BAB 2
TINJAUAN PUSTAKA

2.1 Ilmu Nahwu

Pada Bahasa Arab, terdapat tiga ilmu yang menjadi dasar untuk dapat mengetahui struktur kalimat serta memahami maknanya. Ilmu *Nahwu* merupakan salah satunya, yang membahas bentuk kata dan kedudukannya saat sebelum menjadi kalimat, dan saat ditempatkan dalam kalimat dengan kedudukannya masing-masing. Ilmu *nahwu* fokus pada bagaimana kita merangkai kata-kata menjadi sebuah kalimat yang sempurna, baik dari susunan kata tersebut atau perubahan akhir setiap kata dalam kalimat yang dikenal dengan istilah *I'rob* [4].

Setiap bahasa memiliki kata dengan jenis tertentu yang menjadi penyusun dalam kalimat, begitupun dengan Bahasa Arab. Pada bahasa lainnya kata yang menjadi penyusun kalimat terdiri dari banyak jenis, seperti dalam bahasa Indonesia terdapat kata kerja, kata benda, kata sifat, kata sambung, dan sebagainya, bahasa Inggris terdapat *verb*, *noun*, *adjective*, *adverb*, *determiner*, dan lain sebagainya. Pada bahasa Arab dikelompokkan menjadi tiga kelompok besar, yaitu:

1. *Fi'il* (الفِعْلُ)
2. *Isim* (الإسْمُ)
3. *Harf* (الحَرْفُ)

Meskipun terbagi menjadi tiga, syarat minimal terbentuknya kalimat dalam bahasa arab tidak harus memenuhi ketiga kata penyusun tersebut, melainkan dapat tersusun atas dua buah *isim*, atau tersusun atas *isim* dan *fi'il* [5], seperti:

قَامَ زَيْدٌ	**اَلْقَلَمُ مَكْسُوْرٌ**
الفِعْلُ +الإسْمُ	**الإسْمُ +الإسْمُ**
Zaid telah berdiri	**Pulpen itu Patah**

2.1.1 Fi'il (الفِعْلُ)

Fi'il secara singkat dapat didefinisikan dengan kata yang mengandung makna kerja, akan tetapi didefinisikan sebagai kata yang memiliki makna dan terikat oleh waktu, diantaranya: *madhi* (lampau), *haal* (sedang berlangsung), dan *istiqbal* (yang akan datang). Tanda-tanda *fi'il* diantaranya:

1. Didahului oleh huruf *qad* (قَدْ)
2. Dilekatkan dengan *sin* (سَ)
3. Didahului oleh *Saufa* (سَوْفَ)
4. Diakhiri oleh *Ta Ta'nits* disukunkan (تْ) [5]

Fi'il sebagai kata yang mengandung makna kerja, secara waktu perbuatannya dibagi menjadi tiga, diantaranya:

1. *Fi'il Madhi* (الْفِعْلُ الْمَاضِيْ)

   Fi'il Madhi adalah kata kerja pada masa lampau yang memiliki arti telah melakukan atau perbuatan yang dilakukan telah terjadi, contohnya: كَتَبَ (telah menulis) atau قَرَأَ (telah membaca).
2. *Fi'il Mudhorie'* (الْفِعْلُ الْمُضَارِعُ)

   Fi'il Mudhorie' memiliki arti sedang atau akan terjadinya suatu perbuatan. Contohnya: يَكْتُبُ (sedang menulis), atau يَقْرَأُ (sedang membaca).
3. *Fi'il Amr* (فِعْلُ الأَمْرِ)

   Fi'il Amr adalah kata perintah. Contohnya: اُكْتُبْ (tulislah!), atau اقْرَأْ (bacalah!)

   Pembagian fi'il menjadi tiga diatas serupa dengan tata Bahasa Inggris yaitu: *past tense* untuk *fi'il madhi*, *present continuous tense* dan *future tense* untuk *fi'il mudhorie'*. [4]

Pada kata *fi'il*, terdapat kata yang mengharuskan adanya objek, dan terdapat kata yang tidak mengharuskan adanya objek. Pembagian ini dalam Bahasa Arab dikenal dengan istilah:

1. *Fi'il Lazim* (الْفِعْلُ اللاَّزِمُ)

   Fi'il yang tidak membutuhkan objek karena kalimat dianggap sudah sempurna tanpa adanya objek, seperti:

جَلَسْتُ عَلَ الْكُرْسِيُّ Saya telah duduk di atas kursi

قَمْتُ فِيْ الْمَسْجِدِ Saya telah berdiri di dalam masjid

Kalimat diatas tidak harus menggunakan objek sudah cukup menjadikan kalimat tersebut sempurna.

2. *Fi'il Muta'addiy* (**الْفِعْلُ الْمُتَعَدِّيْ**)

Fi'il yang membutuhkan objek dikarenakan kalimat tidak dapat difahami secara benar jika tidak disertai dengan objek, seperti:

كَتَبَتْ الرِّسَالَةَ Saya telah menulis surat → Kata menulis membutuhkan objek

أَكَلْتُ السَّمَكَ Saya telah memakan ikan → Kata memakan membutuhkan objek

Cara mengetahui perbedaan *fi'il* ini dilakukan dengan penalaran makna dari kalimat yang diucapkan. Jika kalimat sudah sempurna tanpa objek maka akan tergolong *fi'il lazim*, sebaliknya jika tidak sempurna maka akan tergolong sebagai *fi'il muta'addy*. [4]

Selain itu, terdapat *fi'il* yang dapat diukur dari perubahannya yaitu aktif dan pasif, yaitu:

1. *Fi'il Ma'lum* (**الْفِعْلُ الْمَعْلُمُ**)

Fi'il Ma'lum merupakan kata kerja aktif, seperti:

ضَرَبَ زَيْدٌ بَكْرً Zaid telah memukul Bakr

2. *Fi'il Majhul* (**الْفِعْلُ الْمَجْهُوْلُ**)

Fi'il Majhul merupakan kata kerja pasif, seperti:

ضُرِبَ بَكْرٌ Bakr telah dipukul

Fi'il Majhul dalam kaidah Bahasa Arab tidak boleh menggunakan subjek, dengan alasan bahwa pelakunya sudah diketahui; pelakunya memang tidak diketahui; atau pelakunya sengaja disembunyikan. [4]

Jika dilihat dari huruf penyusunnya, *fi'il* dibagi menjadi dua, yaitu:

1. *Fi'il Shahih* (**الْفِعْلُ الصَّحِيْحُ**)

   Fi'il shahih merupakan *fi'il* dimana tidak ada hurufnya yang tersusun dari huruf '*illat*, yaitu *alif* (**ى** atau **ا**), *waw* (**و**), dan *ya* (**ي**).

2. *Fi'il Mu'tal* (**الْفِعْلُ الْمُعْتَلُّ**)

   Fi'il mu'tal merupakan *fi'il* yang diantara huruf penyusunnya terdapat huruf illat, yaitu *alif* (**ى** atau **ا**), *waw* (**و**), dan *ya* (**ي**).

Tabel 2.1 Contoh Fi'il Shahih dan Fi'il Mu'tal

Fi'il Shahih	Fi'il Mu'tal
كَتَبَ	**خَشِيَ**
أَكَلَ	**قَالَ**
سَأَلَ	**وَقَى**
ضَرَبَ	**وَجَدَ**

Pada tabel diatas terlihat bahwa terdapat perbedaan antara *fi'il* shahih dan *fi'il* mu'tal. Dimana pada *fi'il* shahih memiliki huruf penyusunnya selain huruf *alif*, *waw*, dan *ya*. Sedangkan pada *fi'il mu'tal* terlihat terdapat huruf *alif* (**ى** atau **ا**) pada **وَقَى** dan pada **قَالَ**, huruf *waw* (**و**) pada **وَجَدَ**, serta *ya* (**ي**) pada **خَشِيَ**. [4]

2.1.2 Isim (**الإِسْمُ**)

Jika *Fi'il* bergantung pada waktu, maka *isim* merupakan kata yang tidak terikat pada waktu. *Isim* dapat diketahui melalui:

1. *Isnad* (**اِسْنَادْ**) atau *Musnad Ilaih* (**مُسْنَادْ اِلَيْهِ**), yaitu *fa'il* (**فَاعِلٌ**), *mubtada* (**مُبْتَدَأُ**), atau *naibul fa'il* (**نَائِبُ الفَاعِلُ**)
2. Didahului oleh *Harf Khafadh*/*Jar*
3. Ber*tanwin* (ٌ,ٍ,ً)
4. Dilekatkan dengan *Alif-lam* (**ال**)

Isim sendiri dapat ditinjau berdasarkan jenisnya, menjadi *Isim Mudzakkar* (**الإِسْمُ المُذَكَّرُ**) yang secara bahasa memiliki arti laki-laki dimana makna laki-laki bukan secara *dzahir*, dan *Isim Muannats* (**الإِسْمُ المُؤَنَّثُ**) yang secara bahasa memiliki arti perempuan yang bukan secara makna *dzahir*. *Isim* Muannats biasanya

bercirikan mengandung huruf ta marbuthah (ة) seperti , sekolah (مَدْرَسَةُ), Fatimah (فَاطِمَةُ) atau nama perempuan lainnya, sedangkan Mudzakkar tidak, seperti buku (كِتَابٌ), ahmad (أَحْمَدٌ) dan nama laki-laki lainnya. [4]

Isim juga dapat dibagi menjadi tiga jika ditinjau berdasarkan jumlah, artinya berdasarkan berapa banyak kata benda yang sedang dibahas, berikut diantaranya:

1. *Isim Mufrad* (الْإِسْمُ الْمُفْرَدُ)

   Isim Mufrad bermakna tunggal, artinya untuk mengungkapkan *isim* yang jumlahnya tunggal. Contohnya adalah: Masjid (مَسْجِدٌ), Muhammad (مُحَمَّدٌ), Buku (كِتَابٌ).

2. *Isim Mutsanna* (الإِسْمُ المُثَنَّ)

   Isim Mutsanna digunakan ketika ingin mengucapkan dua *isim* dalam satu kata secara sekaligus. Contohnya: Dua orang muslim, dua orang Muslimah (مُسْلِمَتَانِ، مُسْلِمَنِ) atau dua buah pulpen, dua buah buku (قَلَمَنِ، كِتَبَنِ), dan lainnya.

