# BAB II

# LANDASAN TEORI

## 2.1 Automatic Speech Recognition

### 2.1.1 Definisi Automatic Speech Recognition

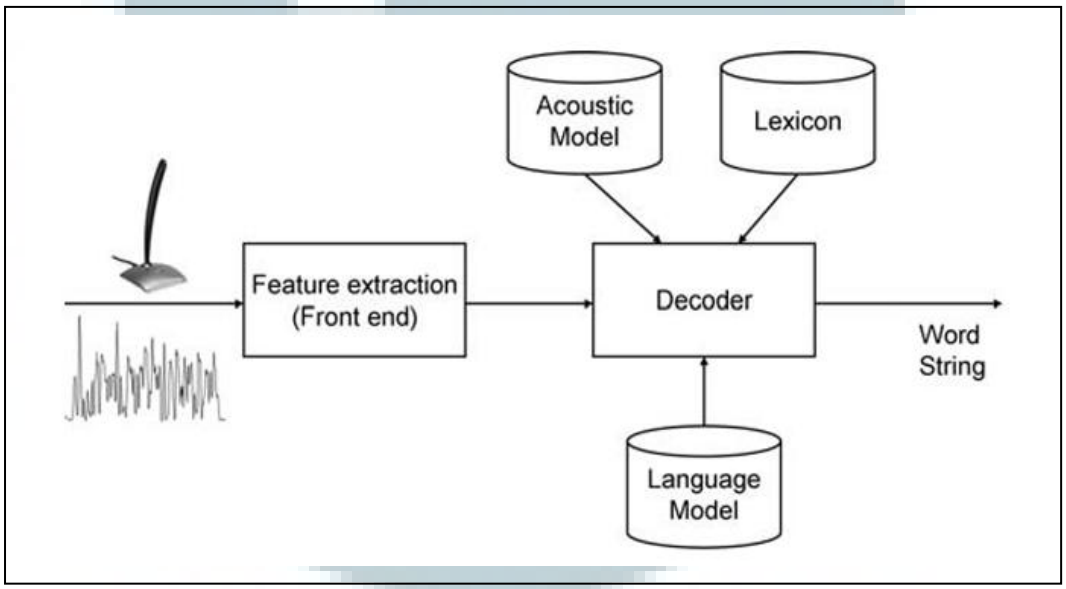
*Automatic Speech Recognition* (ASR) dapat diartikan sebagai suatu teknik mengorganisasikan pengucapan untuk menghasilkan suatu jawaban yang tepat terhadap masukkan yang diberikan (Juari, 2009). ASR juga dapat diartikan sebagai suatu proses dimana suatu sinyal akustik ditangkap oleh *microphone* dan diubah ke dalam sekumpulan kata dengan menggunakan program komputer. ASR termasuk dalam bidang teknologi pemrosesan pengucapan digital /*digital speech processing*, yang di dalamnya juga termasuk sintesis ucapan/ *speech synthesis* (*text – to – speech* dan *language generation*) serta biometrik suara/*voice biometrics* (identifikasi pengucap/*speaker identification* dan verifikasi pengucap/*speaker verification*) (Stouten, 2006).

Teknologi ini sendiri menjadi topik yang menarik perhatian masyarakat semenjak dipopulerkan di beberapa film Blockbuster pada tahun 1960 dan 1970, terutama film tahun 2001 karya Stanley Kubrick berjudul "A Space Odyssey". Pada film ini, sebuah komputer bernama "HAL" mampu berbicara dalam suara yang natural dan mampu memahami kata – kata yang diucapkan secara lancar, serta mampu memberikan respon yang sesuai. Kemampuan "HAL" dalam hal tersebut memberikan kesan terhadap masyarakat luas mengenai potensi adanya

suatu mesin yang memiliki kepandaian memahami perkataan manusia (Juang, 2004).

### 2.1.2 Tahapan Pengenalan Kata Dalam ASR

Tahapan pengenalan kata dalam suatu sistem ASR dapat digambarkan pada gambar berikut.




Gambar 2.1 Gambaran Proses Pengenalan Kata Dalam ASR

Sumber: http://what-when-how.com/video-search-engines/speech-recognition-audio-processing-video-search-engines/ (Diakses tanggal 5 Desember 2012)

*Feature Extraction* pada bagian *front end* atau sering disebut juga tahapan pemrosesan sinyal (*signal processing*) adalah tahapan dimana sinyal suara/sinyal akustik (dalam bentuk sinyal analog) akan diubah ke dalam bentuk sinyal digital, melalui proses *sampling*. Sinyal ini kemudian diukur karakteristik amplitudonya dan dinyatakan dalam suatu nilai *integer* melalui proses kuantisasi (Jurafsky, 2009).

Pada sisi *back end*, terdapat tiga bagian utama, yaitu pemodelan akustik/*acoustic model*, pemodelan bahasa/*language model*¸ dan *lexicon/dictionaries*. Berikut adalah penjelasan dari masing – masing tahapan.

1. Pemodelan Akustik (*Acoustic Model*)

   Pemodelan akustik berfungsi untuk mengetahui karakteristik unit pengenalan suara yang telah diproses oleh *feature extraction*. Unit pengenalan tersebut dapat berada pada tingkatan kata, tingkatan suku kata (*syllable*), ataupun tingkatan fonem (Chitturi, 2005).

2. Pemodelan Bahasa (*Language Model*)

   Pemodelan bahasa dalam ASR biasanya digunakan untuk memperkirakan kemungkinan rangkaian kata – kata yang akan dikenali ataupun dihipotesis. Dengan demikian, dapat dilakukan pembatasan pencarian bagi sistem ASR dalam proses pengenalan ucapannya. Pemodelan bahasa sendiri dapat dibangun melalui dua pendekatan sebagai berikut. (Juari, 2009).

   a. Pemodelan bahasa berbasis aturan (*Rules – Based Language Model*), dimana rangkaian kata – kata yang akan dikenali didefinisikan dalam suatu aturan yang statis.
   b. Pemodelan bahasa berbasis statistik (*Statistical Language Model*), dimana rangkaian kata – kata yang akan dikenali didefinisikan berdasarkan kemungkinan/probabilitas kemunculannya. Pemodelan *N-gram* adalah pemodelan statistik yang sering digunakan dalam kasus ini, seperti pemodelan *bigram* untuk kemungkinan kata – kata yang berpasangan, dan pemodelan *trigram* untuk kemungkinan rangkaian tiga

kata sekaligus. Pemodelan *N-gram* sendiri biasanya digunakan untuk memperkirakan suatu frasa panjang ataupun satu kalimat. (Ferdiansyah, 2012).

3. *Lexicon*/*Dictionaries*

   *Lexicon/Dictionaries* dapat diartikan sebagai kumpulan kata – kata dengan cara pelafalan (*pronounciation*) setiap katanya yang diungkapkan dalam rangkaian fonem (Jurafsky, 2009). Kata – kata yang mampu dikenali oleh ASR akan sangat bergantung pada *lexicon*. (Juari, 2009). Pada beberapa penelitian, seperti yang dilakukan oleh Juari (2009) serta Lestari (2010), daftar kata dalam *lexicon* dapat diperoleh dari koran dan majalah (misalnya Kompas dan Tempo). Kata – kata yang tidak terdaftar dalam *lexicon/dictionaries* ini sering disebut juga dengan kata – kata yang bersifat OOV (*Out of Vocabulary*). Kata – kata OOV biasanya mengacu pada kata – kata tertentu, seperti nama orang atau nama tempat (Juari, 2009).

Dalam membangun suatu aplikasi berbasis ASR, biasanya digunakan suatu *speech recognizer engine* yang dilengkapi dengan berbagai *toolkit* dalam membantu membuat kode – kode program yang dibutuhkan. Di beberapa penelitian, seperti yang dilakukan oleh Chitturi (2005), Juari (2009), serta Ferdiansyah (2012), sarana yang digunakan adalah CMU Sphinx, yaitu suatu *open source toolkit* yang dapat digunakan untuk membangun suatu sistem ASR, yang diciptakan dari hasil kerja sama antara Sphinx *Group* di Universitas Carnegie Mellon, Laboratorium Sun Microsystems, Laboratorium Penelitian Elektronik Mitsubishi/Mitsubishi *Electric Research Labs* (MERL), dan Hewlett Packard

(HP). Pada penelitian ini, penulis akan mencoba menggunakan *engine speech recognizer* lain, yaitu Microsoft *Speech Recognition* dan akan dijelaskan secara lebih lanjut pada bagian 2.2.

### 2.1.3 Faktor – Faktor Penentu Kinerja ASR

Menurut Pratondo (2009), kesalahan pengenalan oleh teknologi berbasis *Speech Recognition* dapat disebabkan oleh beberapa hal sebagai berikut.

1. Kesalahan dalam pengucapan (*misspoken*) dan pembacaan (*misread*) frasa
2. Keadaan emosional yang ekstrim (misalnya stress, sedang marah, atau sedang sedih)
3. Pergantian penempatan *microphone* (*intrasession* atau *intersession*)
4. Kekurangan atau ketidak-konsistenan akustik dari ruangan (misalnya *multipath* dan *noise*)
5. *Channel Mismatch* (misalnya penggunaan *microphone* yang berbeda dalam perekaman dan verifikasi)
6. Sakit (misalnya flu yang dapat merubah *vocal tract*)
7. Penuaan/*aging* (model *vocal tract* dapat berubah berdasarkan usia)

Beberapa parameter lain yang juga turut memengaruhi kinerja ASR antara lain adalah sebagai berikut (Jurafsky, 2009).

