# BAB II KAJIAN PUSTAKA

## Pengertian *Natural Language*

*Natural Language* atau bahasa alami adalah suatu bahasa yang diucapkan, ditulis, atau diisyaratkan (secara visual atau isyarat lain) oleh manusia untuk berkomunikasi. Singkatnya *Natural Language* adalah bahasa yang sering kita gunakan untuk berkomunkasi sehari-hari, seperti bahasa Indonesia, bahasa Inggris, bahasa isyarat dan bahasa lainnya sesuai letak geografisnya (Putra et al., 2017). Bahasa alami menjadi topik yang hangat diperbincangkan akhir-akhir ini, banyak peneliti yang berlomba-lomba untuk menciptakan teknologi sehingga interaksi manusia dan komputer menjadi lebih mudah lagi, salah satu penerapan teknologi adalah *Natural Language Processing* (NLP) (Chowdhury, 2005). Karena erat sekali dengan kaidah atau aturan, maka setidaknya ada tiga aspek utama pada *Natural Language*, yaitu:

1. Sintaks: menjelaskan bentuk atau struktur dari sebuah bahasa. Sintaks biasa direpresentasikan oleh sebuah *grammar* atau tata bahasa. Sebagai contoh, untuk membentuk sebuah kalimat yang valid dalam bahasa kita memakai struktur: Subjek + Predikat + Objek. *Natural language* jauh melebii daripada *formal language* yang digunakan untuk logika kecerdasan buatan dan program komputer
2. Semantik: menggambarkan hubungan antara sintaks dan model komputasi. Meskipun teori Semantik secara umum sudah ada, ketika membangun sistem *natural language understanding* untuk aplikasi tertentu, akan digunakan representasi yang paling sederhana.
3. *Pragmatics*: menjelaskan bagaimana pernyataan yang ada berhubungan dengan dunia. Untuk memahami bahasa, agen harus mempertimbangan lebih dari hanya sekedar kalimat. Agen harus melihat lebih ke dalam konteks kalimat, keadaan dunia, tujuan dari speaker dan listener, konvensi khusus, dan sejenisnya.

## Pengertian *Natural Language Processing*

Ketika seseorang melihat atau membaca sebuat tulisan, orang tersebut akan menggunakan seluruh pengetahuan dan wawasan yang ia miliki untuk memahami tulisan tersebut. Tidak hanya sebatas tata bahasa atau *grammar*, namun lebih dari itu manusia akan mengolah informasi yang ia dapatkan dan menganalisa substansi atau konteks dari tulisan tersebut sehingga didapatkanlah pengetahuan yang baru. Maka, dikembangkan sebuah teknologi Artificial Intelligence (AI) yang berfokus pada bahasa alami, yaitu NLP. Menurut Liddy (2001) NLP adalah teknik-teknik komputasi yang didorong secara teoritis untuk menganalisa dan merepresentasikan bahasa alami pada tingkat analisis linguistik untuk mencapai pemrosesan bahasa seperti manusia. NLP hadir sebagai teknologi agar komputer dapat memahami dan memproses bahasa alami tanpa menghilangkan makna yang terkandung di dalamnya. Selain itu, NLP hadir untuk mempermudah interaksi antara manusia dan komputer. Banyak aplikasi yang menerapkan prinsip-prinsip NLP, salah satunya adalah fitur “Oke, Google” pada smartphone, pengguna kini dapat melakukan perintah-perintah seperti memutar lagu, menyetel alarm, hingga melakukan navigasi hanya cukup dengan perintah suara. Contoh lainnya adalah sentimen analisis, dimana komputer dapat menentukan sentimen dari sebuah respon yang diberikan, apakah itu berupa respon yang positif, negatif, ataupun netral. Contoh lainnya adalah aplikasi penerjemah, dimana komputer dapat menerjemahkan suatu bahasa kedalam bahasa lainnya secara otomatis.

Liddy (2001) mengungkapkan, setidaknya ada enam istilah yang sering digunakan dalam NLP:

1. *Part-of-speech tagging*:Sangatlah sulit untuk menandai istilah-istilah dalam suatu teks yang terkait dengan bagian tertentu dari suatu naskah (misalnya kata benda, kata kerja, kata sifat atau kata keterangan), karena bagian dari naskah tidak hanya bergantung pada definisi istilah tetapi juga pada konteks dimana teks digunakan.
2. *Text segmentation*: Beberapa bahasa tulisan, seperti bahasa mandarin, jepang, dan thai, tidak memiliki batasan kata. Dalam contoh ini, tugas *text-parsing* memerlukan idetifikasi terhadap batasan kata, yang seringkali merupakan tugas yang sangat sulit. Tantangan serupa dalam segmentasi naskah muncul ketika menganalisa bahasa verbal, karena suara menyajikan rangkaian huruf dan kata yang bercampur satu sama lain.
3. *Word sense disambiguation*:Banyak kata yang memiliki lebih dari satu arti. Memilih arti yang paling masuk akal hanya bisa dicapai dengan mempertimbangkan konteks di mana kata digunakan.
4. *Syntactic ambiguity*:Tata bahasa dalam bahasa alami seringkali ambigu yang artinya, ada berbagai struktur kalimat yang memungkinakan yang perlu dipertimbangkan. Memilih struktur yang paling tepat biasanya memerlukan paduan informasi kontekstual dan semantik.
5. *Imperfect or irregular input*:Aksen asing atau lokal dan berbagai hambatan vokal dalam pidato dan kesalahan ketik dan tata bahasa dalam teks-teks menyebabkan pengolahan bahasa bahkan lebih sulit.
6. *Speech acts*:Suatu kalimat seringkali dianggap sebagai suatu aksi oleh si pembicara. Struktur kalimatnya sendiri mungkin tidak berisi cukup informasi untuk mendefinisikan tindakan ini.

Liddy (2001) juga mengungkapkan bahwa penerapan-penerapan NLP seringkali tidak akan terlepas dari bidang-bidang berikut ini:

1. *Information Retrieval*: Ilmu untuk melakukan pencarian terhadap berbagai dokumen yang relevan, menemukan informasi tertentu didalamnya, dan menghasilkan metadata untuk isinya.
2. *Information Extraction* (IE): Sejenis ‘*information retrieval’* yang tujuannya adalah untuk mengekstrak secara otomatis informasi terstruktur, seperti data yang sudah terdefinisi dengan baik secara semantik dan secara kontekstual yang sudah terkelompok dari domain tertentu, dengan menggunakan berbagai dokumen tak-terstruktur yang bisa terbaca oleh mesin.
3. *Question-Answering*: Pekerjaan menjawab secara otomatis suatu pertanyaan yang diajukan dalam bahasa alami; yaitu, menghasilkan jawaban bahasa manusia ketika diberi pertanyaan bahasa manusia. Untuk mendapatkan jawaban terhadap pertanyaan, program computer bisa menggunakan baik database pra-terstruktur atau kumpulan dokumen bahasa alami (suatu ‘*text corpus*’ seperti *world wide web*).
4. *Summarization*: Tingkat NLP yang lebih lanjut, menghasilkan sebuah rangkuman singkat dari sebuah dokumen (dengan teks dalam jumlah besar) dengan waktu yang cepat.
5. *Machine Translation*: Penerjemahan otomatis dari satu bahasa manusia ke bahasa manusia yang lain.
6. *Dialogue Systems*: Berkomunikasi dengan komputer layaknya dengan manusia, yang memungkinkan dikembangkan lebih lanjut mengingat potensi yang dimiliki sistem ini sangat besar.
7. *Speech Recognition*: Mengubah kata-kata verbal menjadi *input* yang bisa terbaca oleh mesin. Dengan adanya sound clip dari orang yang sedang berbicara, sistem menghasilkan dikte dari teks.
8. *Natural Language Generation*: Sistem mengubah informasi dari database komputer (simbolik atau numerik) menjadi bahasa manusia yang bisa dibaca.
9. *Natural Language Understanding*: Sistem mengekstrak informasi yang mewakili makna dari suatu sumber teks (dokumen atau rangkuman).
10. *Speech Synthesis*: Perangkat yang mampu berbicara atau membaca teks.

NLP mempelajari tentang bagaimana memproses dan mengubah bahasa alami kedalam bentuk yang lebih sederhana dan terstruktur (simbol atau numerik) sehingga proses komputasi menjadi lebih mudah. Lebih dari itu, NLP bertujuan agar informasi yang kita dapatkan dari bahasa alami didapatkan juga oleh komputer, atau dengan kata lain komputer dapat memhami bahasa alami kita dengan memperhatikan batasan sintaks, semantik, gramatikal dan konteks.

## Pengertian *Natural Language Generation*

Berbeda dengan *Natural Language Processing* (NLP), *Natural Language Generation* (NLG) merupakan sistem yang mampu membangkitkan bahasa alami sebagai keluaran. NLG dapat diartikan juga sebagai proses penyusunan teks bahasa alami untuk memenuhi tujuan komunikatif tertentu. Sedangkan Bateman & Zock (2012) menuturkan bahwa NLG adalah sistem yang mampu menghasilkan informasi dalam bentuk text (linguistic) dengan berdasarkan data non-linguistic (data raw atau mentah yang terukur atau berasal dari serangkaian kejadian) agar mudah dipahami oleh manusia. Selain itu, hal yang paling membedakan antara NLP dan NLG adalah pemilihan informasi untuk keluaran, dimana sistem NLG harus memilih beberapa pilihan berupa teks yang akan disampaikan, singkatnya jika terdapat pemilihan informasi untuk keluaran berupa teks maka sistem tersebut lebih mendekati ciri-ciri sistem NLG dibandingkankan dengan NLP (Reiter, E, 2010). Seringkali, pemilihan teks keluaran dapat mempermudah proses pada sistem NLG, contohnya ada pilihan teks keluaran sebagai berikut:

1. Saya bertemu dengan Lira di jalan, dan pergi ke sekolah bersamanya.
2. Saya bertemu dengan Lira di jalan, dan pergi ke sekolah bersama Lira.

Kedua pilihan tersebut jika dilihat secara makna, tentu memiliki makna yang sama. Tetapi secara kasat mata tentu kita akan memilih pilihan a, dikarenakan lebih sederhana dan tidak terkesan kaku. Namun dalam perspektif komputer, pilihan a akan lebih lama pemrosesannya dibandingkan pilihan b. Karena, komputer harus mendefinisikan konteks dari kata bersamanya pada pilihan a, tentu sistem harus memiliki pemahaman untuk mendefinisikan konteks dari kata tersebut. Berbeda dengan pilihan b, sistem akan lebih mudah mendefinisikan bahwa subjek pulang pulang bersama Lira.

Maka tak heran, salah satu indikator kesuksesan dalam membangun sistem NLG adalah adanya pengetahuan teknik (menguraikan, merepresentasikan, dan mengatur proses informasi dari input) serta pengetahuan tentang keadaan dan kendala dari pengguna (prosesor informasi yang diterima) (Bateman & Zock, 2012). Bateman & Zock (2012) juga mengungkapkan bahwa setidaknya ada tiga definisi NLG, yaitu:

1. NLG sebagai *mapping problem*
2. NLG sebagai *problem of choice*
3. NLG sebagai *planning problem*

Dimana setiap definisi tersebut mewakili setiap layer dalam arsitektur utamanya. Berikut adalah penjelasan arsitektur utama dalam sistem NLG menurut (Bateman & Zock, 2012):

1. *Macroplanning:*  Penentuan konten, tujuan, dan *knowledge source* dari data mentah, lalu mengorganisir dan melakukan perencaan untuk membangun sebuah teks keluaran.
2. *Microplanning:* Mendeskripsikan kejadian-kejadian lalu mengelompokkan materi yang berkaitan sehingga dapat membangun informasi yang lebih terintegrasi dan ringkas.
3. *Surface realization:* Pengkontruksian gramatikal yang akan dipilih (pemilihan sintaksis), penandaan *Part-of-Speech* (POS), penambahan atribut-atribut seperti preposisi, dan penentuan bentuk akhir sehingga menjadi kata-kata yang tersusun (morfologi).
4. *Physical presentation:* Proses penambahan artikulasi, pungtuasi, dan layout mana yang akan digunakan.

