BAB 2
LANDASAN TEORI

2.1. Kata

Kata merupakan satuan bahasa yang dapat berdiri sendiri atau merupakan satuan bebas yang paling kecil yang dapat berdiri sendiri. Selain itu, kata merupakan unsur bahasa yang diucapkan atau dituliskan dan merupakan perwujudan kesatuan perasaan dan pikiran yang dapat digunakan dalam berbahasa [7].

2.2. Kalimat

Kalimat adalah satuan bahasa terkecil, dalam wujud lisan atau tulis, yang mengungkapkan pikiran yang utuh. Dalam wujud lisan, tuturan (atau kalimat dalam bentuk tulis) diucapkan dengan nada naik-turun, keras-lembut, disela jeda, dan diakhiri intonasi akhir. Dalam wujud tulis, kalimat diawali dengan huruf kapital dan diakhiri dengan tanda baca titik, tanda tanya, atau tanda seru, kadang kala di tengah-tenganya terdapat tanda baca lain, seperti titik dua, titik koma, dan tanda pisah. Tanda titik, tanda tanya, dan tanda seru dalam bahasa tulis sepadan dengan intonasi akhir dalam bahasa lisan, sedangkan tanda baca lain dalam bahasa tulis sepadan dengan jeda dalam bahasa lisan [8].

2.3. Berita

Berita adalah laporan tercepat mengenai fakta atau ide terbaru yang benar, menarik dan atau penting bagi sebagian besar khalayak, melalui media berkala seperti surat kabar, radio, televisi, atau media online internet [9].

Berita dapat didefinisikan sebagai “hard news” atau “soft news”; perbedaan ini kadang jelas di mata konsumen, tetapi kadang juga tidak. Hard news (berita hangat) punya arti penting bagi banyak pembaca, pendengar dan pemirsa karena biasanya berisi kejadian yang terkini yang baru saja terjadi atau akan terjadi di pemerintahan, politik, hubungan luar negeri, pendidikan, ketenagakerjaan, agama,

pengadilan, pasar fiansial, dan sebagainya. Soft news (berita ringan) baiasanya kurang penting karena isinya menghibur, walaupun kadang juga memberi informasi penting. Berita ini sering kali bukan berarti terbaru. Di dalamnya memuat berita *human intesert* atau jenis rubik *feature*. Berita jenis ini lebih menarik bagi emosi ketimbang akal pikiran [10].

2.4. Artikel

Artikel adalah tulisan lepas berisi opini seseorang yang mengupas tuntas suatu masalah tertentu yang sifatnya actual dan atau kontroversial dengan tujuan untuk memberitahu (informatif), memengaruhi dan meyakinkan (persuatif argumentatif), atau menghibur khalayak pembaca (rekreatif). Disebut lepas, karena siapa pun pembaca boleh menulis artikel dengan topik bebas sesuai dengan minat dan keahliannya masing-masing. Selain itu juga artikel yang ditulis tersebut tidak terkait dengan berita atau laporan tertentu. Ditulisnya pun boleh kapan saja, di mana saja, dan oleh siapa saja [9].

2.5. Peringkas Teks Otomatis

Peringkas teks otomatis (*automatic text summarization*) adalah pembuatan bentuk yang lebih singkat dari suatu teks dengan memanfaatkan aplikasi yang dijalankan dan dioperasikan pada komputer. Peringkas teks otomatis mampu menghasilkan ringkasan yang memiliki konten paling penting pada dokumen[13]. Tujuan dari peringkas teks yaitu mengambil intisari informasi dari dokumen sumber dan menjadikannya lebih singkat tanpa menghilangkan topik utama sehingga dapat membantu pengguna lebih cepat memahami makna yang terkandung didalamnya[14].