3. *Isim Jamak* (الإِسْمُ الجَمْعُ)

   Isim Jamak merupakan *isim* yang digunakan untuk menggantikan makna *isim* dengan jumlah tiga atau lebih dalam satu kata. *Jamak taksir* ada tiga jenis, yaitu:

   1) *Jamak Mudzakkar salim* (جَمْعُ المُذَكَّرِ سَلِمٍ)

      Jamak bagi *isim* yang laki-laki. Contohnya: Para lelaki muslim (مُسْلِمُوْنَ atau مُسْلِمِيْنَ)

   2) *Jamak Muannats Salim* (جَمْعُ المُؤَنَّثِ سَلِمٍ)

      Jamak bagi *isim* yang Perempuan. Contohnya: Para perempuan muslim (مُسْلِمَاتٌ)

   3) *Jamak Taksir* (جَمْعُ التَّكْسِيْرِ)

      Jamak taksir adalah jamak yang tidak memiliki aturan tersendiri, sehingga perubahan katanya berbeda dari pada umumnya. Biasanya digunakan untuk kata benda mati atau tidak berakal (لِغَيْرِالعَاقِلِ) seperti pulpen (قَلَمٌ), buku (كِتَابٌ), pintu (بَابٌ), dan sebagainya, atau kata untuk menunjuk kepada yang berakal (لِالعَاقِلِ) seperti laki-laki (رَجُلٌ - رِجَالٌ),

nabi (**نَبِيٌّ - أَنْبِيَاءُ**), rasul (**رَسُوْلٌ - رُسُلٌ**), orang kaya (**غَنِيٌّ - أَغْنِيَاءُ**) dan lainnya. [4]

Isim ditinjau dari keumumannya (**الإِسْمُ النَّكِرَةُ**) dan kekhususannya (**الإِسْمُ المَعْرِفَةُ**) menjadi sebagai berikut:

1. *Isim Nakiroh* (**الإِسْمُ النَّكِرَةُ**)

   Isim nakiroh adalah *isim* yang digunakan ketika obyek yang dibicarakan tidak ditentukan alias umum, artinya mencakup semua kriteria yang masuk dalam pembicaraan. Contohnya: **مَسْجِدٌ ، بَابٌ، كِتَابٌ**

2. *Isim Ma'rifat* (**الإِسْمُ المَعْرِفَةُ**)

   Isim Ma'rifat adalah *isim* yang digunakan ketika obyek yang dibicarakan jelas atau khusus kepada objek tertentu dan diketahui siapa atau apa objek tersebut. *Isim* ini terbagi menjadi 6, diantaranya:

   1) *Dhamir*, yaitu: **هُوَ، هُمَ، هُمْ، هِيَ، هُمَ، هُنَّ، أَنْتَ، أَنْتُمَا، أَنْتُمْ، أَنْتِ، أَنْتُمَا، أَنْتُنَّ، أَنَا، نَحْنُ**

   2) '*Alam*, seperti: **زَيْدٌ، مُحَمَّدٌ، أَحْمَدٌ**

   3) *Isim Isyaroh*, seperti:

Tabel 2.2 Isim Isyaroh

إِسْمُ الإِشَارَةِ لِلْقَرِيْبِ (Kata tunjuk untuk yang dekat)		
العَدَدُ	**المُذَكَّرُ**	**المُؤَنَّثُ**
المُفْرَدُ	**هَذَا**	**هَذِهِ**
المُثَنَّ	**هَذَانِ/ هَذَيْنِ**	**هَتَانِ/ هَتَيْنِ**
الجَمْعُ	**هَؤُلآءِ**	**هَؤُلآءِ**
إِسْمُ الإِشَارَةِ لِلْبَعِيْدِ (Kata tunjuk untuk yang jauh)		
المُفْرَدُ	**ذَلِكَ**	**تِلْكَ**
المُثَنَّ	**ذَانِكَ/ ذَيْنِكَ**	**تَانِكَ/ تَيْنِكَ**
الجَمْعُ	**أُولَئِكَ**	**أُولَئِكَ**

   4) *Isim* yang dilekati alif lam (**ال**), seperti:

   Semua *isim* dalam bahasa arab yang dilekatkan dengan **ال**, seperti: **المَسْجِدُ، القَلَمُ، المَكَّةُ**

5) *Isim Maushul*

Isim maushul adalah kata sambung. *Isim maushul* dibagi dua, yaitu: *Isim maushul* yang umum, yaitu **مَنْ، مَا** dan *isim maushul* yang khusus, diantaranya

الَّذِيْ، الَّذَانِ/ الَّذَيْنِ، الَّتِيْ، الَّتَانِ/ الَّتَيْنِ، الَّاتِيْ/ الَّاتِ/ الَّائِيْ

6) *Isim* yang di-*idhofah*-kan

Isim Idhofah adalah *isim* yang disandarkan pada *isim* lainnya, terbagi menjadi *Idhofah* terhadap *Dhamir*, yaitu **كِتَابُهُ، كِتَابُكَ** ;Idhofah terhadap *Isim 'Alam*, yaitu **كِتَابُ زَيْدٌ** ; *Idhofah* terhadap *Isim Isyarah*, yaitu **أُمُّ هَذِهِ الْمَرْأَةِ** ; *Idhofah* terhadap *Isim Maushul*, yaitu **كِتَابُ الَّذِيْ يَقُوْمُ جَدِيْدٌ** ; *Idhofah* terhadap *isim* yang dilekati *Al*, yaitu **كِتَابُ اللُّغَةِ**. [4]

Isim juga dapat ditinjau dari keberadaan *tanwin* (**الإِسْمُ الْمُنْصَرِفُ**), dan tidak adanya tanwin (**الإِسْمُ غَيْرُامُنْصَرِفِ**). Adapun contoh dari **الإِسْمُ الْمُنْصَرِفُ** adalah sebagai berikut:**مُحَمَّدٌ، بَابٌ، مَسْجِدٌ** . Sedangkan **الإِسْمُ غَيْرُامُنْصَرِفِ** dibagi lagi menjadi beberapa kelompok, diantaranya: Seluruh nama wanita, seperti ; seluruh nama laki-laki yang diakhiri *ta marbuthah* (**ة**), seperti **مُعَاوِيَةُ، أُسَمَةُ، مُيَسَرَةُ** ; Nama yang bukan berasal dari arab atau '*ajam* (**عَجَمُ**), seperti **إِبْرٰهِيْمُ، عِيْسَى،سُلَيْمَنُ**; lalu nama yang berakhiran *alif* dan *nun*, seperti **عَدْنَانُ، عُثْمَانُ، مَرْوَانُ** ; *isim* yang mengikuti pola-pola tertentu seperti berikut:

Tabel 2.3 Isim Maushul dengan pola khusus

No.	Wazan	Pola	Contoh
1	Yuf'il	**يُفْعِلُ**	**يَزِيْدُ**
2	Fu'ala	**فُعَلُ**	**عُمَرُ، زُحَلُ**
3	Fa'lani	**فَعْلَنُ**	**جَوْعَنُ، غَضْبَانُ**
4	Af'al	**أَفْعَلُ**	**أَحْمَدُ، أَخْضَرُ، أَبْيَضُ**
5	Shighot Muntahal Jumu'	**أَفَاعِيْلُ،فَوَاعِلُ ،مَفَاعِلُ**	**أَنَاشِيْدُ، رَسَائِلُ، مَدَارِسُ**

Serta *isim* yang diakhiri *alif ta'nits* maqshurah (**ى**) seperti haus (**عَطْشَى**), lapar (**جَوْعَى**), dan peringatan (**ذِكْرَى**) atau *alif ta'nits mamdudah* (**اء**) seperti hitam (**سَوْدَاءُ**), para penyair (**شُعَرَاءُ**), dan merah (**حَمْرَاءُ**). [4]

2.1.3 Harf (الحَرْفُ)

Harf ialah lafadz yang tidak boleh dan tidak bias memiliki tanda *isim* maupun tanda *fi'il*. Huruf dibagi menjadi beberapa bagian, namun dalam penelitian ini digunakan tiga yang utama:

1. *Harfu Jar* atau *Harfu Khafadh* (حَرْفُ الْجَرِّ atau حَرْفُ الخَفْضُ)
   مِنْ، إِلَى، حَتَّى، خَلاَ، حَاشَا، عَدَا، فِي، عَنْ، عَلَى، مُذْ، مُنْذُ، رُبَّ، ل، كَيْ، ك، ب، لَعَلَّ، مَتَى
2. *Harfu Nashab* (حَرْفُ النَّصْبِ)
   أَنْ، لَنْ، إِذَنْ،كَيْ، لَامُ الْجُحُوْدِ / لِـ، حَتَّى، فَـ
3. *Harfu Jazm* (حَرْفُ الْجَزْمِ)
   لَمْ، لَمَّا، أَلَمْ، أَلَمَّا، لَا، لَامُ الأَمْرِ، إِنْ، مَتَى، مَنْ، مَا

2.1.4 Kalimat (الجُمْلَةُ)

Struktur kalimat inti bahasa arab cukup terdiri dari dua buah kata, struktur kalimat dalam bahasa arab disebut *jumlah* yang terbagi menjadi dua, yaitu:

1. Jumlah Ismiyyah (الجُمْلَةُ الإِسْمِيَّةُ)

   Jumlah ismiyyah adalah kalimat yang didahului oleh *isim.* Pada umumnya pola kalimat *jumlah ismiyyah* adalah seperti berikut:

   الإِسْمُ (*Mubtada*) + (*Khabar*) الإِسْمُ

   Isim pertama disebut dengan *mubtada*, *isim* kedua disebut *khabar*. *Mubtada* adalah kata/ objek dalam bentuk *isim* yang ingin dijelaskan sedangkan *khabar* sesuai dengan namanya adalah kabar atau penjelasan dari kondisi, keadaan, jabatan, atau penjelasan dalam bentuk apapun dari objek yang sedang dijelaskan (*mubtada*). [4]

   Seperti contoh berikut:

   زَيْدٌ مُسْلِمٌ

   Zaid adalah seorang muslim

   Pada *jumlah ismiyyah*, terdapat 3 kaidah dasar agar dapat menjadi kalimat yang sesuai. Diantaranya [4]:

   1) Mubtada dan Khabar harus rofa'

      Mubtada dan *khabar* keduanya harus dalam keadaan *rofa'*. Berikut kaidah *rofa'* yang perlu diperhatikan:

Tabel 2.4 Kaidah Jumlah Ismiyyah 1

Jumlah	Keadaan Marfu'	Contoh
Mufrod	Dhammah	طَالِبٌ
Mutsanna	Alif	طَالِبَانِ
Jamak Mudzakkar Salim	Waw	طَالِبُوْنَ
Jamak Muannats Salim	Dhammah	طَالِبَاتٌ
Jamak Taksir	Dhammah	طُلَّابٌ
Asmaul Khomsah	Waw	أَبُوْكَ

2) *Mubtada* harus *isim ma'rifat*

Mubtada asal hukumnya harus *isim ma'rifat*, sedangkan *khabar* asal hukumnya berupa *isim nakiroh*. Seperti contoh berikut ini:

هَذَاكِتَابٌ

Ini adalah buku

3) *Khabar* harus sama dengan *mubtada* dari sisi jenis dan jumlah

Bila *mubtada mufrod - mudzakkar*, maka *khabar* wajib *mufrod – mudzakkar*. Begitupun bila mubtadanya *mutsanna – muannats*, ataupun *jamak muannats salim*, *khabar*nya harus mengikuti polanya [4]. Seperti contoh berikut:

Tabel 2.5 Kaidah Jumlah Ismiyyah 3

Jumlah	Mudzakkar	Muannats
Mufrod	الطَّالِبُ مُسْلِمٌ	الطَّالِبَةُ مُسْلِمَةٌ
Mutsanna	الطَّالِبَانِ مُسْلِمَانِ	الطَّالِبَتَانِ مُسْلِمَتَانِ
Jamak salim	الطَّالِبُوْنَ مُسْلِمُوْنَ	الطَّالِبَاتُ مُسْلِمَاتُ
Jamak Taksir	الطُّلَّابُ مُسْلِمُوْنَ	

2. Jumlah Fi'liyyah (الجُمْلَةُ الفِعْلِيَّةُ)

Jumlah Fi'liyyah merupakan kalimat yang terdiri dari *Fi'il* dan *isim*, biasanya diawali oleh *fi'il* dalam susunannya [4]. Secara umum, kalimat ini memiliki pola sebagai berikut:

الفِعْلُ + فَاعِلٌ (الإِسْمُ)

Fa'il (subjek) adalah pelaku dari suatu perbuatan, oleh karenanya *fa'il* merupakan *Isim*. Namun, berdasarkan kebutuhannya pada objek, *fi'il* dibagi menjadi *fi'il lazim* (tidak membutuhkan objek) dan *fi'il muta'addiy* (membutuhkan objek) [4], sebagai berikut:

1) Pola Kalimat Fi'il Lazim

Berikut ini merupakan pola dari *fi'il lazim*

الفِعْلُ + فَاعِلٌ

(جَلَسَ زَيْدٌ)

"Zaid telah duduk"

Pada pola diatas, kalimat hanya terdiri dari dua kata yaitu pelaku dan perbuatannya. Pada contoh, kata "**جَلَسَ**" tidak membutuhkan obyek karena kalimat diatas sudah cukup dimengerti dengan dua unsur tersebut saja [4].