1. Ukuran perbendaharaan kata/*vocabulary size*

   Proses pengenalan kata akan menjadi lebih mudah apabila jumlah perbendaharaan kata dalam ASR sedkit. Misalnya dalam pengenalan dua jenis kata berbeda (hanya kata *"ya"* dan "*tidak"*), atau pengenalan 11 jenis

kata berbeda (dari "*nol*" sampai dengan *"sepuluh*"). Di sisi lain, perbendahaan kata dalam jumlah besar (misalnya antara 20.000 sampai dengan 60.000 kata) akan menyebabkan proses pengenalan kata menjadi lebih rumit, misalnya dalam mentranskripsikan percakapan antara dua orang melalui telepon atau siaran berita.

2. Tingkat kelancaran pengucapan

   Pengenalan kata secara terisolasi/*isolated word* (dimana pengucapan setiap kata diselingi jeda singkat) relatif lebih mudah daripada pengenalan kata secara berkesinambungan/*continuous speech* (dimana pengucapan dilakukan secara lancar, seperti berbicara normal). Kelancaran pengucapan juga terkait dengan penerapan ASR. Untuk pendiktean, pengenalan kata biasanya lebih mudah karena manusia cenderung akan secara otomatis menyederhanakan ucapannya, berbicara lebih perlahan, lebih keras, dan jelas apabila berbicara dengan mesin. Untuk transkripsi percakapan antara dua orang atau lebih, proses pengenalan kata akan menjadi jauh lebih sulit.

3. *Channel* dan *Noise*

   Proses pendiktean menggunakan ASR biasanya dilakukan dengan menggunakan *microphone* yang dipasang di kepala (*head mounted microphone*), dengan kualitas yang baik. Jenis *microphone* seperti ini dapat menghilangkan distorsi yang terjadi pada *microphone* meja saat pembicara tanpa sengaja menggerakkan kepalanya. Faktor *noise* dapat membuat proses pengenalan kata menjadi lebih sulit, sehingga pendiktean akan lebih maksimal dilakukan di lingkungan yang sunyi (misalnya ruangan kantor)

daripada di lingkungan yang bising (misalnya di dalam mobil yang melaju di jalan tol dengan jendela terbuka).

4. Karakteristik logat pembicara

   Proses pengenalan kata akan menjadi lebih mudah apabila pembicara mengucapkan kata – kata dalam logat yang standar atau umum. Untuk pembicara yang memiliki logat – logat tertentu (misalnya logat bahasa asing) ataupun masih anak - anak, proses pengenalan kata biasanya akan menjadi lebih sulit, kecuali jika sistem ASR memang sudah dirancang khusus untuk mengakomodasi hal – hal tersebut.

### 2.1.4 ASR dan Phonetic Transcription

Apabila akan digunakan untuk mengenali suatu bahasa tertentu, maka ketiga tahapan di atas harus dilakukan menyesuaikan dengan prinsip – prinsip pada bahasa tersebut. Pada prakteknya, pembangunan pemodelan akustik untuk bahasa tertentu cukup sulit dilakukan, tidak seperti pemodelan bahasa dan *lexicon*, mengingat harus dilakukannya perekaman pengucapan dari berbagai pengucap untuk mendapatkan suatu teknologi ASR yang *speaker independent* (Ferdiansyah, 2012). Salah satu pendekatan yang dapat dilakukan untuk mengatasi kesulitan ini adalah dengan melakukan translasi fonem untuk mendapakan *phonetic transcription* (representasi pelafalan dalam simbol – simbol fonem) yang paling sesuai dari suatu kata. *Phonetic transcription* dan translasi fonem akan dijelaskan secara lebih lanjut pada bagian 2.3.

## 2.2 Microsoft Speech Recognition

### 2.2.1 Definisi Microsoft Speech Recognition

Microsoft *Speech Recognition* adalah suatu *engine* pengenal ucapan/*speech recognizer* berbasis ASR yang dikembangkan oleh Microsoft. *Engine* ini bekerja dengan menerima aliran suara/*audio stream* sebagai inputnya dan menghasilkan keluaran dalam bentuk tulisan/teks. *Engine* ini terdiri dari dua bagian besar, yaitu *front end* dan *back end*. (Microsoft, 2011).

1. *Front End*, berfungsi untuk memroses aliran suara/*audio stream*, mengisolasi segmen – segmen suara yang memiliki kemungkinan sebagai suatu ucapan/*speech* dan mengubahnya ke dalam rangkaian nilai numerik, yang mengkarakteristikan suara vokal dalam sinyal suara tersebut.
2. *Back End*, berfungsi untuk memroses keluaran yang akan dihasilkan dari input yang telah diproses oleh bagian *front end*. Bagian *back end* terdiri dari tiga proses yang sama dengan prinsip dasar ASR, yaitu *acoustic model*, *language model*, dan *lexicon*.
   a. *Acoustic Model*

      Merepresentasikan suara akustik dari bahasa utama yang digunakan sebagai acuan ASR. Dapat dilatih untuk mengenali karakteristik pola suara dari pengucap tertentu dan dalam keadaan akustik tertentu. *Acoustic Model* untuk *engine* ini biasanya telah disediakan dalam suatu paket bahasa (*Runtime Languages*), dan sampai saat ini telah tersedia dalam 26 bahasa yang berbeda, namun tidak mencakup untuk Bahasa Indonesia.

b. *Language Model*

Merepresentasikan cara – cara bagaimana mengkombinasikan kata – kata yang terdapat pada *lexicon*. Pada *engine* ini, *language model* dapat dibuat dengan menggunakan *grammar* yang mendefinisikan aturan – aturan mengenai cara pengucapan yang akan dikenali oleh *engine*. Untuk berbagai penerapan tertentu, *grammar* sebagai *language model* memiliki berbagai keuntungan sebagai berikut.

1) Meningkatkan keakurasian pengenalan ucapan
2) Menjamin bahwa semua hasil pengenalan ucapan akan memiliki makna bagi aplikasi
3) Memungkinkan *engine* mengenali makna semantik yang terdapat dalam hasil pengenalan ucapan

c. *Lexicon*

Merupakan daftar kata yang mampu dikenali oleh aplikasi. *Lexicon* juga dapat berisikan informasi mengenai pelafalan dari setiap kata tersebut, dengan tujuan meningkatkan keakurasian pengenalan suatu kata oleh *engine*. Pada *engine* ini, pelafalan dapat direpresentasikan oleh satu dari tiga jenis simbol pelafalan berikut.

1) Simbol IPA (*International Phonetic Alphabet*)
2) Simbol Microsoft UPS (*Universal Phone Set*)
3) Simbol Speech API (SAPI). Simbol ini biasanya digunakan secara khusus oleh *engine* Microsoft *Speech* untuk bahasa Cina (Taiwan dan

RRC), Inggris (Amerika), Perancis (Standard), Jerman (Standard), Jepang, dan Spanyol.

Proses membangun suatu aplikasi berbasiskan ASR dengan menggunakan *engine* ini dapat dibagi menjadi lima tahapan dasar sebagai berikut.

1. Memulai *Speech Recognizer*
2. Membuat *grammar*
3. Memuat *grammar* ke dalam *Speech Recognizer*
4. Me*register* notifikasi *event* pada *Speech Recognizer*
5. Membuat *handler* untuk *event* tersebut

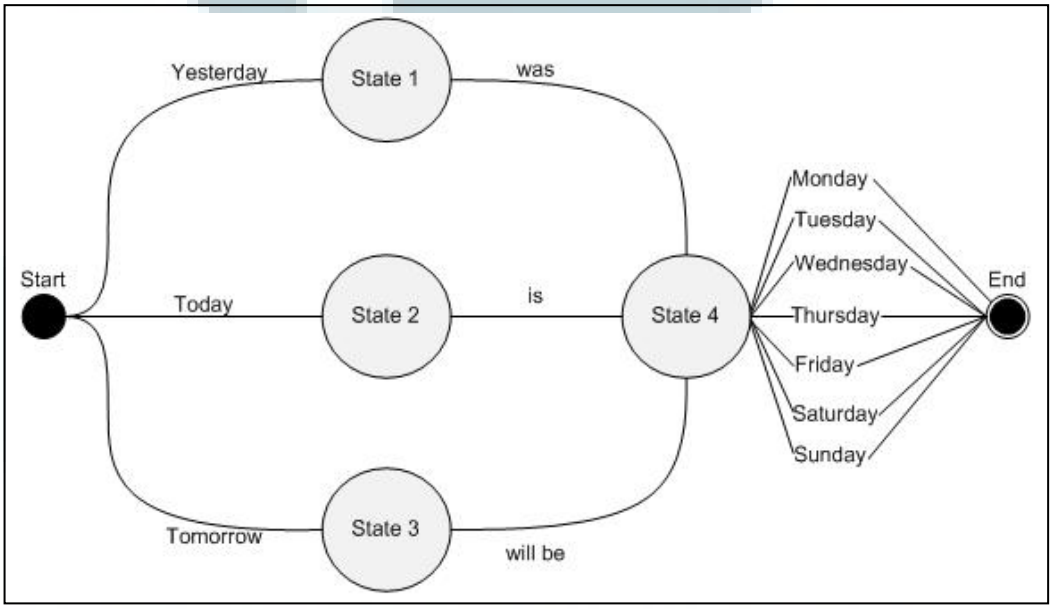
### 2.2.2 Pelafalan

Performa keakurasian dari aplikasi dapat ditingkatkan dengan penyediaan pelafalan dari suatu kata. Biasanya, pelafalan dibutuhkan apabila terdapat kata – kata yang tidak umum pada patokan bahasa yang digunakan. Pemberian pelafalan ini biasanya berkaitan dengan penyediaan *lexicon* sebagai salah satu unsur penting dalam pembangunan ASR, dan disesuaikan dengan simbol pelafalan yang didukung oleh *engine* (yang biasa digunakan adalah simbol UPS). Pemberian pelafalan atau simbol fonem dilakukan secara per huruf penyusun kata dan dapat diperjelas dengan pemenggalan suku kata, atau pemberian tanda penekanan pada huruf – huruf atau bagian – bagian tertentu dari kata. Pelafalan untuk fonem – fonem yang berbunyi komposit (terdiri dari dua fonem) dapat diperjelas dengan pemberian tanda ‘+’. Pemenggalan suku kata ditandai dengan tanda ‘.’ yang diletakkan antar suku kata.

