BAB 2
LANDASAN TEORI

2.1. *Coreference Resolution*

Menurut Andrew Radford tahun 1988, c*oreference* merupakan hubungan antara beberapa istilah yang memiliki rujukan yang sama. *Coreference resolution* merupakan subtugas dari *Natural Language Processing* (NLP) yang bertugas untuk mencari semua ekspresi yang merujuk pada entitas yang sama. Ini merupakan langkah yang penting bagi tugas NLP yang melibatkan pemahaman pada *natural language* [11]. *Coreference resolution* pada dasarnya adalah *clustering* masalah. Kata benda di dalam dokumen dibagi menjadi beberapa *cluster*, dengan setiap cluster yang mengacu pada satu entitas [11].

Agar dapat mengilustrasikan bagaimana *coreference resolution* bekerja, mari kita lihat contoh kalimat berikut.

> ***Budi Angkasa$_1$*** *kemarin hadir di rapat umum.* ***Dia$_2$*** *bertemu dengan* ***Uswandi Pribadi$_3$*** *di Lobby Utama.* ***Uswandi$_4$*** *adalah Manager di PT. Linggar Jaya.*

Seperti contoh kalimat dia atas, ketika dua kata mengalami *coreferential*, biasanya satu kata pertama adalah sebuah kata benda yang merupakan nama seseorang dan kata lainnya adalah bentuk yang disingkat atau kata ganti. Proses penyelesaian *Coreference Resolution* di atas harus mengidentifikasi bahwa Dia$_2$ mengacu pada Budi Angkasa$_1$ dan bahwa Uswandi$_4$ mengacu pada Uswandi Pribadi$_3$. Dengan kata lain, sistem harus menghasilkan dua *cluster* yaitu {Budi Angkasa$_1$ dan Dia$_2$} dan { Uswandi Pribadi$_3$ dan Uswandi$_4$}.

Jadi, penelitian ini memanfaatkan *coreference resolution* untuk menemukan *coreference* yang ada di dalam novel

2.2. Text Mining

Text mining adalah proses ekstraksi pola (informasi dan pengetahuan yang berguna) dari sejumlah besar sumber data yang tidak terstruktur. *Text mining* merupakan hasil perkembangan dari *data mining* namun *text mining* hanya fokus kepada data yang berbentuk dokumen atau teks, dengan menggali informasi yang terdapat pada teks, di mana sumber datanya berasal dari dokumen seperti dokumen *Word*, PDF, kutipan, teks, dan sebagainya. *Text mining* ini bertujuan untuk mencari kata-kata yang mewakili isi dokumen sehingga dapat melakukan analisa keterkaitan antar dokumen [12].

Pada kasus *text mining*, dokumen atau teks yang menjadi sumber data berbentuk *unstructured* data. Pada *data mining*, di asumsikan bahwa data sudah di simpan pada format terstruktur. Oleh karena itu, proses transformasi data dari *unstructured* data menjadi lebih terstruktur, merupakan proses yang penting dan menentukan pada kasus *text mining*. Pada kasus dengan sumber *structured data*, normalisasi data merupakan proses yang menjadi fokus pada *preprocessing*. Sedangkan, pada *text mining* yang menggunakan unstructured data, *preprocessing* fokus pada mentransformasi *unstructured* data pada dokumen menjadi fitur yang lebih representatif.

Jadi, penelitian ini termasuk ke dalam *text mining* dengan menggunakan teks novel sebagai sumber data yang merupakan *unstructured* data.

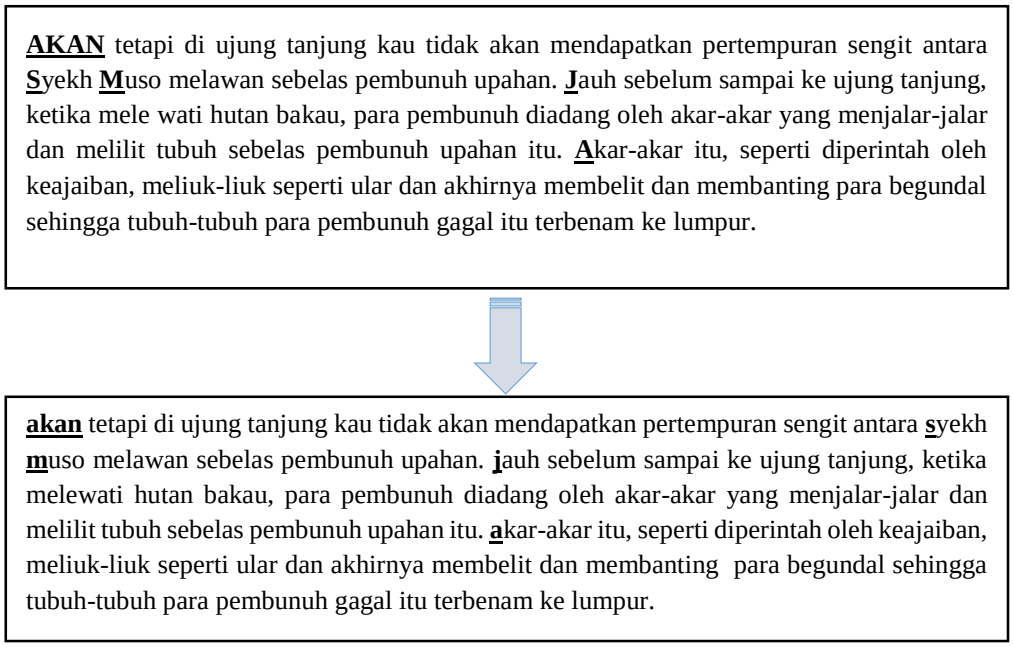
2.3. *Preprocessing*

Preprocessing merupakan proses pengolahan data asli sebelum diolah pada tahap selanjutnya. Pada *text mining* data yang di gunakan berasal dari dokumen atau teks yang tidak terstruktur. Oleh karena itu, dibutuhkan suatu proses yang dapat mengubah bentuk data yang sebelumnya tidak terstruktur menjadi data yang terstruktur. Proses ini bertujuan agar data yang akan digunakan nantinya bersih dari *noise* atau ciri-ciri yang tidak berpengaruh pada klasifikasi seperti tanda baca dan *stopword*. Tahap *preprosesing* juga bertujuan agar data yang digunakan memiliki dimensi yang lebih kecil dan lebih terstruktur, sehingga dapat diolah lebih lanjut.

