# KAMUS BAHASA ARAB – INDONESIA ONLINE DENGAN PEMECAHAN SUKU KATA MENGGUNAKAN METODE PARSING

**Anny Yuniarti, Aris Tjahyanto, Imam Kuswardayan**
Jurusan Teknik Informatika,
Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Kampus ITS, Jl. Raya ITS, Sukolilo – Surabaya 60111, Tel. + 62 31 5939214, Fax. + 62 31 5913804
Email: anny@its-sby.edu, aris@its-sby.edu, imam@its-sby.edu

**ABSTRAK**

*Kebutuhan umat Islam akan fasilitas penunjang belajar bahasa Arab di Indonesia masih belum terpenuhi dengan optimal. Kamus bahasa Arab yang beredar di pasaran sulit dipahami karena minimnya pengetahuan tentang ilmu tata bahasa Arab di kalangan umat Islam.*

*Pada penelitian ini dikembangkan sebuah perangkat lunak yang berfungsi menerjemahkan kata berbahasa Arab dengan metode parsing sehingga dapat mencakup kata-kata yang telah mengalami perubahan bentuk dari bentuk dasarnya. Karena kata bahasa Arab memiliki turunan kata yang jumlahnya cukup besar, dan supaya kamus efisien, maka tidak semua turunan kata disimpan dalam basisdata. Oleh sebab itu diperlukan suatu cara untuk mengenali pola kata, dan cara mengetahui bentuk dasar suatu kata. Keseluruhan perangkat lunak ini diimplementasikan berbasis web sehingga memudahkan pengaksesan pengguna. Dan pengguna tidak memerlukan proses instalasi perangkat lunak atau sistem operasi tertentu.*

*Pembuatan perangkat lunak ini didahului dengan perancangan proses dan perancangan interface. Kemudian rancangan tersebut diimplementasikan menjadi sebuah perangkat lunak yang siap untuk dipakai. Perangkat lunak yang sudah jadi tersebut telah diuji coba sesuai dengan spesifikasi kebutuhan dan kemampuan yang dimiliki yaitu melakukan manajemen pada basisdata rules dan basisdata kamus. Dengan demikian perangkat lunak ini dapat dipakai sebagai kamus bahasa Arab digital.*

***Kata kunci :*** *Parser*, Bahasa Arab, *Unicode*.

## 1. PENDAHULUAN

### 1.1. Latar Belakang Masalah

Pada bahasa Arab, sebuah kata dapat diturunkan menjadi banyak kata. Dan pada kamus bahasa Arab, seringkali turunan kata tersebut tidak dicantumkan atau diselipkan di bagian kata dasarnya, sehingga menyulitkan pengguna yang masih awam dalam bahasa Arab karena pengguna harus mengetahui bentuk dasarnya, baru dicari arti kata yang dimaksud di bagian (indeks) kata bentuk dasar.

Pembuatan perangkat lunak kamus bahasa Arab yang dirancang sedemikian rupa sehingga pengguna tidak perlu mengetahui bentuk dasar suatu kata terlebih dahulu akan sangat memudahkan pengguna. Implementasi berbasis web juga memudahkan pengguna karena pengguna dapat mengakses aplikasi secara online tanpa harus melakukan instalasi perangkat lunak atau sistem operasi tertentu yang mendukung seperti pada aplikasi yang bukan berbasis web.

### 1.2. Masalah

Penelitian ini bertujuan membuat perangkat lunak yang memiliki kemampuan menerjemahkan kata berbahasa Arab ke dalam kata berbahasa Indonesia meskipun kata tersebut telah mengalami perubahan bentuk sehingga kata yang diterima lebih banyak dan kamus lebih efisien.

Permasalahan yang dihadapi dalam pembuatan perangkat lunak ini adalah:

- bagaimana metode/ cara yang efisien untuk mengenali pola kata berbahasa Arab
- bagaimana cara pengenalan pola kata bahasa Arab bisa berkembang dan bisa dimodifikasi andaikata terdapat penyempurnaan
- bagaimana mengambil bentuk dasar dari kata masukan yang telah diketahui polanya, yang kemudian akan dicocokkan dengan data kamus
- bagaimana data kamus bisa berkembang serta bisa dimodifikasi sebagai langkah penyempurnaan
- bagaimana implementasi pengenalan pola kata, pencarian kata dasar, dan penampilan kamus bahasa Arab-Indonesia dalam web.

### 1.3. Batasan Masalah

Untuk lebih memperjelas dan mencapai tujuan utama pembuatan perangkat lunak ini, maka penelitian ini dibatasi oleh hal-hal berikut:

1. Bentuk-bentuk kata yang dimasukkan pola dasarnya tergantung dari aturan (rules) yang terdapat dalam basis data.
2. Kata yang dapat diterjemahkan adalah kata yang ada dalam basisdata aturan dan/ atau basisdata kamus.
3. Internet browser mendukung Javascript dan Arabic Language Support.

## 2. APLIKASI KAMUS BAHASA ARAB-INDONESIA ONLINE

### 2.1. Teori Bahasa dan Teknologi Pemrosesan Bahasa

Pada dasarnya bahasa merupakan suatu bentuk representasi dari suatu pesan yang ingin dikomunikasikan antar manusia. Bentuk utamanya adalah dalam bentuk suara/ucapan (*spoken language*), tetapi sering juga dinyatakan dalam bentuk tulisan.