Secara umum terdapat dua tipe peringkasan yaitu ekstraktif dan abstraktif. Ekstraktif meringkas suatu dokumen dengan memilih sebagian dari kalimat yang ada dalam dokumen asli. Metode ini menggunakan statistik, linguistik, dan heuristik atau kombinasi dari semuanya dalam menetapkan kalimat ringkasannya. Sedangkan metode abstraktif melakukan peringkasan dengan cara menginterpretasi teks asal melaui proses transformasi suatu kalimat asli. Meskipun ringkasan yang

dihasilkan oleh manusia bersifat tidak ekstraktif, akan tetapi kebanyakan penelitian mengenai peringkasan ini adalah ekstraktif yang memberikan hasil yang lebih baik apabila dibandingkan dengan peringkasan abstraktif [15].

2.5.1. Penelitian Terdahulu

Beberapa penelitian terdahulu yang berkaitan dengan penelitian *text summarization*, seperti :

Linda Marlinda dan Harsih Rianto dari AMIK Bina Sarana Informatika tahun 2013 dalam penelitiannya yang berjudul "Pembelajaran Bahasa Indonesia Berbasis Web Menggunakan Metode Maximum Marginal Relevance", menyatakan bahwa siswa dapat lebih memahami penalaran pelajaran Bahasa Indonesia dengan mudah terutama pada soal *essay*. Dengan 5 pertanyaan *essay* diberikan kepada 10 orang responden [2].

Muchammad Mustaqhfiri, Zainal Abidin, dan Ririen Kusumawati dari Universitas Islam Negeri Maulana Malik Ibrahim Malang dalam penelitiannya yang berjudul "Peringkasan Teks Otomatis Berita Berbahasa Indonesia Menggunakan Metode *Maximum Marginal Relevance*" menyatakan bahwa metode *maxixmum marginal relevance* dapat mengurangi redundansi dengan data uji voba dari surat kabar berbahasa Indonesia *online* sebanyak 30 berita. Menghasilkan ringkasan sistem dengan rata-rata *recall* 60%, *precision* 77%, dan *f-measure* 66% dibandingkan dengan ringkasan manual [25].

Jasman Pardede dan Jordy Sinantria dari Itenas Bandung dalam penelitiannya yang berjudul "Implementasi Maximum Marginal Relevance dan Matriks Cosine Similarity pada Aplikasi Peringkas Dokumen" menyatakan bahwa aplikasi automatic summarization mampu menemukan kata yang relevansi dengan *query* yang diinginkan oleh pengguna, dengan hasil akurasi 46%. Hal tersebut disebabkan karena *query* yang tidak dibatasi, sehingga *user* bisa memasukkan apa saja pada *query*[3].

Rasim Alguliev dan Ramiz Aliguliyev dari Azerbaijan National Academy of Sciences yang berjudul "Effective Summarization Method of Text Documents"

menyatakan bahwa metode KNN bisa digunakan untuk meringkas dokumen, dengan hasil recall 63%, precision 100%, f-measure 77%[4].

2.6. Proses Peringkasan Teks Otomatis

2.6.1. *Case Folding*

Case folding adalah mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf "a" sampai dengan "z" yang diterima [16].

2.6.2. *Filtering*

Data teks dalam dokumen yang sebelumnya sudah diubah ke dalam huruf kecil semua. Selanjutnya dilakukan proses *filtering* teks. *Filtering* adalah tahapan pemrosesan teks dimana semua teks selain karakter "a" sampai "z" dan titik "." akan dihilangkan dan hanya menerima spasi [13].

2.6.3. *Tokenizing* 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* [13].

2.6.4. *Tokenizing* Kata

Tokenizing adalah pemotongan *string input* berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan men*scan* kalimat dengan pemisah (*delimeter*) spasi [16].

2.6.5. *Removal Stopword*

Stopword adalah kumpulan kata-kata yang sering muncul dalam dokumen. *Stopword* pada umumnya adalah sebuah kata penghubung yang tidak begitu penting, maka *stopword* dapat diabaikan dan tidak ikut dalam proses pengindeksan. Contoh *stopword* adalah "yang", "dan", "di", "dari", dan seterusnya [17]. Untuk mendeteksi apakah suatu kata merupakan suatu *stopword* atau bukan adalah

menggunakan kamus *stopword* yang sudah ditentukan sebelumnya. Contoh *stopword* pada bahasa Indonesia adalah di, ke, dari, pada, dan lain-lain [13].

