BAB II
LANDASAN TEORI

II.1 *Text Summarizaton*

Text Summarization atau peringkasan teks adalah sebuah proses untuk menghasilkan ringkasan (*summary*) dari suatu artikel tetapi tetap memiliki gambaran yang akurat dari isi suatu artikel. Tujuannya adalah mengambil sumber informasi atau garis besar informasi dengan mengutip sebagian besar isi yang penting dam menampilkan artikel dalam bentuk yang ringkas.

Text summarization atau peringkasan teks berdasarkan hasil ringkasan dapat dibagi menjadi dua, yaitu ringkasan ekstraksi dan abstraksi. Ringkasan ekstraksi adalah ringkasan yang dihasilkan berisi informasi yang mengutip langsung dari teks asli yang tidak mengalami perubahan susunan kata dari kalimat yang terekstrak. Sedangkan ringkasan abstraksi adalah ringkasan yang dihasilkan dibentuk dalam susunan kata yang berbeda dari inputan. Metode *lexical chain* adalah salah satu metode untuk menghasilkan ringkasan ekstraktif yang menghasilkan poin-poin penting atau informasi utama dari inputan [6].

II.2 Metode *Lexical chain*

Metode *lexical chain* adalah suatu metode yang membentuk rantai leksikal dari kata-kata yang saling berelasi dalam teks. Hubungan leksikal dapat dibentuk pada keseluruhan kata yang saling berkaitan. Setiap kata yang dihasilkan dari *noun filtering* akan dianggap sebagai kandidat kata yang kemudian akan dicari relasinya dengan membentuk suatu *synset* yang berdasarkan kepada kamus yang telah tersedia. Berdasarkan *synset* dari setiap kandidat kata yang terbentuk, kemudian melakukan pencocokan dari setiap kandidat kata dan *synset* yang dimiliki.

Hubungan semantik antara dua kata dapat dibentuk jika terdapat kecocokan atau kesamaan antara *synset* yang direpresntasikan oleh kata tersebut. Setiap rantai leksikal yang dibentuk akan memiliki ukuran panjang rantai tertentu. Untuk mengidentifikasi nilai rantai pada setiap artikel dilakukan perhitungan berdasarkan [2] menggunakan rumus (2.1) sebagai berikut:

*Score(chain) = length * homogenityindex*

Homogenityindex = 1 – (*distinctmember*/*length*)

Keterangan:

a. *Length*: merupakan jumlah total dari seluruh kemunculan anggota penyusun rantai dalam teks.
b. *Distinctmember*: banyaknya anggota yang berbeda yang menyusun rantai.
c. *Homogenityindex*: nilai kemiripan dari anggota-anggota penyusun rantai.

Perhitungan *Homogenityindex* digunakan untuk menunjukan tingkat kemiripan anggota-anggota yang membentuk rantai. *Distinctmember* dibagi dengan *length* menyatakan tingkat perbedaan anggota penyusun rantai.

II.3 *Word sense disambiguation*

Word sense disambiguation adalah suatu proses mengidentifikasi makna kata yang digunakan dalam kalimat tertentu ketika kata memiliki sejumlah makna yang berbeda [7].

Dalam tata bahasa komputasi, arti *word sense disambiguation* adalah pemecahan masalah pengolahan bahasa alami, yang terdiri dari proses identifikasi kata yang digunakan dalam sebuah kalimat ketika kata tersebut memiliki makna lebih dari satu. Sebuah proses disambiguasi sangat memerlukan kamus untuk menentukan makna dari kata yang memiliki ambiguitas. *Word sense disambiguation* memiliki dua varian yaitu: semua kata dan sampel leksikal. Yang dimaksud semua kata adalah program harus melakukan disambiguasi terhadap semua kata, sedangkan sampel leksikal melakukan disambiguasi terhadap kata-kata yang sebelumnya dipilih.