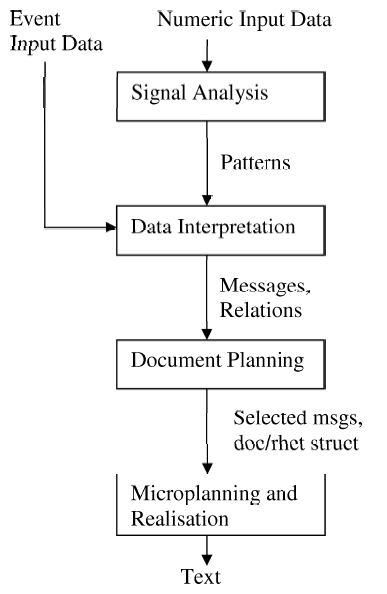
## Arsitektur sistem *Data-to-Text*

*Data-to-text* (D2T) adalah sistem *Natural Language Generation* (NLG) yang mampu menghasilkan teks dari input data *non-linguistic*, seperti data sensor dan serangkaian kejadian (Reiter, Ehud, 2011). Seperti namanya, sistem D2T mengkonversikan data raw (data mentah) dari sensor baik berupa data numerik, data log, ataupun data ­ *non-linguistic* menjadi sebuah teks yang mudah dipahami oleh pembaca data memuat informasi serta *knowledge* sesuai dengan data masukannya.

Berbagai pengaplikasian D2T sudah banyak dgunakan dalam berbagai bidang, seperti pada bidang peramalan cuaca misalnya, Putra *et al.* (2017) membuat sistem bernama *Data-to-text Weather Prediction* (DWP) yang mampu menghasilkan ringkasan berita klimatologis dan cuaca selama satu bulan serta memberikan informasi prediksi untuk satu hari berikutnya. Setelah itu, sistem DWP dikembangkan lagi oleh Abidin *et al.* ( 2018) , sehingga sistem D2T yang dibangun mampu menerima masukan berupa data *streaming.* Pada bidang lain, seperti bidang kesehatan dibangun sistem seperti *BabyTalk* yang mampu menghasilkan ringkasan teks dari data neonatal selama 45 menit kemudian ringkasan tersebut digunakan sebagai bahan pendukung keputusan presentasi modalitas yang terjadi saat itu (Gatt et al., 2009), selain itu terdapat sistem BT-Nurse yang mampu meringkas kejadian selama pergantian shift keperawatan berlangsung, berdasarkan hasil rekaman medis elektronik pasien (Hunter et al., 2011). Pada bilang ekonomi, terdapat *Knowledge-Based Report Generator* yang mampu menghasilkan laporan stok berdasarkan data stok produk (*non-linguistic*) suatu pasar (Kukich, 1983). Beberapa contoh tersebut membuktikan bahwa D2T menjadi pilihan yang tepat dalam berbagai bidang.

### Arsitektur Data-to-Text oleh Reiter (2011)

Seperti yang digambarkan pada Gambar 2.1, Reiter (2011) memaparkan bahwa setidaknya ada empat elemen utama yang diperlukan untuk membangun sebuah sisten D2T, yaitu *Signal Analysis*, *Data Interpretation*, *Document Planning*, dan *Microplanning and Realisation*. Elemen-elemen ini hampir mirip dengan arsitektur sistem NLG yang sudah dijelaskan pada sub-bab sebelumnya. Reiter (2011) memaparkan bahwasannya, perbedaan terbesar antara sistem D2T dan sistem NLG adalah sistem D2T harus menganalisis dan menginterpretasikan data masukannya, begitu juga dengan menentukan bagaimana proses penyampaiannya dari segi linguistik.

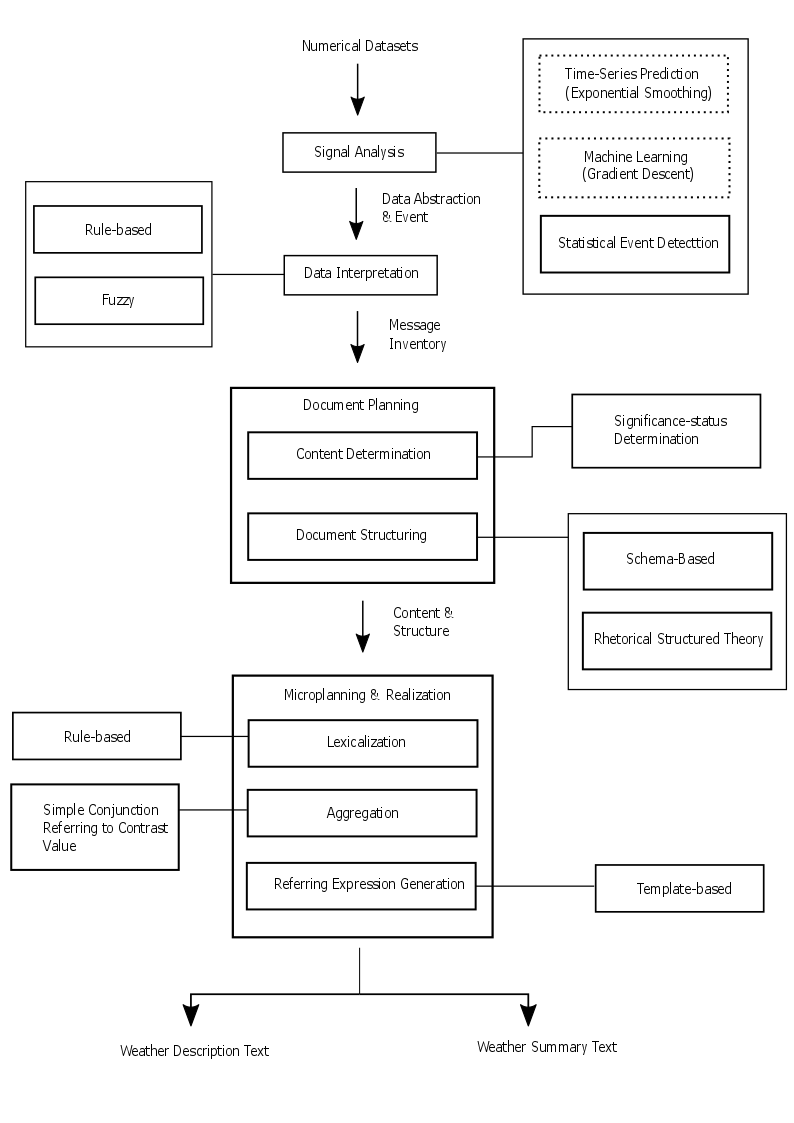


Gambar 2. Arsitektur D2T oleh (Reiter, Ehud, 2011)

Pada gambar 2.1 Reiter (2011) memaparkan bahwa sistem D2T terdiri dari 4 fase utama, yaitu *Signal Analysis*, *Data Interpretation, Document Planning,* dan *Microplanning.* Dimana proses ini diterapkan pada penelitian BT45 (Portet et al., 2009) dan (Hunter et al., 2011).

### Arsitektur Data-to-Text oleh Putra et al., (2017)

Dalam penelitian DWP, dilakukan pengembangan model dengan memodifikasi beberapa elemen, sehingga output yang dihasilkan dibagi menjadi dua kategori berupa *Weather Description Text* dan *Weather Summary Text*. Gambar 2.2 menjelaskan bagaimana sistem DWP yang dikembangkan oleh Putra *et al.*, (2017), dimana ada beberapa penambahanan yaitu penggunaan *Machine Learning* yaitu *Gradient Descent* dan pendekatan *Time Series* yaitu *Exponential Smoothing*, dan juga terdapat pendeteksian *Statistical Event*. Proses *Signal Analysis* menghasilkan *Data Abstraction and Event,* yang menjadi masukan bagi proses *Data Interpretation.* Proses yang diklaim oleh Reiter (2011) sebagai pembeda antara sistem D2T dan sistem NLG ini, merupakan proses untuk menginterpretasikan data masukan. DWP menggunakan 2 cara untuk menginterpretasikan datanya, yang pertama yaitu *Rule-Based* dan *Fuzzy* sehingga dihasilkanlah *Message Inventory* yang akan diproses pada proses selanjutnya yaitu *Document Planning.* Pada proses *Document Planning*, terdapat *Content Determination* dan *Document Structuring*. *Content Determination* atau pemilihan konten dilakukan dengan memilah seberapa pentingnya status suatu pesan atau nama lainnya yaitu *Significance Status Determination.* Setelah ditentukan seberapa pentingnya status suatu pesan, maka pesan-pesan tersebut digabungkan pada sebuah proses bernama *Document Structuring*. Setelah ditentukan statusnya, maka pesan tersebut diproses pada proses *Document Planning,* sehingga dihasilkan *Content* dan *Structure*, yang nantinya akan diproses pada tahap *Microplanning* dan *Realization,* sehingga dihasilkanlah sebuah teks yang terdiri dari *Weather Description Text* dan *Weather Summary Text.*



Gambar 2. Arsitektur sistem DWP (Putra et al., 2017)

Tahapan pada gambar 2.2 akan dijelaskan secara terperinci pada poin-poin berikut:

1. *Signal Analysis*

Tahapan pertama dalam membangun sistem D2T adalah *Signal Analysis.* Tahapan ini bertujuan untuk mendeteksi pola-pola yang terdapat pada data masukan. Sehingga sistem dapat memproses data dengan bentuk simbolis (diskrit) dibandingkan memprosesnya dalam bentuk numerik. Namun, dalam beberapa kasus tahapan ini menjadi opsional, ketika data masukan sudah terstruktur dalam bentuk kejadian-kejadian diskrit, contohnya seperti data rekord kejadian medis, atau *log file*. Jika data masukan sudah berbentuk simbolis (diskrit) sepeti yang sudah dijelaskan sebelumnya, maka proses *Signal* *Analysis* ini tidak perlu dilakukan (McKeown et al., 1994).