Pada penelitian ini, tahap *preprocessing* terdiri dari *filtering*, *case folding*, *tokenizing*, *stemming*, dan POS *tagging*. Adapun tahapan dari *preprocessing* yang akan digunakan adalah sebagai berikut.

2.3.1. *Case Folding*

Case folding adalah proses menyeragamkan semua teks ke dalam case yang sama menjadi huruf kecil (*lowercase*) atau huruf capital (*uppercase*) [12]. Pada penelitian ini huruf dalam teks diseragamkan menjadi huruf kecil. Contoh dari tahap *case folding* dapat dilihat pada gambar 2.1 di bawah ini. Gambar tersebut menjelaskan bahwa semua huruf kapital yang ada pada teks *input* akan diubah menjadi huruf kecil.



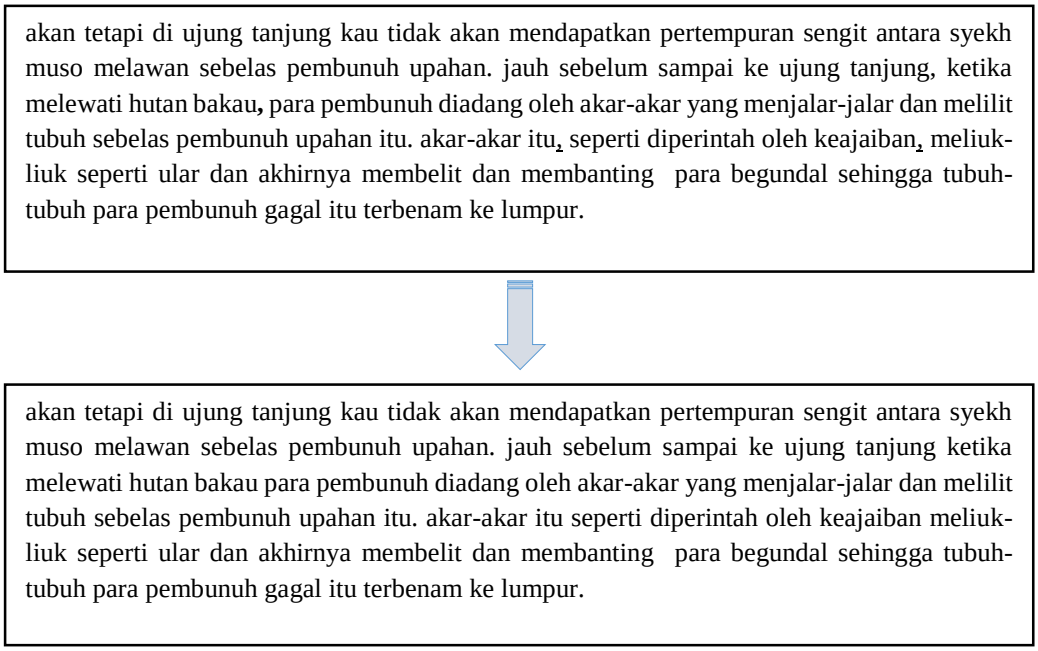
Gambar 2. 1 Contoh Tahap *Case Folding*

Jadi, penelitian ini menggunakan tahap *case folding* untuk menyeragamkan semua teks menjadi huruf kecil (*lowercase*).

2.3.2. *Filtering*

Filtering mengacu pada proses memutuskan istilah mana yang harus digunakan untuk merepresentasikan dokumen sehingga dapat digunakan untuk menggambarkan isi dokumen dan membedakan dokumen dengan dokumen lain

yang ada pada koleksi [12]. Dengan kata lain *filtering* merupakan proses dimana teks selain karakter ‘a’ sampai ‘z’, ‘-‘, ‘.’ dan spasi akan dihilangkan. Berikut adalah contoh tahap filtering



Gambar 2. 2 Contoh Tahap *Filtering*

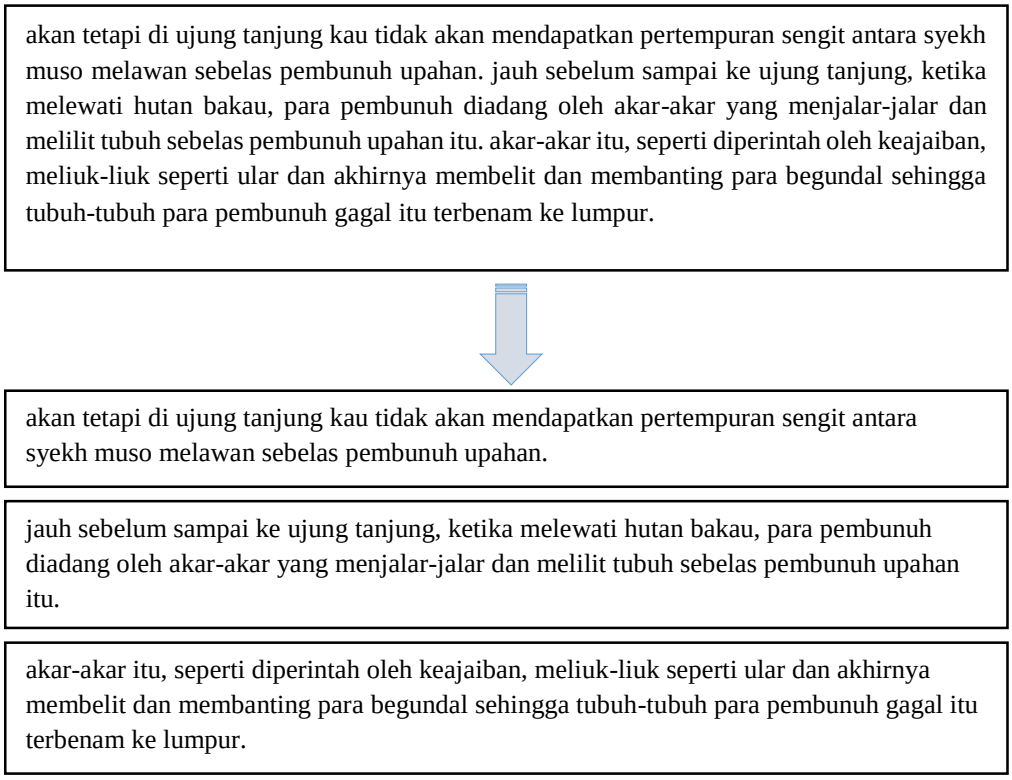
Jadi, penelitian ini menggunakan tahap filtering untuk menghilangkan karakter lain selain karakter “a” sampai “z”, “-“ dan spasi.