Selain bahasa alami yang merupakan bahasa komunikasi antar manusia, juga dikenal bahasa buatan yang dibuat secara khusus untuk kebutuhan tertentu, seperti bahasa pemodelan atau bahasa pemrograman. Dari sinilah lahir bidang ilmu *Natural Language Processing (NLP)* yang melakukan pemrosesan bahasa secara simbolik dengan teknologi komputer.

Saat ini, teknologi yang berkaitan dengan pemrosesan bahasa alami sering disebut sebagai ”*speech and language technology*”, atau dalam beberapa pertemuan ilmiah disepakati penggunaan istilah ”teknologi bahasa” oleh beberapa peneliti di Indonesia.

Suatu sistem pemrosesan bahasa alami dapat diberntuk dari tiga sub-sistem, yaitu:

1. Sub-Sistem Natural Language Processing (NLP), berfungsi melakukan pemrosesan secara simbolik terhadap bahasa tulisan. Beberapa contoh aplikasi sub-sistem ini adalah translator bahasa alami (misalnya dari Bahasa Inggris ke Bahasa Indonesia), sistem pemeriksa sintaks bahasa, dan sebagainya.
2. Sub-Sistem Text to Speech (TTS), berfungsi untuk mengubah teks (bahasa tulisan) menjadi ucapan (bahasa lisan).
3. Sub-Sistem Speech Recognition (SR), berfungsi untuk mengubah atau mengenali suatu ucapan (bahasa lisan) menjadi teks (bahasa tulisan).

### 2.2. Konsep Bahasa Arab

Kata (*kalam*) dalam bahasa Arab terdiri dari tiga bagian, yaitu:

1. Kata Benda (*Isim*)
2. Kata Kerja (*Fi’il*)
3. Huruf

Berdasarkan jumlahnya kata benda terdiri dari:

1. Kata benda tunggal (isim mufrod), yaitu kata benda yang menunjukkan arti tunggal, contoh: مُسْلِمَة (seorang wanita muslim)
2. Kata yang menunjukkan arti dua (isim tatsniyah), yaitu kata benda yang menunjukkan dua benda, contoh: مُسْلِمَتَان (dua orang wanita muslim)
3. Kata benda jamak, yaitu kata benda yang menunjukkan jumlah tiga atau lebih. Contoh: مُسْلِمَاتٌ (wanita wanita muslim)

Kata kerja (*fi’il*) dibedakan berdasarkan banyak hurufnya dan bentuk (*bina’*)nya. Berdasarkan banyak hurufnya, *fi’il* terbagi dua bagian yaitu:

1. Fi’il Tsulatsy (huruf asalnya tiga)
2. Fi’il Ruba’i (huruf asalnya empat)

*Fi’il tsulatsy mujarrad* (tanpa tambahan) terbagi 6 macam sebagai berikut:

1. Fa’ala – yaf’ulu, contoh: نَصَرَ يَنْصُرُ
2. Fa’ala – yaf’ilu, contoh: ضَرَبَ يَضْرِبُ
3. Fa’ala – yaf’alu, contoh: سَأَلَ يَسْأَلُ.
4. Fa’ila – yaf’alu; contoh: عَلِمَ يَعْلَمُ
5. Fa’ila – yaf’ilu, contoh: حَسِبَ يَحْسِبُ
6. Fa’ula – yaf’ulu, contoh: حَسُنَ يَحْسُنُ dan sebagainya.

Macam-macam huruf ada banyak sekali, menurut Djuha ada 16 macam, beberapa diantaranya seperti huruf *jarr*, huruf *athaf*, huruf *nashab*, huruf *jazm* dan sebagainya.

bahasa Arab dikenal adanya perubahan pola kata (*tasrif*). Adapun menurut istilah, *tasrif* berarti mengubah dari bentuk asal (pokok) ke bentuk yang lain..

Perubahan bentuk tersebut berfungsi untuk mendapatkan arti yang berbeda, seperti:

1 نَصَرَ = *fi’il madli*, artinya sudah menolong.

2 يَنْصُرُ = *fi’il mudlari’*, artinya sedang/ akan menolong.

3 نَصْرًا = *masdar*, artinya pertolongan (kata benda).

4 نَاصِرٌ = *isim fa’il*, artinya yang menolong (subyek).

5 مَنْصُوْرٌ = *isim maf’ul*, artinya yang ditolong (obyek).

6 أُنْصُرْ = *fi’il amar*, artinya perintah untuk menolong.

7 لَاتَنْصُرْ = *fi’il nahi*, artinya jangan menolong (menunjukkan larangan).

8 مَنْصَرٌ = *isim makan*, artinya tempat menolong (keterangan tempat).

9 مَنْصَرٌ = *isim zaman*, artinya waktu menolong (keterangan waktu).

10 مِنْصَرٌ = *isim alat*, artinya alat untuk menolong.