### 2.3.3 Grammar

*Grammar* merupakan salah satu unsur penting dalam pembangunan aplikasi berbasis ASR dengan menggunakan *engine* ini. Setiap aturan/*rule* dalam *grammar* dapat digambarkan sebagai suatu *graph* yang mendeskripsikan semua kemungkinan frasa yang mungkin didefinisikan *rule* tersebut. *Graph* yang tersusun dari sekumpulan *state* dengan sejumlah transisi yang ke *state* lainnya. Setiap *rule* didefinisikan oleh suatu *state* awal, suatu *state* akhir, dan sekumpulan *state* lainnya yang berjumlah nol ataupun berada di pertengahan *graph*. (Microsoft, 2011).

Berikut adalah contoh *graph* yang menggambarkan *rule* dalam *grammar* yang mendefinisikan suatu kalimat sederhana yang menyatakan keterangan hari (dalam Bahasa Inggris).




Gambar 2.2 Contoh *Grammar* Pernyataan Hari Dalam Bahasa Inggris
(Sumber: Microsoft, 2011)

Pada gambar tersebut, terlihat bahwa kata pertama yang dapat diucapkan oleh pembicara adalah salah satu dari ketiga kata “Yesterday”, “Today”, dan “Tomorrow”. Kata tersebut akan membawa *grammar* ke salah satu *state* (*state* 1, 2, atau 3 sesuai dengan kata pertama yang diucapkan), dimana pengucap kemudian harus mengucapkan satu kata yang sesuai dengan keadaan *state* nya sekarang. Misalnya pengucap harus mengucap kata “was” ketika kata pertama yang diucapkan adalah “Yesterday”, mengucap kata “will be” ketika kata pertama yang diucapkan adalah “Tomorrow”, dan seterusnya, sehingga *grammar* berada pada *state* 4. Pengucap kemudian dapat mengucap salah satu dari alternatif kata penunjuk hari (“Monday”, “Tuesday”, “Wednesday”, dan seterusnya hingga “Sunday”) agar *grammar* berhenti tepat di *state* akhir. Pengucapan yang mampu membawa *grammar* pada *state* akhir inilah yang akan dikenali oleh *engine* dan ditampilkan hasilnya (Microsoft, 2011).

Pada *engine* ini, *grammar* biasanya berada pada *file* terpisah dalam format dokumen XML dan memiliki ekstensi .grxml. Berikut adalah contoh penulisan *grammar* dalam format tersebut.

```
<grammar version="1.0" xml:lang="en-US" mode="voice" root="mediaMenu"
xmlns="http://www.w3.org/2001/06/grammar">

  <rule id="mediaMenu" scope="public">

  Find

    <one-of>
      <item> albums </item>
      <item> artists </item>
    </one-of>

  in the

    <ruleref uri="#genres"/>

  category.

  </rule>

  <rule id="genres" scope="public">
    <one-of>
      <item> Blues </item>
      <item> Classical </item>
      <item> Gospel </item>
      <item> Jazz </item>
      <item> Rock </item>
    </one-of>
  </rule>
</grammar>
```

Gambar 2.3 Contoh Penulisan *Grammar* Dalam Microsoft *Speech Recognition Engine* (Sumber: Microsoft, 2011)

Dengan masing – masing penjelasan elemen dalam contoh *grammar* tersebut (dan juga yang paling sering digunakan) adalah sebagai berikut (Microsoft, 2011).

1) *Rule*, sebagai penampung sekumpulan kata atau frasa yang dapat digunakan oleh *engine* untuk melakukan pencocokkan dengan input yang diberikan, serta mendefinisikan rangkaian kata yang dapat diucapkan secara berkesinambungan oleh pengguna.
2) *Item*, sebagai penampung kata – kata yang dapat diucapkan, ataupun sebagai penampung elemen – elemen lain, misalnya *ruleref*.

3) *One-of*, sebagai penanda adanya sekumpulan alternatif aturan yang dapat dipilih salah satunya. Elemen ini berfungsi untuk meningkatkan fleksibilitas *grammar* dimana salah satu input hanya dapat dicocokkan dengan salah satu alternatif saja.
4) *Ruleref*, sebagai penunjuk ke *rule* lain yang harus dipenuhi oleh input agar pengenalan kata oleh *engine* dapat dilakukan.

Contoh *grammar* di atas adalah *grammar* yang digunakan untuk mencari artis ataupun album dalam suatu kategori musik. Terlihat bahwa kategori musik yang tersedia (yang direpresentasikan oleh elemen *one-on* pada *rule genres*) adalah *Blues, Classical, Gospel, Jazz,* dan *Rock*. Pertama – tama, pengguna harus mengucap kata "*Find"* lalu diikuti hal apa yang mau dicari, yaitu antara artis ataupun album. Pengguna lalu melanjutkan ucapannya dengan frasa *"in the*" dan memilih salah satu kategori musik yang tersedia, dan ditutup dengan kata "*category*". Contoh kalimat – kalimat yang dapat diterima misalnya adalah "*Find albums in the Blues category", "Find artists in the Gospel category*", dan seterusnya.

## 2.3 Phonetic Transcription

### 2.3.1 Simbol IPA dan ARPAbet

*Phonetic Transcription* berfungsi untuk merepresentasikan hasil transkripsi suatu kata ke dalam bentuk fonetisnya sesuai dengan simbol fonetis yang digunakan. Fonem sendiri dapat diartikan sebagai suara ucapan, yang dalam teknologi ASR biasanya direpresentasikan dalam simbol fonetis. Terdapat dua

simbol fonetis yang umum digunakan, yaitu *International Phonetic Alphabet* (IPA) dan ARPAbet. IPA merupakan standar simbol fonetis yang dikembangkan oleh International Phonetic Association pada tahun 1888 dengan tujuan untuk mentranskripsikan suara dari seluruh bahasa manusia, sedangkan ARPAbet biasanya secara spesifik dirancang untuk digunakan dalam bahasa Inggris Amerika (*American English*) dan menggunakan simbol – simbol ASCII. Dalam beberapa penerapan, seperti kamus pelafalan *speech recognition* dan *speech synthesis*, umumnya ARPAbet lebih umum digunakan daripada IPA karena IPA mengandung huruf – huruf non-ASCII yang secara representasinya tidak praktis digunakan untuk menggambarkan pelafalan. (Jurafsky, 2009).

Berikut adalah tabel *phonetic transcription* dalam ARPAbet untuk konsonan dan vokal, dalam penerapannya menggambarkan pelafalan kata – kata berbahasa Inggris, beserta dengan contoh kata, simbol IPA dan ARPAbet nya.

Tabel 2.1 Tabel *Phonetic Transcription* Untuk Konsonan Bahasa Inggris Dalam ARPAbet (Sumber: Jurafsky, 2009)

| **Simbol ARPAbet** | **Simbol IPA** | **Contoh Kata (Bahasa Inggris)** | **Transkripsi dalam ARPAbet** |
|---|---|---|---|
| [p] | [p] | parsley | [p aa r s l i y] |
| [t] | [t] | tea | [t iy] |
| [k] | [k] | cook | [k uh k] |
| [b] | [b] | bay | [b ey] |

Tabel 2.1 Tabel *Phonetic Transcription* Untuk Konsonan Bahasa Inggris Dalam ARPAbet (Lanjutan)

| **Simbol ARPAbet** | **Simbol IPA** | **Contoh Kata (Bahasa Inggris)** | **Transkripsi dalam ARPAbet** |
|---|---|---|---|
| [d] | [d] | dill | [d ih l] |
| [g] | [g] | garlic | [g aa r l ix k] |
| [m] | [m] | mint | [m ih n t] |
| [n] | [n] | nutmeg | [n ah t m eh g] |
| [ng] | [ŋ] | baking | [b ey k ix ng] |
| [f] | [f] | flour | [f l aw axr] |
| [v] | [v] | clove | [k l ow v] |
| [th] | [θ] | thick | [th ih k] |
| [dh] | [ð] | those | [dh ow z] |
| [s] | [s] | soup | [s uw p] |
| [z] | [z] | eggs | [eh g z] |
| [sh] | [ʃ] | squash | [s k w aa sh] |
| [zh] | [ʒ] | ambrosia | [ae m b r ow zh ax] |
| [ch] | [tʃ] | cherry | [ch eh r iy] |
| [jh] | [ʤ] | jar | [jh aa r] |
| [l] | [ l] | licorice | [l ih k axr ix sh] |
| [w] | [w] | kiwi | [k iy w iy] |
| [r] | [r] | rice | [r ay s] |
| [y] | [j] | yellow | [y eh l ow] |

