BAB 2
LANDASAN TEORI

Landasan teori merupakan definisi, konsep yang telah disusun secara sistematis dan dasar yang kuat dalam sebuah penelitian. Landasan teori yang digunakan dalam penelitian implementasi logika *fuzzy* dan *lexical chains* untuk *automatic text summarization* ini meliputi *text mining*, *automatic text summarization*, *extraction features*, *lexical chains*, metode *fuzzy*, ringkasan teks, *object oriented programming* (OOP), *unified modelling language* (UML), pengujian akurasi, dan perngujian beta.

2.1 *Automatic Text Summarization*

Automatic summarization adalah proses meringkas dokumen teks menggunakan antuan program komputer dengan tetap mempertahankan inti dokumen asli [5]. Pada umumnya, *automatic summarization* memiliki tiga teknik peringkasan, yaitu *extraction-based summarization*, *abstraction-based summarization*, dan *maximum entropy-based summarization*. Teknik yang sering digunakan dalam kasus meringkas teks ini yaitu *extraction-based summarization*. Teknik ini memiliki fungsi untuk memberikan hasil ringkasan yang lebih pendek dari dokumen aslinya. Teknik selanjutnya adalah teknik abstraksi yang merupakan cara mengambil intisari dari teks sumber kemudian membuat ringkasan dengan menciptakan kalimat-kalimat baru yang merepresentasikan intisari teks sumber dalam bentuk berbeda. Teknik ini masih jarang digunakan dalam penelitian ATS.

2.2 *Text Mining*

Text mining adalah proses mencari informasi dengan menggunakan *tools* analisis berupa kategorisasi. Proses *text mining* dapat menjadi solusi dari permasalah pemrosesan data berupa teks yang tidak terstruktur dalam jumlah yang banyak. *Text mining* sering digunakan dalam kasus *information retrieval*. Tujuan adanya *text mining* yaitu untuk mendapatkan informasi yang bermanfaat dari sekumpulan data, salah

satunya adalah dokumen teks. Tahapan-tahapan yang dilakukan pada proses *text mining* adalah sebagai berikut: [6]

2.2.1 Tahap Segmentasi Kalimat

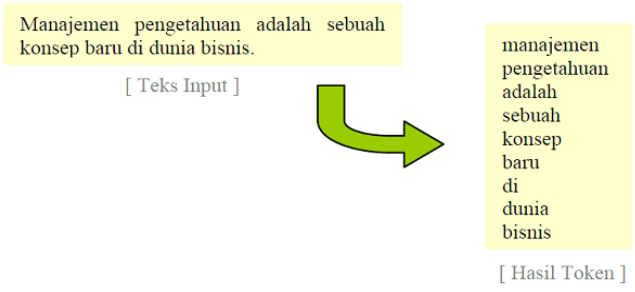
Tahap segmentasi kalimat yaitu proses memisahkan string teks dokumen menjadi kumpulan kalimat- kalimat [6]. Proses pemisahan dokumen menjadi kalimat-kalimat ini menggunakan fungsi split(), dengan tanda titik "." sebagai delimiter untuk memotong string dokumen. Contoh segmentasi kalimat dapat dilihat pada Tabel 2.1.

Tabel 2.1 Contoh Segmentasi Kalimat

Paragraf	Hasil Segmentasi Kalimat
Manajemen pengetahuan adalah sebuah konsep baru di dunia bisnis. Manajemen pengetahuan merupakan suatu cara untuk merencanakan, mengumpulkan dan mengorganisir, memimpin dan mengendalikan sumber daya untuk suatu tujuan.	- Manajemen pengetahuan adalah sebuah konsep baru di dunia bisnis. - Manajemen pengetahuan merupakan suatu cara untuk merencanakan, mengumpulkan dan mengorganisir, memimpin dan mengendalikan sumber daya untuk suatu tujuan.