### 2.3. Konsep Parsing

*Lexical Analyzer (scanner)* bertujuan untuk memisahkan teks yang dimasukkan menjadi bagian-bagian atau token-token. *Syntax analyzer (parser)* menghasilkan sebuah *output* berupa *syntax tree* (pohon sintaks) dimana daunnya adalah token-token. Sebuah grammar (sekumpulan aturan) dapat digunakan *syntax analyzer* untuk menentukan struktur dari *source program*. Proses pengenalan ini disebut *parsing*, oleh karenanya *syntax analyzer* sering disebut sebagai *parser*.

Sebuah grammar terdiri dari sekumpulan aturan *finite nonempty* atau produksi yang menspesifikasikan sintaks suatu bahasa. Grammar juga merepresentasikan struktur kalimat bahasa.

### 2.3.1. Top-down Parsing

*Top-down parser* membentuk sebuah pohon parsing dimulai dari *root* (yaitu simbol teratas dari *grammar*) dan berkembang menggunakan aturan-aturan *grammar*.

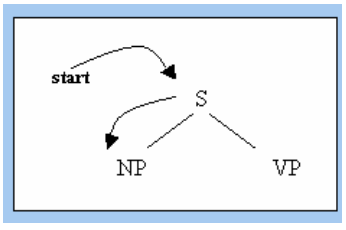
Contoh sebuah *grammar* sederhana:

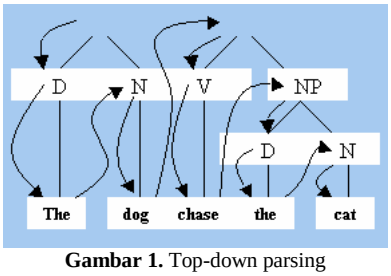
```
S  → NP VP
NP → D N
VP → V NP
VP → V
D  → the, all, every
N  → dog, dogs, cat, cats
V  →chase, chases, see, sees
```

Cara kerja *top-down parsing* dapat diilustrasikan dengan contoh parsing kalimat berikut: *the dog chase the cat*, maka parser akan :

- mencari **S.** Untuk mendapatkan **S**, dibutuhkan sebuah **NP** dan **VP.**
- Untuk mendapatkan **NP**, dibutuhkan sebuah **D** dan **N.**
- Untuk mendapatkan **D,** dapat digunakan kata *the.*
- Untuk mendapatkan **N,** dapat digunakan *dog*, sehingga **NP** didapatkan.
- Untuk mendapatkan **VP,** dibutuhkan **V** dan **NP.**
- Untuk mendapatkan **V,** dapat digunakan *chase*, sehingga **VP** didapatkan,
- Untuk mendapatkan **NP**, dibutuhkan sebuah **D** dan **N.**
- Untuk mendapatkan **D,** dapat digunakan kata *the.*
- Untuk mendapatkan **N,** dapat digunakan *cat*, sehingga **NP** didapatkan, **VP** didapatkan, dan **S** didapatkan pula.

Sebuah pohon digambar untuk mengilustrasikan langkah-langkah *top-down parsing* pada contoh diatas:



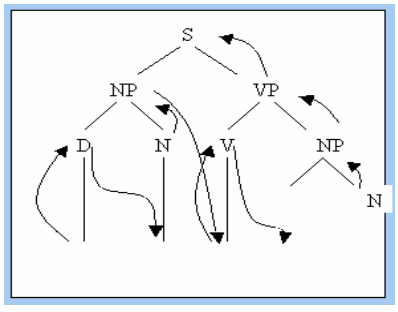

**Gambar 1.** Top-down parsing

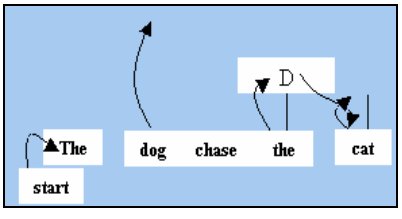
### 2.3.2. Bottom-up Parsing

*Top-down parser* memiliki keterbatasan ketika terdapat *rule* yang bersifat *Left-Recursive* berbentuk ***A → A B*** ("untuk memparsing A, dibutuhkan memparsing A dan ...). Salah satu cara menangani rule yang *left-recursive* adalah dengan *bottom-up parsing*. *Bottom-up parser* menerima kata-kata kemudian mengkom-binasikannya membentuk kesatuan. Pada contoh kalimat "*the dog barked*", *bottom-up parser* akan:

- Menerima sebuah kata: *the.*
- *The* adalah sebuah **D**.
- Kata selanjutnya adalah: *dog.*
- *Dog* adalah sebuah **N**.
- **D** dan **N** membentuk **NP**.
- Kata selanjutnya adalah: *chase.*
- *Chase* adalah sebuah **V**.
- Kata selanjutnya adalah: *the.*
- *The* adalah sebuah **D**.
- Kata selanjutnya adalah: *cat.*
- *Cat* adalah sebuah **N**.
- **D** dan **N** membentuk **NP**.
- **V** dan **NP** membentuk **VP**.
- **NP** dan **VP** membentuk **S**.