Tabel 2.1 Tabel *Phonetic Transcription* Untuk Konsonan Bahasa Inggris Dalam ARPAbet (Lanjutan)

| **Simbol ARPAbet** | **Simbol IPA** | **Contoh Kata (Bahasa Inggris)** | **Transkripsi dalam ARPAbet** |
|---|---|---|---|
| [h] | [h] | honey | [h ah n iy] |
| Fonem yang jarang digunakan | | | |
| [q] | [ʔ] | uh-oh | [q ah q ow] |
| [dx] | [ɾ] | butter | [b ah dx axr] |
| [nx] | [ɾ̃] | winner | [w ih nx axr] |
| [el] | [ɫ] | table | [t ey b el] |

Tabel 2.2 Tabel *Phonetic Transcription* Untuk Vokal Bahasa Inggris Dalam ARPAbet (Sumber: Jurafsky, 2009)

| **Simbol ARPAbet** | **Simbol IPA** | **Contoh Kata (Bahasa Inggris)** | **Transkripsi dalam ARPAbet** |
|---|---|---|---|
| [iy] | [ i ] | lily | [l ih l iy] |
| [ih] | [ɪ] | lily | [l ih l iy] |
| [ey] | [eɪ] | daisy | [d ey z iy] |
| [eh] | [ɛ] | pen | [p eh n] |
| [ae] | [æ] | aster | [ae s t axr] |
| [aa] | [ɑ] | poppy | [p aa p iy] |
| [ao] | [ɔ] | orchid | [ao r k ix d] |
| [uh] | [ʊ] | wood | [w uh d] |
| [ow] | [oʊ] | lotus | [l ow dx ax s] |

Tabel 2.2 Tabel *Phonetic Transcription* Untuk Vokal Bahasa Inggris Dalam ARPAbet (Lanjutan)

| **Simbol ARPAbet** | **Simbol IPA** | **Contoh Kata (Bahasa Inggris)** | **Transkripsi dalam ARPAbet** |
|---|---|---|---|
| [uw] | [u] | tulip | [t uw l ix p] |
| [ah] | [ʌ] | buttercup | [b ah dx axr k ah p] |
| [er] | [ɝ] | bird | [b er d] |
| [ay] | [aɪ] | iris | [ay r ix s] |
| [aw] | [aʊ] | sunflower | [s ah n f l aw axr] |
| [oy] | [ɔɪ] | soil | [s oy l] |
| Fonem yang tereduksi dan tidak umum | | | |
| [ax] | [ə] | lotus | [l ow dx ax s] |
| [axr] | [ɚ] | heather | [h eh dh axr] |
| [ix] | [ɨ] | tulip | [t uw l ix p] |
| [ux] | [ʉ] | dude | [d ux d] |

### 2.3.2 Simbol UPS

Selain simbol IPA dan ARPAbet, terdapat simbol lain yaitu UPS (*Universal Phone Set*). UPS juga merupakan kumpulan alfabet representasi fonem yang didasarkan oleh IPA. UPS dibuat oleh Microsoft dan sama halnya dengan IPA dan ARPAbet, UPS juga dapat digunakan untuk mendefinisikan pelafalan dari kata berbahasa apapun. Berikut adalah tabel contoh simbol UPS dan IPA nya beserta contoh kata dalam Bahasa Inggrisnya (Microsoft, 2011).

Tabel 2.3 Tabel Contoh Representasi UPS dan IPA

(Sumber: Microsoft, 2011)

| **Simbol UPS** | **Simbol IPA** | **Contoh Kata (Bahasa Inggris)** |
|---|---|---|
| P | p | put |
| B | b | big |
| M | m | mat |
| F | f | fork |
| V | v | vat |
| TH | θ | thin |
| DH | ð | then |
| T | t | talk |
| D | d | dig |
| N | n | no |
| DX | ɾ | butter |
| S | s | sit |
| Z | z | zap |
| L | l | lid |
| SH | ʃ | she |
| ZH | ʒ | pleasure |
| R | ɹ | red |
| J | j | yard |
| W | w | with |

Tabel 2.3 Tabel Contoh Representasi UPS dan IPA (Lanjutan)

| **Simbol UPS** | **Simbol IPA** | **Contoh Kata (Bahasa Inggris)** |
|---|---|---|
| K | k | cut |
| G | g | gut |
| NG | ŋ | sing |
| H | h | help |
| CH | ʧ | chin |
| JH | ʤ | joy |
| I | i | feel |
| U | u | too |
| IH | ɪ | fill |
| UH | ʊ | book |
| O | o | go |
| AX | ə | ago |
| EH | ɛ | pet |
| ER | ɜ | bird |
| AH | ʌ | cut |
| AO | ɔ | dog |
| AE | Æ | cat |
| AA | ɑ | father |
| Q | ɒ | hot |

### 2.3.3 Fonem Bahasa Indonesia dan Pemenggalan Suku Kata

Pemerintah Indonesia telah menetapkan aturan mengenai pelafalan yang benar bagi setiap fonem penyusun kata dalam Bahasa Indonesia. Bahasa Indonesia sendiri ditulis dengan menggunakan alfabet Latin yang terdiri dari 26 karakter (A – Z), dengan korespondensi antara suara dengan bentuk tulisannya bersifat umum (serupa). Berikut adalah alfabet penyusun kata – kata dalam Bahasa Indonesia.

Tabel 2.4 Tabel Alfabet Penyusun Kata Bahasa Indonesia
(Sumber: Kuntarto, 2011)

| **Kategori** | **Huruf** | **Contoh Pemakaian Dalam Kata** | | |
|---|---|---|---|---|
| | | **Awal** | **Tengah** | **Akhir** |
| Vokal | a | *a*pi | p*a*di | lus*a* |
| | e* | *e*nak | p*e*tak | sor*e* |
| | | *e*mas | k*e*na | tip*e* |
| | i | *i*tu | s*i*mpan | murn*i* |
| | o | *o*leh | k*o*ta | radi*o* |
| | u | *u*lang | b*u*mi | ib*u* |
| Konsonan | b | *b*ahasa | se*b*ut | ada*b* |
| | c | *c*akap | ka*c*a | - |
| | d | *d*ua | a*d*a | aba*d* |
| | f | *f*akir | ka*f*an | maa*f* |
| | g | *g*una | ti*g*a | gude*g* |
| | h | *h*ari | sa*h*am | tua*h* |
| | j | *j*alan | man*j*a | mikra*j* |

Tabel 2.4 Tabel Alfabet Penyusun Kata Bahasa Indonesia (Lanjutan)

| **Kategori** | **Huruf** | **Contoh Pemakaian Dalam Kata** | | |
|---|---|---|---|---|
| | | **Awal** | **Tengah** | **Akhir** |
| Konsonan | k | *k*ami | pa*k*sa | politi*k* |
| | l | *l*ekas | a*l*as | aka*l* |
| | m | *m*uka | ka*m*i | dia*m* |
| | n | *n*ama | ta*n*ah | dau*n* |
| | p | *p*asang | a*p*a | sia*p* |
| | q** | *q*uran | status-*q*uo | taufi*q* |
| | r | *r*aih | ba*r*a | putar |
| | s | *s*ampel | asli | tangka*s* |
| | T | *t*ali | ma*t*a | rapa*t* |
| | v | *v*aria | la*v*a | - |
| | w | *w*anita | ha*w*a | - |
| | x** | *x*erox | - | sinar–*x* |
| | y | *y*akin | pa*y*ung | - |
| | z | *z*eni | lazim | juz |
| Diftong | ai | *ai*n | mal*ai*kat | pand*ai* |
| | au | *au*la | s*au*dara | harim*au* |
| | oi | - | b*oi*kot | amb*oi* |

Tabel 2.4 Tabel Alfabet Penyusun Kata Bahasa Indonesia (Lanjutan)

| **Kategori** | **Huruf** | **Contoh Pemakaian Dalam Kata** | | |
|---|---|---|---|---|
| | | **Awal** | **Tengah** | **Akhir** |
| Gabungan Huruf Konsonan | kh | *kh*usus | a*kh*ir | tari*kh* |
| | ng | *ng*ilu | ba*ng*un | sena*ng* |
| | ny | *ny*ata | ba*ny*ak | - |
| | sy | *sy*arat | is*y*arat | aras*y* |

Pengelompokkan fonem Bahasa Indonesia sendiri terdiri dari enam buah vokal, tiga buah diftong, satu buah diftong informal, dan 22 konsonan, seperti yang ditunjukkan pada tabel berikut.