Pada dasarnya tahap *Signal Analysis* merupakan proses untuk menganalisa data masukan yang berupa numerik sehingga dihasilkan sebuah informasi berbentuk simbolis yang akan disampaikan. Contohnya untuk kasus pembangkitan berita cuaca pada DWP, data masukan berupa seluruh data klimatologi dan kualitas udara selama satu tahun, lalu data tersebut dianalisa dan dihasilkanlah sinyal-sinyal dari data tersebut seperti curah hujan terbesar, kondisi kualitas udara hari ini, serta bagaimana kondisi curah hujan dan temperatur bulan ini dibandingkan dengan curah hujan dan temperatur satu tahun penuh, dan lain-lain. Pada penmbangunan sistem DWP (Putra et al., 2017) , dilakukan pendekatan *Time Series* dan pendekatan *Statistical Tools* dalam menganalisa data masukan sehingga dihasilkan sebuah pesan diskrit untuk satu bulan. Contoh penggunaan statistik (*mean, min, max, sum*) dalam bahasa R digambarkan pada pada gambar 2.3 di bawah ini.

|  |
| --- |
| ...  **for(**i **in** i**:**n**){** #Loping sebanyak jumlah parameter  max\_amt**[**i**]** **<-** max**(**LM**[**i**])** #Nilai Max pd parameter ke-i  min\_amt**[**i**]** **<-** min**(**LM**[**i**])** #Nilai Min pd parameter ke-i  sum\_amt**[**i**]** **<-** sum**(**LM**[**,i**])** #Total nilai dari parameter ke-i  ....  **}**  #Rata-rata 1 bulan terakhir untuk setiap parameter  LMmean\_result **<-** colMeans**(**xLM**)**  .... |

Gambar 2. Contoh implementasi *Signal Analysis* (Putra et al., 2017)

1. *Data Interpretation*

Langkah ke-dua setelah mendapatkan sinyal-sinyal dari proses *signal analysis*, yang harus dilakukan kemudian adalah menerjemahkan sinyal-sinyal yang telah didapatkan tersebut kedalam pesan dan menganalisis apakah ada relasi antara pesan-pesan yang didapatlan. Jadi, tujuan utama dari *Data Interpretation* ini adalah untuk memetakan pola dan *event* dasar menjadi pesan dan relasi dimana manusia membutuhkannya.

|  |
| --- |
| membership\_check **<-** **function(**partition,v,pname,oname**){**  i**=**1; n**=**length**(**partition**)**  membership\_value **<-** c**(**1**)**  **for(**i **in** i**:**n**){**  **....**  **if((**v**<**a**)||(**v**>**d**)){**  membership\_value**[**i**]** **<-** 0  **}**  **if((**v**>=**a**)&&(**v**<=**b**)){**  membership\_value**[**i**]** **<-** **(** **(**v**-**a**)** **/** **(**b**-**a**)** **)**  **}**  **if((**v**>**b**)&&(**v**<=**c**)){**  membership\_value**[**i**]** **<-** 1  **}**  **if((**v**>**c**)&&(**v**<=**d**)){**  membership\_value**[**i**]** **<-** **(** **(**d**-**v**)** **/** **(**d**-**c**)** **)**  **}**  **}**  i**=**1; biggest**=**0; part**<-**"a"  **for(**i **in** i**:**n**){**  **....**  **if(**biggest**<=**membership\_value**[**i**]){**  biggest **<-** membership\_value**[**i**]**  part**<-**pname**[**i**]**  **}**  **}**  return **(**part**)**  **}**  ....  Temperature\_partition **<-** c**(**"very cold.","cold.","warm.","hot.","very hot."**)**  Temperature\_interval **<-** list**(**very\_cold**=**c**(**a**=**0,b**=**0,c**=**5,d**=**10**)**,cold**=**c**(**a**=**5,b**=**10,c**=**15,d**=**20**)**,  warm**=**c**(**a**=**15,b**=**20,c**=**25,d**=**30**)**, hot**=**c**(**a**=**25,b**=**30,c**=**35,d**=**40**)**,very\_hot**=**c**(**a**=**35,b**=**40,c**=**45,d**=**50**))**  Temperature\_interval **<-** membership\_partition**(**Temperature\_interval,"Temperature"**)**  InterpretationResult\_temperature **<-** membership\_check**(**Temperature\_interval,as.double**(**climatePredictionResult**[**"Average.Temperature"**])**,Temperature\_partition," "**)**  ..... |

Gambar 2. Contoh implementasi data interpretation Rainfall DWP (Putra et al., 2017)

Sebagai contoh, misalnya terdapat data suhu udara hari ini senilai 40oC. Maka dengan melalui serangkaian proses interpretasi data ini, angka 40oC diiterpretasikan menjadi pesan *“very hot”*, seperti implementasi pada DWP (Putra et al., 2017) pada gambar 2.4 dengan berdasarkan pada *fuzzy membership function* (Ramos-Soto et al., 2016a).

1. *Document Planning*

Langkah ke-tiga yang dilakukan dalam arsitektur ini adalah menentukan *event* mana yang akan disebutkan didalam teks, dan juga didalam struktur dokumen. Analisis sinyal dan *Data Interpretation* dapat menghasilkan sejumlah pesan, pola, dan *event* yang banyak, tetapi teks biasanya terbatas untuk mendeskripsikan sebagian kecil pesan. Perencanaan dokumen harus menentukan pesan mana yang sebenarnya dapat dikomunikasikan dalam bentuk teks, pilihan ini didasarkan pada genre dan domain. Dalam langkah ini juga harus direncanakan bagaimana pesan disebutkan dalam sebuah teks yang berkaitan antara satu dengan yang lainnya.

Menurut Reiter (2011) bahwa serangkaian proses *Document Planning* ini diantaranya adalah membagi tugas menjadi beberapa bagian berikut:

1. *Content Determintation*

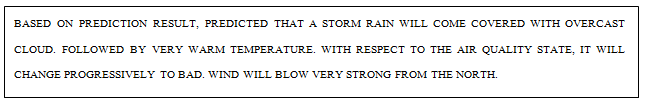
Tahap ini melakukan pemilihan *event* atau pesan yang didapatkan, idenya adalah membagi status pesan menjadi *Routine Message* dan *Significant Event Message*. *Routine Message* merupakan pesan-pesan yang akan selalu disampaikan disetiap pembangkitan kalimat, sedangkan *Significant Event Message* adalah pesan-pesan yang hanya akan disampaikan jika dan hanya jika indikasi pembangkitan dipenuhi. Artinya, *Significant Event Message* hanya disampaikan saat kondisi tertentu. DWP menerapkan *Significant Event Message* dalam menentukan event suatu hujan terjadi berturut turut secara *extreme* atau tidak seperti pada gambar 2.5.

|  |
| --- |
| RainExtremeMessage\_function **<-** **function(**Dataset**){**  **if(**interpreterRainfall **==** "no rain" **||** interpreterRainfall**==**"light rain" **||** interpreterRainfall**==**"moderate rain"**){**  return**(**"x"**)** #artinya tidak extreme  **}else{**  **.....** #get repeated date  **return (date)**  **}**  } |

Gambar 2. Contoh Content Determination Significant Event Message DWP (Putra et al., 2017)

1. *Documenet Structuring*

*Document Structuring* adalah proses penentuan bagaimana struktur pesan yang akan disampaikan. Urutan pesan-pesan ditentukan sesuai dengan relasinya masing-masing. Ada beberapa cara untuk membuat struktur dokumen, salah satunya adalah dengan menggunakan skema. Skema tersebut dibuat berdasarkan *Target Text* yang ingin dicapai atau berdasarkan penalaran (Reiter, Ehud, 1996). Contoh “*Target Text*” dalam DWP dapat dilihat pada gambar 2.6.

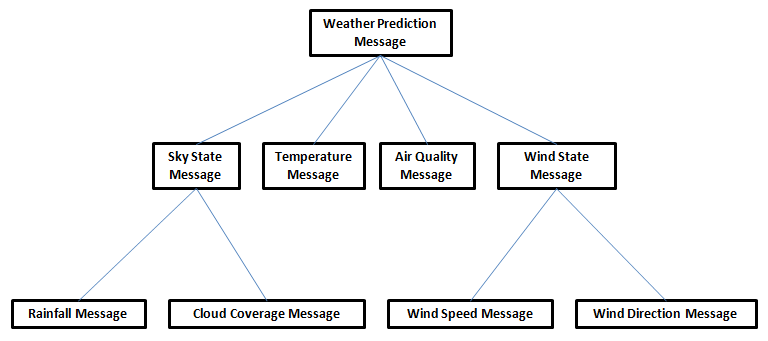


Gambar 2. Contoh Target Text DWP (Putra et al., 2017)

Reiter (1996) menjelaskan bahwa dengan berdasarkan *Target Text* pada gambar 2.6 dilakukan pembuatan skema dengan cara sebagai berikut:

* + - 1. Ambil contoh sejumlah teks yang sama dalam *Target Text*.
      2. Identifikasi terhadap pesan-pesan yang ada, tentukan bagaimana setiap pesan dapat dibangun berdasarkan data.
      3. Mengusulkan aturan atau struktur yang menjelaskan mengapa pesan “x” ada dalam teks A tetapi tidak ada didalam teks B. Penentuan ini lebih mudah jika disusun dalam bentuk seperti taksonomi atau pohon.
      4. Diskusikan hasil analisis bersama pakar.

Dengan berdasarkan tahapan diatas, maka didapat skema dalam bentuk pohon seperti pada gambar 2.7, dimana *Weather Prediction Message* terdiri dari empat komponen, yaitu *Sky State Message*, *Temperature Message*, *Air Quality Message*, dan *Wind State Message*.



Gambar 2. Contoh skema dalam bentuk *tree* berdasarkat   
*Target Text* DWP (Putra et al, 2017)

1. *Microplaning and Realisation*

Langkah ke-empat adalah membangkitkan bahasa alami dalam bentuk teks didasarkan pada konten dan struktur yang dipilih pada tahap perencanaan dokumen. Tahap *Microplanning* dan realisasi harus menentukan bagaimana sebenarnya mengekspresikan apa yang telah disusun pada tahap-tahap sebelumnya (*signal analysis, data interpretation,* dan *document planning*).

Dalam proses *Microplanning*, pesan-pesan yang disampaikan akan melalui serangkaian proses berikut:

1. *Lexicalisation*

Proses *lexicaisation* adalah bagaimana melakukan pemilihan kata atau frase yang akan digunakan dalam mendeksripsikan segala hal, contohnya mendeskripsikan relasi, tren, dan kemungkinan. DWP menjelaskan tren yang terjadi pada kualitas udara dengan membandingkan dua baris data, yakni data ke-*n* dan data ke-(*n-1*), dimana *n* merupakan jumlah baris data, seperti pada gambar 2.8.

|  |
| --- |
| ....  TrendDesc\_template **<-** **function** **(**IVL,data**){**  **if((**IVL**[**1**]==**"0"**)&&(**IVL**[**2**]==**"0"**)){**  TrendDesc **<-** change\_word\_bank\_AQ**(**"stable"**)**  **}**  **if(((**IVL**[**1**]==**"+"**)&&(**IVL**[**2**]==**"-"**))||((**IVL**[**1**]==**"-"**)&&(**IVL**[**2**]==**"+"**))){**  TrendDesc **<-** change\_word\_bank\_AQ**(**"mediumChange"**)**  **}**  **....**  return**(**TrendDesc**)**  **}**  .... |

Gambar 2. Contoh implementasi Lexicalisation tren DWP (Putra et al., 2017)

1. *Aggregation*

Proses *aggregation* adalah bagaimana setiap kata digabungkan menjadi frase, bagaimana frase dihubungkan menjadi kalimat, dan bagaimana kalimat digabungkan menjadi paragraf. Intinya, proses *Aggregation* adalah menghubungkan pesan yang didapat dengan menggunakan beberapa teknik. Ada beberapa teknik yang dapat dilakukan untuk proses *aggregation,* salah satu diantaranya adalah dengan menggunakan *simple conjunction*, seperti dalam pada gambar 2.9.

|  |
| --- |
| ....  Contrast\_lexicalisation1 **<-** **function(**msg1,msg2**){**  **if(**msg1**[**2**]==**msg2**[**2**]){**  return**(**"and"**)**  **}else{**  return**(**"but"**)**  **}**  **}**  .... |

Gambar 2. Contoh *Simple Conjunction Referring to  
 Contrast Value* (Putra et al., 2017)

1. *Referring Expression Generation*

Proses ini berisi mengenai bagaimana sistem dapat merujuk informasi tertentu kepada sebuah subjek. Contohnya: “Suhu hari ini tergolong sangat panas”, sistem dikondisikan agar dapat menyampaikan bahwa informasi “sangat panas” adalah penjeasan informasi dari subjek “Suhu”. Salah satu contoh penerapan dengan cara *hardcode* pada kode program seperti pada gambar 2.10, dimana “*The wind for the month was*” merupakan subjek dari kalimat yang akan ditampilkan.

|  |
| --- |
| ....  MonthlyMsg4\_aggregation**<-function(**msg1,msg2**){**  **if(**msg2**==**"false"**){**  msg**<-**paste**(**"The wind for the month was",msg1,"in average."**)**  **}**  **else{**  msg**<-**paste**(**"The wind for the month was",msg1,"in average, but",msg2**)**  **}**  return**(**msg**)**  **}**  .... |

Gambar 2. Contoh Referring Expression Generation (Putra et al., 2017)