Diagram pohon dari langkah-langkah diatas kemudian dapat dibentuk sebagai berikut:

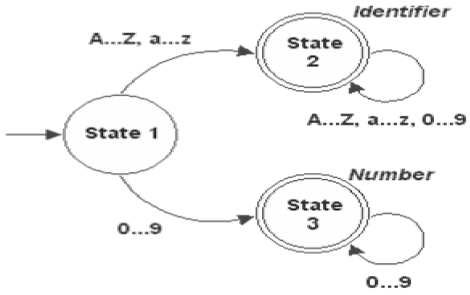



**Gambar 2.** Bottom-up parsing

Parser *bottom-up* modern menggunakan *Deterministic Finite Automaton* (DFA) untuk mengimplementasikan *tokenizer* dan *state machine* untuk memparsing token yang terbentuk. Algoritma ini mudah diimplementasikan karena bergantung pada tabel-tabel untuk menentukan aksi yang diambil dan transisi *state*/keadaan, sehingga komputasi tabel-tabel inilah yang menghabiskan waktu dan cukup kompleks.

DFA bersifat deterministic yang artinya dari titik manapun hanya ada satu jalur menuju titik yang lain, dengan kata lain tidak ada ambiguitas dalam jalurnya. DFA juga bersifat finite/ terbatas, yang artinya terdapat sejumlah titik (yang disebut state) dan percabangan yang tetap jumlahnya. DFA membaca input dan memutuskan apakah sekumpulan karakter dapat dikenali sebagai sebuah token.




**Gambar 3.** Contoh DFA

Gambar 3 adalah sebuah contoh DFA dengan tiga state. Misal dimasukkan kata “parser”. Dari State 1 (state inisial), DFA berpindah ke State 2 ketika membaca “p”. Untuk karakter selanjutnya, “a”, “r”, “s”, “e”, dan “r”, DFA berjalan terus (loop) ke State 2.

Secara desain, *tokenizer* akan berusaha mencocokkan sepanjang mungkin kumpulan karakter untuk diputuskan sebagai sebuah token. Setiap kali sebuah token berhasil diidentifikasi, akan segera dialihkan ke *parser engine* dan *tokenizer* akan kembali ke state awal.

Salah satu metode *bottom-up parsing* yakni *LR parsing*. *LR parsing* membaca masukan dari kiri ke kanan dan bertujuan untuk menemukan *rightmost derivation*. (L mengindikasikan *left*, R mengindikasikan *rightmost derivation*). Pencarian dilakukan dari kiri ke kanan sampai ditemukan sebuah *handle*, yakni sebuah frase untuk me-*reduce* selanjutnya.

Sebuah parser LR mengkonstruksi kebalikan dari *rightmost derivation* sebuah string input. Sebuah grammar G memiliki simbol start S. Untuk sebuah string input x, maka *rightmost derivation* dari input ini adalah:

$$S \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_{m-1} \rightarrow \alpha_m = x$$

dimana nonterminal paling kanan pada masing-masing $\alpha_i$, untuk setiap $1 \le i < m$, adalah yang dipilih untuk ditulis. Representasi dari langkah derivasi ini adalah dalam bentuk:

$$\Phi B t \rightarrow \Phi \beta t$$

dimana $\alpha_i = \Phi B t$ dan $B \rightarrow \beta$ adalah produksi yang telah dibaca. Oleh karena *rightmost derivation*, t haruslah string dari simbol-simbol terminal.

Sebuah grammar dikatakan LR(k) jika untuk suatu input string yang diberikan, pada setiap tahap derivasi *handle* β dapat dideteksi dengan memeriksa string Φβ dan membaca simbol-simbol k pertama dari string input t yang tidak dapat digunakan.

Beberapa definisi pada konsep LR parser adalah:

- *Item.* Yaitu produksi dengan sebuah titik (·) di produksi sisi kanan. Sebuah item dikatakan *complete* jika (·) adalah simbol yang paling kanan.
- *viable prefixes*, yaitu string dari simbol-simbol inisial pada bentuk sederhana paling kanan yang tidak meliputi simbol pada sisi lain handle. *Viable prefix* dari bentuk Φβt, dimana β menandakan *handle*, adalah suatu prefiks atau string kepala dari Φβ. Oleh karenanya jika $\Phi\beta = u_1u_2 \ldots u_r$, string $u_1u_2 \ldots u_i$, dimana $1 \le i \le r$, adalah *viable prefix* dari bentuk Φβt. *Viable prefix* tidak dapat berisi simbol-simbol yang ada di sebelah kanan *handle* (misalnya simbol-simbol dalam t). Selama dalam *viable prefix*, pencarian akan dilanjutkan dengan simbol selanjutnya. Ketika telah sampai pada akhir *viable prefix* terpanjang, maka posisi tersebut adalah posisi kanan paling akhir dari *handle*. Pada titik ini akan dilakukan reduksi.

Parser LR, seperti kebanyakan tipe parser yang lain, adalah sebuah mesin *pushdown* yang mempunyai input sebuah string, *stack*, dan mekanisme *finite-control*. Mekanisme ini adalah mesin *finite-state* dengan banyak *state*. *State read* menyebabkan terjadinya transisi dari satu *state* ke *state* yang lain ketika membaca simbol terminal atau nonterminal.