Tabel 2.5 Tabel Fonem Bahasa Indonesia Beserta Contoh Katanya
(Sumber: Lestari, 2010)

| **Kategori** | **Fonem** | **Contoh Kata** | **Representasi Fonem Penyusun** |
|---|---|---|---|
| Vokal | /a/ | saya | /s *a* y *a*/ |
| | /e/ | enak | /*e* n a k/ |
| | /E/ | kEmana | /k *E* m a n a/ |
| | /i/ | ingin | /*i* n g *i* n/ |
| | /o/ | orang | /*o* r a ng/ |
| | /u/ | untuk | /*u* n t *u* k/ |
| Diftong | /ai/ | sungai | /s u ng *ai*/ |
| | /au/ | danau | /d a n *au*/ |
| | /oi/ | amboi | /a m b *oi*/ |

Tabel 2.5 Tabel Fonem Bahasa Indonesia Beserta Contoh Katanya (Lanjutan)

| Kategori | Fonem | Contoh Kata | Representasi Fonem Penyusun |
|---|---|---|---|
| Diftong Informal | /ei/ | hei | /h *ei*/ |
| Semi-vokal | /w/ | wanita | /*w* a n i t a/ |
| | /y/ | saya | /s a *y* a/ |
| Konsonan Plosif | /b/ | berapa | /*b* e r a p a/ |
| | /p/ | petani | /*p* e t a n i/ |
| | /d/ | dia | /*d* i a/ |
| | /t/ | teman | /*t* e m a n/ |
| | /g/ | giat | /*g* i a t/ |
| | /k/ | kamu | /*k* a m u/ |
| | /kh/ | khairul | /*kh* a i r u l/ |
| Konsonan Afrikat | /j/ | juga | /*j* u g a/ |
| | /c/ | cinta | /*c* i n t a/ |
| Konsonan Frikatif | /f/ | maaf<br>video | /m a a *f*/<br>/*f* i d e o/ |
| | /z/ | jenazah | /j e n a z a h/ |
| | /s/ | saya | /*s* a y a/ |
| | /sy/ | syahdu | /*sy* a h d u/ |
| | /h/ | hujan | /*h* u j a n/ |
| Konsonan Likuida | /r/ | ramai | /*r* a m a i/ |
| | /l/ | lambat | /*l* a m b a t/ |

Tabel 2.5 Tabel Fonem Bahasa Indonesia Beserta Contoh Katanya (Lanjutan)

| Kategori | Fonem | Contoh Kata | Representasi Fonem Penyusun |
|---|---|---|---|
| Konsonan Nasal | /m/ | mana | /*m* a n a/ |
| | /n/ | mana | /m a *n* a/ |
| | /ny/ | nyanyian | /*ny* a *ny* i a n/ |
| | /ng/ | lambang | /l a m b a *ng*/ |

Pemenggalan suku kata atau sering disebut juga *syllabification* adalah suatu tindakan memisah – misahkan suatu rangkaian fonem ke dalam kumpulan suku kata. Pemenggalan suku kata ini memiliki peranan penting dalam berbagai aplikasi berbasis ucapan, seperti *speech synthesis* untuk menentukan fonem tertentu berdasarkan letaknya dalam suatu suku kata. Untuk *speech recognition* sendiri, pemenggalan suku kata ini sangat berguna bagi *engine recognizer* yang memungkinan representasi pelafalan dalam bentuk suku kata (Jurafsky, 2009).

Untuk pemenggalan suku kata dalam Bahasa Indonesia sendiri, ada beberapa aturan yang harus diperhatikan sebagai berikut. (Kuntarto, 2011).

1. Pemenggalan kata pada kata dasar dilakukan sebagai berikut.
   a. Jika di tengah kata ada huruf vokal yang berurutan, pemenggalannya dilakukan di antara kedua huruf vokal itu. Misalnya: b*u*-*a*h, m*a*-*i*n, n*i*-*a*t, s*a*-*a*t.
   b. Huruf diftong *ai, au,* dan *oi* tidak dipenggal. Misalnya: pan-d*ai, au*-la, s*au*-da-ra, am-b*oi*.

c. Jika di tengah kata dasar ada huruf konsonan (termasuk gabungan huruf konsonan) di antara dua buah huruf vokal, pemenggalannya dilakukan sebelum huruf konsonan itu. Misalnya: b*a-pa*k, l*a-wa*n, d*e-nga*n, k*e-nya*ng, m*u-ta-khi*r, m*u-sya-wa-ra*h.

d. Jika di tengah kata dasar ada dua huruf konsonan yang berurutan, pemenggalannya dilakukan di antara kedua huruf konsonan itu. Misalnya: A*p-r*il, ca*p-l*ok, ma*kh-l*uk, ma*n-d*i, sa*ng-g*up, so*m-b*ong, swas*-t*a.

e. Jika di tengah kata dasar ada tiga huruf konsonan atau lebih yang masing – masing melambangkan satu bunyi, pemenggalannya dilakukan di antara huruf konsonan yang pertama dan huruf konsonan yang kedua. Misalnya: u*l-tr*a, i*n-fr*a, be*n-tr*ok, i*n-str*u-men.

   Catatan:

   (1) Gabungan huruf konsonan yang melambangkan satu bunyi tidak dipenggal. Misalnya: ba*ng*-krut, ba*ng*-sa, ba-*ny*ak, i*kh*-las, ko*ng*-res, ma*kh*-luk, mas*y*-hur, sa*ng*-gup.

   (2) Pemenggalan kata tidak boleh menyebabkan munculnya huruf (vokal) di awal atau akhir baris. Misalnya: itu → i-tu, setia →se-ti-a.

2. Pemenggalan kata dengan awalan, akhiran, atau partikel dilakukan di antara bentuk dasar dan imbuhan atau partikel itu. Misalnya: *ber*-jalan, *mem*-bantu, *di*-ambil, *ter*-bawa, *per*-buat, makan-*an*, letak-*kan*, *me*-ra-sa-*kan*, pergi-*lah*, apa-*kah*, *per*-bu-at-*an*, ke-ku-at-*an*.

   Catatan:

(1) Pemenggalan kata berimbuhan yang bentuk dasarnya mengalami perubahan dilakukan seperti pada kata dasar. Misalnya: *me-nu*-tup, *me-ma*-kai, *me-nya*-pu, *me-nge*-cat, *pe-no*-long, *pe-mi*-kir, *pe-nga*-rang, *pe-nye*-but, *pe-nge*-tik.

(2) Akhiran –*i* tidak dipisahkan pada pergantian baris.

(3) Pemenggalan kata bersisipan dilakukan seperti pada kata dasar. Misalnya: ge-lem-bung, ge-mu-ruh, ge-ri-gi, si-nam-bung, te-lun-juk.

(4) Pemenggalan tidak dilakukan pada suku kata yang terdiri dari satu vokal. Misalnya: m*au*.

3. Jika sebuah kata terdiri atas dua unsur atau lebih dan salah satu unsurnya it dapat bergabung dengan unsur lain, pemenggalan dilakkan di antara unsur – unsur itu. Tiap – tiap unsur gabungan dipenggal seperti pada kata dasar. Misalnya: bi-o-gra-fi, bi-o-da-ta, fo-to-gra-fi, fo-to-ko-pi, in-tro-spek-si, in-tro-jek-si, ki-lo-gram, ki-lo-me-ter, pas-ca-pa-pen, pas-ca-sar-ja-na.
4. Nama orang, badan hukum, atau nama diri lain yang terdiri dari dua unsur atau lebih dipenggal pada akhir baris di antara unsurnya (tanpa tanda pisah). Unsur nama yang berupa singkatan dipisahkan.

### 2.3.4 Translasi Fonem

Translasi fonem biasanya dilakukan apabila terdapat perbedaan pemodelan akustik dengan bahasa tujuan (misalnya ASR dengan pemodelan akustik Bahasa Inggris untuk mengenali Bahasa Indonesia). Hasil translasi ini telah disajikan dalam penelitian yang dilakukan Ferdiansyah (2012). Teknik ini serupa dengan

pemetaan fonem/*phoneme mapping*, dengan dilakukannya pemetaan simbol – simbol fonem dari kata – kata berbahasa asing tersebut ke dalam simbol – simbol fonem bahasa tujuan (Lestari, 2010).

Pemetaan fonem/*phoneme mapping* sendiri dapat dilakukan melalui dua jenis pendekatan sebagai berikut (Schultz, 2001).

1. Pemetaan fonem berbasiskan pengetahuan/*Knowledge – Based Phoneme Mapping*

   Pendekatan ini dilakukan dengan asumsi tidak adanya data *training* yang tersedia pada bahasa tujuan, sehingga diterapkanlah pengetahuan a priori (*a priori knowledge*). Sejauh ini, belum terdapat suatu *database* yang memuat data *training* untuk Bahasa Indonesia, sehingga pendekatan ini dapat dikatakan sebagai pendekatan yang paling intuitif dan tepat sasaran. Pemetaan didapatkan dengan memperkirakan kesamaan bunyi antar bahasa (Lestari, 2010).

2. Pemetaan fonem berbasiskan data/*Data - Driven Phoneme Mapping*

   Pendekatan ini dilakukan apabila data *training* sudah tersedia. Latihan/*training* terhadap pemodelan akustik dilakukan dengan menggunakan *phonetic transcription* dari kata – kata asing yang diinginkan. Pada kasus Bahasa Indonesia, data yang tersedia masih sangat terbatas, dikarenakan diperlukannya contoh pengucapan bahasa asing oleh orang – orang Indonesia dalam jumlah banyak, untuk membedakan antara *training* dan pengujian/*testing*. Berdasar pada masalah yang ada pada poin (1),

pendekatan ini tidak terlalu cocok untuk diterapkan pada Bahasa Indonesia (Lestari, 2010).

Hasil translasi fonem untuk ASR Bahasa Indonesia dengan pemodelan akustik Bahasa Inggris yang dilakukan oleh Ferdiansyah (2012) dapat dilihat pada tabel berikut.