II.4 Metode *Lexical chain* dengan *Word sense disambiguation*

Metode *lexical chain* dengan *Word sense disambiguation* merupakan metode pengembangan dari metode *original lexical chain*. Dalam metode ini setiap kandidat kata yang digunakan untuk membangun rantai leksikal akan dilakukan pengecekan ambiguitas makna terlebih dahulu. Pengecekan ini dilakukan untuk

memperoleh keakuratan dalam menentukan hubungan leksikal yang sesuai dengan konteks kalimat [8].

Perbedaan mendasar dari metode *lexical chain* dengan *word sense disambiguation* dibandingkan dengan *original lexical chain* yaitu jika metode *lexical chain* dengan *word sense disambiguation* dilakukan pengecekan ambiguitas sebelum menentukan rantai leksikal sedangan metode *original lexical chain* tidak dilakukan pengecekan ambiguitas. Kemudian jika metode *lexical chain* dengan *word sense disambiguation* hanya menggunakan *synset* yang sesuai dengan konteks kalimat untuk membentuk rantai leksikal, berbeda dengan metode original *lexical chain* yang menggunakan semua *synset* dari makna yang dimiliki oleh kata yang membentuk rantai leksikal.

Selain perbedaan yang dijelaskan tersebut, semua proses dalam metode ini sama dengan proses pada metode original *lexical chain*.

II.5 *Noun filtering*

Noun filtering merupakan tahapan untuk melakukan pemilihan kata benda (*noun*) dari kumpulan token hasil *preprocessing* yang kemudian dianggap sebagai masukan dalam membentuk rantai leksikal. *Noun filtering* dapat meningkatkan performansi dari peringkasan teks karena komponen subjek suatu kalimat biasanya bertipe kata benda, dan predikatnya menggunakan kata kerja sehingga kata benda dianggap *head* dari kalimat [2]. Selain itu kata benda lebih banyak memiliki relasi semantik dibanding dengan kata kerja.

II.6 Algoritmas *Lesk*

Algoritma *lesk* adalah algoritma yang digunakan untuk menghilangkan ambiguitas makna kata. Algoritma *lesk* merupakan salah satu algoritma untuk menyelesaikan masalah *word sense disambiguation* dengan berbasis kamus. Algoritma ini bekerja dengan membandingkan definisi dari kata yang berambigu dengan definisi. Definisi dari kata tetangganya berdasarkan definisi kamus [4]. Berikut ini merupakan pseude code algoritma lesk:

for every word w[i] in the phrase

```
        let BEST_SCORE = 0
        let BEST_SENSE = null
        for every sense sense[j] of w[i]
                let SCORE = 0
        for every other word w[k] in the phrase, k != i
                for every sense sense[l] of w[k]
                SCORE = SCORE + number of words that occur in the
                gloss
        of
        both sense[j] and sense[l]
                end for
        end for
        if SCORE > BEST_SCORE
                BEST_SCORE = SCORE
               BEST_SENSE = w[i]
        end if
        end for
        if BEST_SCORE > 0
                output BEST_SENSE
        else
        output "Could not disambiguate w[i]"
        end if
end for
```

II.7 *Lexical chain*

Lexical chain adalah rantai yang menggambarkan hubungan semantik dari suatu kata dengan kata yang lain dengan penghubung berupa *synset (glossary)* yang dibentuknya. Suatu *synset (glossary)* yang terbentuk dari suatu kata berasal dari suatu kamus berupa *wordnet*.

Pembentukan rantai leksikal didapat dari kandidat kata yang telah terpilih dari proses *noun filtering serta* telah didisambiguasi pada proses word sense disambuguation [2].