Kaidah yang berlaku dalam *Fi'il Lazim* adalah:

(1)*Fi'il* harus sesuai jenisnya dengan fa'il

Bila *fa'il*nya *mudzakkar*, maka *fi'il* wajib *mudzakkar*. Jika *muannats*, maka *fa'il*nya wajib *muannats*. Misalnya :

Tabel 2.6 Kaidah Fi'il Lazim 1

Mudzakkar	Muannats
جَلَسَ أَحْمَدٌ	**جَلَسَتْ فَاطِمَةُ**
يَجْلِسُ أَحْمَدٌ	**تَجْلِسُ فَاطِمَةُ**

(2)*Fi'il* harus dalam bentuk *mufrod*

Fi'il akan selalu berbentuk *mufrod*, meskipun *fa'il*nya dalam bentuk *mufrod*, *mutsanna*, maupun *jamak*.

Tabel 2.7 Kaidah Fi'il Lazim 2

Mudzakkar	Muannats	'Adad
جَلَسَ المُسْلِمُ	**جَلَسَتْ المُسْلِمَةُ**	Mufrad
جَلَسَ المُسْلِمَانِ	**جَلَسَتْ المُسْلِمَتَانِ**	Mutsanna
جَلَسَ المُسْلِمُوْنَ	**جَلَسَتْ المُسْلِمَاتُ**	Jamak

(3)*Fa'il* harus dalam keadaan *rofa'* (*marfu'*)

Berikut kaidah *fi'il* dan *fa'il*, dimana *fa'il* harus selalu dalam kondisi *rofa'*

Tabel 2.8 Kaidah Fi'il Lazim 3

Jenis Kata	Syakl Rofa'	Contoh	Keterangan
Mufrod	Dhammah	ذَهَبَ الطَّالِبُ	
Mutsanna	Alif	ذَهَبَ الطَّالِبَانِ	Bukan الطَّالِبَيْنِ
Jamak Mudzakkar Salim	Waw	ذَهَبَ الطَّالِبُوْنَ	Bukan الطَّالِبِيْنِ
Jamak Muannats Salim	Dhammah	ذَهَبَ الطَّالِبَاتُ	
Jamak Taksir	Dhammah	ذَهَبَ الطُّالاَّبُ	
Asmaul Khomsah	Waw	ذَهَبَ أَبُوْكَ	Bukan أَبِيْكَ / أَبُوْكَ

2) Pola Kalimat Fi'il Muta'addiy

Fi'il muta'addiy adalah fi'il yang membutuhkan objek (*maf'ul bih*). Hal ini disebabkan oleh kurang jelasnya makna yang dikandung oleh sebuah *jumlah fi'liyyah* tersebut apabila tidak dilengkapi dengan objek [4]. Seperti contoh berikut ini:

قَرَأَ زَيْدٌ الْقُرْآنَ

Prediket Subjek Objek

Sehingga pola kalimat *fi'il muta'addiy* adalah sebagai berikut:

الفِعْلُ + فَاعِلٌ + مَفْعُلٌ بِه

Kata "قَرَأَ" yang merupakan kata kerja masa lampau (*fi'il madhi*), sebagai *fa'il* membutuhkan objek untuk memperjelas kondisi

perbuatan sehingga didatangkan kata "**الْقُرْآنَ**" sebagai *maf'ul bih* [4]. Adapun kaidah yang berlaku untuk *fi'il muta'addiy* adalah sebagai berikut:

1. *Fi'il* harus sesuai dengan jenis *fa'il*nya

   Jika *fa'il*nya mudzakkar, maka *fa'ilnya* wajib *mudzakkar*. Sebaliknya jika *fa'ilnya muannats*, maka *fi'ilnya* wajib *muannats*. Contohnya:

   Mudzakkar = **قَرَأَ عَلِيٌّ الْقُرْآنَ**

   Muannats = **قَرَأَتْ فَاطِمَةُ الْقُرْآنَ**

2. *Fi'il* harus dalam bentuk *mufrod*

   Kaidah ini sama seperti pada *fi'il lazim*, dimana *fi'il* akan selalu berbentuk *mufrod*, meskipun *fa'il*nya dalam bentuk *mufrod*, *mutsanna*, maupun *jamak*

Tabel 2.9 Kaidah Fi'il Muta'addiy 1

Mudzakkar	Muannats	'Adad
قَرَأَ الْمُسْلِمُ الْقُرْآنَ	**قَرَأَتْ الْمُسْلِمَةُ الْقُرْآنَ**	Mufrad
قَرَأَ الْمُسْلِمَانِ الْقُرْآنَ	**قَرَأَتْ الْمُسْلِمَتَانِ الْقُرْآنَ**	Mutsanna
قَرَأَ الْمُسْلِمُوْنَ الْقُرْآنَ	**قَرَأَتْ الْمُسْلِمَاتُ الْقُرْآنَ**	Jamak

3. *Fa'il* harus dalam keadaan *rofa'* (*marfu'*)

   Pada kondisi *isim* menjadi fa'il (subjek), makai sim akan menempati mahal *rofa'* . Kondisi *isim* sebagai fa'il biasanya muncul setelah fi'il. Alamat *syakl rofa'* pada dasarnya adalah dhommah, akan tetapi hal ini berubah bergantung jenis *isim*nya. Berikut kaidah *rofa'* untuk *mufrod*, *mutsanna*, dan *jamak*

Tabel 2.10 Kaidah Fi'il Muta'addiy 2

Jenis Kata	Syakl Rofa'	Contoh
Mufrod	Dhammah	قَرَأَ الطَّالِبُ الْقُرْآنَ
Mutsanna	Alif	قَرَأَ الطَّالِبَانِ الْقُرْآنَ
Jamak Mudzakkar Salim	Waw	قَرَأَ الطَّالِبُونَ الْقُرْآنَ
Jamak Muannats Salim	Dhammah	قَرَأَ الطَّالِبَاتُ الْقُرْآنَ
Jamak Taksir	Dhammah	قَرَأَ الطَّلَّابُ الْقُرْآنَ
Asmaul Khomsah	Waw	قَرَأَ أَبُوْكَ الْقُرْآنَ

4. *Maf'ul bih* harus dalam keadaan *nashab* (*manshub*)

   Berikut keadaan beberapa kelompok kata Ketika *manshub*

Tabel 2.11 Kaidah Fi'il Muta'addiy 3

Jenis Kata	Syakl Nashab	Contoh
Mufrod	Dhammah	رَأَيْتُ الطَّالِبُ
Mutsanna	Alif	رَأَيْتُ الطَّالِبَيْنِ
Jamak Mudzakkar Salim	Waw	رَأَيْتُ الطَّالِبِيْنَ
Jamak Muannats Salim	Dhammah	رَأَيْتُ الطَّالِبَاتِ
Jamak Taksir	Dhammah	رَأَيْتُ الطُّلَّابَ
Asmaul Khomsah	Waw	رَأَيْتُ أَبَاكَ

5. *Maf'ul bih* bisa dilihat dari jenis atau jumlah apa saja (tergantung pada konteks kalimat)

   Meskipun pada *fi'il* maupun *fa'il* satu sama lain harus terkait, akan tetapi *maf'ul bih* sama sekali tidak terkait dengan kondisi *fi'il* dan *fa'il* karena *maf'ul bih* memang menyesuaikan pada maksud pembicaraan [4]. Contohnya kalimat:

   قَرَأَ زَيْدٌ الكِتَابَيْنِ

   (Zaid membaca dua buku)

2.1.5 Bina (الْبِنَاءُ) dan I'rab (الإعْرَبُ)

Proses yang dilakukan untuk mengetahui kedudukan kata dalam Bahasa Arab adalah proses menentukan apakah suatu kata termasuk kedalam *Mu'rob* (**مُعْرَبٌ**) atau *Mabni* (**مَبْنِيٌّ**). *Mabniy* (*Bina*) ialah kata yang huruf akhirnya senantiasa tetap (tidak berubah), baik harokat maupun sukunnya. Harokatnya terbagi menjadi empat yaitu dhammah, fath-hah, kasroh, dan sukun, seperti [5]:

1. **حَيْثُ** Mabni Dhammah
2. **أَيْنَ** Mabni Fathah
3. **أَمْسِ** Mabni Kasroh
4. **هَلْ** Mabni Sukun

Bina biasanya ditemukan pada kata yang tergolong huruf*, fi'il madhi*, dan *fi'il amr*, meskipun dalam kondisi lainnya bisa ditemukan pada *fi'il mudharie'*, dan yang lainnya. Sedangkan, *I'rob* adalah perubahan akhir kata karena perbedaan *amil* yang memasukinya, baik secara lafazh ataupun secara diperkirakan keberadaannya, seperti [5]:

1. **جَاءَ زَيْدٌ** = Zaid telah datang
2. **رَأَيْتُ زَيْدًا** = Aku telah melihat Zaid
3. **مَرَرْتُ بِزَيْدٍ** = Aku telah bertemu dengan Zaid

I'rob terbagi menjadi empat, yaitu *I'rob Rofa'*(**الرَّفْعُ**), *I'rob Nashob*(**النَّصْبُ**), *I'rob Khafadh*(**الخَفْضُ**), dan *I'rob Jazm*(**الْجَزْمُ**). Pada *isim* hanya terjadi tiga kondisi yaitu *Rofa'*, *Nashob*, dan *Khafadh*, sedangkan pada *Fi'il* terjadi tiga kondisi yaitu *Rofa'*, *Nashob*, dan *Jazm*. [4]

2.2 Natural Language Processing

Natural language processing (NLP) jika diuraikan secara bahasa menjadi terpisah yaitu *natural language* yang diterjemahkan sebagai bahasa yang digunakan sehari-hari sebagai alat komunikasi bagi manusia, dan processing yang dimaksud adalah proses yang dilakukan untuk memberikan instruksi atau perintah kedalam

komputer dengan bahasa yang dipahami oleh komputer. Sehingga, NLP secara sederhana dapat dipahami sebagai metode yang digunakan untuk membuat komputer mampu memahami bahasa alami manusia. NLP merupakan bidang disiplin ilmu *computational linguistics*, *computing science, cognitive science,* dan *artificial intelligence.* [10]

Tujuan NLP berdasarkan perspektif ilmiah adalah mempelajari dan mengembangkan mekanisme kognitif yang ada pada bahasa alami manusia, sedangkan dalam persepektif teknik, berfokus pada pengembangan sistem untuk memfasilitasi interaksi komputer dengan manusia menggunakan bahasa alami. [10] Secara umum NLP bertujuan untuk memudahkan interaksi antara manusia dengan komputer, sehingga terlihat seolah-olah manusia sedang berkomunikasi dengan manusia lainnya, padahal salah satu dari mereka merupakan komputer.

Alat *preprocessing* yang tepat dalam banyak studi *natural language processing* sangat penting untuk memberikan akurasi yang lebih baik [16]. Beberapa aplikasi yang menggunakan NLP tercakup dalam kategori *speech recognition* (pengenalan perkataan), *spoken language understanding* (Pemahaman bahasa lisan), *dialogue systems* (sistem percakapan), *lexical analysis*, *parsing*, *machine translation* (mesin penerjemah), *knowledge graph* (grafik pengetehuan), *information retrieval* (pencarian informasi), *question answering* (tanya jawab), *sentiment analysis* (analisis sentimen), *social computing* (komputasi sosial), *natural language generation*, dan *natural language summarization* (Ringkasan bahasa alami) [10].