Aksi yang dapat dilakukan dalam parser LR antara lain:

1. *shift*, mentransfer input selanjutnya ke dalam *stack*
2. *reduce*, untuk $X \rightarrow w$, gantikan w dengan X pada *stack*
3. *accept*, string yang dimasukkan termasuk dalam grammar bahasa
4. *error*, string yang dimasukkan tidak termasuk dalam grammar bahasa

Sebuah grammar G adalah grammar LR(0) jika:

a. Simbol *start* dari grammar ini tidak terdapat di semua produksi sebelah kanan
b. Untuk setiap *viable prefix* γ dari G, jika $A \rightarrow \alpha \cdot$ adalah item yang *complete* dan valid untuk γ, maka tidak terdapat item *complete* yang lain dengan sebuah terminal di sebelah kanan titik (·) yang valid untuk γ.

Sebuah parser LR(*k*) membaca string input yang diberikan dari kiri ke kanan dan mengkonstruksi kebalikan dari *rightmost derivation* string tersebut. Sebuah parser LR(*k*) membuat semua keputusan

parsing berdasarkan isi dari *stack parse* dan simbol *k* berikutnya pada string input.

Item-item dari LR(k) terdiri dari sebuah item LR(0) diikuti dengan *lookahead set* yang didalamnya terdapat terminal dan/ atau simbol spesial $. Item LR(1) mempunyai bentuk sebagai berikut:

$$A \rightarrow \alpha \cdot \beta , \{a_1, \ldots, a_n\}$$

Ada tiga algoritma yang umum dipakai untuk membuat parsing tabel sebuah parser LR, yaitu:

1. SLR(1) atau LR(0) dengan tambahan menggunakan FOLLOW set untuk memilih satu diantara beberapa *action.* Beberapa kelebihan algoritma LR(0) antara lain: memiliki kelas grammar terkecil, tabel (jumlah state) terkecil, dan konstruksi tabel yang sederhana dan cepat.
2. LR(1), mampu menangani grammar LR(1), memiliki tabel (jumlah state) terbesar, konstruksi tabel lambat dan besar.
3. LALR(1) atau LR(0) dengan analisa *lookahead* untuk memilih satu diantara beberapa *action*. Menangani grammar dengan ukuran menengah, memiliki jumlah state yang sama dengan SLR(1), konstruksi *canonical* lambat dan besar (tetapi teknik konstruksi lebih baik).

## 2.4. Aplikasi Berbasis Web

Aplikasi kamus bahasa Arab-Indonesia ini dirancang untuk diimplementasikan berbasis web. Aplikasi database berbasis web memiliki bagian dasar atau lapisan sebagai berikut:

- *Client*. Yang termasuk didalamnya antara lain: *web browser* dari *user*, *java applet*, aplikasi *java*, atau platform dari program *client* yang berdiri sendiri.
- Aplikasi Lojik, meliputi: algoritma pengkodean pada *script* CGI, modul khusus dari *web server*, atau aplikasi *server* yang berdiri sendiri.
- Konektifitas Basis data, meliputi: API dari basis data, protokol konektifitas umum seperti ODBC atau JDBC.
- *Server* untuk basis data, seperti: RDBMS, ODBMS dan lain-lain.

Implementasi dari beberapa aplikasi dapat menggunakan model *multi-tier*, karena satu atau beberapa lapisan dapat dipakai bersama-sama. Namun implementasi secara umum yang biasa dipakai adalah sistem *three-tier* (tiga lapisan), yang terdiri atas tiga komponen utama, yaitu :

- Lapisan pertama, yang merupakan aplikasi dari *client*, contohnya : *browser* dari *user*
- Lapisan kedua, merupakan aplikasi *Web Server*, *Script* CGI dan API koneksi untuk basis data, seperti Apache *Web Server* dengan modul PHPnya, yang mendukung basis data MySQL, dan *script* PHP
- Lapisan ketiga adalah *server* untuk basis data.

### 2.4.1. Karakter Unicode

Pada pertengahan pertama tahun 1980-an *Xerox Corporation* berhasil mengimplementasi-kan hasil penelitiannya mengenai sistem pengkodean karakter *multilingual* dan berhasil mengeluarkan sebuah proyek yang disebut *Unification Code* atau *Unicode*, yang memiliki tujuan utama menyatukan semua karakter yang ada di dunia menjadi sebuah kumpulan karakter.

*Unicode* adalah standard internasional untuk mengkodekan semua bahasa di dunia pada komputer. *Unicode* dapat mencegah ambiguitas yang muncul ketika menampilkan skrip-skrip kompleks seperti bahasa Jepang, Arab atau Cina pada sistem komputer. *Unicode* dapat berjalan lebih cepat di Windows NT, 2000 dan XP.

Aplikasi yang menggunakan *Unicode* mampu menyimpan dan memproses semua karakter di dunia. Sehingga lebih mudah untuk meng-internasional-kan suatu aplikasi.

Sebagian besar bahasa pemrograman saat ini telah memiliki tipe data spesial untuk karakter Unicode/ ISO 10646-1, misalnya Ada95, Java, TCL, Perl, Python, C# dan lain sebagainya.