2.6.6. *Stemming*

Stemming merupakan proses pencarian akar (*root*) kata dari tiap kata, yaitu dengan mengembalikan suatu kata berimbuhan ke bentuk dasarnya (*stem*). Untuk pemrosesan pada bahasa Indonesia, proses *stemming* dilakukan dengan menghilangkan imbuhan yang mengawali dan mengakhiri kata sehingga diperoleh bentuk dasar dari kata tersebut [13]. Tahap ini kebanyakan dipakai untuk teks berbahasa Inggris dan lebih sulit diterapkan pada teks berbahasa Indonesia. Hal ini dikarenakan bahasa Indonesia tidak memilki rumus bentuk baku yang permanen [16].

Algoritma *stemmer* yang diperkenalkan Nazief dan Adriani (1996) didefinisikan sebagai berikut:

1. Di awal proses *stemming* dan setiap langkah yang selanjutnya dilakukan, lakukan pengecekan hasil proses stemming kata yang di*input*kan pada langkah tersebut ke kamus kata dasar. Jika kata ditemukan, berarti kata tersebut sudah berbentuk kata dasar dan proses stemming dihentikan. Jika tidak ditemukan, maka langkah selanjutnya dilakukan.
2. Hilangkan *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”). 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. Hilangkan *derivation prefixes.*
    a. Langkah 4 berhenti jika:
        i. Terjadi kombinasi awalan dan akhiran yang terlarang.
        ii. Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
        iii. Tiga awalan telah dihilangkan.
    b. Identifikasikan tipe awalan dan hilangkan. Awalan terdiri dari dua tipe:
        i. Standar (“di-”, “ke-”, “se-”) yang dapat langsung dihilangkan dari kata.
        ii. Kompleks (“me-”, “be-”, “pe”, “te-”) adalah tipe-tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya. Oleh karena itu, gunakan aturan pada Tabel 2.1 untuk mendapatkan hasil pemenggalan yang tepat.
    c. Cari kata yang telah dihilangkan awalannya ini di dalam kamus kata dasar. Apabila tidak ditemukan, maka langkah 4 diulangi kembali. Apabila ditemukan, maka keseluruhan proses dihentikan.
5. Apabila setelah langkah 4 kata dasar masih belum ditemukan, maka proses *recoding* dilakukan dengan mengacu pada aturan pada Tabel 2.1. *Recoding* dilakukan dengan menambahkan karakter *recoding* di awal kata yang dipenggal. Pada Tabel 2.1, karakter *recoding* adalah huruf kecil setelah tanda hubung („-“) dan terkadang berada sebelum tanda kurung. Sebagai contoh, kata “menangkap” (aturan 15), setelah dipenggal menjadi “nangkap”. Karena tidak valid, maka recoding dilakukan dan menghasilkan kata “tangkap”.
6. Jika semua langkah gagal, maka input kata yang diuji pada algoritma ini dianggap sebagai kata dasar.

Tabel 2. 1 Aturan Pemenggalan *Stemmer* Nazief dan Andriani

Aturan	Format Kata	Pemenggalan
1	berV...	ber-V... | be-rV...
2	berCAP...	ber-CAP... dimana C!=“r“ & P!=“er“