II.8 *Sentence extractor*

Dalam tahap *sentence extractor* memiliki tahap *sentence scoring* dan *sentence extraction*. Dimana *sentence scoring* memiliki tujuan untuk melakukan pembobotan pada setiap kalimat yang menyusun artikel berdasarkan metode yang digunakan. Pembobotan kalimat yang digunakan pada metode *lexical chain* yaitu

dilihat dari nilai rantai-rantai leksikal yang telah dibentuk sebelumnya. berdasarkan [9] Pembobotan kalimat ditentukan dengan rumus (2.2) sebagai berikut:

$$score(sen_j) = \sum_{i=1}^{m} \left(\frac{score(ChainMember_i, Sen_j)}{S_i} \right)$$

Dimana $score(sen_j)$ adalah nilai dari kalimat j, *score(ChainMember$_i$,Sen$_j$)* adalah nilai dari *ChainMember$_i$* yang anggotanya muncul pada kalimat *Sen$_j$*, dan m adalah jumlah maksimal rantai yang terbentuk pada artikel dengan S_i merupakan jumlah kalimat yang mengandung satu atau semua anggota dari rantai leksikal *ChainMember$_i$* diseluruh teks.

Setiap kalimat akan dihitung bobotnya, kemudian dilakukan *rerangking* secara *descanding*, yang kemudian akan dilakukan proses ekstraksi kalimat pada proses *sentence extraction*. Dimana tujuan dari *sentence extraction* adalah untuk menemukan kalimat yang akan di ekstrak sebagai penyusun hasil ringkasan. Kalimat tersebut masing-masing akan diekstraksi dari artikel tekstual yang asli dan disusun kembali sebagai satu kesatuan ringkasan. Kalimat yang akan diektraksi dipilih berdasarkan nilai bobot masing-masing kalimat kemudian diurutkan berdasarkan urutan kalimat [9].

II.9 *Preprocessing*

Preprocessing adalah tahap dimana sebelum masuk ke proses utama dalam sistem. *Preprocessing* dilakukan terlebih dahulu agar menghasilkan informasi yang siap digunakan pada proses selanjutnya. Beberapa proses yang terjadi saat melakukan *preprocessing* yaitu:

a. *Cleaning*

Tahap pembersihan karakter-karakter yang tidak diperlukan dalam proses selanjutnya.

b. *Break into token*

Tahap pemotongan teks input berdasarkan tiap kata yang menyusunnya.

c. *Case folding*

Pada tahap *Case folding* ini merupakan tahap dimana semua huruf akan diubah menjadi lowercase atau huruf kecil

d. *Stoplist*

Token-token yang telah dihasilkan saat proses *Break into token* kemudian diproses kembali dengan melakukan pembuangan kata yang termasuk *stoplist* yaitu kata-kata yang dianggap tidak memiliki bobot dalam dokumen. Artinya, kata-kata ini tidak memiliki makna penting dalam dokumen seperti "adalah", "yang", "dari", dan sebagainya. Hasil dari proses pembuangan *stoplist* inilah yang akan digunakan untuk proses selanjutnya yaitu proses *stemming*. Hasil pembuangan *stoplist* disebut dengan *non-stopword tokens*.

e. *Stemming*

Non-stopword tokens dari proses *stoplist* akan difilter kembali melalui proses *stemming* yaitu mengubah kata-kata berimbuhan menjadi kata dasar pembentuknya. Misalnya, kata "dicari" diubah menjadi kata "cari", kata ”kesakitan” diubah menjadi kata ”sakit”.

II.10 UML

UML (*Unified* Modeling *Language*) merupakan salah satu *tools* untuk memodelkan analisis dan desain berorientasi objek. UML adalah sebuah bahasa pemodelan standar untuk perangkat lunak dan pengembangan sistem. Berikut adalah beberapa diagram pada UML untuk desain suatu sistem [10]:

1. *Use Case Diagram*

Use case Diagram dapat digunakan untuk menggambarkan fungsionalitas dari sebuah sistem. Sebuah *Use case diagram* dapat merepresentasikan sebuah interaksi antara aktor dengan sistem.