### 2.4.2. Cara Penyimpanan Data Kamus

Seperti diketahui, sebuah kamus yang baik adalah kamus yang efektif dalam penggunaan serta penyimpanan kata-katanya. Dalam sebuah kamus tidak semua kata disimpan sebagaimana bentuk kata tersebut. Kamus yang efektif adalah kamus yang kata turunannya tidak disimpan secara eksplisit, misalnya kata “kebaikan”, kata tersebut tidak akan disimpan dalam bentuk “kebaikan”, tetapi yang disimpan adalah kata “baik” dimana dalam penjelasannya barulah dijelaskan tentang imbuhan atau kata-kata yang bisa diturunkan dari kata “baik” tersebut.

Penyimpanan kata dan pola kata dalam kamus dilakukan dengan cara membagi kata atau pola dalam beberapa segmen, misalnya dalam wazan fa’ala, kata atau pola dibagi menjadi segmen-segmen fa’ fi’il, fatha, ‘ain fi’il, fatha, lam fi’il, dan fatha.

Contoh:

pola : فَعَلَ

→ fa’ fi’il : ف

→ ‘ain fi’il : ع

→ lam fi’il : ل

kata نَصَرَ termasuk dalam pola فَعَلَ dengan

→ fa’ fi’il : ن

→ ‘ain fi’il : ص

→ lam fi’il : ر

kata دَرَسَ termasuk dalam pola فَعَلَ dengan

→ fa’ fi’il : د

→ ‘ain fi’il : ر

→ lam fi’il : س

dan seterusnya.

Adapun teknik penyimpanan rule dilakukan dengan cara masing-masing huruf hijaiyyah, harakat, fa' fi'il, 'ain fi'il, lam fi'il, dan sebagainya dikodekan menjadi suatu simbol. Sehingga gabungan dari pola akan membentuk suatu *right production* pada suatu grammar.

Pada kasus penelitian yakni aplikasi kamus bahasa Arab, misal dengan aturan pola untuk wazan fa'ala-yaf'ulu, didapatkan kumpulan *right production* grammar sebagai berikut:

**Tabel 1.** *Grammar* untuk *wazn fa'ala yaf'ulu*

| No | Pola Wazn | Ketr. | Grammar |
|---|---|---|---|
| 1 | فَعَلَ | fiil madhi | K → akbkck |
| 2 | يَفْعُلُ | fiil mudhorik | K → dkambncn |
| 3 | فَعْلًا | mashdar | K → akbmceo |
| 4 | مَفْعَلًا | mashdar mim | K → fkambkroe |
| 5 | فَاعِلٌ | fa'il (pelaku) | K → akebpcq |
| 6 | مَفْعُولٌ | maf'ul | K → fkambngmcq |
| 7 | اُفْعُلْ | fi'il amr | K → enambncm |
| 8 | لَاتَفْعُلْ | fi'il nahi | K → rekhkambncm |
| 9 | مَفْعَلٌ | isim makan / zaman | K → fkambkcq |
| 10 | مِفْعَلٌ | isim alat | K → fpambkcq |

Dengan grammar seperti di atas, disusun sebuah *parsing table* dengan algoritma yang sederhana, yakni LR(0).

Dengan *parsing table* LR(0) tersebut, misalkan dimasukkan sebuah kata فَاعِلٌ yang memiliki pola "akebpcq". Proses *shift-reduce parsing* dan isi stack yang terjadi adalah:

| | Stack | Input | Action |
|---|---|---|---|
| (1) | # | akebpcq# | shift 2 |
| (2) | # a | kebpcq# | shift 7 |
| (3) | # ak | ebpcq# | shift 15 |
| (4) | # ake | bpcq# | shift 24 |
| (5) | # akeb | pcq# | shift 33 |
| (6) | # akebp | cq# | shift 43 |
| (7) | # akebpc | q# | shift 53 |
| (8) | # akebpcq | # | reduce by K → akebpcq |
| (9) | # K | # | accept |

Dengan demikian, kata فَاعِلٌ termasuk dalam pola kata bahasa Arab.

## 3. PERANCANGAN PERANGKAT LUNAK

Perangkat lunak yang dibuat ini bertujuan untuk membangun dan mengimplementasikan kamus bahasa Arab online. Pada tahap pembangunan, dilakukan segala sesuatu yang diperlukan jalannya aplikasi seperti penentuan aturan pola kata jadian, bentuk dasar kata jadian yang diterima, dan pemasukan data kamus berupa kata dasar bahasa Arab beserta arti katanya.

Dengan demikian, sistem aplikasi kamus bahasa Arab online ini memiliki dua sub-sistem, yaitu:

- sub-sistem pemasukan data
- sub-sistem penerjemahan kata

**Gambar 4.** Diagram sistem aplikasi kamus bahasa Arab online

Seperti terlihat pada Gambar 3, terdapat dua proses pada sub-sistem yang pertama, yakni pemasukan pola kata dan kata dasar serta pemasukan kata dasar dan artinya. Data pola dan bentuk dasar yang dimasukkan disimpan dalam basisdata *rule*. Sedangkan data kata dasar dan artinya disimpan di basis data kamus.