3	berCAerV...	ber-CaerV... dimana C!="r"
4	Belajar	bel-ajar
5	beC1erC2...	be-C1erC2... dimana C1!={"r"|"l"}
6	terV...	ter-V... | te-rV...
7	terCerV...	ter-CerV... dimana C!="r"
8	terCP...	ter-CP... dimana C!="r" dan P!="er"
9	teC1erC2...	te-C1erC2... dimana C1!="r"
10	me{l|r|w|y}V...	me-{l|r|w|y}V...
11	mem{b|f|v}...	mem-{b|f|v}...
12	mempe{r|l}...	mem-pe...
13	mem{rV|V}...	me-m{rV|V}... | me-p{rV|V}...
14	men{c|d|j|z}...	men-{c|d|j|z}...
15	menV...	me-nV... | me-tV
16	meng{g|h|q}...	meng-{g|h|q}...
17	mengV...	meng-V... | meng-kV...
18	menyV...	meny-sV…
19	mempV...	mem-pV... dimana V!=„e"
20	pe{w|y}V...	pe-{w|y}V...
21	perV...	per-V... | pe-rV...
22	perCAP	per-CAP... dimana C!="r"danP!="er"
23	perCAerV...	per-CAerV... dimana C!="r"
24	pem{b|f|V}...	pem-{b|f|V}...
25	pem{rV|V}...	pe-m{rV|V}... | pe-p{rV|V}...
26	pen{c|d|j|z}...	pen-{c|d|j|z}...
27	penV...	pe-nV... | pe-tV...
28	peng{g|h|q}...	peng-{g|h|q}...
29	pengV...	peng-V... | peng-kV...
30	penyV...	peny-sV…
31	pelV...	pe-lV... kecuali "pelajar" yang menghasilkan "ajar"
32	peCerV...	per-erV... dimana C!={r|w|y|l|m|n}
33	peCP...	pe-CP... dimana C!={r|w|y|l|m|n} dan P!="er"

Keterangan simbol huruf :

C: huruf konsonan

V: huruf vokal

A: huruf vokal atau konsonan

P: partikel atau fragmen dari suatu kata, misalnya “er”

2.6.7. Metode TF-IDF (*Term Frequency – Inversed Document Frequency*)

TF-IDF (*Term Frequency – Inversed Document Frequency*) digunakan rumus untuk menghitung bobot (w) masing-masing dokumen terhadap kata kunci. TF-IDF banyak digunakan sebagai faktor bobot (w) dalam pencarian informasi dan *text mining*. Pembobotan diperoleh dari jumlah kemunculan *term* dalam sebuah dokumen *term frequency* (*tf*) dan jumlah kemunculan *term* dalam koleksi dokumen *inverse document frequency* (*idf*). Variasi dari skema pembobotan TF-IDF sering digunakan oleh mesin pencari sebagai alat eksekusi. Bobot suatu istilah semakin besar jika istilah tersebut sering muncul dalam suatu dokumen dan semakin kecil jika istilah tersebut muncul dalam banyak dokumen [19].

Dalam *tf* frekuensi *term* pilihan paling sederhana adalah dengan menggunakan frequensi baku dalam dokumen, yaitu berapa kali *term (t)* terjadi dalam dokumen *(d)* [3]. Nilai *idf* sebuah *term* (kata) dapat dihitung menggunakan dalam persamaan (2.1).

$$IDF = \log\left(\frac{N}{df}\right) \quad \ldots (2.1)$$

D adalah jumlah dokumen yang berisi *term* (*t*) dan *idf* adalah jumalh kemunculan (frekuensi) *term* terhadap D. Adapun algoritma yang digunakan untuk menghitung bobot (w) masing-masing dokumen terhadap kata kunci (*query*) menggunakan persamaan (2.2).

$$w_{d.t} = tf_{d.t} * IDF_t \quad \ldots (2.2)$$

Dimana :

D : dokumen ke-*d*

t : term ke-t dari kata kunci

tf : term frekuensi/ frekuensi kata

w : bobot dokumen ke-*d* terhadap term ke-*t*

2.6.8. *Cosine Similarity*

Cosine similarity digunakan untuk menghitung pendekatan relevansi *query* terhadap dokumen. Semakin besar nilai kesamaan vektor *query* dengan vektor dokumen maka *query* tersebut dipandang semakin relevan dengan dokumen. Cosinus dari 0^0 adalah 1, dan kurang dari 1 untuk setiap sudut lainnya. Dengan demikian dua vektor dengan orientasi yang sama memiliki kesamaan cosinus dari 1, dua vektor pada 90^0 memiliki kesamaan 0. *Cosine similarity* terutama digunakan dalam ruang positif, dimana hasilnya dibatasi (0,1). *Cosine similarity* kemudian memberi tolak ukur seberapa mirip dua dokumen [3].