2.3.3. Tokenisasi

Proses tokenisasi adalah proses pemotongan *string input* dengan memecah dokumen menjadi beberapa bagian. Tokenisasi dapat dilakukan dengan melihat delimiternya seperti tipe kapitalisasi, keberadaan digit, tanda baca, karakter special dan sebagainya. Pemecahan dokumen menjadi kata-kata tunggal dilakukan dengan men-*scan* dokumen dan setiap kata terindentifikasi atau terpisahkan dengan kata yang lain oleh pemisah spasi [13].

2.3.3.1. Tokenisasi Kalimat

Pemisahan kalimat merupakan proses pemisahan teks pada dokumen menjadi kumpulan kalimat. Teknik yang digunakan dalam pemisahan kalimat adalah memisahkan kalimat dengan tanda titik (.) sebagai *delimeter*.Pada penelitian ini tokenisasi kalimat berguna untuk proses penentuan coreference. Gambar 2.3 di bawah ini adalah contoh tokenisasi kalimat.



Gambar 2. 3 Contoh Tahap Tokenisasi Kalimat

Jadi, penelitian ini menerapkan tokenisasi kalimat untuk mengetahui jarak antara kalimat satu dan kalimat lainnya.

2.3.2.2. Tokenisasi Kata

Tokenisasi kata adalah pemotongan *string input* berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan men-*scan* kalimat dan memisahkan kata demi kata dengan spasi sebagai *delimiter*. Hasil keluaran dari proses tokenizing akan digunakan sebagai masukan

dalam tahap transformasi teks. Gambar 2.4 di bawah ini merupakan contoh proses tokenisasi kata.

akan tetapi di ujung tanjung kau tidak akan mendapatkan pertempuran sengit antara syekh muso melawan sebelas pembunuh upahan



akan	tetapi	di	ujung	tanjung
kau	tidak	akan	mendapatkan	pertempuran
sengit	antara	syekh	muso	melawan
sebelas	pembunuh	upahan.		

Gambar 2. 4 Contoh Tahap Tokenisasi Kata

Jadi, penelitian ini menggunakan tahap tokenisasi dengan memotong teks keseluruhan menjadi kalimat dengan delimiter titik (.).

2.3.4. *Stemming*

Stemming merupakan suatu proses yang mentransformasi kata-kata yang berimbuhan ke kata-kata dasarnya (*root word*) [14]. Sebagai contoh, kata menyenangkan, menyenangi, disebut dapat dikatakan serupa atau satu kelompok dan dapat diwakili oleh satu kata umum senang. Proses *stemming* pada teks berbahasa Indonesia berbeda dengan stemming pada teks berbahasa Inggris. Pada teks berbahasa Inggris, proses yang diperlukan hanya proses menghilangkan sufiks. Sedangkan pada teks berbahasa Indonesia, selain sufiks, prefiks, dan konfiks juga dihilangkan. Pada umumnya kata dasar pada bahasa Indonesia terdiri dari kombinasi:

Prefiks 1 + Prefiks 2 + Kata dasar + Sufiks 3 + Sufiks 2 + Sufiks 1

Pada penelitian ini, algoritma yang digunakan dalam proses *stemming* yaitu nazief dan andriani yang telah dikembangkan oleh sastrawi. Algoritma *Stemming* Bahasa Indonesia M. Adriani dan B. Nazief ini mempunyai aturan imbuhan sendiri dengan model sebagai berikut.

[[[AW+]AW+]AW+] Kata-Dasar [[+AK][+KK][+P]]

Dimana:

AW : Awalan

AK : Akhiran

KK : Kata Ganti kepunyaan

P : Partikel

Algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahapan sebagai berikut.

1. Mencari kata yang akan distem dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah root word. Maka algoritma berhenti.
2. Inflection Suffixes (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa particles (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus Possesive Pronouns (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus Derivation Suffixes (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a.
   a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
   b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.
4. Hapus Derivation Prefix. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
   a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
   b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
5. Melakukan *Recoding*.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Tipe awalan ditentukan melalui langkah-langkah sebagai berikut.

1. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
2. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-” maka berhenti.
4. Jika tipe awalan adalah “none” maka berhenti. Jika tipe awalan adalah bukan “none” maka awalan awalan jika ditemukan.

Tabel 2. 1 Kombinasi Yang Tidak Diajukan

Awalan	Akhiran yang Tidak Diijinkan
be-	-i
di-	-an
ke-	-i, -kan
Se-	-i, -kan

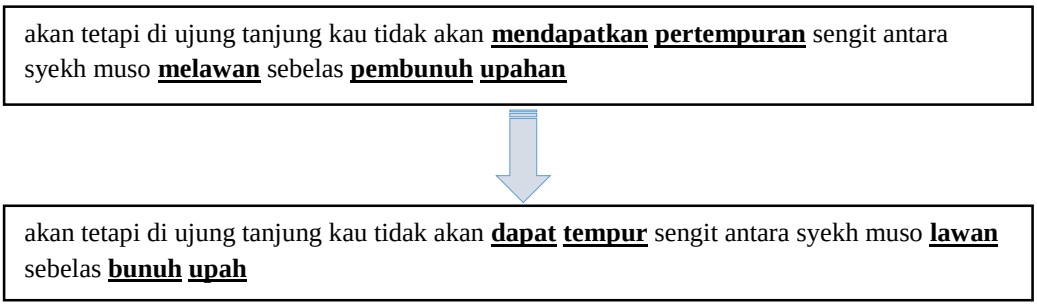
Tabel 2. 2 Cara Menentukan Awalan untuk Kata Yang Tidak Diajukan