1. *Structure Realisation*

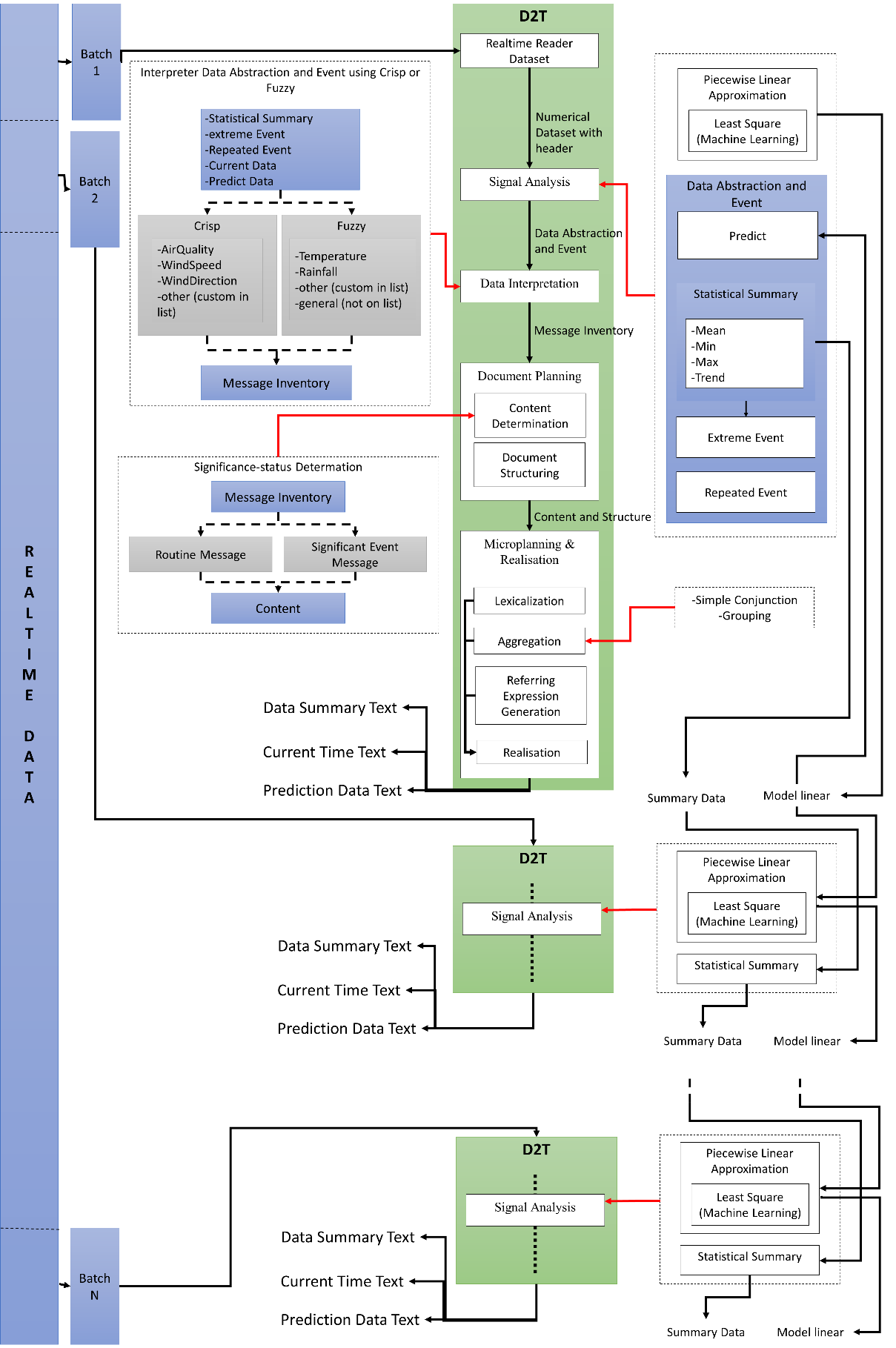
Pada proses ini, setiap struktur yang telah dibuat dalam proses *dokumen planning* direalisasikan sehingga menghasilkan teks dalam bentuk aktual (Reiter, Ehud, 1996). Contohnya, merealisasikan struktur teks dalam bahasa pemrograman menjadi teks aktual dalam HTML, LaTeX, RTF, SABLE, dan lain-lain, seperti pada gambar 2.11, yang merealisasikan sturktur dalam bahasa R dengan menggabungkan hasil seleksi konten, kemudian menampilkan hasil kedalam HTML dengan bantuan *package* Shiny.

|  |
| --- |
| Structure\_Realization\_predict **<-** **function(){**  ....  Sky\_State **<-** Sky\_Agg**(**Rain\_State,Cloud\_State**)**  Sky\_Intro **<-** Prediction\_Intro**()**  Sky\_Sentence **<-** paste**(**Sky\_Intro,Sky\_State**)**  Temperature\_Intro **<-** Temperature\_Intro**()**  Temperature\_Sentence **<-** paste**(**Temperature\_Intro, Temperature\_State**)**  ....  **}**  ....  library**(**shiny**)**  .....  tags**$**div**(**class**=**"col-lg-8",  tags**$**h2**(**"Weather Prediction News"**)**,  tags**$**p**(**align**=**"justify",HTML**(**'&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;'**)**,  Prediction\_Result  **)**,  tags**$**hr**(**class**=**"style13"**)**,  tags**$**p**(**align**=**"justify",HTML**(**'&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;'**)**,    MonthlyMsg  **)**  **)**  ..... |

Gambar 2. Contoh implementasi Structure Realisation DWP (Putra et al., 2017)

### Arsitektur Data-to-Text oleh (Abidin et al., 2018)

Pada sistem D2T yang dikembangkan oleh (Abidin et al., 2018). Diperkenalkan sebuah sistem yang dapat menerima masukan berupa data *streaming*. Seperti yang bisa kita lihat pada gambar 2.12 terdapat perbedaan pada tahap *Signal Analysis* dan terdapatnya proses bernama *Realtime Reader Dataset,* dimana data yang masuk akan terlebih dahulu dihandle pada tahap *Realtime Reader Dataset* sebelum diproses pada tahap *Signal Analysis*. Setelah data masukan diproses pada tahap *Realtime Reader Dataset*, data masukan selanjutnya akan diproses pada tahap *Signal Analysis,* dimana data *streaming* akan diproses dan dipecah berdasarkan *batch-batch*. Sehingga proses prediksi data dilakukan menggunakan teknik Piecewise Linear Approximation (PLA) dengan metode Least Square yang diperkenalkan oleh Palpanas *et al.*, (2004).



Gambar 2. Arsitektur D2T untuk data *streaming* (Abidin et al., 2018)

Abidin *et al.*, (2018) menuturkan bahwa pada proses ini dihasilkan model linear dari setiap batch data dan model linear untuk keseluruhan data (batch pertama hingga batch ke-N), hal ini dilakukan karena data yang digunakan merupakan data Time Series sehingga antara batch satu dan lainnya memiliki keterikatan. Penerapan PLA dilakukan untuk merepresentasikan data streaming lampau dan menggabungkannya menjadi sebuah garis linear, mengingat data yang digunakan hanya dalam satu kali proses dan selanjutnya data terhapus untuk mengatasi masalah storage berlebihan dikarenakan kemunculan data sangat cepat dan memiliki ukuran yang tidak sedikit bila kita menggabungkan setiap data stream.

Untuk mengimplemetasikan *Realtime Reader Dataset* proses ini (Abidin et al., 2018) menggunakan Javascript animation dan AJAX dalam proses pengecekan dataset baru (fungsi checkFile), bila terdapat dataset baru, maka sinyal akan dikirimkan melalui fungsi RunR() dan kemudian D2T\_Main.R akan dijalankan. Pada gambar 2.13 terlihat animasi dijalankan setiap 1000 *milisecond* untuk melakukan proses pengecekan dataset, jika terdapat dataset maka akan dikirim sinyal “*run*” kepada contrlorrer RunR dengan metode *post*, jika tidak terdapat dataset baru, maka cek konten yang ditampilkan apakah sesuai dengan konten terakhir yang dihasilkan oleh D2T\_Main.R atau tidak.

|  |
| --- |
| setTimeout**(**checkFile**,**1000**);** //Animation Checker  ***function*** checkFile**(){**  // Checking Dataset  $**.**ajax**({**  type**:** 'HEAD'**,**  url**:** 'http://localhost/D2T/DatasetsRealTime/Dataset.csv'**,**  success**:** ***function*()** **{**  // This is a new dataset  // Send signal to controller RunR for running D2T\_Main  $**.**ajax**({**  url**:**"http://localhost/D2T/RunR"**,** //the with php script  type**:** "post"**,** //request type,  dataType**:** 'json'**,**  data**:** **{**exec**:** "run"**}**  **});**  setTimeout**(**checkFile**,** 5000**);**  **},**  error**:** ***function*()** **{**  // No dataset found  // Check last update file content  $**.**ajax**({**  type**:** 'HEAD'**,**  url**:** 'http://localhost/D2T/Result/tempTime.json'**,**  success**:** ***function*()** **{**  // Validation Content in Web  checktempTime**();**  **},**  error**:** ***function*()** **{**  console**.**log**(**"nothing change"**);**  **}**  **});**  setTimeout**(**checkFile**,**1000**);**  **}**  **});**  **}** |

Gambar 2. *Realtime animation for check file* (Abidin et al., 2018)

Sedangkan pada gambar 2.14 terlihat bahwa sinyal “*run*” diproses untuk menjalankan fungsi yang dapat mengeksekusi D2T\_Main.R untuk membaca dataset.

|  |
| --- |
| **public** **function** executeR(){  // Execute D2T\_Main.R  exec ("\".\\R-3.4.0\\bin\\Rscript.exe\" .\\D2T\_Main.R 2>&1", $output);  // echo '<pre>', join("\r\n", $output), "</pre>\r\n";  }  **public** **function** Index()  {  **if**($\_POST["exec"] == "run"){  $this->executeR();  }  } |

Gambar 2. *Execute* D2T\_Main.R (Abidin et al., 2018)

Setelah proses membaca *file* selesai, maka dataset dihapus untuk meminimalisir penggunaan *hardisk* yang besar seperti pada gambar 2.15.

|  |
| --- |
| # Read data  dataset **<-** read.table**(**file**=**"DatasetsRealTime/Dataset.csv", sep**=**",", header**=TRUE)**  fn **<-** "DatasetsRealTime/Dataset.csv"  # Delete data  **if** **(**file.exists**(**fn**))** file.remove**(**fn**)** |

Gambar 2. *Read and Remove Dataset* *in* R (Abidin et al., 2018)

Sedangkan untuk menyimpan hasil, Abidin et al., (2008) menggunakan library “*jsonlite*” dengan menggunakan fungsi toJSON() kemudian menuliskan data pada file dengan fungsi write() dan write.csv() seperti pada gambar 2.16

|  |
| --- |
| library**(**jsonlite**)**  timeInterval **<-** toJSON**(**timeInterval**)**  resumeResult **<-** toJSON**(**resumeResult**)**  currentResult **<-** toJSON**(**currentResult**)**  predictResult **<-** toJSON**(**predictResult**)**  columnName **<-** toJSON**(**columnName**)**  write.csv**(**statisticalResume, file **=**  "Result/statisticalResume.csv",row.names**=FALSE)**  write**(**columnName, file**=**'Result/columnName.json'**)**  write**(**timeInterval, file**=**'Result/timeInterval.json'**)**  write**(**now, file**=**'Result/tempTime.json'**)**  write**(**resumeResult, file**=**'Result/resumeResult.json'**)**  write**(**currentResult, file**=**'Result/currentResult.json'**)**  write**(**predictResult, file**=**'Result/predictResult.json'**)** |

Gambar 2. *Write Result* JSON and csv *in* R (Abidin et al., 2018)

Kemudian untuk menampilkan data pada web, Abidin et al., (2018) menggunakan AJAX $.getJSON() dan menampilkan hasilnya dengan append() seperti pada gambar 2.17.

|  |
| --- |
| $**.**getJSON**(**'http://localhost/D2T/Result/tempTime.json'**,** ***function*(**data**)** **{**  $**(**'#tempTime'**).**val**(**data**);**  **});**  $**.**getJSON**(**'http://localhost/D2T/Result/resumeResult.json'**,** ***function*(**data**)** **{**  $**(**'#newsResume'**).**append**(**"<p>&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp"**+**data**+**"</p>"**);**  **});**  $**.**getJSON**(**'http://localhost/D2T/Result/currentResult.json'**,** ***function*(**data**)** **{**  $**(**'#newsCurrent'**).**append**(**"<p>&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp"**+**data**+**"</p>"**);**  **});**  $**.**getJSON**(**'http://localhost/D2T/Result/predictResult.json'**,** ***function*(**data**)** **{**  $**(**'#newsPredict'**).**append**(**"<p>&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp"**+**data**+**"</p>"**);**  **});** |

Gambar 2. *Get* JSON *in* AJAX (Abidin et al., 2018)

## Penelitian Terkaitsistem *Data-to-Text*

Penelitian terkait dengan sistem *Data-to-text* akhir-akhir ini telah menjadi perhatian tertentu bagi para peneliti, ditunjukan dengan banyaknya penelitian baru terkait dengan bidang ini (D2T dan NLG). Beberapa penelitian sejauh ini mengenai *Data-to-text* dapat dilihat pada Tabel 2.1.