Beberapa aplikasi diatas pada umumnya menerima inputan data berupa teks yang didapatkan dari sosial media, dan website lainnya. Data ini kemudian diolah untuk mendapatkan hasil prediksi klasifikasi, kecocokan, serta prediksi yang dapat digunakan untuk pengolahan data. Beberapa aplikasi lainnya menggunakan aturan dasar (*rule-based*) yang akan dibahas pada sub bab selanjutnya, salah satunya mesin penerjemah. Berikut merupakan *tahapan* dari NLP yang biasa dilakukan:

1. Tokenisasi

   Tokenisasi merupakan pembuatan token untuk setiap suatu kumpulan kalimat, paragraph dalam suatu teks tertentu. Tokenisasi ditujukan untuk

membagi suatu teks kedalam paragraf-paragraf, paragraf menjadi kalimat-kalimat, dan seterusnya. Sebagai contoh terdapat suatu paragraf sebagai berikut:

> "Jakarta merupakan ibu kota dan kota terpadat di Indonesia. Jakarta merupakan satu-satunya kota di Indonesia yang memiliki status setingkat provinsi. Jakarta terletak di pesisir bagian barat laut Pulau Jawa. Dahulu pernah dikenal dengan beberapa nama di antaranya Sunda Kelapa, Jayakarta, dan Batavia."

Paragraf diatas akan dibagi dan proses tokenisasi dilakukan sebanyak dua kali, yaitu:

2. Tokenisasi kalimat

   Langkah pertama dalam tokenizing adalah memisahkan teks dalam paragraf menjadi kalimat per kalimat. Sebagai contoh menggunakan paragraf diatas, kemudian dipecah dan dibagi menjadi sebagai berikut:

   > "Jakarta merupakan ibu kota dan kota terpadat di Indonesia"
   >
   > "Jakarta merupakan satu-satunya kota di Indonesia yang memiliki status setingkat provinsi"
   >
   > "Jakarta terletak di pesisir bagian barat laut Pulau Jawa"
   >
   > "Dahulu pernah dikenal dengan beberapa nama di antaranya Sunda Kelapa, Jayakarta, dan Batavia"

   Asumsinya adalah setiap kalimat pada bahasa Indonesia dibagi menjadi suatu gagasan atau ide yang terpisah. Pemisahan ini untuk mempermudah saat melakukan pemrograman dan mempercepat eksekusi, dibandingkan menggunakan paragraph langsung. Pemisahan kalimat lebih mudah dan sederhana untuk dibuat dibandingkan dengan melihat tanda baca itu sendiri. Akan tetapi, NLP juga dapat menggunakan Teknik yang lebih kompleks yang mampu bekerja meski format dokumen berantakan.

3. Tokenisasi kata

   Setelah melakukan tahapan tokenisasi kalimat, selanjutnya dapat melakukan proses pemisahan kata dalam kalimat satu persatu. Sebagai contoh pada kalimat pertama, maka:

   "Jakarta merupakan ibu kota dan kota terpadat di Indonesia"

   Kalimat diatas akan menjadi token baru dalam bentuk kata yang terpisah, seperti:

   "Jakarta", "merupakan", "ibu kota" dan "kota" "terpadat" "di" "Indonesia".

   Tokenisasi mudah dilakukan apabila kata didalamnya merupakan kata yang dapat beridir sendiri, bukan merupakan gabungan dari dua kata, seperti kata "ibu kota" dapat berbeda makna dari kata "ibu" dan kata "kota".

4. Memprediksi jenis kata pada setiap token

   Pada tahapan selanjutnya, setiap token akan diklasifikasikan kedalam kata apakah itu, kata benda, kata kerja, kata sifat, dan sebagainya. Melalui cara ini, kita dapat mengetahui apa kalimat yang sedang dibicarakan. Hal ini dapat dilakukan dengan memasukkan setiap kata (dan beberapa kata tambahan di sekitarnya untuk konteks) ke dalam model klasifikasi yang disebut *pre-trained part-of-speech*.

5. Lemitisasi Teks

   Pada bahasa manusia, beberapa kata muncul dalam bentuk yang berbeda, padahal memiliki kata dasar yang sama. Sebagai contoh dalam bahasa inggris:

   I had a pony.

   I had two ponies.

   Kedua kalimat berbicara tentang hal yang sama, yaitu kepemilikian kuda poni .tetapi memiliki perubahan bentuk yang berbeda. Komputer membaca kedua kata tersebut, yakni "pony" dan "ponies" merupakan dua kata yang sangat berbeda. Sehingga untuk mempermudah komputer dalam

mengetahui bahwa kedua kalimat diatas pada dasarnya bercerita tentang satu hal yang sama, maka dibutuhkan kata dasar sebagai acuannya.

Kita dapat melakukan lemitisasi dengan mencari tahu bentuk kata paling dasar dari setiap kata dalam kalimat. Hal yang sama juga diterapkan pada kata kerja. Kata kerja dapat dilemitisasi dengan mencari akar kata mereka. Lemitisasi biasanya dilakukan dengan melihat tabel bentuk kata dasar, dengan memperhatikan bagian kata-katanya dan mungkin memiliki beberapa aturan khusus untuk menangani kata-kata yang belum pernah Anda lihat sebelumnya.

6. Mengidentifikasi Stop Words

   Selanjutnya, kita perlu mempertimbangkan pentingnya setiap kata dalam kalimat. Bahasa manusia memiliki banyak kata-kata yang muncul dengan makna yang kurang. Sebagai contoh dalam bahasa inggris terdapat "and", "the", dan "a", pada bahasa Arab pun terdapat waw ibtida “وَ”, sebagai huruf pengawal saja, dan tidak memiliki makna. Ketika diteliti secara statistik pada teks, kata-kata ini muncul amat banyak dan lebih sering dibandingkan kata lainnya. Pada NLP kita dapat melakukan penandaan terhadap kata tersebut.

2.3 Sistem Rule-Based

Sistem *rule-based* merupakan sistem yang terdiri dari seperangkat aturan. Dengan kata lain, aturan-aturan yang ada pada dasarnya disimpan dalam satu set, yang disebut sebagai *rule set*. Selain itu, data yang digunakan untuk diolah menggunakan *machine learning* biasanya disebut sebagai *data set*. [11]. *Rule-based* sangat bersinggungan dengan logika Boolean, dan logika matematika adalah rumpun ilmu yang menjadi dasar dalam logika Boolean.

Logika matematika didalamnya memuat konjungsi, disjungsi, negasi, ,dan implikasi. Berikut Ilustrasi untuk mempermudah pemetaan logika Boolean seperti konjungsi, disjungsi, negasi berdasarkan input tertentu.

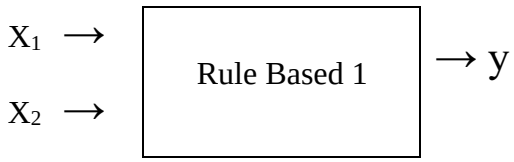
Tabel 2.12 Logika Matematika

Input 1	Input 2	Negasi Input 1	Negasi Input 2	Konjungsi	Disjungsi	Implikasi
a	b	~a	~b	a ˆ b	a ˇ b	a → b
TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE

Logika matematika diatas amat banyak digunakan pada *rule-based*, khususnya dalam membuat *rule* set. Logika menurut Rose dalam Han Liu, merupakan bagian kecil dari kemampuan pertimbangan yang dimiliki manusia dan digunakan untuk membantu mereka dalam pengambilan keputusan[11].

Rule set digunakan untuk menyimpan rules dan setiap elemen diolah melalui rules, sedangkan *data set* digunakan untuk menyimpan data, dan setiap elemen merepresentasikan data poin setiap set dapat terdiri atas elemen (2^n), dimana n adalah jumlah elemen pada satu set. Terdapat bermacam cara untuk merepresentasikan cabang keilmuan *Artificial Intelligence*, dan diantara yang paling popular adalah bentuk *if-then rules* yang dinodai dengan ekspresi IF – PENYEBAB (antecedent) THEN – AKIBAT (consequent)[11].

If-*then* r*ules* menunjukkan bahwa suatu penyebab dapat memberikan efek tertentu. Terkadang, penyebab ini muncul dala beberapa kondisi, sehingga menghasilkan *rules* yang berbeda, dan hal ini mengindikasikan bahwa *rules* bersifat *inconsistent*[11]. Sebagai contoh, berikut merupakan gambaran mengenai *rule-based*:



Jika $X_1 = 0$ dan $X_2 = 0$ maka y € (0,1)

Jika $X_1 = 0$ dan $X_2 = 1$ maka y € (0,1)

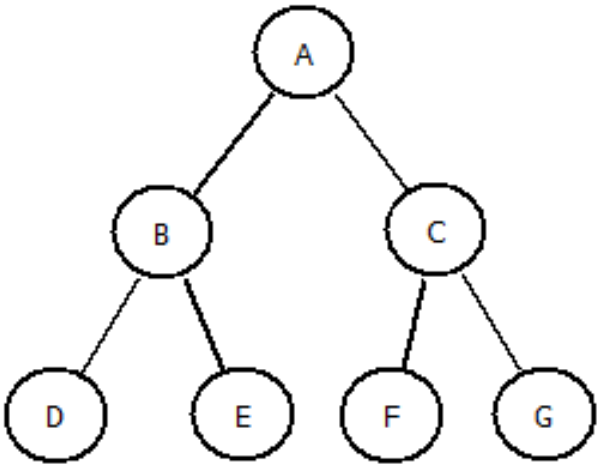
Jika $X_1 = 1$ dan $X_2 = 0$ maka y € (0,1)

Jika $X_1 = 1$ dan $X_2 = 1$ maka y € (0,1)

Rule base dapat digunakan untuk mengatur rules secara efektif dan efisien dari sisi *storage* dan *retrieval*. Hal ini sejalan dengan efektifitas algoritma menurut Han Lin, bahwa Algoritma yang baik harus memenuhi aspek [11]:

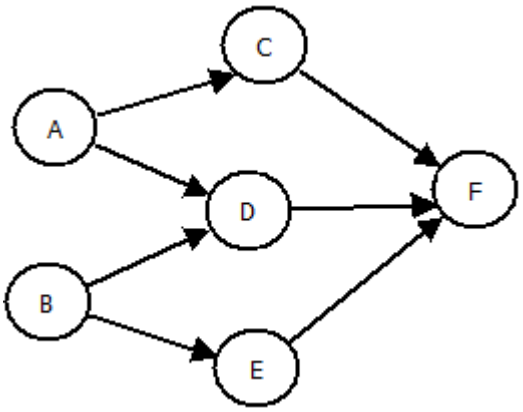
1. *Acuration*, yang mengacu pada ketepatan dalam hal korelasi antara input dan output
2. *Effectivity*, yang mengacu pada biaya yang dibutuhkan untuk komputasi
3. *Robustness*, yang mengacu pada toleransi terhadap kesalahan input
4. *Readability*, yang mengacu pada interpretabilitas pada orang lain.

Han Liu menyebutkan bahwa terdapat graph theory yang direpresetasikan dengan *binary trees*. Menurutnya, *binary trees* terbagi menjadi dua, *directed* dan *undirected*. Berikut gambaran umum *binary trees* [11]:

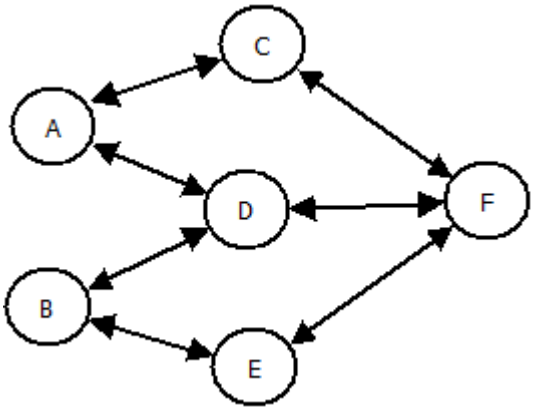


Gambar 2.1 Binary Tree

Sedangkan berikut ini merupakan gambaran *binary trees directed* dan *binary trees undirected*:



Gambar 2.2 Directed binary trees



Gambar 2.3 Undirected binary trees

2.3.1 Pendekatan Rule Based

Han Liu menyebutkan bahwa dalam sistem *rule-based* dibagi menjadi dua pendekatan, yaitu *Divide and conquer*, dan *Separate and conquer*[11].