Penjelasan dari sub-sistem penerjemahan kata adalah sebagai berikut:

- Kata berbahasa Arab yang dimasukkan user diproses dalam proses parsing kata. Sebelum diproses karakter Arab yang dimasukkan dikonversi terlebih dahulu menjadi simbol-simbol.
- Dari proses parsing diperoleh output berupa status *accepted* atau *not accepted*. Apabila *not accepted* kata yang dimasukkan diterjemahkan secara langsung dengan mencocokkan kata yang dimasukkan dengan basis data kamus.
- Apabila hasil proses parsing adalah *accepted* maka kata yang dimasukkan dikategorikan dalam pola tertentu dan ditentukan kata dasarnya. Selanjutnya berdasarkan pola dan kata dasarnya tersebut maka dicari artinya di basis data rule dan basis data kamus.
- Setelah itu kata yang dimasukkan juga dicocokkan secara langsung ke basis data kamus untuk mendapatkan arti kata yang lain.

## 4. IMPLEMENTASI PERANGKAT LUNAK

Implementasi aplikasi ini meliputi tiga hal, yaitu implementasi data, implementasi proses dan implementasi perangkat lunak. Setelah rancangan data konseptual selesai dibuat, maka dilanjutkan dengan pengimplementasian rancangan data tersebut ke dalam bentuk tabel beserta tipe datanya.

Berikut ini implementasi beberapa proses yang ada dalam sistem:

a. Proses konversi input *Arabic* ke simbol

Proses ini bertujuan untuk mengkonversi masukan yang berupa kata bahasa Arab ke simbol tertentu sehingga dapat diolah dalam proses selanjutnya. Yang dilakukan adalah membaca setiap karakter masukan untuk kemudian membentuk suatu string simbol yang sesuai dengan masukan tersebut dengan mencocokkannya pada table konversi yang telah dibuat sebelumnya.

b. Proses parsing

Proses ini merupakan proses pengenalan pola dari masukan pengguna. Masukan berupa hasil konversi simbol pada proses konversi yang telah dibahas sebelumnya dan *parsing table* yang tersimpan dalam file JavaScript yang di-*include*-kan. Sedangkan keluaran berupa nomor-nomor produksi yang telah digunakan untuk reduksi dalam proses parsing. Jika parsing menghasilkan error, maka keluaran menghasilkan status *not accepted*.

c. Proses konstruksi *state machine*

Proses ini dilakukan untuk membuat suatu *state machine* berdasarkan *production rule* yang ada. Proses diawali dengan mencari input yang unik dari *right production* grammar. Kemudian dilanjutkan dengan membuat item-item LR parsing beserta closure-nya. Langkah selanjutnya adalah membuat kumpulan *goto item* sehingga dapat diketahui item tertentu dengan input tertentu akan mengalami transisi ke item yang mana.

d. Proses konstruksi *parsing table*

Dalam implementasinya *parsing table* yang dikonstruksi berupa matriks 2x2, jadi menggunakan Array dua dimensi dalam JavaScript. Yang dilakukan pertama kali adalah inisialisasi *parsing table* dimana semua diset error. Kemudian mulai dilakukan pembacaan dari setiap state dengan input tertentu akan menuju ke mana. Jika transisi yang dialami adalah menuju state lain maka *parsing table* diisi dengan *shift* atau *goto*. Jika tanda (·) telah berada di ujung paling kanan maka dilakukan *reduce* dengan produksi yang sesuai. Apabila karakter terakhir adalah "#" maka *parsing table* diisi *accept*.

e. Proses penambahan aturan pola kata

Pada proses ini selain dilakukan *update* pada data *production rule* dalam basis data, juga dilakukan konstruksi file prod.js, yaitu sebuah file *javascript* pada server yang digunakan dalam proses parsing. Setiap kali dilakukan proses penambahan data *production rule* pada basis data, bersamaan dengan itu pula dilakukan update isi file prod.js yang digunakan pada proses konstruksi parsing table. Untuk dapat menulis/ mengubah isi file yang terdapat pada server, digunakan obyek *FileSystemObject* dari ASP.

f. Proses penerjemahan kata

Pada proses penerjemahan kata yang pertama kali dilakukan adalah membaca hasil proses parsing, kata masukan, pola kata masukan, dan kata dasarnya. Proses penerjemahan disini meliputi dua bagian, yakni untuk kata-kata yang mengalami perubahan bentuk, dan penerjemahan secara langsung apabila hasil proses parsing ditolak untuk kata tersebut (kata tidak mengalami perubahan bentuk).

Proses penerjemahan kata yang mengalami perubahan bentuk, dilakukan apabila nilai variabel *statusnya* adalah "Accepted", dan meliputi langkah-langkah sebagai berikut:

1. Baca data tentang pola kata dari tabel *tbrule*. Data itu berupa kode kategori *rule* dan kode jenis *rule*. Kemudian baca nama pola dan arti pola kata dari tabel *jenispola* berdasarkan kode jenis *rule*.
2. Susun bentuk dasar dari masukan kata dengan menggabungkan isi variabel *katadasar* dengan *harokat* yang bersesuaian dengan pola.
3. Langkah selanjutnya adalah menentukan arti kata dasar dari masukan kata berdasarkan kata dasar yang telah dibentuk pada langkah sebelumnya. Kemudian ditentukan arti kata dari masukan kata yang merupakan gabungan dari arti pola dengan arti kata dasarnya.

Untuk mendapatkan arti yang lain berdasarkan data kamus, dicari pula arti masukan kata secara langsung pada basis data kamus.