Following Characters				Tipe Awalan
Set 1	Set 2	Set 3	Set 4	
‘r’	‘r’			none
‘-r-‘	vowel	-	-	ter-luluh
‘-r-‘	not (vowel or ‘-r-‘)	‘-er-‘	vowel	ter
‘-r-‘	not (vowel or ‘-r-‘)	‘-er-‘	not vowel	ter-
not (vowel or ‘-r-‘)	not (vowel or ‘-r-‘)	not ‘-er-‘	-	ter
not (vowel or ‘-r-‘)	‘-er-‘	vowel	-	none
not (vowel or ‘-r-‘)	‘-er-‘	not vowel	-	te

Tabel 2. 3 Jenis Awalan untuk Kata yang Diawali "te-"

Tipe Awalan	Awalan yang Harus Dihapus
di-	di-
ke-	ke-
se-	se-
te-	te-
ter-	ter-
ter-luluh	ter

Sebagai contoh kita dapat lihat pada gambar 2.5 kata "mendapatkan" akan diubah menjadi kata "dapat" karena pada kata "mendapatkan" terdapat imbuhan me- dan –kan [15].



Gambar 2. 5 Contoh Tahap *Stemming*

Pada penelitian ini digunakan library *stemming* bahasa Indonesia untuk PHP dari Sastrawi. Karena *library* ini menggunakan beberapa algoritma yaitu algoritma Nazief dan Adriani, algoritma CS *Stemmer* dan algoritma CS *Stemmer*. Dimana algoritma CS *Stemmer* dan algoritma ECS Stemm CS *Stemmer* memodifikasi beberapa aturan yang terdapat di algoritma Nazief dan Adriani.

2.3.5. POS *Tagging*

Part of Speech Tagging (POS-Tagging) adalah suatu proses yang memberikan label kelas kata secara otomatis pada suatu kata dalam kalimat[16]. Hasil dari POS ini sangat berpengaruh terhadap keluaran dari proses *Parsing* [17].

Saat ini telah banyak penelitian yang dilakukan terkait dengan POS *Tagging* untuk teks berbahasa Indonesia. Metode-metode yang digunakan dalam penelitian-penelitian tersebut pun memiliki akurasi yang cukup tinggi, diantaranya Brill *tagger* yang memiliki akurasi 98% [18] *Conditional Random Field (CRF)* dan *Maximum Entropy Method* memiliki akurasi 97,57% [19] dan *Hidden Markov Model* (HMM) yang memiliki akurasi 96,5.POS *tagger* yang dipakai pada penelitian ini adalah POS *tagger* dengan metode HMM [20]. Berikut adalah daftar *tagset* yang dapat diidentifikasi oleh POS *tagger* yang bersangkutan [20].

Tabel 2. 4 Daftar *Tag Set* POS *Tagger*

POS	KETERANGAN	CONTOH
OP	Open Paranthesis	({[
CP	Close Paranthesis	]})
GM	Slash	/
;	Semicolon	;
:	Colon	:
“	Quotation	“’
.	Sentence Terminator	.!?
,	Comma	,
-	Dash	-
…	Elipsis	…
JJ	Adjective	Kaya, manis
RB	Adverb	Nanti, sementara
NN	Common Noun	Mobil
NNP	Proper Noun	Bekasi, Indonesia
NNG	Genitive Noun	Bukunya
VBI	Intrasitive Verb	Pergi
IN	Preposition	Di, ke, dari
MD	Modal	Bisa
CC	Coo-conjunction	Dan, atau, tetapi
SC	Subor-conjunction	Jika, ketika

DT	Determinan	Para, ini, itu
UH	Interjection	Wah, aduh, oi
CDO	Ordinal numerals	Bertiga
CDC	Colective numerlas	Bertiga
CDP	Primary numerals	Satu, dua
CDI	Irregular numerals	Beberapa
PRP	Personal pronouns	Saya, kamu
WP	WH-pronouns	Apa, siapa
PRN	Number pronouns	Kedua-keduanya
PRL	Locative pronouns	Sini, situ, sana
NEG	Negation	Bukan, tidak
SYM	symbols	@#$%^&
RP	Particles	Pin, kah
FW	Foreign words	Foreign, word

Tabel 2. 5 Contoh Proses POS *Tagging*

Kalimat	**Hasil POS *Tagging***	
	Kata	**POS Tag**
Ibu pergi ke pasar.	Ibu	NN
	pergi	VB
	ke	IN
	pasar	NN

Dari contoh kalimat pada Tabel 2.1, kalimat "*ibu pergi ke pasar"* terdiri dari empat kata di mana setiap katanya memiliki POS tag yang berbeda beda, yaitu *ibu*/NN, *pergi*/VB, *ke*/IN, dan *pasar*/NN.

Pada penelitian ini digunakan library POS *tagging* bahasa Indonesia untuk PHP dari Alfan Farizki Wicaksono, karena pada POS *tag* yang dibuatnya sudah memiliki akurasi cukup tinggi. Tag set yang dimilikinya cukup lengkap serta mudah untuk menambahkan tag set baru untuk menyesuaikan kebutuhan pada penelitian *coreference.*

2.6. Ekstraksi Fitur

Ekstraksi fitur adalah sebuah proses untuk mengambil atau mengekstrak karakteristik atau informasi yang penting dari suata data untuk digunakan dalam proses klasifikasi.

2.6.1. Fitur *Coreference Resolution*

Untuk membangun *machine learning* dibutuhkan fitur untuk menentukan apakah dua ekspresi merupakan *coreference* atau tidak. Pada penelitian yang dilakukan sebelumnya yang berjudul ”Research of Noun Phrase Coreference Resolution” menerapkan beberapa fitur untuk mendeteksi *corference resolution* diantaranya adalah *Distance, String Match, Alias, I-pronoun, J-pronoun, Demonstrative Noun Phrase, Semantic Class, I-propernoun, J-propernoun, I-arg0, I-arg1, J-arg0, J-arg1* dan *Similarity.* Semua fitur tersebut akan diekstrak menjadi nilai-nilai yang akan digunakan sebagai ciri untuk tahap pengklasifikasian [1].