1. Divide and Conquer

   Divide and conquer juga dikenal dengan istilah *Top Down Induction of Decision Trees* (TDIDT) karena proses klasifikasi *rules* yang digunakan dalam pendekatan ini dalam bentuk decision tree. Bentuk dasar *procedural* dari TDIDT diilustrasikan pada gambar berikut[11].

   Pada pendekatan ini, hal terpenting dari prosedurnya adalah pemilihan atribut yang digunakan untuk melatih subset pada *node* dari *decision tree*. Secara khusus, penentuan atribut dapat dilakukan melalui seleksi random atau menggunakan hasil pengukuran statsitik seperti *information gain*,

gain ratio, dan *Gini index*. Pada dasarnya, metode ini memecahkan masalah dengan membagi terlebih dahulu masalah utama mmenjadi rumusan masalah yang lebih kecil, Kemudian, rumusan masalah tersebut diselesaikan secara *independent*. Setelahnya menggabungkan kembali solusi dari masing-masing rumusan masalah, sehingga mendapatkan solusi dari masalah utama.

2. Separate and Conquer

   Pendekatan *separate and conquer* juga dikenal dengan pendekatan *covering* karena pendekatan ini melibatkan *sequential generation* dari *if-then rules*. Secara khusus, pendekatan ini bertujuan untuk menghasilkan aturan yang mencakup sejumlah *instance* yang berasal dari kelas yang sama dan kemudian aturan berikutya dibuat berdasarkan sisa *training instance* yang tidak tercakup oleh aturan yang dibuat sebelumnya. Prosedur dasar dari pendekataan ini sebagai berikut[11]:

```
Input: satu set training instances, atribut Ai, dimana i merupakan
index dari atribut A, nilai Vj , dimana j merupakan index dari nilai V
Output: Sebuah decision tree.
  if kriteria terpenuhi then
        buat cabang yang sesuai dengan training instances yang tersisa
  else
        pilih yang terbaik (berdasarkan beberapa heuristik) atribut Ai
        label node ini dengan Ai
        for each nilai Vj dalam atribut Ai do
              label outgoing edge dengan nilai Vj
              buat subtree secara rekursif dengan menggunakan subset
training
              instances yang sesuai
        end for
end if
```

   Perbedaan pendekatan separate *and conquer* dengan pendekatan *divide and conquer* adalah jika dalam pendekatan divide and conquer memecah atau membagi satu masalah utama kedalam beberapa masalah, maka dalam pendekatan ini masalah tersebut pada dasarnya bukan merupakan

satu kesatuan, tetapi masalah yang berkaitan satu sama lain, lalu kemudian digabungkan.

2.3.2 Teknik-Teknik dalam Rule Based

Representasi aturan yang tepat diperlukan untuk meningkatkan efisiensi dan interpretabilitas model. Menurut Han Liu, terdapat tiga teknik untuk representasi *classification rules* yaitu, *decision tree, linear lists* dan *rule-based network*. Secara khusus, representasi ini diilustrasikan menggunakan contoh dalam hal penentuan aturan. Teknik-teknik dibahas secara komparatif dari sisi kompleksitas komputasi dan interpretabilitas.

1. *Decision Tree*

   Decision tree adalah representasi otomatis untuk *classification rules* yang dihasilkan oleh pendekatan *'divide and conquer*'. Ini menunjukkan bahwa jika metode berbasis aturan yang mengikuti pendekatan yang disebutkan di atas diadopsi untuk menghasilkan aturan, maka aturan secara otomatis diwakili dalam struktur *tree*. Representasi aturan yang tepat adalah signifikansinya untuk tujuan penelitan dan *predictive modelling*.

   Representasi *decision tree* dikritik oleh Cendrowska dan diidentifikasi sebagai penyebab utama overfitting karena masalah *sub-tree* yang direplikasi sedangkan Cendrowska dibenarkan pendapatnya bahwa aturan-aturan yang tidak memiliki atribut umum tidak dapat masuk dalam struktur tree dan bahwa masalah *sub-tree* yang direplikasi akan muncul jika aturan tersebut dipaksakan untuk masuk dalam struktur *tree*. Juga dikemukakan bahwa diperlukan pemeriksaan seluruh *tree* untuk mengekstraksi aturan tentang klasifikasi tunggal dalam kasus terburuk. Kelemahan ini membuatnya sulit untuk memanipulasinya untuk sistem pakar dan dengan demikian secara serius menurunkan efisiensi komputasi dalam memprediksi instance yang tidak terlihat. Efisiensi komputasi dalam tahap pengujian sangat penting terutama ketika sistem pakar yang akan dibangun dalam kondisi kritis, jika tujuan menggunakan sistem tersebut adalah untuk *predictive modelling* [11].

Menurut Han Liu, *decision tree* seringkali cukup kompleks dan sulit dipahami. Bahkan jika *decision tree* disederhanakan dengan menggunakan algoritma *pruning*, masih sulit untuk menghindari fakta bahwa *decision tree* terlalu rumit, dan sulit dipahami. Hal Ini jelas menurunkan interpretasi *decision tree* dan dengan demikian ini menjadi kelemahan serius dalam ranah penelitian [11].

2. *Linear List*

Representasi *decision tree* memiliki keterbatasan serius untuk penemuan pengetahuan dan *predictive modelling*. Oleh karena itu, penggunaan langsung aturan *if-then* cukup bagus. Berbeda dengan *decision tree*, *linear list* tidak perlu memiliki batasan seperti semua aturan harus memiliki setidaknya satu atribut yang sama, dan dengan demikian keberadaan istilah yang berlebihan ini berkurang dalam satu *rule set*. Namun, seolah-olah rule diwakili dalam struktur *linear list*, proses prediksi *instances* tak tampak dalam representasi ini ditunjukkan dalam *linear search* dengan kompleksitas waktu dari O (n), di mana jumlah total *rule terms* yang digunakan sebagai ukuran input (n). Hal ini terjadi karena representasi *linear list* dilakukan berdasarkan *linear search* dengan menelusuri keseluruhan *set rule* demi *rule* yang pada *outer loop*; dan melalui istilah demi istilah dalam *inner loop*. Proses *linear search* dapat diilustrasikan dengan menggunakan contoh *rule set* di bawah ini [11]:

Rule 1: Jika $X_1 = 0$ dan $X_2 = 0$ maka y = 0;

Rule 2: Jika $X_1 = 0$ dan $X_2 = 1$ maka y = 0;

Rule 3: Jika $X_1 = 1$ dan $X_2 = 0$ maka y = 0;

Rule 4: Jika $X_1 = 1$ dan $X_2 = 1$ maka y = 1;

Berdasarkan *rule set* di atas, jika sebuah instance dengan dua input ($X_1 = 1$ dan $X_2 = 1$), maka perlu terlebih dahulu melalui *Rule* 1 memeriksa nilai-nilai X_1dan X_2 dan kemudian pindah ke *Rule* 2 melakukan pemeriksaan yang sama lagi hingga *Rule* 4 diperiksa dan ditemukan hasil.

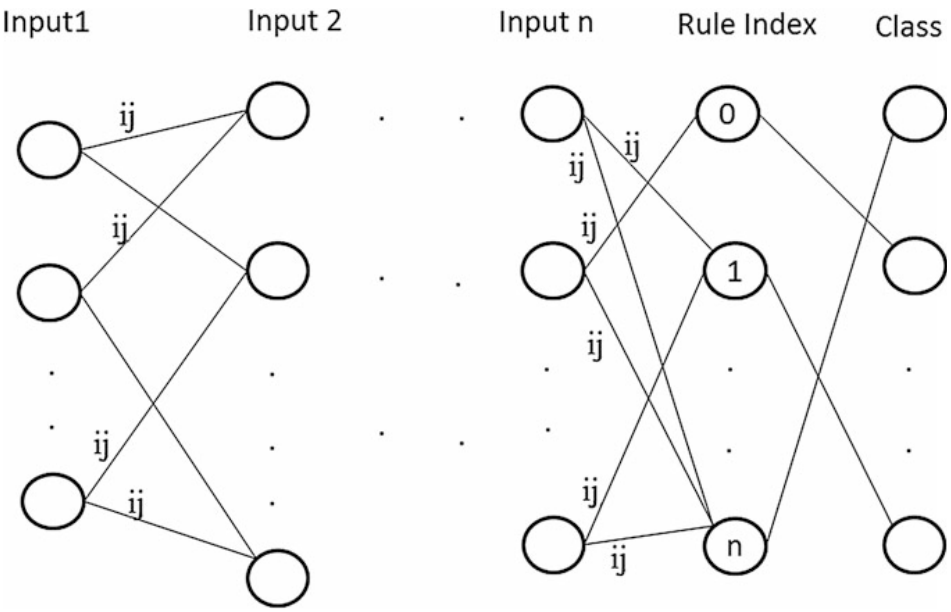
Menurut Han Liu, deskripsi di atas menyiratkan bahwa dapat terjadi kondisi dimana input harus melalui seluruh *rule set* untuk mendapatkan

hasil *rule* yang pertama. Sehingga, hal ini akan menyebabkan biaya komputasi yang besar ketika representasi ini digunakan untuk mewakili *rule set* yang dibuat dengan menggunakan data latih yang besar (*large training data*). Oleh karena itu, untuk tujuan *predictive modelling*, *linear list* masih belum dapat memenuhi tujuan untuk mencari rules yang efisien. Sehingga, menurutnya diperlukan untuk mengembangkan teknik representasi rules lain yang menunjukkan tingkat efisiensi komputasi yang lebih tinggi daripada *linear list* [11].

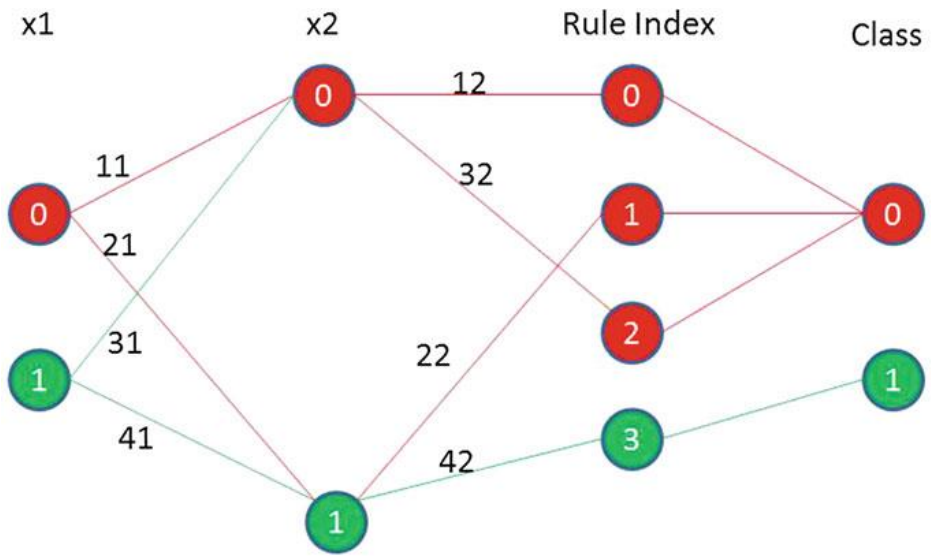
3. *Decision Tree*

   Han Liu dan rekannya mengembangkan salah satu teknik baru untuk representasi rules. Mereka menyebut Teknik ini dengan sebutan *rule based* network, dimana mereka mengklaim bahwa teknik ini lebih efisien daripada *linear search*. Mereka juga mengklaim bahwa teknik ini dapat digunakan untuk memberikan representasi pengetahuan yang lebih dapat ditafsirkan daripada *decision tree* dan *linear list* lakukan.