Tabel 2. Penelitian terkait D2T dan NLG

| Referensi | Metode *Content Selection* | Domain | Sumber Data |
| --- | --- | --- | --- |
| (Kukich, 1983) | Rule-Based | Market | Database |
| (Boyd, 1998) | No Content Selection | Weather | Database |
| (Sripada, S.G. et al., 2001) | Two Stage model: (1) Domain Reasoner; (2) communicative reasoner | Weather, Oil Rigs | Sensor data, Numerical Data |
| (Sripada, Somayajulu G. et al., 2003) | Gricean Maxims | Weather, Gas Turbins, Heatlh | Sensor data |
| (Hallett et al., 2006) | Rule-Based | Heatlh | Database |
| (Yu et al., 2007) | Rules derived from corpus analysis and main knowledge | Gas Tourbines | Sensor |
| (Sripada, Somayajulu G. & Gao, 2007) | Decompression Models | Dive | Sensor |
| (Turner et al., 2008) | Decision Tree | Georeferenced Data | Database |
| (Gatt et al., 2009) | Rule-Based | Health | Sensor |
| (Thomas et al., 2012) | Document Schema | Georeferenced Data | Datatbase |
| (Demir et al., 2012) | Rule-based | Domain Indepentent | Graph-database |
| (Reddington & Tintarev, 2011) | Threshold-based rules | Assitive Technology | Sensor |
| (Banaee et al., 2013) | Rule-based | Health | Grid of sensor |
| (Schneider et al., 2013) | Rule-based | Health | Sensor |
| (Ramos-Soto et al., 2016b) | *Fuzzy*-sets | Weather | Database |
| (Gkatzia et al., 2016) | Rule-based | Weather | Numerical data with assigned probabilities |
| (Putra et al., 2017) | Rule-Based and Fuzzy | Weather | Numerical data |
| (Abidin et al., 2018) | Rule-Based and Fuzzy | General | Data Streaming |

## *Machine Learning*

*Machine Learning* termasuk dalam bagian dari ilmu komputer yang dapat membelajarkan komputer sehingga memiliki kemampuan untuk belajar tanpa diprogram secara eksplisit (Samuel, 1959). *Machine Learning* merupakan bagian dari kecerdasan buatan yang berfokus dalam mempelajari, mendesain, dan membuat sebuah algoritma yang memiliki kemampuan untuk belajar dari data yang ada. Agar sebuah perangkat memiliki kecerdasan, maka komputer atau mesin tersebut harus dapat belajar. Dengan kata lain, *Machine Learning* berisi tentang keseluruhan proses pembelajaran komputer atau mesin sehingga mesin menjadi cerdas dan dapat belajar seiring dengan berkembangnya data masukan. *Machine Learning* sudah ada dan mulai digunakan sejak 50 tahun yang lalu, pengaplikasiannya pun sudah digunakan dalam berbagai bidang seperti bidang ekonomi, keilmuan, industri dan sebagainya*.*

Salah satu implementasi *Machine Learning* yang pernah dilakukan oleh Arthur Samuel sekitar 59 tahun yang lalu yaitu pembuatan permainan catur dengan computer (Samuel, 1959). Catur dipilih karena permainan sangat mudah tetapi memerlukan strategi yang bagus. Samuel membuat permainan catur ini berdasarkan pohon penyelesaian. Pencarian penyelesain dilakukan dengan menyusuri pohon permasalahan sampai mendapatkan solusinya*.*

Awal ditemukannya *Machine Learning* yaitu pada tahun 1914, seorang ilmuan dari Spanyol, Torres y Quevedo, membuat sebuah mesin catur yang dapat mengalahkan atau melakukan skakmat pada raja lawan dengan sebuah ratu dan raja (Shannon, 1950). Perkembangan secara sistematis kemudian dimulai segera setelah diketemukannya komputer digital.

Artikel ilmiah pertama tentang Kecerdasan Buatan ditulis oleh Alan Turing pada tahun 1950 (Turing, 1950), dan kelompok riset pertama dibentuk tahun 1954 di Carnegie Mellon University oleh Allen Newell and Herbert Simon. Namun bidang Kecerdasan Buatan baru dianggap sebagai bidang tersendiri di konferensi Dartmouth tahun 1956, di mana 10 peneliti muda memimpikan mempergunakan komputer untuk memodelkan bagaimana cara berfikir manusia. Mereka berhipotesis bahwa mekanisme berfikir manusia dapat secara tepat dimodelkan dan disimulasikan pada komputer digital.

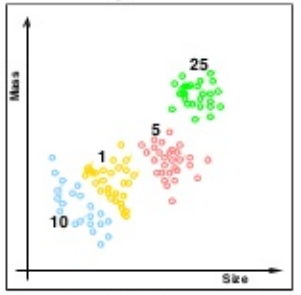
*Machine Learning* memiliki beberapa tipe dengan proses pembelajaran yang berbeda, tipe-tipe tersebut akan dijelaskan pada sub-bab berikutnya.

### *Supervised Learning*

Tugas dari *Supervised Learning* terdiri dari pembangunan model yang memetakan nilai *input* pada nilai ouput dimana *data* training tersedia (Riza, 2015). *Supervised Learning* adalah *Machine Learning* yang membutuhkan label sebagai tujuan dari pelatihan data atau *data training* (Mohri et al., 2012). *Supervised Learning* merupakan suatu pembelajaran yang terawasi, dimana jika *output* yang diharapkan telah terdapat pada daftar yang diketahui sebelumnya. Pada metode ini, setiap pola yang diberikan kedalam model *Machine Learning* telah diketahui *output*nya. Contoh algoritma dari salah satu bagian dari *Machine Learning* yaitu jaringan saraf tiruan yang mernggunakan metode *Supervised Learning* adalah hebbian (hebb rule), perceptron, adaline, boltzman, hapfield, dan backpropagation*.*

Berikut ini adalah beberapa contoh penerapan tipe *Machine Learning*, *Supervised Learning*:

1. Klasifikasi: adalah sebuah metode untuk menyusun data secara sistematis menurut aturan-aturan yang telah ditetapkan sebelumnya (Athoillah et al., 2015). Dengan melakukan klasifikasi, dari data yang telah ada dapat dibuat sebuah model prediksi dengan *output* kelas. Beberapa algoritma klasifikasi yang cukup terkenal adalah k-Means, SVM, EM, Naïve Bayes, dan kNN.
2. Regresi: analisis regresi adalah salah satu metode statistik untuk memprediksi nilai dari satu atau lebih variabel respon/dependen dari satu set variabel prediktor/independen (Härdle & Simar, 2007).



Gambar 2. Contoh *Supervised Learning* pada pengenalan koin

Pada Gambar 2.18, diperlihatkan bagaimana klasifikasi dari pengenalan koin, terlihat sangat jelas lokasi bagian dari tiap kelas, seperti koin dengan nilai sepuluh terpisah dipaling bawah dengan warna biru, koin dengan nilai satu yang berwarna kuning tidak bercampur dengan yang lainnya, dan seterusnya. Beberapa contoh penerapan *Supervised Learning* yaitu pada kasus klasifikasi sentimen (Ye et al., 2009), prediksi virus hepatitis B (Hospital, 2003), dan *Gait Event Detection* (Williamson & Andrews, 2000).

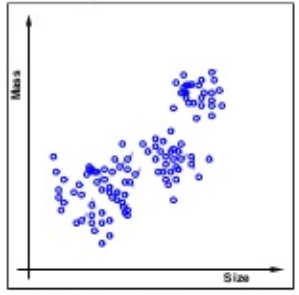
* + - 1. **Algoritma Gradient Descent**

Algoritma *Gradient Descent*adalah algoritma optimasi untuk menemukan *minimum* lokal dari fungsi menggunakan *gradien descent*, diambil  langkah sebanding dengan negatif dari gradien (atau perkiraan gradien) dari fungsi pada titik sekarang (Riza et al., 2016). Jika diambil langkah sebanding dengan gradien positif, maka akan didapatkan maksimum lokal fungsi tersebut; prosedur ini kemudian dikenal sebagai *gradient ascent*. Algoritma *Gradient Descent* yang digunakan dalam penelitian ini mengacu pada penerapan algoritma *Gradient Descent* pada penelitian DWP untuk mengatasi *missing value* sehingga sistem tetap berkerja tanpa galat, meskipun pada data masukan terdapat nilai NA atau *missing value* (Putra et al., 2017)*.* Dalam penelitian ini, *missing value* *handling* dilakukan saat pra-proses data menggunakan *package mice* dalam R.

### *Unsupervised Learning*

*Unsupervised Learning* terdiri dari pembangunan model dari *data training* dengan tidak mengandung nilai *output* (Riza, 2015). *Unsupervised Learning* merupakan pembelajan yang tidak terawasi dimana tidak memerlukan target *output*. Teknik ini menggunakan prosedur yang berusaha untuk mencari partisi dari sebuah pola. *Unsupervised Learning* mempelajari bagaimana sebuah sistem dapat belajar untuk merepresentasikan pola *input* dalam cara yang menggambarkan struktur statistikal dari keseluruhan pola *input*. Berbeda dari *Supervised Learning*, *Unsupervised Learning* tidak memiliki target *output* yang eksplisit atau tidak ada pengklasifikasian *input.*

Dalam *Machine Learning*, teknik *Unsupervised* sangat penting. Hal ini dikarenakan cara kerjanya mirip dengan cara bekerja otak manusia. Dalam melakukan pembelajaran, tidak ada informasi dari contoh yang tersedia. Oleh karena itu, *Unsupervised Learning* menjadi esensial. Pada metode ini tidak dapat ditentukan hasil seperti apa yang diharapkan selama proses pembelajaran, nilai bobot yang disusun dalam proses range tertentu tergantung pada nilai *output* yang diberikan. Tujuan metode *Unsupervised Learning* ini agar kita dapat mengelompokkan *Unit*-*Unit* yang hampir sama dalam satu area tertentu. Pembelajaran ini biasanya sangat cocok untuk klasifikasi pola. Contoh algoritma jaringan saraf tiruan yang menggunakan metode *Unsupervised* ini adalah competitive, hebbian, kohonen, *Learning Vector Quantizatio (*LVQ), dan neocognitron.



Gambar 2. Contoh *Unsupervised Learning* dalam pengenalan koin.