2.5. Metode *Support Vector Machine*

Support Vector Machine (SVM) merupakan salah satu metode *machine learning* untuk mengubah text menjadi data vector. Metode ini dikembangkan oleh Boser, Guyon, Vapnik pada tahun 1992. SVM merupakan suatu teknik untuk menemukan *hyperplane* yang bisa memisahkan dua set data dari dua kelas yang berbeda (Vapnik, 1999). *Hyperplane* adalah garis batas pemisah data antar kelas, sedangkan margin adalah jarak antara *hyperplane* dengan data terdekat pada masing-masing kelas. Adapun data terdekat dengan *hyperplane* pada masing-masing kelas inilah yang disebut support vector [21].

Dalam linear SVM pemisah merupakan fungsi *linear*. Data latih dinyatakan oleh ($x_{i,} y_i$) dan x_i = { x_1 , x_2 , …, x_{iq} } merupakan fitur set untuk data latih kelas ke-i. Untuk y_i € {-1,1} menyatakan label kelas. Pendefinisian persamaan suatu *hyperplane* pemisah yang dituliskan dengan :

$$wx_i + b = 0 \quad (2.1)$$

Data x_i yang terbagi ke dalam dua kelas, yang termasuk kelas -1 (sampel negatif) didefinisikan sebagai vektor yang memenuhi pertidaksamaan 2.2 berikut ini.

$$wx_i + b < 0 \ untuk\, y_i = -1 \quad (2.2)$$

Sedangkan yang termasuk kelas +1 (sampel positif) memenuhi pertidaksamaan 2.3 berikut.

$$wx_i + b > 0 \ untuk\, y_i = +1 \quad (2.3)$$

Dimana :

x_i = data *input*

y_i = label yang diberikan

w = nilai dari bidang normal

b = posisi bidang relative terhadap pusat koordinat

Parameter w dan b adalah parameter yang akan dicari nilainya. Bila label data y_i = -1, maka pembatas menjadi persamaan 2.4 berikut.

$$wx_i + \mathrm{b} \leq -1 \tag{2.4}$$

Bila label data yi = +1, maka pembatas menjadi persamaan 2.5 berikut.

$$wx_i + \mathrm{b} \geq +1 \tag{2.5}$$

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antar bidang pembatas kedua kelas dan titik terdekatnya, yaitu 2/|w|. Hal ini dirumuskan sebagai permasalahan *quadratic programming* (QP) *problem* yaitu mencari titik minimal persamaan 2.6 dengan memperhatikan persamaan 2.7 berikut.

$$min\,(w) = \frac{1}{2}\,||w||^2 \tag{2.6}$$

$$y_i\,(wx + b) - 1 \geq 0,\ (i = 1,..,) \tag{2.7}$$

Permasalahan ini ini dapat dipecahkan dengan berbagi teknik komputasi. Lebih mudah diselesaikan dengan mengubah persamaan 2.6 ke dalam fungsi *Lagrangian* pada persamaan 2.8, dan menyederhanakannya menjadi persamaan 2.9 berikut.

$$(w, b, a) = \frac{1}{2}\,||w||^2 \quad - \sum_{i=1}^{n} a_i\,(\,((w^T x_i + b) - 1)) \tag{2.8}$$

$$(w, b, a) = \frac{1}{2}\,||w||^2 \quad - \sum_{i=1}^{n} a_i y_i\,(w^T x_i + b) + - \sum_{i=1}^{n} a_i \tag{2.9}$$

Dimana a_i adalah *lagrange multiplier* yang bernilai nol atau positif ($a_i \geq 0$). Nilai optimal dari persamaan 2.9 dapat dihitung dengan meminimalkan L terhadap w, b dan a. Dapat dilihat pada persamaan 2.10 sampai 2.12 berikut.

$$\frac{\partial \mathrm{L}}{\partial \mathrm{w}} = w - \sum_{i=1}^{n} a_i y_i x_i = 0 \tag{2.10}$$

$$\frac{\partial \mathrm{L}}{\partial \mathrm{w}} = \sum_{i=1}^{n} a_i y_i = 0 \tag{2.11}$$

$$\frac{\partial \mathrm{L}}{\partial \mathrm{w}} = w - \sum_{i=1}^{n} a_i y_i (w^T x_i + b) + \sum_{i=1}^{n} a_i = 0 \quad (2.12)$$

Maka masalah *Lagrange* untuk klasifikasi dapat dinyatakan pada persamaan 2.13 berikut.

$$Min\ L(w, a, b) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} a_i y_i (w^T x_i + b) + \sum_{i=1}^{n} a_i = 0 \quad (2.13)$$

Dengan memperhatikan persamaan 2.14 dan 2.15 berikut.

$$w - \sum_{i=1}^{n} a_i y_i a_i = 0 \quad (2.14)$$

$$\sum_{i=1}^{n} a_i y_i = 0 \quad (2.15)$$

Model persamaan 2.13 di atas merupakan model primal *Lagrange.* Sedangkan dengan memaksimalkan L terhadap ai, persamannya menjadi persamaan 2.16 berikut.

$$Max \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1,j=1}^{n} a_i\, y_i {y_j}^T x_i x^T \quad (2.16)$$

Dengan memperhatikan persamaan 2.17 berikut.

$$\sum_{i=1}^{n} a_i y_i = 0, a_i > 0\ (i, j = 1, \dots, n) \quad (2.17)$$

Untuk mencari nilai x_i dan y_i dapat dilakukan ketika sudah didapatkan nilai tiap pasangan dari hasil ekstraksi fitur dan inisialisasi kelas . Hasil dari ekstraksi fitur diubah ke dalam bentuk format data svm (x), sedangkan data kelas menjadi label data svm (y).