2.2.2 Tahap Tokenization

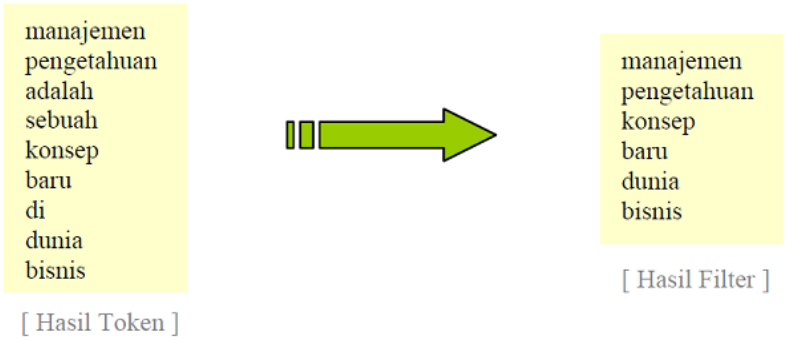
Tahap *tokenization* merupakan tahap pemotongan *string input* menjadi kumpulan kata yang menyusunnya [6]. Contoh *tokenization* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Contoh *Tokenization*

2.2.3 Tahap *Removing Stop Words*

Tahap *removing stop words* merupakan tahap penghilangan kata-kata yang tidak terlalu penting [6]. Setiap kata hasil dari *tokenization* dilakukan pengecekan dengan *stop list* yang berisi kata-kata yang tidak terlalu penting. Jika ditemukan kata pada hasil *tokenization* sama dengan kata yang ada di dalam *stop list*, maka kata tersebut akan hilang. Contoh *removing stop words* dapat dilihat pada Gambar 2.1.



Gambar 2.2 Contoh *Removing Stop Words*

2.2.4 Tahap *Stemming*

Tahap *stemming* merupakan tahap mengubah kata dari hasil *removing stop words* menjadi kumpulan kata dasar [6]. Tahap ini tidak jauh berbeda dengan hasil *stop words* karena jika hasil kata dasar sudah ada dan tidak ada perubahan maka hasilnya akan sama.

2.3 *Extraction Features*

Extraction feature merupakan proses pengolahan kalimat menggunakan beberapa fitur sehingga menghasilkan nilai-nilai yang nantinya akan digunakan untuk meringkas dokumen. *Extraction feature* yang digunakan meliputi panjang kalimat, bobot kalimat, posisi kalimat, dan kemiripan antar kalimat [7].

a. Panjang Kalimat

Panjang kalimat merupakan proses untuk mengetahui rasio dalam jumlah kata yang terjadi dalam kalimat atas jumlah kata yang terjadi dalam kalimat dokumen terpanjang. Rumus panjang kalimat adalah sebagai berikut:

$$\text{S_F1}(S) = \frac{\text{Jumlah Kata Yang Terjadi Dalam Kalimat}}{\text{Kalimat Terpanjang}} \qquad (2.1)$$

b. Bobot Kalimat

Bobot kata merupakan proses menghitung frekuensi kejadian di dalam dokumen yang telah sering digunakan untuk menghitung kata yang terdapat di dalam kalimat. Bobot yang dilakukan adalah dengan menggunakan perhitungan *ts-isf* (*term frequency*, *inverse sentence frequency*). Rumus bobot kata adalah sebagai berikut:

$$S_F2(S) = \ w_i = tf_i \times isf_i = \ tf_i \times \left(log\frac{N}{n_i}\right) \qquad (2.2)$$

dimana, tf_i = frekuensi banyak kata di dalam dokumen

N = jumlah kalimat

n_i = jumlah kalimat yang mengandung kata

c. Posisi Kalimat

Posisi kalimat merupakan posisi kalimat di dalam sebuah paragraf dengan mengusulkan kalimat pertama merupakan posisi dengan peringkat tertinggi. Rumus posisi kalimat adalah sebagai berikut:

$$\text{S_F3(S)} = \frac{\text{Kalimat}}{\text{Paragraf}} \qquad (2.3)$$

d. Kemiripan Antar Kalimat

Kemiripan antar kalimat merupakan kata yang muncul di dalam sebuah kalimat sama dengan kata yang muncul di dalam kalimat lain. Rumus kemiripan antar kalimat adalah sebagai berikut:

$$S_F4(S) = \frac{\sum Sim(S_i,S_j)}{Max(\sum Sim(S_i,S_j))} \quad (2.4)$$

dimana, $Sim(S_i, S_j)$ = kemiripan antara kalimat ke-i dan kalimat ke-j

$\sum Sim(S_i, S_j)$ = jumlah kemiripan antara kalimat ke-i dan kalimat ke-j

Dengan S_i,S_j diperoleh dari rumus jarak antar kalimat yaitu:

$$S_i,S_j = \frac{\sum_{t=1}^{n} W_{it} \times W_{ij}}{\sqrt{\sum_{t=1}^{n} {W_i}^2} \times \sqrt{\sum_{t=1}^{n} {W_j}^2}} \quad (2.5)$$

dimana, $\sum_{t=1}^{n} W_{it} \times W_{ij}$ = jumlah perhitungan scalar dari pembobotan kalimat