Cosinus dari dua vektor dapat diturunkan dengan menggunakan *dot product* Euclidean yaitu 1.b = ||a|| ||b|| cos Θ dimana vektor atribut, A dan B kesamaan dan besarnya ditukan dengan persamaan (2.3).

$$CS(b_1, b_2) = \frac{\sum_{t=1}^{n} W_{t.b1} W_{t.b2}}{\sqrt{\sum_{t=1}^{n} W_{t.b1}^2} \sqrt{\sum_{t=1}^{n} W_{t.b2}^2}} \quad \ldots (2.3)$$

Dimana :

t : term dalam Kalimat

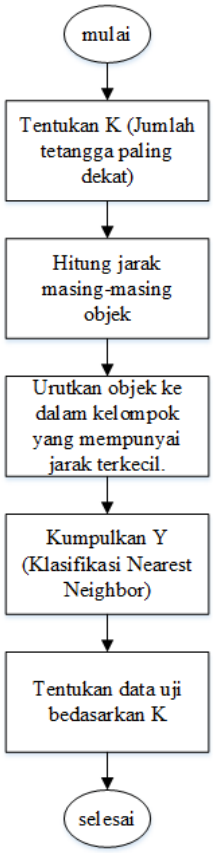
$W_{t.b1}$: bobot term t dalam blok b_1

$W_{t.b2}$: bobot term t dalam blok b_2

2.6.9. Metode KNN (K-*Nearest Neighbor*)

K-*Nearest Neighbor* (KNN) merupakan sebuah metode untuk melakukan klasifikasi terhadap obyek baru berdasarkan (K) tetangga terdekatnya. KNN termasuk algoritma *supervised learning*, dimana hasil dari *query instance* yang baru, diklasifikasikan berdasarkan mayoritas dari kategori pada KNN. Kelas yang paling banyak muncul yang akan menjadi kelas hasil klasifikasi.

Nearest Neigbor adalah pendekatan untuk mencari kasus dengan menghitung kedekatan antara kasus baru dengan kasus lama, yaitu berdasarkan pada pencocokan bobot dari sejumlah fitur yang ada [18].



Gambar 2. 1 *Flowchart K-Nearest Neighbors* [18]

2.6.10. Metode MMR (*Maximum Marginal Relevance*)

Maximum Marginal Relevance (MMR) adalah sebuah algoritma yang digunakan untuk metode ringkasan ekstrasi yang dapat meringkas satu dokumen atau lebih. Metode ini pertama kali dikemukakan oleh Carbonell dan Goldstein pada tahun 1998. Metode MMR digunakan untuk memilih kalimat dengan mempertimbangkan aspek kerelevanan kalimat dengan query [20]. Cara kerja algoritma MMR meringkas kalimat dengan menghitung kasamaan (*similarity*) antar bagian kalimat seperti yang dinyatakan pada persamaan (2.4).

$$MMR = argmax\left[\lambda * Sim_1^{(D_i,Q)} - (1-\lambda) * Sim_2^{(D_i,D)}\right] \quad \ldots (2.4)$$

Dimana :