Tabel 2.6 Tabel Hasil Translasi Fonem Indonesia ke Inggris
(Sumber: Ferdiansyah, 2012)

| **Fonem Indonesia** | **Hasil Translasi** | **Fonem Indonesia** | **Hasil Translasi** |
|---|---|---|---|
| /a/ | /ah/ | /o/ | /ow/ |
| /b/ | /b/ | /p/ | /p/ |
| /c/ | /ch/ | /q/ | /k/ |
| /d/ | /d/ | /r/ | /r/ |
| /E/ | /ah/ | /s/ | /s/ |
| /e/ | /eh/ | /t/ | /t/ |
| /f/ | /f/ | /u/ | /uh/ |
| /g/ | /g/ | /w/ | /w/ |
| /h/ | /hh/ | /y/ | /y/ |
| /i/ | /ih/ | /z/ | /z/ |
| /j/ | /jh/ | /ai/ | /ah ih/ |
| /k/ | /k/ | /kh/ | /k hh/ |
| /l/ | /l/ | /ny/ | /n y/ |
| /m/ | /m/ | /sy/ | /s y/ |
| /n/ | /n/ | | |

Tabel 2.7 Tabel Contoh Representasi Translasi Fonem Pada Beberapa Kata Berbahasa Indonesia (Sumber: Ferdiansyah, 2012)

| Kata Bahasa Indonesia | Bentuk Representasi Fonem | Representasi Fonem Sesudah Translasi |
|---|---|---|
| ABSOLUTISME | /a b s o l u t i s m E/ | /ah b s ow l uh t ih s m ah/ |
| AKHMAD | /a kh m a d/ | /ah k hh m ah d/ |
| ANEKA | /a n e k a/ | /ah n eh k ah/ |
| ANTARANYA | /a n t a r a n y a/ | /ah n t ah r ah n y ah/ |
| BERLANTAI | /b E r l a n t a i/ | /b ah r l ah n t ah ih/ |
| BERMASYARAKAT | /b E r m a sy a r a k a t/ | /b ah r m ah s y ah r ah k ah t/ |

## 2.4 Finite Automata

### 2.4.1 Definisi Finite Automata

*Finite Automata* adalah suatu pemodelan yang dapat digunakan terhadap berbagai *hardware* ataupun *software* yang terdiri dari sejumlah *state* hingga (*finite state*) sehingga dapat diimplementasikannya suatu sistem dengan sekumpulan *resource* yang terbatas. *Finite Automata* melibatkan sejumlah *state* serta transisi antar *state* sebagai bentuk respon terhadap input yang diberikan. (Hopcroft, 2007). *Finite Automata* dapat juga disebut sebagai *finite-state machine* yang dapat diartikan sebagai suatu mesin komputasi abstrak dan merupakan suatu teknik komputasi yang sangat penting dalam pemrosesan pengucapan bahasa (*spoken language processing*) (Coleman, 2005).

### 2.4.2 Deterministic Finite Automata dan Nondeterministic Finite Automata

Secara garis besar, *Finite Automata* terbagi menjadi 2 jenis, yaitu *Deterministic Finite Automata* (DFA) dan *Nondeterministic Finite Automata* (NFA) (Mozgovoy, 2010). Berikut adalah penjelasan dari kedua jenis *Finite Automata* tersebut.

1. *Deterministic Finite Automata*

   *Deterministic Finite Automata* (DFA) adalah suatu representasi *Finite Automata* dimana sistem hanya dapat berada pada suatu *state* tunggal setelah diberikan rangkaian input. Istilah ”*deterministic*” sendiri mengacu pada fakta bahwa untuk setiap input, hanya terdapat satu *state* dimana sistem dapat bertransisi dari *state*nya sekarang. Notasi dari DFA (disebut dengan *five tuple notation*) adalah sebagai berikut.

   $$A = (Q, \Sigma, \delta, q_0, F) \quad \text{.....(2.1)}$$

   Dengan masing – masing penjelasan setiap simbol adalah sebagai berikut.

   Q = kumpulan *state* berhingga (*finite state*)

   $\Sigma$ = kumpulan simbol – simbol input berhingga

   $\delta$ = fungsi transisi

   $q_0$ = *state* awal (*start state*)

   F = *state* akhir (*final state*)

   Suatu *string* dikatakan di”terima” oleh DFA apabila setelah ditelusuri, *string* tersebut habis terbaca dan DFA berhenti di *state* akhir (*final state*).

Istilah *language* digunakan untuk menunjuk kumpulan *string* yang dapat diterima oleh DFA.

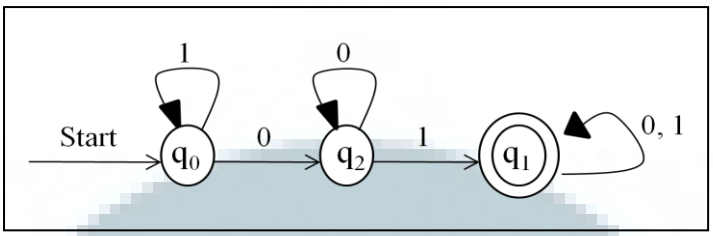
Terdapat dua representasi lain yang dapat digunakan untuk mendefinisikan suatu DFA, yaitu diagram transisi *(transition diagrams*) dan tabel transisi (*transition table*).

a. Diagram transisi

   Adalah suatu *graph* yang menggambarkan notasi *five tuple notation*, dengan rincian sebagai berikut.

   1) Untuk setiap *state* di Q direpresentasikan dengan sebuah *node.*
   2) Untuk setiap *state* q di Q dan simbol input a dalam Σ, sehingga terdapat fungsi transisi δ(q, *a*) = p, maka diagram transisi akan memiliki garis penghubung/*arc* dari *node* q ke *node* p dengan label a. Jika terdapat beberapa simbol input yang menyebabkan terjadinya transisi dari q ke p, maka diagram transisi dapat digambarkan memiliki satu *arc* dengan labelnya adalah simbol – simbol input tersebut.
   3) Ada suatu anak panah ke *state* awal $q_0$ yang diberi label *Start*. Anak panah ini tidak berasal dari *node* manapun.
   4) *Node* – *node* yang berada pada *state* penerima (yang terdapat dalam *state* akhir/F) ditandai dengan lingkaran ganda. *State* yang tidak terdapat dalam F ditandai dengan lingkaran tunggal.

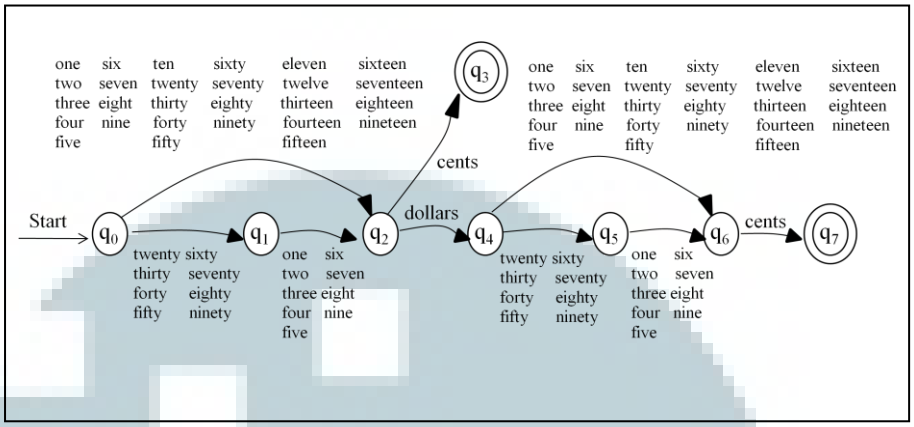
   Berikut adalah ilustrasi sederhana DFA yang menerima semua *string* dengan *substring* 01.




Gambar 2.4 Diagram Transisi DFA Penerima *String* Dengan *Substring* 01 (Sumber: Hopcroft, 2007)

Pada diagram tersebut, dapat dilihat bahwa untuk mencapai *state* akhir, dibutuhkan dua *substring* mutlak, yaitu "01", karena hanya input '0' yang akan menyebabkan transisi dari $q_0$ ke $q_2$, dan hanya input '1' yang akan menyebabkan transisi dari $q_2$ ke $q_1$ sebagai *state* akhir (dengan asumsi simbol input adalah 0 dan 1). Di *state* akhir sendiri ($q_1$), *string* akan tetap diterima apapun inputnya, karena di *state* tersebut, tujuan dari DFA telah tercapai, yaitu menerima *string* dengan *substring* 01.

DFA juga dapat digunakan sebagai pemodelan kombinasi kata – kata. Berikut adalah contoh pemodelan DFA untuk kata – kata penyusun nilai mata uang Inggris.




Gambar 2.5 DFA Untuk Kata – Kata Penyusun Nilai Mata Uang Inggris (Sumber: Jurafsky, 2009)

Diagram DFA tersebut menggambarkan kombinasi kata penyusun mata uang Inggris yang dibatasi pada *dollar* dan *cents*. Dapat dilihat bahwa pengucapan mata uang dapat berupa sekian *cent* saja atau sekian *dollars* sekian *cent*. Pecahan nilai mata uangnya sendiri dibatasi antara satu sampai dengan 99 dan digambarkan pada rangkaian *state* $q_0$ sampai dengan $q_2$ dan $q_4$ sampai dengan $q_6$. Transisi langsung antara $q_0$ ke $q_2$ dan antara $q_4$ ke $q_6$ berfungsi untuk menerima pecahan satu sampai 20 dan puluhan bulat (30, 40, 50, dan seterusnya hingga 90). Transisi $q_0$-$q_1$-$q_2$ dan $q_4$-$q_5$-$q_6$ berfungsi untuk menerima pecahan puluhan tidak bulat (21, 22, 23, 24, dan seterusnya hingga 99).

b. Tabel transisi

Tabel transisi adalah representasi tabular dari fungsi transisi δ yang membutuhkan dua argumen dan mengembalikan suatu nilai. Baris tabel menggambarkan *state* dan kolom tabel menggambarkan inputnya. Nilai isi dari baris *state* q dan kolom input a adalah *state*

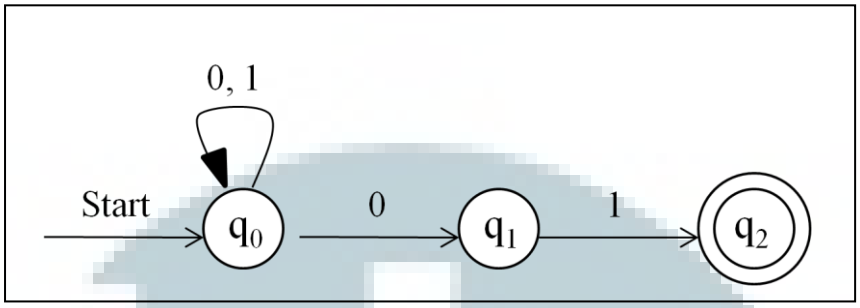
hasil dari δ(q, a). Berikut adalah contoh tabel transisi dari diagram transisi yang terdapat pada gambar 2.4.