Untuk mendapatkan nilaia_i, langkah pertama adalah mengubah setiap data *input* menjadi nilai vektor (*support vector*) = $\begin{pmatrix} x \\ y \end{pmatrix}$. Kemudian nilai vektor dari setiap data *input* dimasukkan ke persamaan (2.18) kernel *trick phi* (ϕ) berikut.

$$\phi \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{cases} \sqrt{{x_n}^2 + {y_n}^2} > 2, maka \begin{bmatrix} \sqrt{{x_n}^2 + {y_n}^2} - x_i + |x_i - y_i| \\ \sqrt{{x_n}^2 + {y_n}^2} - y_i + |x_i - y_i| \end{bmatrix} \\ \sqrt{{x_n}^2 + {y_n}^2} \leq 2, maka \begin{bmatrix} x_i \\ y_i \end{bmatrix} \end{cases} \quad (2.18)$$

Nilai x didapatkan dari persamaan 2.19 kernel linear untuk x berikut.

$$\sum_{i=1}^{n} {x_i}^T x_j\ \ (i, j = 1, \dots, n) \quad (2.19)$$

Nilai y didapatkan dari persamaan 2.20 kernel linear untuk y berikut.

$$\sum_{i=1}^{n} {y_i}^T y_j\ \ (i, j = 1, \dots, n) \quad (2.20)$$

Untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, maka hasil perhitungan dari substitusi nilai x dan nilai y ke persamaan (2.18) diberi nilai bias = 1 . Kemudian cari parameter ai, dengan terlebih dahulu mencari nilai fungsi setiap abstrak menggunakan persamaan (2.21), lalu mencari nilai ai pada persamaan linear menggunakan persamaan (2.22) dengan memperhatikan i,j=1,..,n berikut:

$$\sum_{i=1,j=1}^{n} a_i {S_i}^T S_j \quad (2.21)$$

$$\sum_{i=1,j=1}^{n} a_i {S_i}^T S_j = y_i \quad (2.22)$$

Setelah parameter a_i didapatkan, kemudian masukkan ke persamaan (2.23) berikut:

$$\widetilde{W} = \sum_{i=1,j=1}^{n} a_i S_i \quad (2.23)$$

Hasil yang didapatkan menggunakan persamaan (2.23), selanjutnya digunakan persamaan (2.24) untuk mendapatkan nilai w dan b:

$$y = wx + b \quad (2.24)$$

Sedemikian sehingga didapatkanlah nilai w dan nilai b atau nilai *hyperplane* untuk mengklasifikasikan kedua kelas [17].

Sebuah fungsi bisa menjadi fungsi kernel jika memenuhi Teorema Mercer, yang menyatakan bahwa matriks kernel yang dihasilkan harus bersifat semi *positive* semi *definite* [22]. berikut ini adalah beberapa fungsi kernel yang umum digunakan yaitu:

a. Kernel linier

$$K(x_i, x) = {x_i}^T x \quad (2.25)$$

b. Polynomial

$$K(x_i, x) = (y.{x_i}^T x + r)^p, y > 0 \quad (2.26)$$

c. Radial basis function

$$K(x_i, x) = exp(-y|x_i - x|^2), y > 0 \quad (2.27)$$

d. Signoid kernel

$$K(x_i, x) = \tanh(y{x_i}^T + r) \quad (2.28)$$

Pada penelitian ini, kernel yang digunakan ialah kernel linier dan proses klasifikasi Support Vector Machine menggunakan library LIBSVM PHP yang dibuat oleh Gabriel Couto.

2.6. Akurasi

Untuk menghitung akurasi dari SVM dilakukan perhitungan dengan rumus berikut [23].

$$Akurasi = \frac{Jumlah\ yang\ diklasifikasi\ dengan\ benar}{Jumlah\ sampel\ data\ uji}\ x\ 100\% \quad (2.29)$$

Penelitian ini menggunakan perhitungan akurasi untuk mengetahui tingkat akurasi dari klasifikasi *coreference resolution.*

2.7. *Black Box Testing*

Black Box adalah metode pengujian perangkat lunak yang menguji fungsionalitas aplikasi yang bertentangan dengan struktur internal atau kerja (lihat pengujian *white-box*). Pengetahuan khusus dari kode aplikasi atau struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar spesifikasi dan persyaratan, yakni, aplikasi apa yang seharusnya dilakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih input yang valid dan tidak valid dan menentukan output yang benar. Tidak ada pengetahuan tentang struktur internal benda uji itu.

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit *testing* juga.

Pengujian pada *black box* berusaha menemukan kesalahan seperti:

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

2.8. Pemodelan Sistem

Pemodelan sistem merupakan proses pengembangan model abstrak dari sistem, dengan tiap model memperlihatkan pandangan berbeda dari sistem. Adapun yang digunakan dalam pemodelan sistem pada penelitian tugas akhir ini adalah sebagai berikut.

2.8.1. *Flowchart*

Flowchart adalah penggambaran secara grafik dari langka-langkah dan urut-ututan prosedur dari suatu program. *Flowchart* menolong *analyst* dan *programmer* untuk memecahkan masalah kedalam segmen-segmen yang leboh kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut.

Dalam penulisan *flowchart* dikenal dua model, yaitu sistem *flowchart* dan program *flowchart* [24]*:*

a. Sistem *flowchart*

Sistem *flowchart* merupakan diagram alir yang menggambarkan susatu sistem peralatan komputer yang digunakan dalam proses pengolahan data serta hubungan antar peralatan tersebut.

Sistem *flowchart* ini tidak digunakan untuk menggambarkan urutan langkah untuk memecahkan masalah, tetapi hanya untuk menggambarkan prosedur dalam sistem yang dibentuk.

Dalam menggambarkan *flowchart* biasanya digunakan simbol-simbol yang standar, tetapi pemrogram juga dapat membuat simbol-simbol sendiri apabila simbol-simbol yang telah tersedia dirasa masih kurang. Dalam kasus ini pemrogram harus melengkapi gambar *flowchart* tersebut dengan kamus simbol untuk menjelaskan arti dari masing-masing simbol yang digunakan agara pemrogram lain dapat mengetahui maksud dari simbol-simbol tersebut .

Simbol bisa dilihat pada daftar simbol Tabel 2. Simbol sistem *flowchart*.

b. Program *flowchart* merupakan diagram alir yang menggambarkan urutan logika dari suatu prosedur pemecahan masalah. Untuk menggambarkan program *flowchart* telah tersedia simbol-simbol standar, namun demikian seperti halnya pada sistem *flowchart*, pemrogram dapat menambah khasanah simbol-simbol tersebut asalkan pemrogram melengkapinya dengan penggambaran program *flowchart* dengan kamus simbol.

