maka akan menampilkan jawaban atas pertanyaan yang diberikan jika memenuhi pola pertanyaan dan mampu melewati tahapan evaluator. Dimana tahapan evaluator akan melakukan evaluasi terhadap hasil proses translator, sebagai contoh hasil akhir dari sebuah pertanyaan yang menghasilkan jawaban adalah seperti tertampil pada gambar 11.

```
wa2n@debian:~/program$ python nlpdb.py -t
"Deskripsikan Surat Ibrahim?"
Cacah token adalah => 3
Memenuhi bentuk ke 1a
Hasil ===> Nama surat adalah Ibrahim ,
surat tersebut termasuk surat Makkiyah
yang memiliki 52 ayat.
```

Gambar 11. Menampilkan hasil akhir

KESIMPULAN DAN SARAN

Penelitian ini menghasilkan sebuah aplikasi yang mampu melakukan *query* terhadap data yang disimpan dalam basis data relasional dengan menggunakan bahasa Indonesia. Pertanyaan yang diajukan dalam aplikasi ini masih terbatas dan disesuaikan dengan aturan produksi yang dibangun. Pertanyaan yang tidak sesuai dengan aturan produksi yang ditetapkan akan diberikan pesan kesalahan atau diabaikan. Membebaskan pertanyaan menjadi lebih luas adalah saran yang bisa dilakukan untuk penelitian selajutnya sehingga pengguna dapat memiliki variasi pertanyaan yang lebih banyak.

DAFTAR PUSTAKA

Androutsopoulos, I.G., Ritchie, D., dan Thanisch, P. 1995, "Natural language interfaces to databases-an introduction," *arXiv preprint cmp-lg/9503016*, 1995.

Copestake, A. dan Jones, K.S. 1989, "Natural Language Interfaces to Databases," 1989.

Hartati, S. dan Zuliarso, E. 2008, “Aplikasi Pengolah Bahasa Alami untuk Query Basisdata XML,” *Dinamik-Jurnal Teknologi Informasi*, vol. 13, no. 2, 2008.

Lewis, D.D dan Jones, K.S. 1996, “Natural language processing for information retrieval,” *Communications of the ACM*, vol. 39, no. 1, pp. 92–101, 1996.

Naur, P., Backus, J.W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B. dkk. 1963, “Revised report on the algorithmic language Algol 60,” *Communications of the ACM*, vol. 6, no. 1, pp. 1–17, 1963.

Popescu, A.-M. Etzioni, O. dan Kautz, H. 2003, “Towards a theory of natural language interfaces to databases,” in *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 2003, pp. 149–157.

Utami, E. dan Hartati, S. 2008, “Pendekatan Metode Rule Based dalam Mengalihbahasakan Teks Bahasa Inggris Ke Teks Bahasa Indonesia,” *Jurnal Informatika*, vol. 8, no. 1, pp. pp–42, 2007.

Wicaksana, I.W.S, Wulandari, L, dan Wirawan, S. 2005, “Pentingnya Peranan Bahasa dalam Interoperabilitas Informasi berbasiskan Komputer karena Keragaman Semantik.” Prosiding Seminar Ilmiah Nasional (PESAT 2005), Universitas Gunadarma, Jakarta, halaman S9-S16, 2005