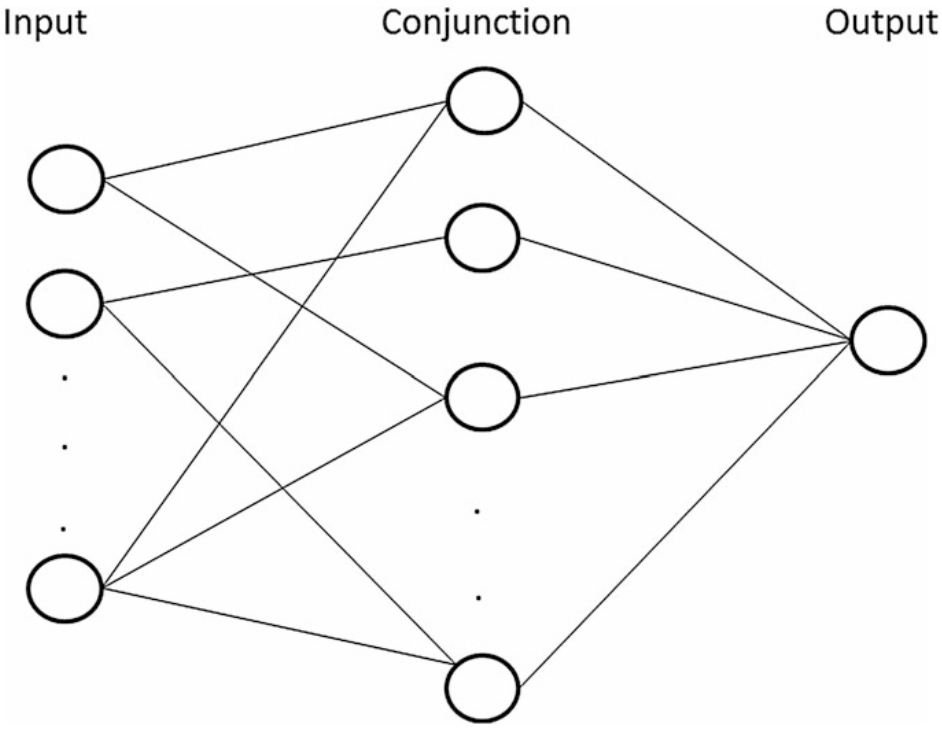
   Mereka menyebutkan terdapat beberapa versi Deterministic rule based network, diantaranya:



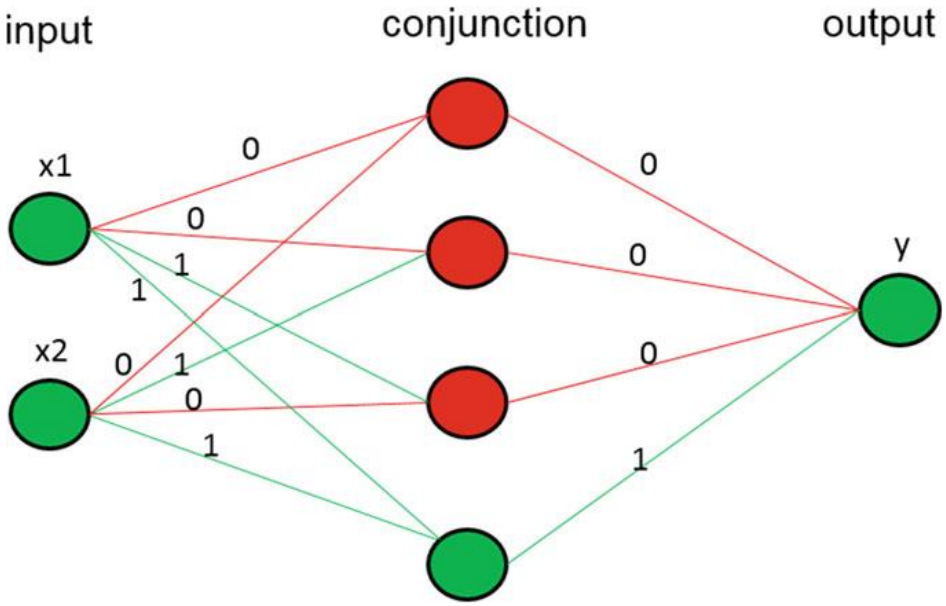
Gambar 2.4 Rule-based network (versi 1)



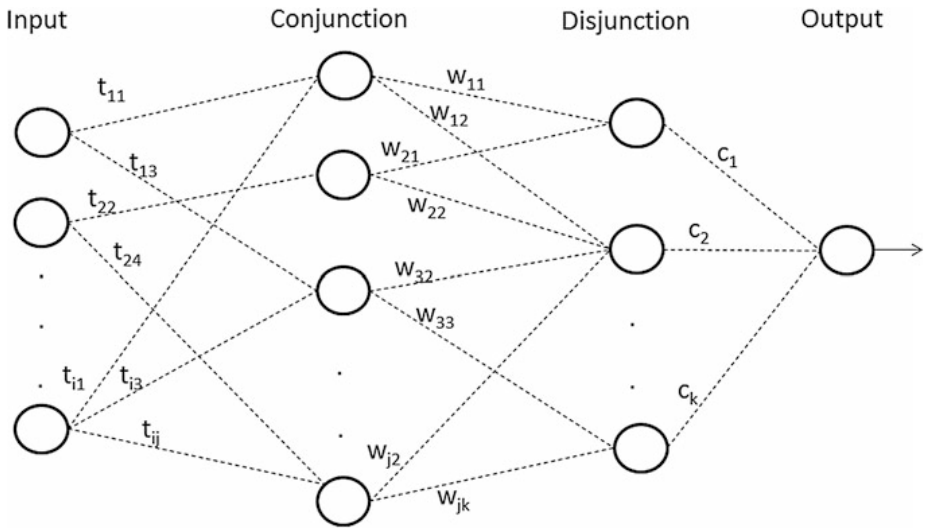
Gambar 2.5 Contoh rule based network (versi 1)



Gambar 2.6 Deterministic rule-based network (versi 2)



Gambar 2.7 Contoh deterministic rule-based network (versi 2)



Gambar 2.8 Unified rule-based network

Representasi *rule-based network* di atas didasarkan pada hubungan antara nilai-nilai atribut dan label kelas, dengan demikian disebut sebagai *rule-based network* yang berorientasi pada atribut-nilai. Secara umum, ini adalah *layer network* n + 2 untuk masalah klasifikasi. n pertama mewakili atribut ke-n dari input. Pada setiap *layer*, setiap simpul mewakili nilai atribut yang sesuai. Oleh karena itu, jumlah node di setiap lapisan n

tergantung pada jumlah nilai atribut yang sesuai. Lapisan kedua terakhir mewakili indeks *rules* yang sama dengan urutan *rules* ini dikurangi satu. Misalnya, jika indeks *rules* adalah 0, ini menunjukkan bahwa ini adalah *rules* pertama dalam *rule set*. Jumlah node di lapisan ini tergantung pada jumlah *rules* [11].

Lapisan terakhir dalam *network* mewakili output kelas dan jumlah node tergantung pada jumlah kelas. Ada juga koneksi antara berbagai lapisan, yang harus dijelaskan lebih lanjut menggunakan contoh-contoh spesifik. Namun, secara umum, koneksi bisa terjadi antara dua lapisan yang tidak berdekatan satu sama lain. Sebagai contoh, node di lapisan pertama dapat memiliki koneksi dengan node lain di lapisan ketiga. Persis seperti rute perjalanan yang mencakup sejumlah kota. Dalam konteks ini, setiap kota adalah *rule term* dan setiap rute seperti *rules*. Ada kemungkinan bahwa ada kota-kota yang tidak berdekatan satu sama lain tetapi termasuk dalam rute perjalanan yang sama. Selain itu, dua node mungkin memiliki lebih dari satu koneksi [11].

2.4 Bahasa Pemrograman

Bahasa Pemrograman merupakan instruksi standar yang dilakukan oleh para programmer untuk menerjemahkan bahasa manusia agar dapat dipahami oleh komputer. Komputer dapat memahami instruksi dengan menggunakan sintaks dan semantik yang disusun menggunakan bahasa yang diberi istilah bahasa komputer. Terdapat sangat banyak bahasa pemrograman yang digunakan di dunia ini. Berikut diantaranya merupakan bahasa pemrograman yang dapat digunakan dalam merancang aplikasi dengan metode *natural language programming* (NLP):

2.4.1 Python

Python merupakan bahasa pemrograman yang dikembangkan tahun 1990 oleh Guido van Rossum di Amsterdam. Bahasa Python dibuat dengan harapan mempermudah para programmer dalam membuat aplikasi. Python memiliki tata bahasa yang jernih dan mudah dipelajari, karena bahasanya yang tidak rumit, membuat python banyak digunakan saat ini. Bahasa Python diminati karena para programmer lebih dapat berfokus pada algoritma aplikasi, daripada terus mencari

eror pada sintaks. Python juga memiliki aturan layout kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang kode sumber.

Pyhton juga dapat digunakan di berbagai platform diantaranya Windows, MacOS X, Linux/Unix, Java Virtual Machine, Palm, Symbian, OS/2, Amiga. Python memiliki berbagai macam distribusi dengan lisensi yang beragam pula, namun demikian semuanya dapat digunakan secara bebas (gratis) untuk tujuan penggunaan pribadi, maupun tujuan komersil. Python juga memiliki fasilitas pengumpulan sampah dan pengelolaan memori otomatis, seperti halnya pada bahasa pemrograman Java. Python juga memiliki komunitas yang besar, sehingga untuk pemula dapat menanyakan kepada para programmer senior di komunitas python.

Terdapat beberapa Framework yang biasa digunakan oleh para programmer yang menggunakan bahasa Python, diantaranya:

1. Django
2. Web2Py
3. CherryPy
4. Flask
5. Pyramid
6. Turbo Gears, dan masih banyak lagi.

2.4.2 JavaScript

JavaScript merupakan sebuah bahasa pemrograman yang dikembangkan diawal oleh Netscape dengan programmer bernama Brendan Eich. Brendan Eich berhasil menyelesaikan pembuatan bahasa ini dalam 10 Hari denga nama Mocha. Mocha lalu diimplementasikan kedalam Netscape Navigator, yaitu sebuah browser yang dikembangkan oleh Netscape dan diganti namanya menjadi Livescript, sebelum akhirnya mereka membuat browser baru, yaitu Mozilla Firefox.

Netscape dengan ambisi mengalahkan perusahaan besar saat itu, Microsoft, Netscape bekerja sama dengan sun microsystems. Perusahaan Sun adalah perusahaaan yang mengembangkan bahasa Java. Sun menyepakati kerja sama dengan Netscape, dengan syarat menggantikan livescript yang digunakan pada Netscape Navigator dengan Java. Akan tetapi, Netscape menolak, dan pada

akhirnya kedua pihak menyepakati perubahan nama LiveScript menjadi JavaScript dimana Sun memegang lisensi dari JavaScript., sedangkan Microsoft setelah mengetahui hal tersebut mulai melakukan *reverse engineering* dengan meniru JavaScript dan menghadirkan Jscript sebagai pesaingnya.

Netscape mengeathui bahwa Microsoft adalah perusahaan besar, sehingga amat mudah untuk membuat ISO dari Jscript, sehingga Netscape ingin membuat ISO atas *side-scripting* mereka melalui W3C, akan tetapi pihak W3C menolak. Netscape kemudian mencari organisasi lain yang melakukan standarisasi dan bertemu d dengan salah satu organisasi, yaitu ECMA (European computer manufacture association) International. JavaScript pertama kali distandarkan pada tahun 1996 dengan nama ECMAScript. ECMAScript terus berkembang dari tahun1997 dengan ES 1-nya, lalu tahun 1998 ES 2, 1999 ES 3 dan mulai diluncurkan, 2008 ES 4 (gagal), 2009 ES 5, 2015 ES 6, 2016 ES 7, 2017 ES 8, 2028 ES 9, 2019 ES 10, hingga tahun 2020, saat ini sedang dikembangkan ES 11.