λ : Koefisiensi nilai penekanan kalimat relevan

$Sim1$: *similarity* kalimat Si, terhadap vektor *query*

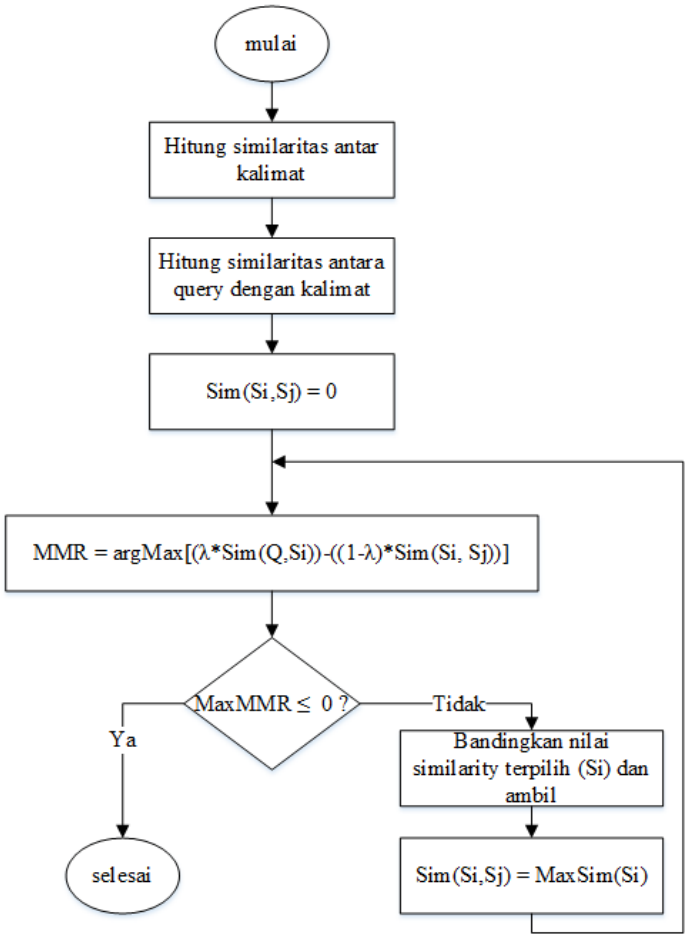
*Sim*2 : matrik *similarity* kalimat *Si*, terhadap setiap kalimat

Di : kalimat dalam dokumen

D : Kalimat yang telah diekstrak

Q : Query

Sim adalah kosinus kesamaan antara dua vektor fitur. λ adalah kofisien untuk mengatur relevansi kalimat dan mengurangi redundansi. Nilai parameter λ adalah 1 atau 0 atau diantaranya ($0<\lambda<1$). Pada saat parameter $\lambda=1$ maka nilai MMR yang diperoleh cenderung relevan terhadap dokumen asli. Ketika $\lambda=0$ maka nilai MMR yang diperoleh akan cenderung releven terhadap kalimat yang diekstrak sebelumnya yang akan dibandingkan. Oleh sebab itu, sebuah kombinasi linear dari kedua kriteria dioptimalkan ketika nilai λ terdapat pada interval $0<\lambda<1$. Untuk peringkasan dengan dokumen yang kecil, seperti artikel berita akan menghasilkan hasil ringkasan yang baik, jika nilai parameter $\lambda=0,7$ atau $\lambda=0,8$ [21].



Gambar 2. 2 Proses *Maximum Marginal Relevance* (MMR)

2.7. Pengujian Akurasi

Pengujian yang dilakukan dalam penelitian ini menggunakan pengujian akurasi. Standar pengukuran yang biasa digunakan dalam penelitian *text summarization* yaitu *recall*, *precision*, dan *f-measure* [25].

Recall adalah tingkat keberhasilan ringkasan. Perhitungan *recall* dapat dilihat pada persamaan sebagai berikut:

$$recall = \frac{tp}{tp + fn} \quad \text{... (2.5)}$$

dimana

recall : tingkat keberhasilan

tp : jumlah kalimat yang berhasil di ekstrak sistem sesuai dengan kalimat yang diekstrak manusia

fn : jumlah kalimat yang diekstrak manusia tetapi tidak terdapat dalam kalimat yang diekstrak sistem

Precision adalah tingkat ketepatan hasil ringkasan. Perhitungan *precision* dapat dilihat pada persamaan berikut:

$$precision = \frac{tp}{tp + fp} \quad \text{... (2.6)}$$

dimana,

precision : tingkat ketepatan

tp : jumlah kalimat yang berhasil di ekstrak sistem sesuai dengan kalimat yang diekstrak manusia

fp : jumlah kalimat yang diekstrak sistem tetapi tidak terdapat dalam kalimat yang diekstrak manusia

F-measure adalah gabungan antara *recall* dan *precision*. Perhitungan *f-measure* dapat dilihat pada persamaan berikut:

$$F - meansure = \frac{2\ x\ recall\ x\ precision}{recall + \ precision} \quad \ldots (2.7)$$

2.8. Bahasa Pemrograman

2.8.1. PHP

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 [11].