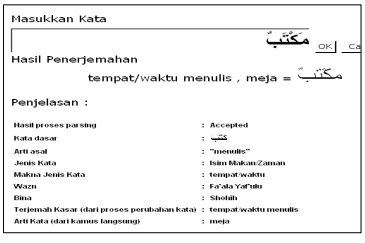
## 5. UJICOBA PERANGKAT LUNAK

Pada uji coba ini diberikan dua buah skenario yang ditujukan untuk mengetahui fungsionalitas dari aplikasi yang dibuat, yaitu :

1. Memasukkan kata yang terdapat dalam basis data kamus, dan termasuk dalam pola tertentu
2. Memasukkan kata yang terdapat dalam basis data kamus, tetapi tidak termasuk dalam pola tertentu.

Uji Coba Skenario Pertama

Pada ujicoba kali ini akan dicoba untuk memasukkan kata مَكْتَبٌ.



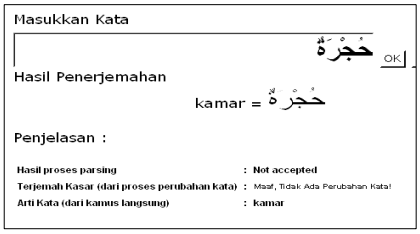

**Gambar 5.** Hasil uji coba 1

Hasil ujicoba kali ini adalah sistem dapat mengenali pola kata yang dimasukkan, yaitu pola kata مَفْعَلٌ, dengan kata dasar كَتَبَ yang berarti menulis. Karena pola مَفْعَلٌ merupakan jenis pola "isim makan/ zaman" yang berarti "tempat/ masa", maka arti kata مَكْتَبٌ adalah "tempat/ masa menulis".

Seperti telah dijelaskan pada bagian sebelumnya, bahwa selain arti dari hasil proses parsing, aplikasi juga mencari arti kata dengan mencocokkan secara langsung masukan kata dengan basis data kamus. Dengan penerjemahan langsung didapatkan arti kata مَكْتَبٌ adalah “meja”. Sehingga hasil akhir arti kata مَكْتَبٌ adalah “tempat/waktu menulis”, atau “meja”.

Uji Coba Skenario Kedua

Pada ujicoba kali ini akan dicoba untuk memasukkan kata حُجْرَةٌ.




**Gambar 6.** Hasil uji coba 2

Hasil ujicoba kali ini adalah sistem tidak dapat mengenali pola kata yang dimasukkan (*Not accepted*), sehingga sistem langsung mencocokkan masukan kata dengan basis data kamus dan didapatkan arti kata “kamar”.

Analisa Hasil

Secara umum aplikasi kamus bahasa Arab-Indonesia online ini telah berhasil diimplementasikan dengan baik. Sesuai dengan tujuan pembuatan aplikasi bahwa penerjemahan kata bahasa Arab dapat dilakukan tanpa harus meng-install program aplikasi tertentu atau harus memakai sistem operasi tertentu dan dapat menerjemahkan kata yang telah mengalami perubahan bentuk. Selain itu juga untuk penyempurnaan aplikasi maka kosakata dapat ditambah secara fleksibel, begitupun pola kata yang diterima.

## 6. KESIMPULAN DAN SARAN

### 6.1. Kesimpulan

a. Penggunaan konsep parsing berdasarkan data rule dan data kamus berisi bentuk dasar mampu menghasilkan perangkat lunak yang dapat menerjemahkan kata bahasa Arab meskipun telah mengalami perubahan bentuk kata dari bentuk dasarnya.
b. Pembuatan aplikasi kamus bahasa Arab online yang hemat dapat dilakukan dengan menyimpan bentuk dasarnya saja.
c. Penerjemahan yang dilakukan oleh perangkat lunak mampu menampilkan hasil maksimal sehingga pengguna tidak harus mengetahui bentuk dasar suatu kata.
d. Penggunaan karakter Unicode mampu menghasilkan perangkat lunak yang tidak tergantung pada Sistem Operasi dengan karakter tertentu seperti Windows Arabic.

### 6.2. Saran

a. Menambahkan basisdata *rule* sehingga bentuk kata yang dapat diterima aplikasi kamus dapat lebih banyak lagi.
b. Menambahkan basisdata kamus sehingga kosakata yang tersedia dalam aplikasi lebih lengkap.
c. Menyimpan kata masukan pengguna yang tidak berhasil diketahui terjemahannya sehingga isi basis data kamus dapat segera ditambah/diperbarui dengan kata tersebut.

## 7. DAFTAR PUSTAKA

[2] **Anwar, Moch., K.H.** *Ilmu Nahwu*, Sinar Baru Algesindo, Bandung, 2002.
[3] **Anwar, Moch., K.H.,** *Ilmu Sharaf*, Sinar Baru Algesindo, Bandung, 2002.
[4] **Djuha, Djawahir, Drs.,** *Tatabahasa Arab (Ilmu Nahwu)*, Sinar Baru Algesindo, Bandung, 2002.
[5] **Gerhards, Rainer,** *Why Unicode?* at http://www.eventreporter.com 2001.
[6] **Searle, Steven J.,** *A Brief History of Character Codes in North America, Europe, and East Asia*, Sakamura Laboratory, University Museum, University of Tokyo, 1999.