Java Script merupakan bahasa yang dikembangkan untuk *client side-scripting* dimana biasanya digunakan untuk melakukan desain UI (*User Interface*) dan UX (*User Experience*) yang lebih interaktif. Saat ini JavaScript merupakan bahasa yang cukup digemari didunia, selain bahasa Java, Python, dan beberapa lainnya. JavaScript dianggap cukup *powerful* untuk digunakan. JavaScript banyak digunakan pada pembangunan web, sebagaimana perkembangan awalnya yaitu sebagai bahasa yang digunakan pada browser Netscape Navigator. Penggunaan JavaScript juga sangat mudah dan efisien cukup dengan text editor dan browser.

2.4.3 HTML dan CSS

HTML (*Hypertext Markup Language*) merupakan bahasa standar pada markup dokumen. Tujuan HTML adalah untuk memuat informasi di dalam suatu website dan melakukan format ASCII agar dapat menghasilkan dokumen yang terintegrasi satu sama lain. Sehingga mempermudah bagi para pembaca untuk mengakses informasi apa yang ingin mereka cari, dan informasi mana yang sudah mereka ketahui.

HTML dibuat oleh seorang ahli fisika CERN, salah satu lembaga penelitian di Swiss, Tim Berners Lee. Ide awalnya adalah dia ingin membuat sistem hypertext

yang dapat digunakan pada internet. Hypertext sendiri merupakan teks yang dapat menyimpan referensi (link) menuju teks lain yang bisa diakses langsung oleh *user*. HTML pada tahun 1991 pertama kali dirilis, dimana saat itu terdapat 18 tag HTML.

Kemudian HTML terus dikembangkan dan setiap kali mengeluarkan dan mengupdate versi terbarunya, HTML selalu menambahkan tag dan attribute (tag modifier) terbaru. Pada tahun 2014 HTML melakukan upgrade besar, dengan sebutan yang samapai saat ini dikenal, yaitu HTML5. Pada upgrade tersebut, HTML menghadirkan semantik baru yang mana mampu memberikan kelompok dari kontennya sendiri, seperti <artcile>, <header>, dan <footer>.

CSS (*Cascading Style Sheet*) biasanya digunakan untuk mengatur tampilan elemen yang tertulis dalam bahasa markup, seperti HTML. CSS berfungsi untuk memisahkan konten dari tampilan visualnya di situs. CSS dibuat dan dikembangkan oleh W3C (World Wide Web Consortium) pada tahun 1996 untuk alasan yang sederhana. Dulu HTML tidak dilengkapi dengan tags yang berfungsi untuk memformat halaman. Anda hanya perlu menulis markup untuk situs.

Tags, seperti <font>, diperkenalkan di HTML versi 3.2, dan ketika itu menyebabkan banyak masalah bagi developer. Karena website memiliki berbagai font, warna background, dan style, maka untuk menulis kembali (rewrite) kode memerlukan proses yang sangat panjang dan sulit. Oleh sebab itu, W3C membuat CSS untuk menyelesaikan masalah ini.

HTML dan CSS memiliki keterikatan yang erat. Karena HTML adalah bahasa markup (fondasi situs) dan CSS memperbaiki style (untuk semua aspek yang terkait dengan tampilan website), maka kedua bahasa pemrograman ini harus berjalan beriringan.

2.5 Library

Python memiliki *library* besar yang sangat dan programmer dan memangkas waktu mereka dalam memprogram, selain itu memiliki banyak fasilitas pendukung sehingga mudah dalam pengoperasiannya. Berikut beberapa Library yang sering digunakan oleh programmer Python yang mengembangkan *Natural Language Programming*:

2.5.1 Natural Language Tool Kit (NLTK)

Natural Language Toolkit merupakan salah satu library yang digunakan untuk membuat program menggunakan Python berhubungan dengan bahasa manusia yang diterapkan pada NLP. NLTK memuat *library* untuk proses *tokenization*, *parsing*, *classification*, *stemming*, *tagging*, dan *semantic reasoning*. Didalamnya juga terdapat Sampel data set dan juga *cook book* yang menjelaskan prinsip kerja NLP pada NLTK.

NLTK pertama kali dibuat pada tahun 2001 sebagai bagian dari pelajaran komputasi bahasa untuk Fakultas Ilmu Komputer dan Informatika di University of Pennsylvania. Sejak saat itu, NLTK terus dikembangkan dan diperbaharui dengan bantuan puluhan kontributor. Hingga saat ini diadopsi sebagai mata kuliah di puluhan Universitas dan dijadikan dasar dalam setiap proyek penelitian . Berikut diantara modul NLTK: [6]

Tabel 2.13 Modul NLTK dan kegunaannya

Tugas NLP	Modul NLTK	Kegunaannya
Accessing corpora	nltk.corpus	Antarmuka standar untuk akses corpora dan lexicons
String processing	nltk.tokenize, nltk.stem	Tokenisasi kata, Kalimat, Stemmers
Collocation discovery	nltk.collocations	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	nltk.tag	n-gram, backoff, Brill, HMM, TnT
Classification	nltk.classify, nltk.cluster	Decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	nltk.chunk	Regular expression, n-gram, named entity
Parsing	nltk.parse	Chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	nltk.sem, nltk.inference	Lambda calculus, first-order logic, model checking
Evaluation metrics	nltk.metrics	Precision, recall, agreement coefficients

2.5.2 TextBlob

TextBlob merupakan llibrary yang harus digunakan oleh *developer* yang ingin memulai belajar NLP di Python dan pertama kali pula menggunakan NLTK. Pada dasarnya, *TextBlob* menyediakan tampilan yang ramah bagi pemula untuk mempermudah mereka mempelajari dasar NLP seperti: analisis sentimen, *pos-tagging*, atau ekstraksi kata. Meskipun demikian, TextBlob memiliki kelemahan utama yang sama dengan NLTK yakni dianggap tidak mampu dalam membantu *developer* untuk pengembangan aplikasi NLP Python yang memiliki banyak permintaan.

2.5.3 CoreNLP

CoreNLP awalnya dikembangkan di Stanford University dan ditulis dalam bahasa Java. Akan tetapi, library ini dilengkapi *wrapper* yang membuatnya dapat dibaca berbagai bahasa lainnya, termasuk Python. Hal ini yang menjadi alasan mengapa *library* ini dapat digunakan oleh developer yang tertarik untuk membuat NLP Python. *Library* ini juga sangat cepat dan bekerja baik dalam lingkungan *product development*. Terlebih lagi, beberapa komponen CoreNLP dapat diintegrasikan dengan NLTK agar bias membuat NLP yang lebih efisiens.

2.5.4 SpaCy

SpaCy adalah *library* yang digunakan untuk *Natural Language Processing* tingkat lanjut pada bahasa Python dan Cython. Sejak awal SpaCy dibuat dengan tujuan untuk dapat digunakan langsung pada produk nyata, dan telah dikembangkan dengan menggunakan hasil riset terbaru. SpaCy muncul dengan model *pretrained statistical* dan *word vectors*, dan saat ini mendukung tokenisasi untuk lebih dari 50 bahasa. *Library* ini juga memiliki fitur State-of-the-art yang cepat, model *Convolutional Neural Network* untuk *tagging, parsing* dan pengenalan *named entity* dan integrasi *deep learning* yang mudah. *Library* ini bersifat *commercial open-source software*, dirilis dengan lisensi MIT.

2.5.5 Pandas

Pada tahun 2008, pandas mulai dibuat oleh AQR Capital Management. Pada akhir tahun 2009 diluncurkan dengan lisensi *open source*, dan telah dibantu oleh kumpulan orang yang memliki tujuan sama di seluruh dunia untuk mengembangkan

nya hingga saat ini. Mereka memberikan waktu dan tenaga berharga mereka untuk berkontribusi didalamnya. Sejak 2015, pandas menjadi proyek yang disponsori oleh NumFOCUS untuk membantu kesusksesan dalam pengembangan panda sebagai proyek *open-source* kelas dunia.

Pandas cukup cepat dan efisien untuk melakukan manipulasi data, memiliki *tools* untuk membaca dan menulis data pada struktur data dan format lainnya seperti CSV dan text files, Microsoft Excel, SQL databases, and format HDF5, mampu melakukan insert data dan delete dari struktur data pada data yang berubah-ubah, dan masih banyak lagi.

2.5.6 PyArabic

PyArabic merupakan l*ibrary* khusus untuk bahasa arab di Python, yang menyediakan fungsi untuk memanipulasi huruf dan teks bahasa arab, seperti melakukan deteksi huruf arab, kelompok huruf arab dan karakteristiknya. Membuang diakritik dan lain sebagainya. Berikut diantaranya beberapa fitur dalam PyArabic:

1. Klasifikasi huruf Arab
2. Tokenisasi teks
3. Strip Harokat (semuanya, kecuali tasydid, tathwil, harokat akhir)
4. Memisah dan menggabungkan kata dan harokat
5. Mengurangi Tasykil
6. Menyesuaikan tasykil
7. Normalisasi huruf (*Ligatures* dan Hamza)
8. Mengubah angka ke dalam kata
9. Mengekstrak kalimat yang maksudnya adalah angka

2.6 Database

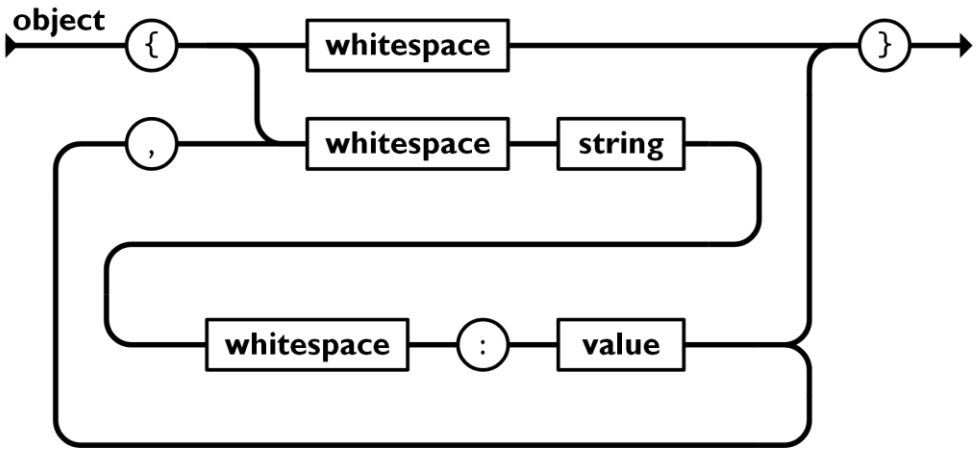
Salah satu database yang sering digunakan adalah JSON (JavaScript Object Notation)yang merupakan format pertukaran data yang ringan, karena mudah untuk dibaca dan ditulis. JSON dapat melakukan *parse* dan *generate*. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON memiliki format kode yang sepenuhnya merupakan bahasa independent, tetapi menggunakan kaidah yang mirip dengan kaidah pada bahasa C,

termasuk C, C++, C#, Java, JavaScript, Perl, Python, dan banyak lagi. Hal ini yang membuat JSON menjadi bahasa pertukaran data yang ideal. JSON dibangun pada dua struktur::

1. Kumpulan nama/nilai yang berpasangan. Dalam berbagai bahasa, dalam beberepa bahasa, hal ini ditampilkan sebagai sebuah *object*, *record*, *struct*, *dictionary*, *hash table*, *keyed list*, atau *associative array*.
2. Daftar nilai yang berurutan. Pada kebanyakan bahasa lain, hal ini ditampaikan sebagai *array*, *vector*, *list*, atau *sequence*.