$\sqrt{\sum_{t=1}^{n} {W_i}^2} \times \sqrt{\sum_{t=1}^{n} {W_j}^2}$ = jumlah perhitungan vector dari pembobotan kalimat

2.4 *Lexical Chains*

Lexical chains adalah sebuah teknik penting dalam pengolahan bahasa alami. *Lexical chains* berkaitan dengan kalimat yang berelasi dengan kata benda, kata kerja, kata sifat [8]. Jika kalimat tidak berkaitan satu sama lain di dalam teks, maka relasi antar leksikal berelasi satu sama lain. Hal ini merupakan konsep yang penting. Skor dari sebuah rantai leksikal secara langsung berkaitan dengan panjang rantai yang hubungannya dengan kata benda. Peringkas menggunakan *lexical chains* untuk memilih kalimat yang paling representatif.

Langkah-langkah untuk membangun *lexical chains* adalah sebagai berikut:

1. Memilih serangkaian calon kata-kata.
2. Untuk setiap calon kata, menemukan sebuah *chain* yang bergantung pada keterkaitan antar anggota dari chain.
3. Jika ditemukan, sisipkan kata di dalam *chain* dan perbaharui sesuai hal tersebut.

Perhitungan *lexical chains* diperoleh dari rumus sebagai berikut: [9]

$$C_{Strength} = C_{length} \times \left\{ \sum \frac{\sum((Tf - Isf_{wi} \times S_{wi})) + (Tf - Isf_{wj} \times S_{wj})}{distance\ between\ wi\ and\ wj} \right\} + TotalNumberOfPairs \quad (2.6)$$

2.5 Logika *Fuzzy*

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. Logika *fuzzy* dapat dianggap sebagai kotak hitam yang menghubungkan antara ruang *input* menuju ruang *output* [10].

Menurut Cox, beberapa alasan menggunakan logika *fuzzy* sebagai berikut:

1. Konsep logika *fuzzy* mudah dimengerti, karena logika *fuzzy* menggunakan dasar teori himpunan, maka konsep matematis yang mendasari penalaran *fuzzy* tersebut cukup mudah untuk dimengerti.
2. Logika *fuzzy* sangat fleksibel, artinya mampu beradaptasi dengan perubahan-perubahan dan ketidakpastian yang menyertai permasalahan.
3. Logika *fuzzy* memiliki toleransi terhadap data yang tidak tepat, artinya jika diberikan sekelompok data yang cukup homogeny, dan kemudian ada beberapa data yang "eksklusif, maka logika *fuzzy* memiliki kemampuan untuk menangani data eksklusif tersebut.
4. Logika *fuzzy* mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks.
5. Logika *fuzzy* dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.
6. Logika *fuzzy* dapat bekerja sama dengan teknik-teknik kendali secara konvensional.
7. Logika *fuzzy* didasarkan pada bahasa alami, maksudnya menggunakan bahasa sehari-hari sehingga mudah dimengerti.

2.5.1 Himpunan *Fuzzy*

Ada beberapa hal yang perlu diketahui dalam sistem *fuzzy*, diantaranya: [10]

a. Variabel *Fuzzy*

Variabel *Fuzzy* adalah variabel yang hendak dibahas dalam suatu sistem variabel *fuzzy*.

Contoh : umur,, temperatur, permintaan, dsb.

b. Himpunan *fuzzy*

Himpunan Variabel *fuzzy* adalah suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel *fuzzy*. Himpunan *fuzzy* memiliki dua atribut, yaitu:

1. Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami.
   Contoh : MUDA, PAROBAYA, TUA.
2. Numeris, yaitu suatu nilai yang menunjukkan ukuran dari suatu variabel.
   Contoh : 40, 25, 50, dsb.

c. Semesta Pembicaraan

Semesta pembicaraan adalah himpunan bilangan *real* yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif atau negatif.

d. Domain

Domain adalah himpunan bilangan *real* yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif atau negatif.

2.5.2 Fungsi Keanggotaan

Logika *fuzzy* memiliki fungsi keanggotaan. Fungsi keanggotaan adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam nilai keanggotaannya (derajat keanggotaan) yang memiliki interval antara 0 sampai 1 [10]. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi.