2.9. *Software* Pendukung

2.9.1. XAMPP

XAMPP adalah perangkat lunak (free software) bebas, 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.

2.9.2. MySQL dan PhpMyAdmin

MySQL termasuk dalam kategori *database management system*, yaitu suatu database yang terstruktur dalam pengolahan dan penampilan datanya. MySQL merupakan *database* yang bersifat *client server*, di mana data diletakkan di *server* yang bisa diakses melalui komputer *client.* Pengaksesan dapat dilakukan apabila komputer telah terhubung dengan *server*. Berbeda dengan *database* desktop, di mana segala pemrosesan data harus dilakukan pada komputer yang bersangkutan.

Bila diinstal pada sistem operasi Microsoft Windows, maka MySQL berlisensi *shareware*, tetapi tidak mempuyai *expired date* (batas waktu). Sedangkan bila diinstal pada system operasi lainnya, maka berlisensi *free* sesuai dengan *General Public Licence* (GPL).

MySQL merupakan *database* yang dikembangkan dari bahasa SQL (*Structured Query Language*). SQL merupakan bahasa terstruktur yang digunakan untuk interaksi antara script program dengan *databse server* dalam hal pengolahan data. Dengan SQL, maka dapat dibuat tabel yang akan diisidata, memanipulasi data seperti menambah, menghapus dan meng-update data, serta membuat suatu perhitungan berdasarkan data yang ditemukan. [12]

Setiap RDBMS (*Relation database management System*) seperti Orcale, SQL Server, MySQL dan lain-lain, pasti memiliki tool yang dapat digunakan untuk mempermudah pengoperasian database. Oracle memiliki TOAD. SQL Server memiliki *Enterprise Manager dan SQL Query Analyzer.* Sedangkan MySQL memiliki tool atau aplikasi yang disebut PhpMyAdmin.

PhpMyAdmin merupakan aplikasi berbasiskan web yang dikembangakan menggunakan bahasa PHP. Melalui PhpMyAdmin, user dapat melakukan perintah *query* tanpa harus mengetikkan seperti pada MS DOS. Perintah tersebut misalnya administrasi *user* dsan *privileges, export* dan *import* darabase, manajemen database, manajemen table dan struktur table, dan sebagainya. PhpMyAdmin sangat *user friendly*, sehingga mudah untuk digunakan walaupun pengguna baru (*newbie*). [12]

2.10. Model Perangkat Lunak

2.10.1. DFD (*Data Flow Diagram*)

Diagram aliran data merupakan model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil [22]. Diagram aliran data dipakai untuk menunjukkan bagaimana data mengalir melalui serangkaian langkah pemrosesan. Data ditransformasikan pada setiap langkah sebelum ke tahap berikutnya. Langkah-langkah pemrosesan atau transformasi ini merupakan fungsi program ketika diagram aliran data digunakan untuk mendokumentasikan desain perangkat lunak. Namun demikian, pada model analisis, pemrosesan dapat dilakukan oleh orang atau komputer [5].

Salah satu keuntungan menggunakan diagram aliran data adalah memudahkan pemakai atau *user* yang kurang menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan [22].

Beberapa komponen yang digunakan di DFD adalah:
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 [23].

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*) [23].

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 [23].

d. Penyimpanan Data (*Data Store*)

   Simpanan data merupakan simpanan dari data yang dapat berupa *file* atau *database* di sistem komputer, arsip atau catatan manual, agenda atau buku.

Simpanan data DFD dapat di simbolkan dengan sepasang garis horizontal [23].