   Simbol bisa dilihat pada daftar simbol. Simbol program *flowchart*.

   Secara umum, proses penyelesaian suatu permasalahan terdiri dari lima langkah utama, yaitu:

   1. Dimulainya suatu proses
   2. Membaca data masukkan
   3. Proses penyelesaian permasalahan
   4. Mencetak hasil pengolahan/informasi
   5. Diakhirinya semua proses pengolahan



Gambar 2. 6 Contoh *Flowchart* Sistem Akademik

2.8.2. Diagram Konteks

Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan seluruh input ke sistem atau output dari sistem. Ia akan memberi gambaran tentang keseluruan sistem. Sistem dibatasi oleh boundary (dapat digambarkan dengan garis putus). Dalam diagram konteks hanya ada satu proses serta tidak boleh ada *store* dalam diagram konteks [25].

2.8.3. *Data Flow Diagram*

Data flow diagram adalah suatu grafik yang menjelaskan sebuah sistem dengan menggunakan bentuk-bentuk dan simbol-simbol untuk menggambarkan aliran data dari proses-proses yang saling berhubungan. *Data flow diagram* ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, data flow diagram adalah alat

pembuatan model yang memberikan penekanan hanya pada fungsi sistem. Data flow diagram ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program. Beberapa komponen yang digunakan di DFD adalah sebagai berikut.

Simbol bisa dilihat pada daftar simbol Tabel 1. Simbol DFD.

a. Entitas Luar (*Boundary*)

   Entitas luar merupakan kesatuan (*entity*) di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan *input* atau menerima *output* dari sistem [25].

b. Arus Data (*Data Flow*)

   Arus data diberi simbol suatu panah. Arus data mengalir diantara proses (*process*), simpanan data (*data store*) dan kesatuan luar (*external entity*) [25].

c. Proses (*Process*)

   Suatu proses adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang masuk kek dalam proses untuk menghasilkan arus data yang akan keluar dari proses [25].

d. Penyimpanan Data (*Data Store*)

   Simpanan data merupakan simpanan dari data yang dapat berupa *file* atau *database* di sistem komputer, arsip, agenda atau buku. Simpanan data DFD dapat di simbolkan dengan sepasang garis horizontal[25].

Diagram aliran data merupakan model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil. Untuk memudahkan analisa dimulai dengan:

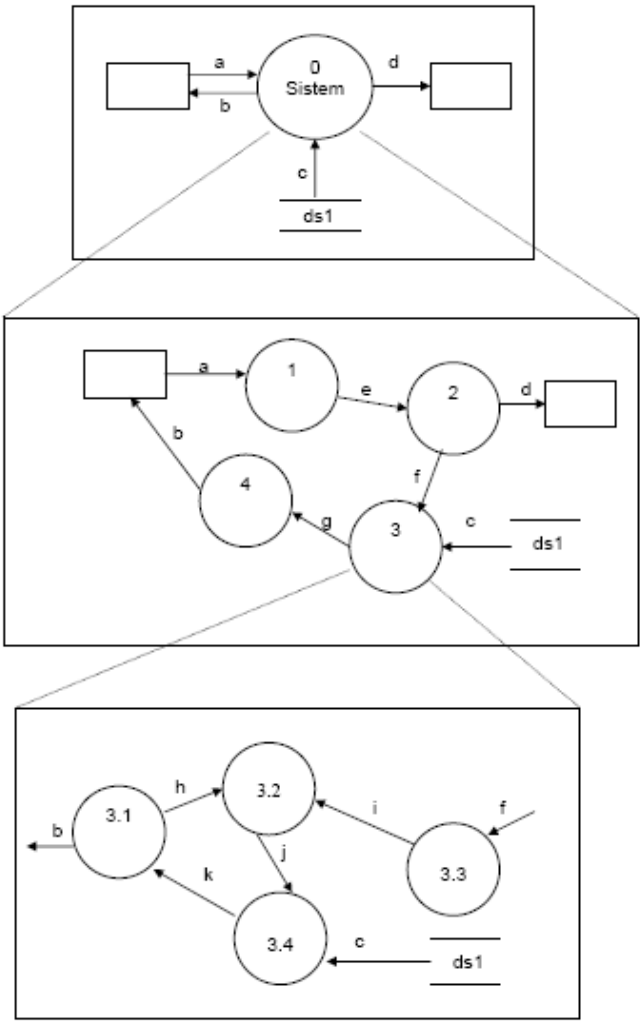
a. Diagram Konteks (*Context Diagram)*

   Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan

level tertinggi dari DFD yang menggambarkan seluruh *input* ke sistem atau *output* dari sistem.

b. Diagram Rinci (*Level Diagram*)

Diagram rinci adalah diagram yang menguraikan proses apa yang ada dalam diagram *zero* atau diagram level atasnya.



Gambar 2. 7 Levelisasi DFD [26]

2.9. Bahasa Pemrograman

Bahasa pemrograman adalah *software* bahasa komputer yang digunakan dengan cara merancang atau membuat program sesuai dengan struktur dan metode yang dimiliki oleh bahasa program itu sendiri. Komputer mengerjakan transformasi

data berdasarkan kumpulan perintah program yang telah dibuat oleh program. Kumpulan perintah ini harus dimengerti oleh komputer, berstruktur tertentu (*syntax*), dan bermakna. Bahasa pemrograman merupakan notasi untuk memberikan secara tepat program komputer [22].

Pada penelitian ini bahasa pemograman yang digunakan adalah PHP karena bahasa pemrograman PHP tidak memerlukan *Compile* dalam penggunaannya.

2.9.1. PHP (*Personal Home Page)*

PHP adalah kependekan dari PHP Hypertext Preprocessor, bahasa interpreter yang mirip dengan bahasa C dan Perl yang memiliki kesederhanaan dalam perintah, PHP dapat digunakan bersama dengan HTML sehingga memudahkan dalam pembangunan aplikasi web dengan cepat. PHP dapat digunakan untuk meng-*update* basis data dan menciptakan basis data. *Interpreter* adalah sebuah program yang digunakan untuk membaca *file* yang berisi kode program yang akan dijalankan kemuadian *interpreter* tersebut akan meminta CPU untuk menlakukan perintah yang diterimanya.