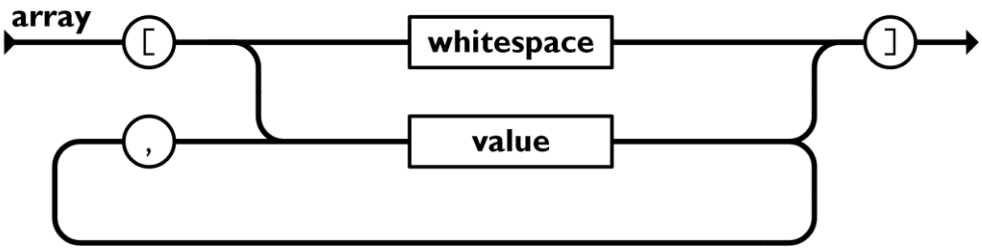
Ini merupakan struktur data universal, Hampir semua bahasa pemrograman modern menggunakannya. JSON menggunakan format sebagai berikut:

Sebuah *object* sebagai pasangan nama/nilai yang tidak beraturan. Sebuah object dimulai dengan kurung kurawal kiri "{" dan diakhiri dengan kurung kurawal kanan "}". Setuap nama diikuti dengan titik dua ":" dan pasangan nama/nilai dipisahkan oleh koma ",".



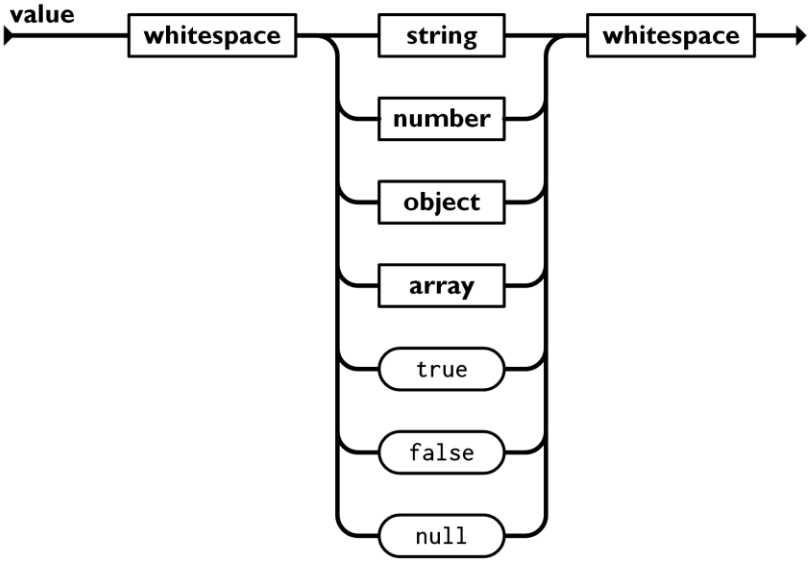
Gambar 2.9 Object pada JSON

Sebuah *array* merupakan kumpulan nilai yang beraturan. Array dimulai dengan kurung siku kiri "[" dan diakhiri dengan kurung siku kanan "]". Nilai dipisahkan dengan menggunakan koma ",".



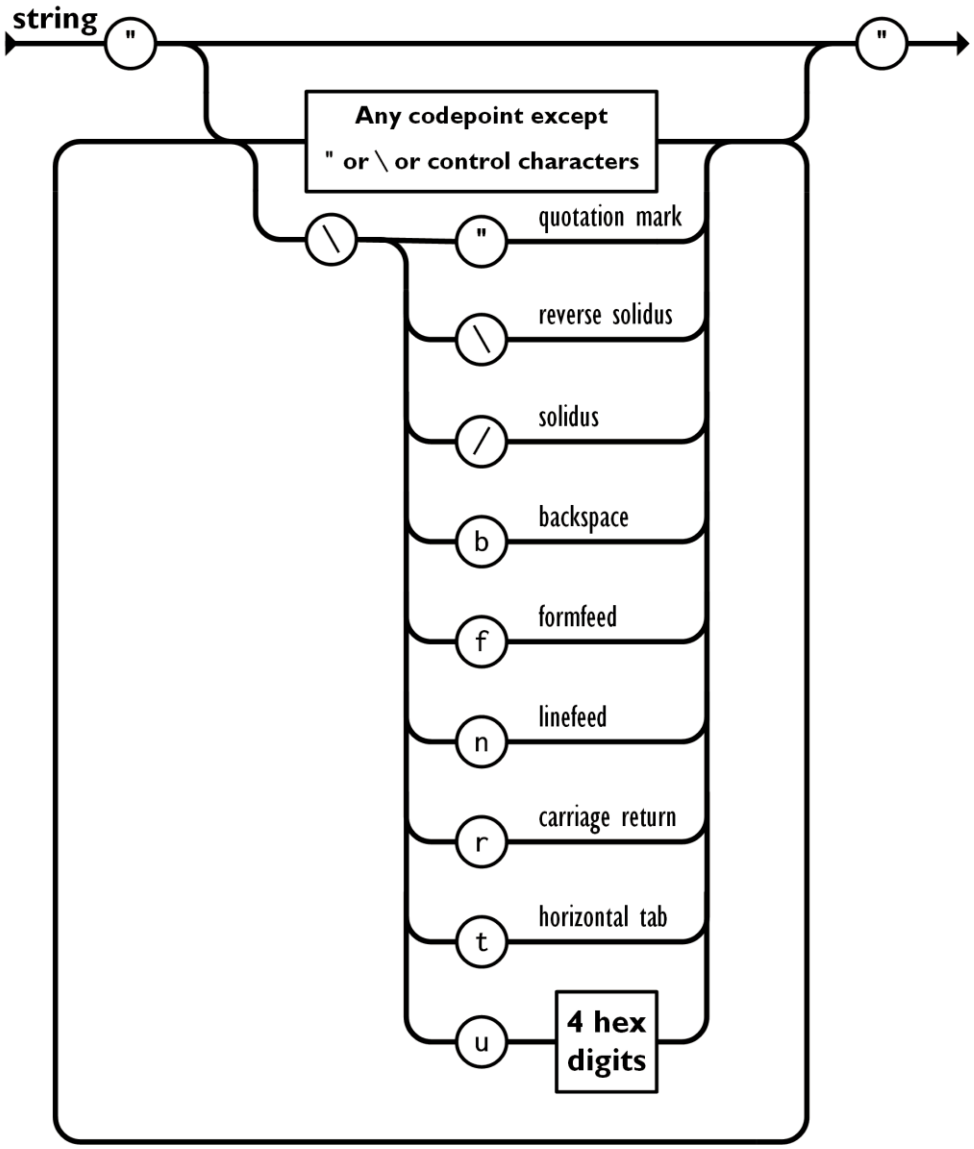
Gambar 2.10 Array pada JSON

Sebuah nilai adalah string dengan tanda kutip dua, nomor, boolean (true/false), null, *object*, sebuah *array*. Strukturnya juga dapat bersarang.



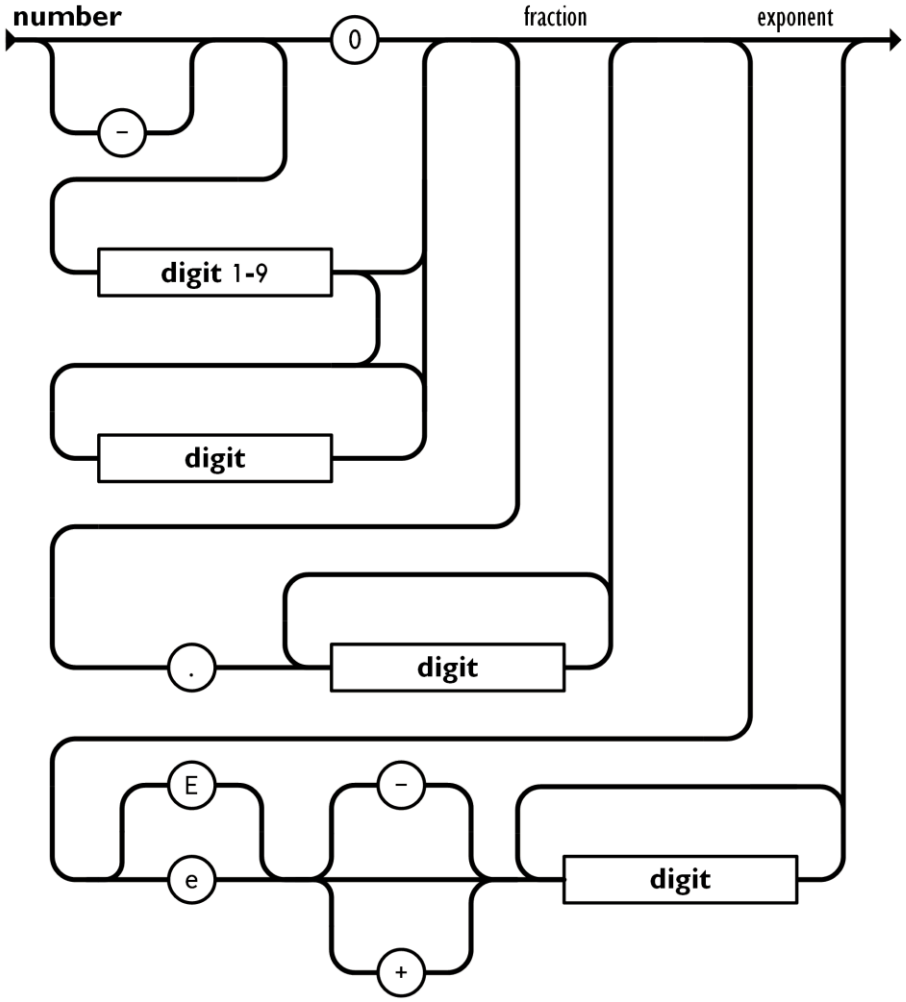
Gambar 2.11 Value pada JSON

String merupakan rangkaian karakter Unicode yang dimulai dari nol, diapit oleh tanda kutip dua, menggunakan backslash. Sebuah karakter direpresentasikan sebagai karakter tunggal. Stringnya mirip dengan string pada bahasa C atau Java.



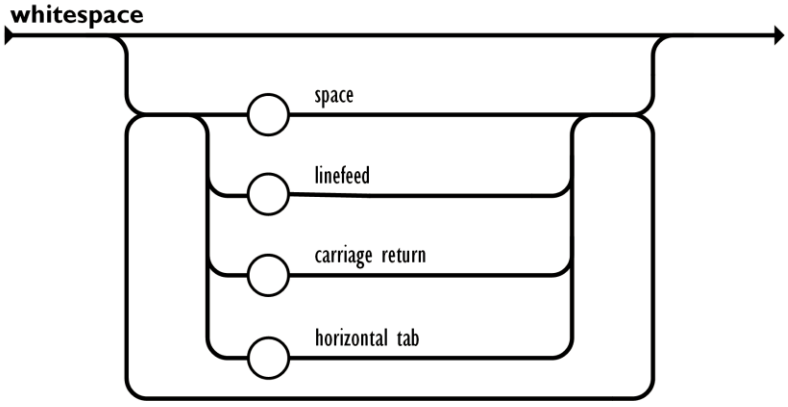
Gambar 2.12 String pada JSON

Angka pada JSON mirip dengan angka pada C atau Java, hanya saja tidak terdapat format Oktal atau Heksadesimal.



Gambar 2.13 Number pada JSON

Whitespace dapat dimasukkan dimanapun, kecuali pada pengkodean yang dapat mempengaruhi hasil.



Gambar 2.14 Whitespace pada JSON