Tabel 2.8 Tabel Transisi DFA Penerima *String* Dengan *Substring* "01" (Sumber: Hopcroft, 2007)

| | 0 | 1 |
|---|---|---|
| → $q_0$ | $q_2$ | $q_0$ |
| *$q_1$ | $q_1$ | $q_1$ |
| $q_2$ | $q_2$ | $q_1$ |

2. *Nondeterministic Finite Automata*

*Nondeterministic Finite Automata* (NFA) adalah representasi lain dari *Finite Automata* dimana suatu sistem mampu berada di beberapa *state* dalam satu waktu. Kemampuan ini seringkali diungkapkan sebagai kemampuan untuk "mengira – ngira" input yang diberikan. Sebagai contoh, ketika digunakan untuk mencari sekumpulan rangkaian karakter (misalnya: suatu kata kunci/*keyword*) dalam suatu *string* yang panjang, dapat dilakukan "perkiraan" bahwa kita sedang berada di awal salah satu huruf penyusun *string* tersebut dan menggunakan serangkaian *state* untuk melakukan pengecekan kemunculan *string* secara per karakter. Notasi yang digunakan untuk merepresentasikan NFA adalah *five tuple notation* begitu pula dengan diagram transisi dan tabel transisinya. Berikut adalah contoh diagram transisi dan tabel transisi dari NFA yang menerima *string* berakhiran "01".




Gambar 2.6 Diagram Transisi NFA Penerima *String* Berakhiran 01 (Sumber: Hopcroft, 2007)

Pada diagram tersebut dapat dilihat bahwa di *state* awal inputan 0 dapat menyebabkan transisi dari $q_0$ kembali ke $q_0$ ataupun ke $q_1$ sehingga tidak dapat dipastikan. Akan tetapi, dapat dipastikan bahwa agar suatu *string* diterima, maka 2 *substring* terakhir dari *string* tersebut haruslah 0 dan 1, terlepas dari sekian kombinasi *substring* sebelumnya.

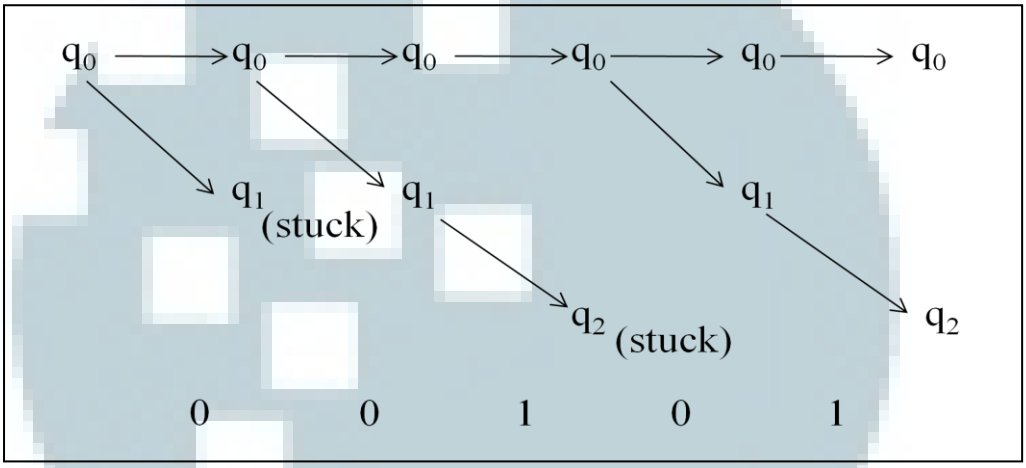
Tabel 2.8 Tabel Transisi NFA Penerima *String* Berakhiran "01" (Sumber: Hopcroft, 2007)

| | 0 | 1 |
|---|---|---|
| → $q_0$ | $\{q_0, q_1\}$ | $\{q_0\}$ |
| $q_1$ | Ø | $\{q_2\}$ |
| *$q_2$ | Ø | Ø |

Pembangunan tabel transisi dari diagram transisi NFA sama dengan DFA, dimana baris tabel menggambarkan *state* dan kolom tabel menggambarkan inputnya. Nilai isi dari baris *state* q dan kolom input a adalah *state* hasil dari δ(q, a). Pada NFA, karena *state* hasil transisi bisa saja lebih dari satu,

maka *state* – *state* tersebut ditulis dalam kurung kurawal {}. Tanda Ø menggambarkan tidak ada *state* hasil transisi dari δ(q, a).

Proses pembacaan *string* dalam NFA pada gambar 2.6 (dengan contoh input *string* "00101") digambarkan dalam skema berikut.




Gambar 2.7 Skema Proses Pembacaan *String* "00101" Oleh NFA
Sumber: (Hopcroft, 2007)

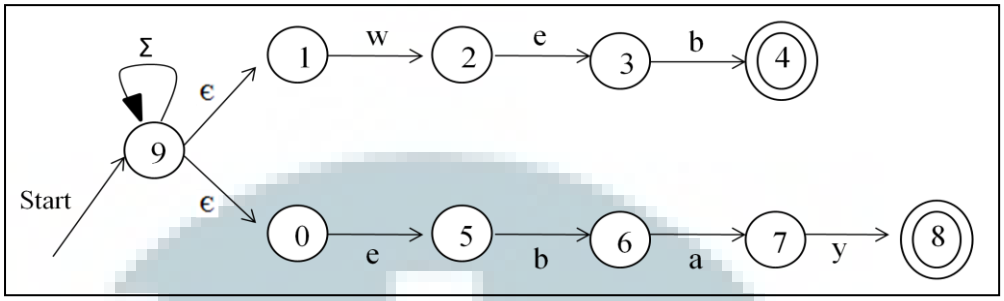
- Saat '0' pertama dibaca, NFA dapat berpindah ke *state* $q_0$ ataupun $q_1$, sehingga perpindahan akan terjadi ke kedua *state* tersebut.
- Lalu '0' kedua akan dibaca. *State* $q_0$ dapat berpindah kembali ke *state* $q_0$ ataupun $q_1$. Di sisi lain, *state* $q_1$ tidak memiliki transisi untuk '0', sehingga proses terhenti/"*dies*"
- Input ketiga, yaitu '1' kemudian dibaca. *State* $q_0$ hanya akan berpindah ke *state* $q_0$, sedangkan *state* $q_1$ hanya akan berpindah ke *state* $q_2$. Pada proses ini (setelah pembacaan "001"), NFA akan berada pada *state* $q_0$ dan $q_2$. Karena *state* $q_2$ adalah *state* akhir, maka NFA akan menerima *string* "001".

- Tetapi proses pembacaan *string* tetap berjalan karena inputnya belum selesai dibaca. Input keempat, yaitu '0' menyebabkan *state* $q_2$ "*dies*", sedangkan *state* $q_0$ berpindah ke *state* $q_0$ dan $q_1$.
- Input terakhir, yaitu '1' menyebabkan perpindahan dari $q_0$ ke $q_0$ dan $q_1$ ke $q_2$. Karena *state* $q_2$ merupakan *state* akhir dan input "00101" telah selesai dibaca, maka dapat disimpulkan *string* "00101" diterima.

### 2.4.3 Finite Automata Dengan Transisi Epsilon

Merupakan pemodelan *Finite Automata* yang memungkinkan adanya input berupa *string* kosong/*epsilon* ($\epsilon$). Dengan input ini, suatu *Finite Automata* (misalnya NFA) dapat melakukan transisi secara langsung tanpa menerima input apapun. Pemodelan ini tidak mengubah kemampuan suatu *Finite Automata* dalam menerima *string*, tetapi sangat membantu dalam kemudahan pemrograman. NFA yang mampu menerima *string* kosong/*epsilon* ini sering disebut sebagai $\epsilon$ – NFA. Walaupun berlaku sebagai simbol input, dalam notasi *five tuple*, $\epsilon$ tidak termasuk ke dalam $\Sigma$.

Berikut adalah contoh diagram transisi dari suatu $\epsilon$ – NFA yang digunakan untuk mencari kata kunci "web" dan "ebay"




Gambar 2.8 Diagram $\epsilon$- NFA Untuk Mencari Kata Kunci "web" dan "ebay"
(Sumber: Hopcroft, 2007)

Pada gambar tersebut, terlihat dari *state* awal, NFA dapat langsung bergerak ke *state* 1 ataupun *state* 0. Karena *string* $\epsilon$ adalah suatu *string* kosong, maka dapat diasumsikan *state* awal NFA berada pada *state* 1 dan *state* 0. Keberadaan *string* sendiri tidak mengurangi atau menambahkan fungsi NFA di atas untuk mencari kata kunci "web" dan "ebay".

### 2.4.4 Penerapan Finite Automata

Beberapa penerapan *Finite Automata*, khususnya dalam bidang *sofware* antara lain adalah sebagai berikut (Hopcroft, 2007).

1. *Software* untuk merancang dan melakukan pengecekan terhadap perilaku (*behavior*) sirkuit digital.
2. *Software compiler* yang berfungsi sebagai "*lexical analyzer*", yang memisahkan teks input ke dalam unit – unit logikal, seperti *identifiers*, kata kunci/*keywords*, dan tanda baca.