Salah satu contoh dari *Unsupervised Learning* adalah clustering, sistem diharapkan mampu untuk memisahkan data serupa ke dalam kelompoknya masing-masing, seperti pada Gambar 2.19, belum diketahui kelas dari masing-masing data, mesinlah yang menentukan berdasarkan kedekatannya. Beberapa contoh penerapan *Unsupervised Learning* diantaranya sistem pendeteksi intrusi (Zanero & Savaresi, 2004), menemukan komunitas pengguna di Internet sesuai dengan kriteria (Paliouras et al., 2002), dan pengembangan strategi pengendalian manufaktur (Bowden & Bullington, 1996).

### *Semi Supervised Learning*

*Semi Supervised Learning* adalah penggabungan dari *Supervised* dan *Unsupervised Learning*. Dimana hasil keluaran sistem ada yang termasuk dalam kategori yang sudah ditetapkan namun ada juga yang tidak. Beberapa contoh penerapannya yaitu kasus representasi kata (Turian, J., Ratinov, L., & Bengio, 2010), *Co-Tracking* (Tang et al., 2007), dan identifikasi peptida (Käll et al., 2007)*.*

### *Reinforcement Learning*

Pada *Reinforcement Learning* model yang dihasilkan terus berkembang seiring pemakaian oleh pengguna, dimana model terus menerus diperbaiki sesuai kondisi penerapan. Beberapa contoh penerapannya yaitu pada simulasi sepak bola dalam *RoboCup* (Stone et al., 2005) dan penerbangan helikopter terbalik secara otonom (Ng et al., 2006)*.* Selain itu, sistem D2T pernah dikembangkan menggunakan model *Reinforcement Learning* oleh Gkatzia, Hastie, Janarthanam, & Lemon, (2013), dimana sistem yang dikembangkan mampu menghasilkan teks keluaran yang adaptif berdasarkan *feedback* dari dosen atau staff kependidikan, adapun teks keluaran yang dihasilkan merupakan rangkuman dari perilaku mahasiswa Ilmu Komputer selama pembelajaran lab berlangsung.

## *Time-series* Data

Kumpulan data yang tercatat dalam periode waktu mingguan, bulanan, kuartalan, atau tahunan(Mishra dan Jain, 2014).  Ada 4 faktor yang mempengaruhi data Time Series.  Dalam data ekonomi biasanya didapatkanadanya fluktuasi atau variasi dari waktu ke waktu atau disebut dengan variasi Time Series.  Variasi ini biasanya disebabkan oleh adanya faktor *Trend* (trend factor), Fluktuasi siklis (cyclical fluktuation), Variasi musiman (seasonal variation), dan pengaruh *random* (irregular atau random influences).

***Trend*** adalah keadaan data yang menaik atau menurun dari waktu ke waktu.  Contoh yang menunjukkan trend menaik yaitu pendapatan per-kapita, jumlah penduduk.   **Variasi musiman** adalah fluktuasi yang muncul secara reguler setiap tahun yang biasanya disebabkan oleh iklim, kebiasaan (mempunyai pola tetap dari waktu ke waktu).  Contoh yang menunjukan variasi  musiman seperti penjualan pakaian akan meningkat pada saat hari raya, penjualan buku dan tas sekolah akan meningkat pada saat awal sekolah.

**Variasi siklis** muncul ketika data dipengaruhi oleh fluktuasi ekonomi jangka panjang, variasi siklis ini bisa terulang setelah jangka waktu tertentu. Variasi siklis biasanya akan kembali normal setiap 10 atau 20 tahun sekali, bisa juga tidak terulang dalam jangka waktu yang sama.  ini yang membedakan antara variasi siklis dengan musiman. Gerakan siklis tiap komoditas mempunyai jarak waktu muncul dan sebab yang berbeda-beda, yang sampai saat ini belum dapat dimengerti.  Contoh yang menunjukkan variasi siklis seperti industri konstruksi bangunan mempunyai gerakan siklis antara 15-20 tahun sedangkan industri mobil dan pakaian gerakan siklisnya lebih pendek lagi.

**Variasi *random*** adalah suatu variasi atau gerakan yang  tidak teratur (irregular).  Variasi ini pada kenyataannya sulit diprediksi. Contoh variasi ini dalam data Time Series karena adanya perang, bencana alam dan sebab-sebab unik lainnya yang sulit diduga.  Total variasi dalam data Time Series adalah merupakan hasil dari keempat faktor tersebut yang mempengaruhi secara bersama-sama.  Dalam tulisan ini hanya akan dianalisa dua variasi pertama, sedangkan dua variasi terakhir tidak dianalisa karena memang pola variasi tersebut tidak tersistem dengan baik selain membutuhkan waktu yang sangat lama untuk mendapatkan data yang panjang.

Model Time Series adalah suatu peramalan nilai-nilai masa depan yang didasarkan pada nilai-nilai masa lampau suatu variabel dan atau kesalahan masa lampau.  Model Time Series biasanya lebih sering digunakan untuk suatu peramalan/prediksi. Dalam tehnik peramal an dengan Time Series ada dua kategori utama yang perlu dilakukan pengujian, yaitu pemulusan (*smoothing*) dan dekomposisi (decomposition).  Metode pemulusan mendasarkan ramalannya dengan prinsip rata-rata dari kesalahan masa lalu (Averaging smoothing past errors) dengan menambahkan nilai ramalan sebelumnya dengan persentase kesalahan (percentage of the errors) antara nilai sebenarnya (actual value) dengan nilai ramalannya (forecasting value).  Metoda dekomposisi mendasarkan prediksinya dengan membagi data Time Series menjadi beberapa komponen dari Trend, Siklis, Musiman dan pengaruh *Random*. Kemudian mengkombinasikan prediksi dari komponen-komponen tersebut (kecuali pengaruh *random* yang sulit diprediksi).  Pendekatan lain untuk peramalan  adalah metoda causal atau yang lebih dikenal dengan sebutan regresi. Tehnik pemulusan dan regresi akan dibahas pada sesi tulisan yang lain.

## *ExponentialSmoothing*

*ExponentialSmoothing* adalah suatu prosedur yang secara terus menerus memperbaiki peramalan dengan merata-rata (menghaluskan = *smoothing*) nilai masa lalu dari suatu data runtut waktu dengan cara menurun (*exponential*). Menurut (Trihendadi, 2005) analisis *exponential smoothing*merupakan salah satu analisis deret waktu, dan merupakan metode peramalan dengan memberi nilai pembobot pada serangkaian pengamatan sebelumnya untuk memprediksi nilai masa depan. Pada penelitian ini, algoritma *exponential smoothing* digunakan untuk memperdiksi nilai dari setiap parameter dengan tipe *numerical* seperti yang sudah dilakukan pada penelitian sistem DWP (Putra et al., 2017).

### *Single Exponential Smoothing*

*Single Exponential Smoothing* atau biasa disebut sebagai *Simple Exponential Smoothing* adalah metode yang digunakan untuk peramalan jangka pendek. Model mengasumsikan bahwa data berfluktuasi di sekitar nilai mean yang tetap, tanpa trend atau pola pertumbuhan konsisten. Tidak seperti *Moving Average*, *Exponential Smoothing* memberikan penekanan yang lebih besar kepada *Time Series* saat ini melalui penggunaan sebuah konstanta *smoothing* (penghalus). Konstanta *smoothing* mungkin berkisar dari 0 ke 1. Nilai yang dekat dengan 1 memberikan penekanan terbesar pada nilai saat ini sedangkan nilai yang dekat dengan 0 memberi penekanan pada titik data sebelumnya.  Pada penelitian ini digunakan *Single Exponential Smoothing* untuk proses prediksi data seperti pada penelitian DWP (Putra et al., 2017).

Rumus untuk Simple exponential *smoothing* adalah sebagai berikut:

**Ft+1 = αAt + ( 1-α )Ft**

dimana:

Ft+1 = Peramalan untuk periode t +1.

Ft = Peramalan untuk periode t (sebelumnya).

At = Nilai aktual Time Series

α = Monstanta perataan antara 0 dan 1

Misalnya, jika kita menggunakan contoh data seperti pada tabel 2.2, dimana At merupakan kolom penjualan, dan Ft merupakan hasil prediksi, dengan α bernilai 0.2, maka perhitungan *exponential smoothing* adalah sebagai berikut.

Tabel 2. Contoh penggunaan *Exponential Smoothing*

| Minggu | Penjualan  (At) | Prediksi  (Ft) | Keterangan |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7 | 39  44  40  45  38  43  39 | 39  39  40  40  41  40.40  40.92  40.54 | F1 = A1 = 39  F2 = 0.2(39) + 0.8(39) = 39  F3 = 0.2(44) + 0.8(39) = 40  F4 = 0.2(40) + 0.8(40) = 40  F5 = 0.2(45) + 0.8(40) = 41  F6 = 0.2(38) + 0.8(41) = 40.40  F7 = 0.2(43) + 0.8(40.40) = 40.92  F8 = 0.2(39) + 0.8(40.92) = 40. 54 |

## *String Matching*

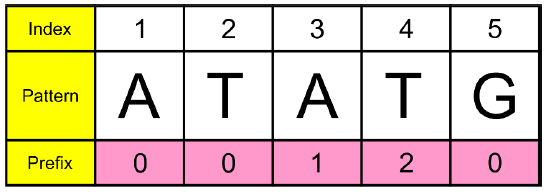
*String Matching* merupakan salah satu teknik yang digunakan dalam *information retrieval,*  dimana metode ini digunakan untuk mengambil informasi dari sesuatu yang ingin diketahui. Sebutan lain untuk teknik *string matching* diantaranya *pattern matching,* atau *pattern searching*, dimana teknik *string matching* ini sering digunakan untuk berbagai hal terutama dalam kemanan informasi, bioinformatika, deteksi plagiarism, pemrosesan teks dan pencocokan dokumen (Vijayarani & Janani, 2016).

Dalam bidang keamanan informasi seperti yang sudah dijelaskan sebelumnya, teknik *string matching* ini digunakan dalam bidang keamanan jaringan misalnya dalam perbaikan data paket HTTP secara *realtime*. Dimana kebutuhan akan algoritma *string mathcing*  yang efisien untuk mereduksi data pada protokol HTTP menjadi sangan penting (Zhang et al., 2015). Selain itu, bidang pengaplikasian *string matching* yang mulai hangat diperbincangkan akhir-akhir ini adalah *bioinformatics.* Dimana teknik ini digunakan untuk menganalisa sekuen DNA sehingga dapat memberikan informasi yang dibutuhkan secara tepat, seperti penelitian yang dilakukan oleh (Rahman, 2017), dimana penelitian tersebut menggunakan algoritma ini untuk mendeteksi *genomic repeats* pada sekuenDNA. Sehingga teknik *string matching* ini menjadi topik riset yang menarik dan penting dalam bidang ilmu komputer (Chen & Wu, 2016).

Pada penelitian ini, teknik *string matching* digunakan untuk mencari motif pada parameter *categorical*, dimana motif beberapa data terakhir akan dicocokkan dengan seluruh data pada parameter tersebut. Algoritma *string matching* yang digunakan dalam penelitian ini adalah algoritma *Knuth-Morris-Prat* (KMP) yang akan dijelaskan pada sub-bab selanjutnya.