Beberapa fungsi yang dapat digunakan adalah pemetaan *input* ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Ada 2 keadaan himpunan *fuzzy* yang linear. Pertama, representasi linear naik merupakan kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi. Berikut ini merupakan fungsi keanggotaannya:

$$\mu(x) = \begin{cases} 0; & x \leq a \\ \frac{(x-a)}{(b-a)}; & a \leq x \leq b \\ 1; & x \geq b \end{cases} \qquad (2.7)$$

Kedua, representasi linear turun merupakan kebalikan yang pertama. Garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah. Berikut ini merupakan fungsi keanggotaannya:

$$\mu(x) = \begin{cases} \frac{(b-x)}{(b-a)}; & a \leq x \leq b \\ 0; & x \geq b \end{cases} \qquad (2.8)$$

2.5.3 Fungsi Implikasi

Metode penalaran secara monoton digunakan sebagai dasar untuk teknik implikasi *fuzzy* [10]. Tiap-tiap aturan (preposisi) pada basis pengetahuan *fuzzy* akan berhubungan dengan suatu relasi *fuzzy*. Bentuk umum dari aturan yang digunakan dalam fungsi implikasi adalah:

IF x is A THEN y is B

dengan x dan y adalah skalar, dan A dan B adalah himpunan *fuzzy*. Preposisi yang mengikuti IF disebut sebagai anteseden, sedangkan preposisi yang mengikuti THEN disebut sebagai konsekuen. Secara umum, ada 2 fungsi implikasi yang dapat digunakan, yaitu :

a. Min (*Minimum*)

   Fungsi ini akan memotong *output* himpunan *fuzzy*.

b. Dot (*Product*)

   Fungsi ini akan menskala *output* himpunan *fuzzy*.

2.6 Ringkasan Teks

Ringkasan berasal dari kata *precis* yang berarti memotong atau memangkas. Ringkasan merupakan suatu cara untuk menyajikan tulisan yang panjang menjadi

tulisan yang singkat. Keindahan gaya bahasa, ilustrasi, serta penjelasan-penjelasan yang rinci dihilangkan sehingga jadilah sari tulisan menjadi hiasan. Peringkas langsung menyusun ringkasan bacaan dalam rangkaian kalimat, alinea. Bacaan yang diringkas dapat berupa buku, bab di dalam buku/artikel, atau skripsi. Secara umum, cara yang digunakan untuk membuat ringkasan yang baik, yaitu: [11]

a. Peringkas membaca naskah asli seluruhnya beberapa kali untuk mengetahui kesan umum, maksud, serta sudut pandang penulis.
b. Peringkas mencatat gagasan utama atau gagasan yang penting.

Peringkas menyusun kembali suatu bacaan secara singkat (ringkasan) berdasarkan gagasan utama.

2.6.1 Kohesi

Sejumlah kalimat dapat menjadi sebuah wacana yang utuh apabila mempunyai hubungan makna di antara kalimat-kalimat tersebut [11]. Kohesi atau pertalian bentuk adalah kepaduan sbentuk unsur-unsur internal di dalam tulisan. Unsur-unsur tersebut saling bergantung baik secara gramatikal maupun leksikal.

A. Kohesi Gramatikal

Kohesi gramatikal terdiri atas:

a. Referensi (Pengacuan)

   Referensi adalah pengacuan di dalam teks (pengacuan ke kiri) dan di luar teks (pengacuan ke kanan).

b. Subtitusi (Penggantian)

   Substitusi adalah sesuatu yang digunakan untuk mrnggantikan pengulangan.

c. Elipsis (Pelesapan)

   Elipsisi adalah unsur yang tidak hadir di dalam kalimat.

d. Konjungsi

   Konjungsi adalah kata yang menghubungkan antarunsur di dalam kalimat atau antara kalimat satu dan kalimat lainnya.

B. Kohesi Leksikal

Kohesi leksikal merupakan kepaduan maka dilihat juga dari segi kosa kata. Kohesi leksikan terdisi atas:

a. Repetisi

   Repetisi adalah pengulangan kata yang sama.

b. Sinonim

   Sinonim adalah persamaan makna kata.

c. Antonim

   Antonim adalah perlawanan makna kata.

d. Hiponim

   Hiponim adalah pencakupan makna.

e. Meronim

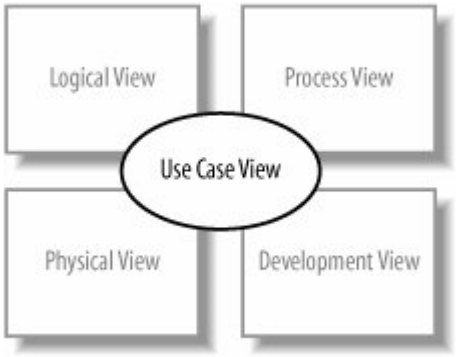
   Hubungan bagian keseluruhan.