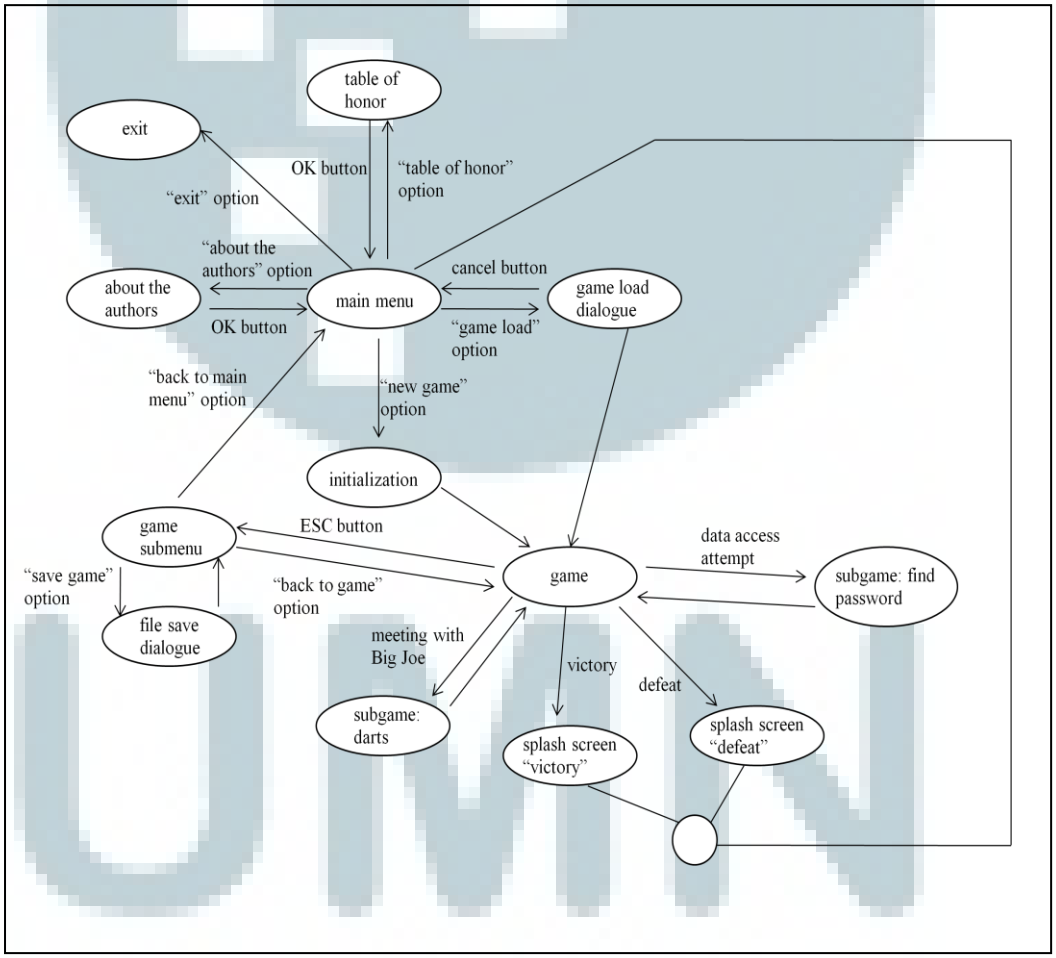
3. *Software* untuk melakukan *scanning* terhadap teks dalam jumlah banyak (misalnya halaman *Web* untuk mencari kata, frasa, ataupun kumpulan kata dengan pola – pola tertentu).
4. *Software* untuk melakukan verifikasi terhadap segala jenis sistem yang memiliki sejumlah *state* berbeda, seperti protokol komunikasi atau protokol keamanan pertukaran data.

Terdapat 2 notasi penting yang juga memiliki peranan penting terhadap penerapan *Finite Automata*.

1. *Grammars*, yang merupakan pemodelan penting dalam merancang *software* yang memroses data yang memiliki struktur rekursif, misalnya ”*parser*” yang merupakan komponen *compiler* yang menangani struktur pemrograman yang bersifat rekursif, misalnya ekspresi aritmatik, kondisional, dan sebagainya. Sebagai contohnya adalah aturan E => E + E, dimana aturan tersebut dapat dibentuk dengan mengambil dua ekspresi apapun dan menghubungannya dengan tanda “+”.
2. *Regular Expressions* yang juga mendifinisikan struktur data, terutama teks/*string*. Contoh *Regular Expression* adalah UNIX *Style Regular Expression* dalam bentuk ‘[A-Z][a-z]*[ ][A-Z][A-Z]’ yang merepresentasikan suatu kata yang diawali huruf kapital dengan diikuti dengan spasi dan dua huruf kapital, misalnya nama suatu kota diikuti dengan negara (contoh: Ithaca NY).

*Finite Automata* dalam bidang pemrograman, juga dapat digunakan sebagai suatu pemodelan dalam struktur pemrograman. Setiap *state* tidak hanya

berfungsi sebagai penanda keadaan sistem sekarang, tetapi juga berfungsi untuk menjalankan sejumlah operasi tertentu sesuai dengan *state* yang bersangkutan. Contoh penerapan *Finite Automata* dalam hal ini adalah untuk memodelkan suatu permainan *game* yang paling tidak memiliki beberapa *state* dasar, seperti memulai permainan (*start*), me*load* permainan (*load game*), menyimpan permainan (*save game*), dan *state* yang menandari akhir permainan (baik kemenangan/*victory* ataupun kekalahan/*defeat*). Berikut adalah gambaran penerapan *Finite Automata* dalam memodelkan suatu permainan. (Mozgovoy, 2010).




Gambar 2.9 Deskripsi *Game* Menggunakan *Finite Automata*

(Sumber: Mozgovoy, 2010)

Gambar tersebut memperlihatkan struktur suatu permainan dalam bentuk *Finite Automata*, dimana simbol input direpresentasikan dalam tombol – tombol tertentu (OK atau *Cancel*) dan pilihan – pilihan/*option*. Setiap tombol dan pilihan akan membawa permainan dari suatu *state* ke *state* lain (terjadinya transisi), dimana dalam *state* hasil transisi tersebut akan dijalankan sejumlah operasi tertentu, misalnya menampilkan informasi mengenai pembuat *game* pada *state "about the authors*", menggambar *sprite*, memroses inputan dari *keyboard* dan sebagai di *state* "*game*", menampilkan pilihan utama/*main menu* dan menunggu pilihan dari pemain pada *state* "*main menu*" dan sebagainya. (Mozgovoy, 2010).

## 2.5 Pendiktean dan Kata

Pendiktean, berasal dari kata dasar "dikte" yang menurut Kamus Besar Bahasa Indonesia (2007) berarti "diucapkan atau dibaca keras – keras supaya ditulis orang lain". Bentuk kata kerja dari "dikte" adalah "mendikte" yang berarti "menyuruh orang menulis apa yang dibacakan atau dikatakan" atau "menyuruh berbuat dan menurut saja seperti yang dikatakannya (dengan tidak boleh membantah)".

Sedangkan kata "kata" sendiri berarti "unsur bahasa yang diucapkan atau dituliskan yang merupakan perwujudan kesatuan perasaan dan pikiran yang dapat digunakan dalam berbahasa". Perulangan kata "kata" menjadi "kata – kata" (dalam bentuk jamak) dapat diartikan sebagai "lebih dari satu kata". Berdasar pada definisi – definisi tersebut, maka dapat disimpulkan bahwa "pendiktean kata

– kata" dapat diartikan sebagai "menyuruh menuliskan kata – kata yang diucapkan (dengan pengucapan mengacu pada lebih dari satu kata)".

## 2.6 Word Error Rate (WER)

Pengukuran performa/kinerja dari ASR sendiri biasanya dinyatakan dalam tingkat kesalahan pengenalan kata/*Word Error Rate* (WER). Pengukuran WER dinyatakan dalam rumus berikut.

$$\text{Word Error Rate} = 100 \times \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{Jumlah Kata Sumber yang Sebenarnya}} \quad \text{.....(2.2)}$$

Dengan *insertions* menyatakan jumlah penyisipan, *deletions* menyatakan jumlah penghapusan, dan *substitutions* menyatakan jumlah penggantian. Hasil evaluasi didapatkan dengan menghitung ketiga aspek tersebut dengan membandingkan antara kata – kata yang dihipotesis oleh *engine* ASR dengan kata – kata sebenarnya yang diucapkan. Berikut adalah contoh penghitungan WER seperti dikutip dari Jurafsky (2009).

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REF: | I | *** | ** | UM | the | PHONE | IS | I | LEFT | THE | portable | **** | PHONE | UPSTAIRS | last | night |
| HYP: | I | GOT | IT | TO | the | ***** | * | FULLEST | I | LOVE | TO | portable | FORM | OF | STORES | last | night |
| Eval: | | I | I | S | | D | | S | | S | S | | I | S | S | | |

Gambar 2.10 Contoh Analisis Penghitungan WER
(Sumber: Jurafsky, 2009)

Dari contoh di atas, terlihat bahwa hasil hipotesis kata oleh *engine* ASR menunjukkan enam *substitutions* (S), tiga *insertions* (I), dan satu *deletion* (D)

terhadap kata – kata sumber yang sebenarnya. Dengan demikian, nilai WER nya dapat diperoleh dengan menerapkan rumus (2.2) dengan rincian sebagai berikut.

$$\text{Word Error Rate} = 100\ \frac{6+3+1}{13} = 76.9\ \%$$

Gambar 2.11 Contoh Penghitungan WER

(Sumber: Jurafsky, 2009)