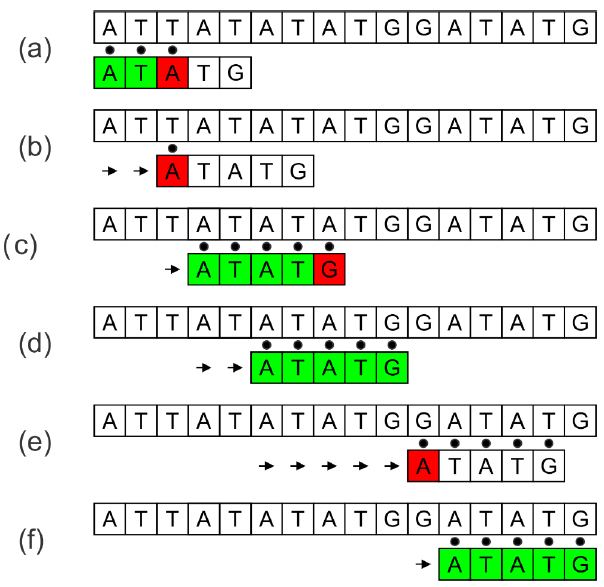
### Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt ditemukan oleh ketiga ilmuwan bernama Knuth, Morris dan Pratt untuk menemukan posisi string yang diberikan pada sebuah *text editing program.* Berbeda dengan algoritma pencarian *brute force* dimana pada algoritma KMP ini akan ditentukan informasi *prefix* dari pattern yang akan dicari sebelum dilakukan pencarian. Pada penelitian ini, digunakan algoritma string Matching yang digunakan pada penelitian (Rahman, 2017). Misalnya jika kita mempunyai *pattern ‘*ATATG’ yang akan dicari pada teks ‘ATTATATATGGATATG’, algoritma *brute force* akan mencocokan setiap *pattern* pada teks dan akan bergeser satu indeks untuk mencocokkan *pattern* tersebut, sehingga kompleksitas dari algoritma pencarian *brute force* ini adalah O(mn). Sedangkan Algoritma KMP akan menyimpan informasi pada pencocokan sebelumnya untuk menghindari pencocokan yang sia-sia. Praproses pada pattern dengan memberikan *prefix* dari tiap karakter pattern akan menjadi petunjuk untuk melakukan pengabaian pencocokkan yang dinilai tak perlu karena sudah pasti tak cocok atau sudah pasti cocok. Sehingga didapatkan *prefix* seperti pada gambar 2.20.



Gambar 2. Pemberian prefix pada pattern 'ATATG' (Rahman, 2017)

Gambar 2.21 adalah skenario *string matching* dengan algoritma KPM pada permasalahan yang sama seperti sebelumnya. Terlihat pada gambar 2.17 bagian (b) melakukan pergeseran dua indeks karena informasi yang teredia pada *prefix*, perhitungan pergeseran terebut ada sebagai berikut: Prefix(3 - 1) *+* 1 = 0 + 1 = 1, maka karakter akan langsung melakukan pengecekan dengan bergeser langsung ke indeks ke-1.



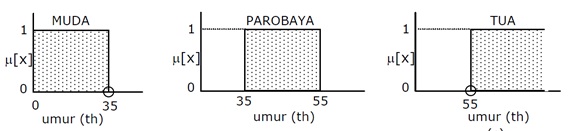
Gambar 2. Skenario *Knuth-Morris-Pratt* pada *String Matching*

## Logika *Fuzzy*

Logika *Fuzzy* adalah logika *multivalued* yang memungkinkan untuk mendefinisikan nilai menengah diantara dua logika atau evaluasi konvensional yang berbeda, seperti benar atau salah, iya atau tidak, tinggi atau rendah, panas atau dingin, dan lain-lain. Logika *Fuzzy* pertama kali diperkenalkan oleh **Prof. Lotfi A. Zadeh** pada tahun 1965. Dasar logika *Fuzzy* adalah teori himpunan *Fuzzy*. Pada teori himpunan *Fuzzy*, peranan derajat keanggotaan sebagai penentu keberadaan elemen dalam suatu himpunan sangatlah penting.

Nilai keanggotaan atau derajat keanggotaan atau *membership function* menjadi ciri utama dalam penalaran dengan logika *Fuzzy* tersebut. Logika *Fuzzy* dapat dianggap sebagai kotak hitam yang berhubungaan antara ruang *input* menuju ruang *output*. Kotak hitam tersebut berisi cara atau metode yang dapat digunakan untuk mengolah data *input* menjadi *output*dalam bentuk informasi yang baik. himpunan*Fuzzy* adalah himpunan yang menyatakan suatu obyek dapat menjadi anggota dari beberapa himpunan dengan nilai keanggotaan (µ) yang berbeda. untuk lebih jelasnya, perhatikan contoh dibawah:

Misalnya, variable umur dibagi 3 kategori, yaitu: Muda < 35 tahun, Parobaya 35 ≤ umur ≤ 55 tahun, dan Tua > 55 tahun. Secara grafis, dapat dilihat pada Gambar 2.22.

[](http://2.bp.blogspot.com/-CMU-0ZYftqA/UVeXOecedPI/AAAAAAAAAKA/hPvnfoyO5oQ/s1600/himpunan.jpg)

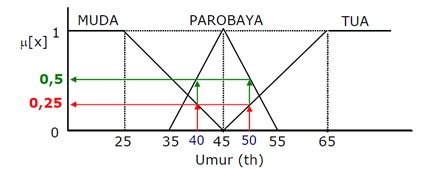
Gambar 2. Contoh himpunan *Crisp* pada kasus umur (Putra et al., 2017)

Jika menggunakan himpunan *crisp* yang dapat dilihat pada Gambar 2.10, dapat diambil kesimpulan bahwa:

* Usia 34 tahun, dikatakan Muda → Muda [34]=1
* usia 35 tahun kurang 1 hari, dikatakan Muda → Muda [35th-1hr]=1
* Usia 35 tahun, dikatakan Tidak Muda → Muda [35]=0
* Usia 55 tahun, dikatakan Parobaya → µParobaya[55]=1
* Usia 55 tahun lebih 1 hari, dikatakan Tidak Parobaya → µParobaya[55th+1hr]=0 atau
* Usia 55 tahun lebih 1 hari, dikatakan Tua → µTua[55th+1hr]=1

Dari kesimpulan diatas, himpunan crisp menyatakan umur seseorang kedalam suatu kategori secara tidak adil, karena orang yang berusia 35 tahun dikatakan parobaya, sedangkan orang yang berusia 35 tahun kurang 1 hari dikatakan tidak parobaya (karena masuk kategori muda). selisih 1 hari saja menimbulkan berbedaan kategori yang signifikan.

Himpunan *Fuzzy* digunakan untuk mengatasi hal tersebut, sehingga dengan menggunakan himpunan *Fuzzy*, seseorang dapat masuk ke dua kategori secara bersamaan, misalnya seseorang yang berusia 35 tahun kurang 1 hari dapat masuk kategori Muda dan Parobaya sekaligus, tetapi dengan nilai keanggotaan yang berbeda. Contohnya seperti pada Gambar 2.23.

[](http://4.bp.blogspot.com/-fWZLplfVti0/UVebt0KSkpI/AAAAAAAAAKI/4sqfVkm8I3Q/s1600/himp+fuzzy.jpg)

Gambar 2. Contoh himpunan *Fuzzy* pada kasus umur (Putra et al., 2017)

Sebagai contoh, seseorang yang berumur 40 tahun termasuk dalam himpunan Muda dengan µmuda[40]=0,25, namun dia juga termasuk dalam himpunan Parobaya dengan µParobaya[50]=0,5.

## Pemrograman R

Bahasa R merupakan sebuah proyek yang dirancang sebagai bahasa pemrograman yang gratis, *open source*, yang dapat digunakan sebagai pengganti dari bahasa pemrograman Splus, pada mulanya dikembangkan sebagai bahasa S di *AT&T Bell Labs*, dan sekarang dipasarkan oleh *Insightful Corporation of Seattle,* di Washington. R adalah sistem untuk komputasi statistik dan grafik. Sebagai sebuah sistem, R memiliki banyak sekali fitur. Sebagai bahasa pemrograman, R memiliki visualisasi grafik yang *high level,* antarmuka ke bahasa pemrograman lain, dan fasilitas *debugging* (Spector, 2004)*.* Logo dari bahasa pemrograman R sendiri dapat dilihat pada Gambar 2.24*.*



Gambar 2. Logo bahasa pemrograman R, sumber: https://www.r-project.org.

Berikut adalah kelebihan dari penggunaan bahasa R (Ihaka & Gentleman, 2012):

1. Serba guna (*versatile*)

R adalah bahasa pemrograman, sehingga tidak ada batasan bagi pengguna untuk memakai prosedur yang hanya terdapat pada paket-paket yang standar. Bahkan pemrograman R adalah berorientasi obyek dan memiliki banyak library yang sangat bermanfaat yang dikembangkan oleh kontributor. Pengguna bebas menambah dan mengurangi *library* tergantung kebutuhan. R juga memiliki antarmuka pemrograman C, phyton, bahkan java yang tentu saja berkat usaha serta kerja keras para kontributor aktif proyek R. Jadi selain bahasa R ini cukup pintar, penggunaannya pun bisa menjadi lebih pintar dan kreatif. Beberapa analisis yang membutuhkan fungsi lanjutan memang ada yang belum tersedia dalam R. Tidak berarti R tidak menyediakan fasilitas tersebut, namun lebih karena faktor waktu. Jadi hanya menunggu waktu saja *package* lanjutan tersebut tersedia

1. Interaktif (*interactive*)

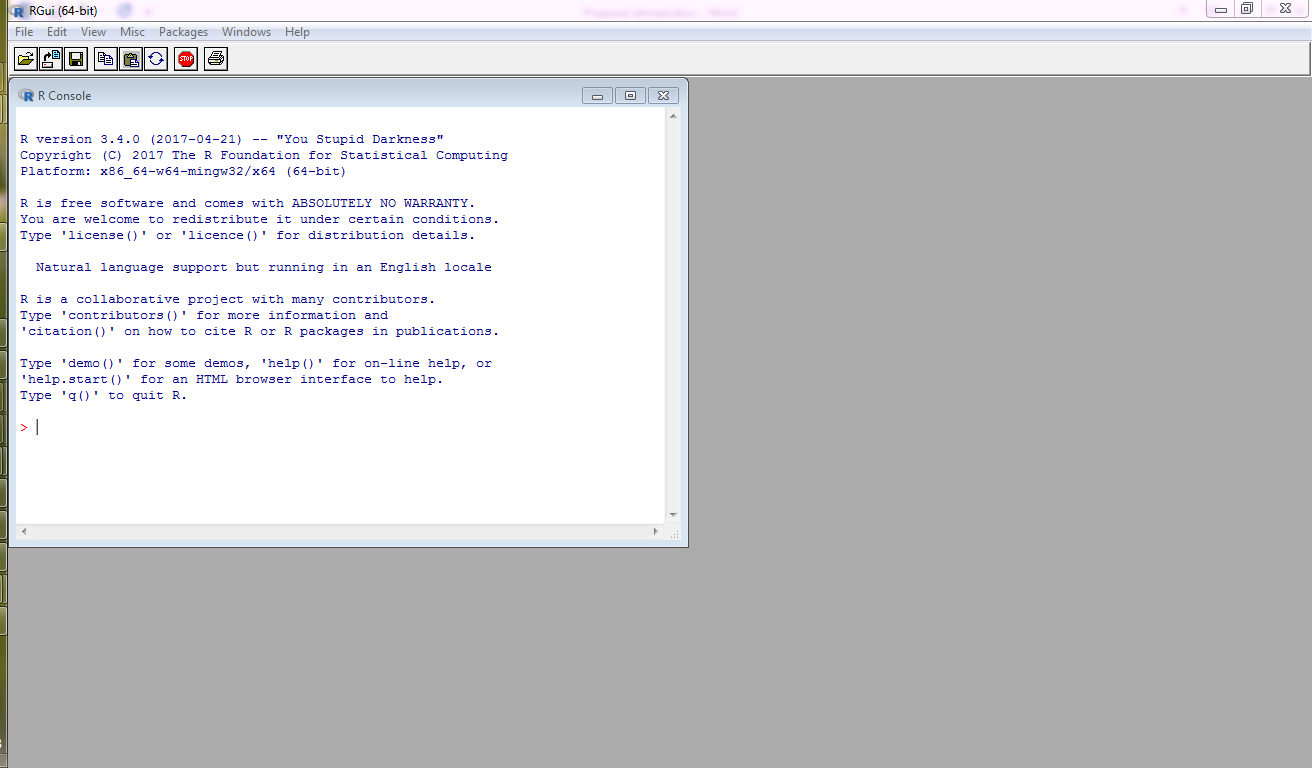
Pada saat ini analisis data membutuhkan pengoperasian yang interaktif. Apalagi jika data yang dianalisis adalah data yang bergerak. R dilengkapi dengan konektivitas ke database server, olap, maupun format data web service seperti XML, spreadsheet dan sebagainya. Sehingga apabila data set berubah hasil analisis pun dapat segera ikut berubah (*real time*).