2.7 *Object Oriented Analysis And Design* (OOAD)

Berikut akan dijelaskan empat macam diagram yang paling sering digunakan dalam pembangunan aplikasi berorientasi objek, yaitu *use case* diagram, *sequence* diagram, *collaboration* diagram, dan *class* diagram [4].

1. *Use Case Diagram*

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *use case* diagram terdiri atas diagram untuk *use case* dan *actor*. *Actor* mempresentasikan orang yang akan mengoperasikan atau orang yang berinterkasi dengan sistem aplikasi. *use case* mempresentasikan operasi-operasi yang dilakukan oleh aktor. *use case* digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *use case*. *Use case* mewakili proses, logika, fisik, dan pembangunan sistem seperti yang terlihat pada Gambar 2.3.



Gambar 2.3 *Use Case View*

2. *Sequence Diagram*

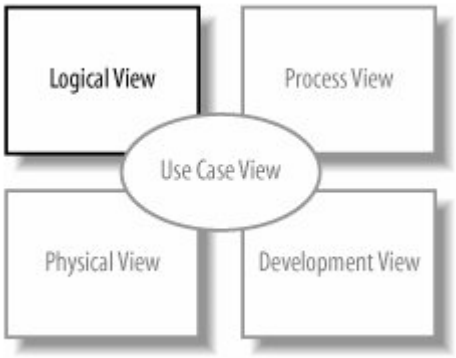
Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

3. *Collaboration Diagram*

Collaboration diagram dipakai untuk memodelkan interaksi antar objek di dalam sistem. Berbeda dengan *sequence* diagram yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan. *Collaboration* diagram lebih fokus pada pemahaman atas keseluruhan operasi yang dilakukan oleh objek.

4. *Class* Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. *Class diagram* sering dikenal dengan *logical view* seperti yang terlihat pada Gambar 2.4.



Gambar 2.4 *Logical View*

2.8 *Unified Modeling Language* (UML)

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalh dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Desain setiap sistem berskala cukup besar. Dasar pembuatan adalah objek yang merupakan kombinasi antara struktur data dan perilaku dalam suatu entitas. Pengertian berorientasi objek berarti cara mengorganisasikan perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur dan perilakunya. Salah satu yang termasuk berorientasi objek adalah *Unified Modelling Language* (UML) adalah standar untuk pembangunnan sistem dan *software*. Berikut ini adalah bagian dari UML yang digunakan: [12]

a. *Use Case Diagram*

Use case diagram menggambarkan sejumlah eksternal aktor dan hubungannya ke use case yang diberikan oleh sistem.

b. *Activity Diagram*

Activity diagram menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainya seperti *use case*.

c. *Class Diagram*

Class diagram menggambarkan struktur dan deskripsi class, package (paket) dan objek beserta hubungan satu sama lain seperti containment (penahanan), pewarisan, asosiasi dan lain-lain.

d. *Sequence Diagram*

Sequence diagram menggambarkan kolaborasi dinamis antara sejumlah objek dan untuk menunjukan rangkaian pesan yang dikirim antara objek juga interaksi. Suatu proses yang terjadi pada titik tertentu dalam eksekusi sistem.

2.9 Pengujian Akurasi

Pengujian yang dilakukan dalam penelitian ini menggunakan pengujian akurasi. Pengujian dilakukan berdasarkan kalimat hasil ringkasan manual yang bersumber dari responden dan hasil ringkasan sistem ATS. Standar pengukuran yang biasa digunakan dalam penelitian *automatic text summarization* yaitu *recall*, *precision*, dan *f-measure* [13]. *Recall* adalah tingkat keberhasilan ringkasan. Perhitungan *recall* dapat dilihat pada persamaan sebagai berikut:

$$recall = \frac{\#kalimat\ ringkasan\ sistem \cap ringkasan\ manual}{kalimat\ ringkasan\ manual} \qquad (2.9)$$

dimana, *recall* = tingkat keberhasilan

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

$$Precision = \frac{\#kalimat\ ringkasan\ sistem \cap ringkasan\ manual}{kalimat\ ringkasan\ sistem} \qquad (2.10)$$

dimana, *precision* = tingkat ketepatan

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

$$F - measure = \frac{2 \times recall \times precision}{recall + precision} \qquad (2.11)$$