Seperti halnya program *open source* lainnya, PHP dibuat di bawah lisensi GNU, *General Public License*, yang dapat di download gratis melalui situs http://www.php.net. Awalnya, PHP diciptakan oleh Andi Gutmans untuk menghitung jumlah pengunjung yang mengakses *homepage* yang dbuatnya. PHP banyak mendukung basis data, seperti MYSQL, PostgresSQL, Interbase, ODBC, mSQL, Oracle, dan Sybase [27].

Pada penelitian ini PHP digunakan sebagai bahasa pemograman utama dalam membangun sistem *Word Sense Disambiguation* dimana kode-kode PHP hanya akan dieksekusi di server kemudian hasilnya dikirim ke browser dalam format HTML.

2.7.2. *JavaScript*

Javascript adalah bahasa skrip yang ditempelkan pada kode HTML dan diproses di sisi klien. Dengan adanya bahasa ini, kemampuan dokumen HTML menjadi semakin luas. Sebagai contoh, dengan menggunakan *JavaScript*

dimungkinkan untuk memvalidasi masukan- masukan pada formulir sebelum formulir dikirimkan ke server. *Javascript* bukanlah bahasa Java dan merupakan dua bahasa yang berbeda. *Javascript* diinterpretasikan oleh klien (kodenya bisa dilihat pada sisi klien), sedangkan kode Java dikompilasi oleh pemrogram dan hasil kompilasinyalah yang dijalankan oleh klien.

Pada penelitian ini *Javascript* diimplementasikan pada pembuatan antarmuka tambah data, apabila *user* ingin menambah data cerpen baru sehingga tampilan lebih interaktif dan dinamis

2.10. Bahasa *Markup*

Bahasa *Markup* merupakan informasi tambahan yang ditempatkan pada teks untuk menjelaskan bagaimana teks tersebut diinterpretasi [25]. Dalam penelitian ini HTML digunakan sebagai tambahan dalam pengimpelementasian program.

2.10.1. HTML

HTML adalah sebuah bahasa *markup* berupa kode-kode tag yang menginstruksikan browser untuk menghasilkan tampilan sesuai dengan yang diinginkan. HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web dan HTML kini merupakan standar *Internet* yang saat ini dikendalikan oleh *World Wide Web Consortium (W3C*) [25].

2.11. Format Pertukaran Data

Pertukaran data elektronik (bahasa inggris: EDI atau *Electronic Data Interchange*, juga *Electronic Document Interchange*) adalah proses *transfer* data yang terstruktur, dalam format standar yang disetujui dari satu sistem komputer ke sistem komputer lainnya dalam bentuk elektronik [26].

Format pertukaran data yang digunakan dalam penelitian ini adalah JSON untuk mengambil data yang dibutuhkan dari kamus kateglo yaitu data kata dan makna kata dalam Bahasa Indonesia.

2.11.1. JSON (*Java Script Object Notation*)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data [26].

JSON terbuat dari dua struktur diantaranya adalah sebagai berikut.

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel *hash* (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data *universal*. Pada dasarnya, semua bahasa pemrograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah ditukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan struktur data ini.

Pada penelitian ini JSON digunakan dalam mengambil data dari *database online* kateglo.

2.12. *Software* Pendukung

Berikut beberapa software pendukung dalam pembuatan program dalam penelitian ini.

2.12.1. XAMPP

XAMPP adalah perangkat lunak bebas (*free software*), yang mendukung untuk banyak sistem operasi, yang merupakan kompilasi dari beberapa program. Fungsi XAMPP sendiri adalah sebagai server yang berdiri sendiri (localhost), yang terdiri beberapa program antara lain : Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP sendiri merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU General Public License dan bebas, merupakan web server yang mudah untuk digunakan yang dapat menampilkan halaman web yang dinamis. Untuk mendapatkanya XAMPP anda dapat mendownload langsung dari web resminya [27].

Pada penelitian ini XAMPP berguna sebagai web server.

2.12.2. *Macromedia Dreamweaver* 8

Macromedia dreamweaver 8 merupakan HTML *editor professional* yang berfungsi mendesain, melakukan editng dan mengembangkan aneka website. Salah satu kelebihan dreamweaver 8 yaitu ruang kerja dreamweaver 8 beserta tools yang tersedia dapat digunakan dengan sangat mudah dan cepat sehingga anda bisa membangun suatu *website* dengan cepat dan tanpa harus melakukan *coding*. Selain itu, *dreamweaver* 8 juga mempunyai integrasi dengan produk *macromedia* lainnya, seperti *flash* dan *firework, flash* sudah sangat terkenal sebagai sebagai program untuk membuat animasi yang berbasis *web* dengan perkembangan kebutuhan dan teknologi, *flash* akhir-akhir ini juga digunakan untuk membuat animasi dan *video* [28]. Fungsi *coding dreamweaver* 8 tidak hanya mendukung *coding* HTML tetapi juga CSS (*Cascading Style Sheet*), *Javascript, Coldfusion*, ASP (*Active Server Sheet*), JSP (*Java Server Page*), dan *Dreamweaver* juga memungkinkan untuk membangun *website* dengan *server* berbahasa CFML (*Colfusion Markup Language*), ASP.net, dan PHP.

Pada penelitian ini *Macromedia Dreamweaver* digunakan sebagai *tools* dalam membantu membuat kode-kode program.

2.12.3 Web Browser

Web browser adalah suatu program perangkat lunak yang berfungsi untuk menerima dan menyajikan sumber informasi di Internet. *Web browser* memudahkan setiap orang untuk menjelajah dan menemukan berbagai informasi di di dunia maya. Pada awalnya *web browser* hanya dapat menampilkan teks tetapi pada perkembangan berikutnya *web browser* sudah bisa menampilkan teks, *video* dan suara .

Contoh *web browser* yang digunakan saat ini :

1. Internet Explorer
2. Mozila Firefox
3. Safari
4. Chrome
5. Opera

Pada penelitian ini web browser yang dipilih untuk diuji adalah *chrome* untuk menampilkan dan memproses program *coreference resolution.*