1. Berbasis S yaitu turunan dari tool statitistik komersial S-Plus

R hampir seluruhnya kompatibel dengan S-Plus. Artinya sebagian besar kode program yang dibuat oleh S dapat dijalankan di S-plus kecuali fungsi-fungsi yang sifatnya *add-on packages* atau tambahan yang dibuat oleh kontributor proyekR.

1. Populer

Secara umum SAS adalah *software* statistika komersial yang populer, namun demikian R atau S adalah bahasa yang paling populer digunakan oleh peneliti di bidang statistika. Beberapa tulisan berupa jurnal statistika mengkonfirmasi kebenaran hal ini. R juga populer untuk aplikasi kuantitatif dibidang keuangan.

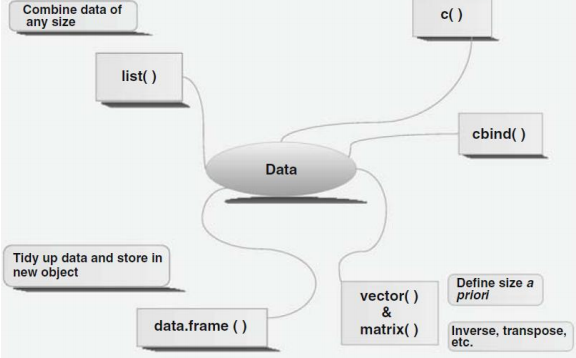


Gambar 2. Antarmuka R Graphical User Interface (RGui).

RGui merupakan tools dalam pemrograman R, antar muka tools ini dapat dilihat pada Gambar 2.25, diperlihatkan bahwa dalam antarmuka RGui terdapat layar *console* yang berfungsi untuk memasukan perintah, terdapat *menu-bar*, *tool-bar*dan lain-lain sesuai dengan fungsinya masin-masing.

### Model Data dalam R

Pada bahasa R, data dipandang sebagai suatu objek yang memiliki suatu atribut dan berbagai fungsionalitas (Budiharto, 2013). Sifat data ditentukan oleh type data dan mode data. Ada berbagai type data yang dikenal oleh R, antara lain vektor, matriks, list, data frame, *array*, *factor* dan fungsi *built in*. Berikut ini beberapa model data yang umum digunakan serta contoh penerapan fungsi *built in*. Untuk menyimpan data di R ada berbagai metode seperti menggunakan fungsi c(), list(), cbind() dan data.frame() seperti Gambar 2.26.



Gambar 2. Model data dalam pemrograman R (Budiharto, 2013).

### Contoh Kode Program Bahasa R

Berikut adalah beberapa contoh kode program yang dapat dilakukan oleh bahasa pemrograman R.

1. Menggabungkan data

Untuk menggabungkan data dalam bahasa R, dapat menggunakan fungsi *concantenate* (c). contoh dari penggunaan fungsi ini dapat dilihat pada Gambar 2.27.

|  |
| --- |
| x **<-** c**(**1,2,3,4,5**)**  y **<-** c**(**6,7,8,9**)**  z **<-** c**(**x,y**)**  z  # output  # [1] 1 2 3 4 5 6 7 8 9 |

Gambar 2. Operator *concantenate* dalam R.

Pada Gambar 2.28 berikut adalah contoh untuk menampilkan dua data pertama dalam vektor z yang telah dibuat.

|  |
| --- |
| x**[**1**:**2**]**  # output  # [1] 1 2 |

Gambar 2. Menampilkan data pertama hingga ke-dua dalam R.

Selain itu, untuk menampilkan jumlah dari seluruh elemen, dapat digunakan fungsi sum. Implementasi dari fungsi sum ini dapat dilihat pada Gambar 2.29.

|  |
| --- |
| sum**(**x**)**  # output  # [1] 15 |

Gambar 2. Penggunaan fungsi *sum* dalam R.

Contoh lainnya, untuk memasukan data string, dapat dilihat pada Gambar 2.30.

|  |
| --- |
| Z **<-** c**(**"muhammad", "ridwan", "UPI"**)**  Z  # output  # [1]"muhammad" "ridwan" "UPI" |

Gambar 2. Penggunaan fungsi *concantenate* untuk *string* dalam R

1. Membuat matriks

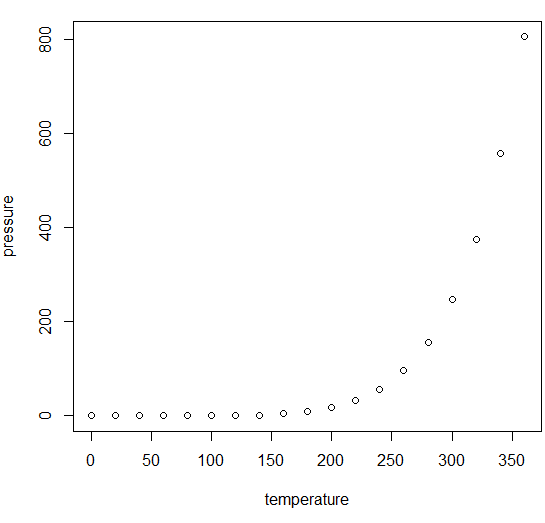
Matriks adalah data dua dimensi dimana sebagian besar fungsi‐fungsi statistik dalam R dapat dianalisis dengan menggunakan bentuk matriks. Bentuk matriks ini juga banyak digunakan pada operasi fungsi‐fungsi built‐in untuk aljabar *linear* dalam R, seperti untuk penyelesaian suatu persamaan *linear*. Argumen yang diperlukan adalah elemen‐elemen dari matriks, dan argumen optional yaitu banyaknya baris dan banyaknya kolom. Berikut contohnya ada pada Gambar 2.31.

|  |
| --- |
| M **<-** matrix**(**c**(**1**:**6**)**,nrow**=**2,ncol**=**3**)**  m  # output  # [,1] [,2] [,3]  # [1,] 1 3 5  # [2,] 2 4 6 |

Gambar 2. Pembuatan matriks dalam R*.*

1. Membuat *visualisasi* data

Salah satu keunggulan dalam bahasa pemrograman R adalah visualisasi data dapat disajikan dengan mudah. Data yang berhasil dientri atau diimport dari aplikasi lain selayaknya divisualisasikan pada grafik untuk analisa. Sebagai contoh, kita dapat menggunakan data dari R yaitu variabel pressure, dengan command “*plot(pressure)”* maka akan menghasilkan grafik seperti pada Gambar 2.32.



Gambar 2.Contoh *Visualisasi* grafis dalam R.

1. Membuat perulangan

Salah satu cara yang paling populer hampir diseluruh bahasa pemrograman dalam melakukan perulangan adalah fungsi FOR. Contoh implementasi fungsi FOR dalam bahasa R dapat dilihat pada Gambar 2.33.

|  |
| --- |
| i **<-** 1  n **<-** 10  **for(**i **in** i**:**n**){**  print**(**"Hello World!"**)**  **}**  # output  # [1] "Hello World!"  # [1] "Hello World!"  # [1] "Hello World!"  # [1] "Hello World!"  # [1] "Hello World!"  # [1] "Hello World!"  # [1] "Hello World!"  # [1] "Hello World!"  # [1] "Hello World!"  # [1] "Hello World!" |

Gambar 2.Contohperulangan dalam R*.*

1. Membuat *decision*

Membuat *decision* dalam dunia pemrograman adalah hal yang paling utama. Dalam bahasa R, membuat decision identik dengan bagaimana bahasa C melakukannya. Dapat dilihat pada Gambar 2.34.

|  |
| --- |
| i **<-** 1  n **<-** 10  **if(** n **>** i**){**  print**(**"Hello World!"**)**  **}**  # output  # [1] "Hello World!" |

Gambar 2. Contohimplementasi *decission* dalam R*.*

1. Membuat fungsi

Dalam pemrograman terstruktur, salah satu hal yang penting adalah membuat fungsi. Dalam bahasa R, contoh pembuatan fungsi dapat dilihat pada Gambar 2.35.

|  |
| --- |
| Penjumlahan **<-** **function(**a,b**){**  X **<-** a**+**b  return**(**X**)**  **}**  Penjumlahan**(**1,2**)**  # output  # [1] 3 |

Gambar 2. Contohfungsi dalam R.

## Package Dalam R

Secara konseptual, R *package* adalah kumpulan fungsi, objek data, dan dokumentasi yang secara koheren mendukun operasi analisis data. R adalah bahasa pemrograman *open-source* dan lingkungan analisis yang mengandung lebih dari 8000 *packages* untuk statistik, *bio-informatics,* visualisasi, *Machine Learning,* ekonomi, dan lain-lain (Ihaka & Gentleman, 2012). Bahkan, sampai Desember 2018 bini anyak *packages* yang terpublish dalam *cran-r project* lebih dari 12800 *packages*. Agar mudah digunakan dan untuk menjaga kualitasnya serta untuk terus mempertahankannya, kebanyakan R *package* disimpan di repositori berikut: Jaringan Arsip R Komprehensif (CRAN, http://cran.r-project.org/) dan BioconductorProject (http://www.bioconductor.org/) (Riza et al., 2016).

Dalam penelitian ini digunakan beberapa package, diantaranya:

1. *Shiny, package* ini berfungsi untuk menampilkan hasil keluaran dari konsol R kedalam bentuk web. Dimana pada penelitian ini konfigurasi *package* *shiny* terdapat dalam file *app.R*
2. *Data.Table, package* ini digunakan dalam proses pembacaan data, dimana sistem dapat melakukan proses pembacaan data dengan mode *force* atau sistem akan melakukan pembacaan dataset dan dapat membedakan mana dataset yang memiliki *header* maupun tidak.
3. *Smooth* dan *greybox, package* ini digunakan dalam proses prediksi data, dimana proses prediksi data dilakukan menggunakan algoritman *exponential smoothing.*
4. *Xts, package* ini berfungsi sebagai konverter data, dimana data masukan yang masih berbentuk tabel akan dikonversikan menjadi *time-series* sebelum nantinya dilakukan proses prediksi.
5. *Corrplot, package* ini berfungsi sebagai visualisator tabel matriks korelasi parameter.
6. *Mice, package* ini berfungsi sebagai *Missing Value Handling* dimana data akan diproses dan digunakan beberapa algoritman yang dapat memproses nilai NA dari data tersebut, seperi *random forrest*,

Proses instalasi *package* pada R ini cukup mudah, dimana pengguna cukup mengetikan perintah install.package(“nama package”) pada konsol. Pastikan koneksi internet sudah terhubung sebelum melakukan proses instalasi *package*, untuk penjelasan lebih lengkapnya dapat dilihat pada gambar 2.36.

|  |
| --- |
| **install.packages(**“Shiny”**)** |

Gambar 2. Contoh Instalasi *Package* dalam R

Setelah proses instalasi selesai, proses selanjutnya adalah meng-*load* *package* yang sudah diinstal, caranya pengguna cukup mengeksekusi perintan library(“nama package”) seperti pada gambar 2.37.

|  |
| --- |
| **library(**Shiny**)** |

Gambar 2. Cara menggunakan *package* yang sudah diinstal