Diagram aliran data merupakan model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil. Untuk memudahkan analisa dimulai dengan [22]:

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 [22].

b. Diagram Rinci (*Level Diagram*)

   Diagram rinci adalah diagram yang menguraikan proses apa yang ada dalam diagram *zero* atau diagram level atasnya [22].

2.10.2. Flowchart

Flowchart dapat diartikan sebagai suatu alat atau sarana yang menunjukkan langkah-langkah yang harus dilaksanakan dalam penyelesaian suatu permasalahan untuk komputasi dengan cara mengekspresikannya ke dalam serangkaian simbol-simbol grafis khusus [20].

Tujuan utama dari penggunaan *flowchart* adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai, rapi, dan jelas dengan menggunakan simbol-simbol yang standar. Tahap penyelesaian masalah yang disajikan harus jelas, sederhana, efektif, dan tepat. Dalam penulisan *flowchart* dikenal dua model, yaitu sistem *flowchart* dan program *flowchart* [1].

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 [1].

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

b. Program *flowchart*

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 [1].

Simbol bisa dilihat pada daftar simbol Tabel 3. Simbol program *flowchart*.

Manfaat menggunakan *flowchart* dalam mengembangkan prosedur pemecahan masalah komputasi.

1. Akan terbiasa berpikir secara sistematis dan terstruktur dalam setiap kesempatan
2. Akan lebih mudah mengecek dan menemukan bagian-bagian prosedur yang tidak valid dan bertele-tele.
3. Prosedur yang dikembangakan akan lebih mudah dipahami oleh orang lain, sehingga tidak menimbulkan kesalahan interpretasi apabila mau menerapkan prosedur yang dikembangkan [20].

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 [20]

2.10.3. ERD

ERD (*Entity Relationship Diagram*) adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak [22]. Komponen-komponen yang digunakan dalam diagram ini adalah:
Simbol bisa dilihat pada daftar simbol Tabel 4. Simbol ERD.

a. Entitas

   Digambarkan dengan sebuah bentuk persegi panjang. *Entity* adalah suatu apa saja yang ada di dalam sistem, nyata maupun abstrak dimana data tersimpan atau dimana terdapat data. Entitas diberi nama dengan kata benda dan dapat dikelompokkan dalam empat jenis nama, yaitu orang, benda, lokasi, kejadian (terdapat unsur waktu didalamnya).

b. Relasi atau Hubungan

   Digambarkan dengan bentuk kotak berbentuk *diamond* dengan garis yang menghubungkan ke *entity* yang terkait. Hubungan atau relasi menunjukkan abstraksi dari sekumpulan hubungan yang mengakibatkan antara *entity* yang berbeda.

c. Atribut

   Atribut menunjukkan karakteristik dari entitas atau sesuatu yang menjelaskan entitas atau hubungan. Dari setiap atribut-atribut entitas terdapat satu atribut yang akan dijadikan sebagai kunci (*key*).

d. *Cardinality*

Kardinalitas menunjukkan jumlah maksimum tupel yang dapat berelasi dengan entitas pada entitas yang lain [22]. Terdapat tiga macam kardinalitas relasi, yaitu:

a) *One to One*

Tingkat hubungan satu ke satu, dinyatakan dengan satu kejadian pada entitas pertama, hanya mempunyai satu hubungan dengan satu kejadian pada entitas yang kedua dan sebaliknya. Yang berarti setiap tupel pada entitas A berhubungan paling banyak satu tupel pada entitas B, dan begitu juga sebaliknya setiap tupel pada entitas B berhubungan dengan paling banyak satu tupel pada entitas A.

b) *One to Many* atau *Many to One*

Tingkat hungan satu ke banyak adalah sama dengan banyak ke satu. Tergantung dari arah mana hubungan tersebut dilihat. Untuk satu kejadian pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua. Sebaliknya satu kejadian pada entitas yang kedua hanya dapat mempunyai satu hubungan dengan satu kejadian pada entitas yang pertama.

c) *Many to Many*

Tingkat hubungan kebanyalan terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada entitas lainnya. Baik dilihat dari segi entitas pertama maupun dilihat dari sisis yang kedua.