2. *Activity Diagram*

   Activity diagram menggambarkan berbagai alur aktivitas suatu sistem, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana akhir dari alur suatu aktivitas. Diagram aktivitas dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang melakukan aktivitas tertentu

3. *Sequence diagram*

   Sequence diagram menggambarkan interaksi pada sistem dan digambarkan terhadap waktu. Oleh karena itu *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan keluaran tertentu.

4. *Class Diagram*

   Class Diagram adalah sebuah diagram yang menggambarkan objek, atribut, dan method pada suatu sistem. Diagram kelas juga menjelaskan hubungan antar objek secara keseluruhan di dalam sebuah sistem yang sedang dibuat dan bagaimana caranya agar mereka saling berkolaborasi untuk mencapai sebuah tujuan.

II.11 Pengujian

Pengujian adalah suatu proses mengevaluasi suatu sistem atau komponen dengan maksud untuk menemukan apakah itu memenuhi persyaratan yang ditentukan atau tidak. Dengan kata lain pengujian ialah mengeksekusi sistem untuk mengidentifikasi kesenjangan, atau kesalahan. Dalam penelitian ini dilakukan dua pengujian yaitu pengujian black box dan pengujian akurasi.

1. Pengujian Black box

Pengujian yang mengabaikan mekanisme internal sistem atau komponen dan fokus semata-mata pada output yang dihasilkan yang merespon input yang dipilih dan kondisi eksekusi. Pengujian yang dilakukan untuk mengevaluasi pemenuhan sistem atau komponen dengan kebutuhan fungsional tertentu [11].
Tujuan *Black Box* diantaranya adalah menemukan fungsi yang tidak benar atau hilang, Kesalahan *interface, Error* pada struktur data atau akses database external, *Error* pada kinerja, *Error* pada saat inisialisasi dan terminasi, Kesensitifan sistem terhadap nilai input tertentu, Batasan dari suatu data.

2. Pengujian Akurasi

Pengujian Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya. Pengujian akurasi ini dilakukan untuk mengetahui akurasi dalam mengidentifikasi makna kata yang berambigu. Tingkat akurasi diperoleh dengan perhitungan sesuai dengan persamaan(2.3) berikut [12] :

$$Akurasi\ Keberhasilan = \frac{\sum Pengujian\ Data\ Benar}{\sum Banyak\ Data\ Uji} \times 100\%$$

Sedangkan untuk akurasi peringkasan teks terdapat banyak metode untuk pengujian akurasi hasil peringksan. Untuk pengujian peringksan teks dapat dikategorikan menjadi dua, yaitu metode intrinsik dan metode ekstrinsik. Dimana metode instrinsik adalah metode yang menguji kualitas ringkasan yang dihasilkan, sedangkan metode ekstrinsik adalah metode untuk menguji performansi hasil ringkasan dalam memenuhi task tertentu. Metode pengujian yang kebanyakan digunakan dalam peringkasan teks adalah metode intrinsik. Pendekatan dalam pengujian intrinsik yang biasanya dilakukan adalah membandingkan hasil peringkasan sistem dengan ringkasan yang diperoleh dari ringkasan-ringkasan yang dibuat oleh manusia [13].

Dalam tugas akhir ini pengujian akan dilakukan dengan metode *Precision recall*. Pengujian dengan metode ini diharapkan bisa memberkan hasil untuk dasar pengambilan kesimpulan terhadap hasil ringkasan sistem. *Precision* menunjukan perbandingan jumlah data relevan yang terambil dengan seluruh jumlah data terambil. Sedangkan *recall* adalah perbandingan data yang relevan terambil dengan jumlah data seluruh dalam dokumen. Nilai *Precision* dan *recall* yang tinggi menunjukan tingkat keakuratan. *Precision* dan *recall* diperoleh dari persamaan (2.4) sebagai berikut [14] [15]:

$$Precision = \frac{A \cap B}{B}$$

$$Recall = \frac{A \cap B}{C}$$

Dimana A adalah kalimat hasil ringkasan manusia, B adalah kalimat yang dihasilkan sistem dan $A \cap B$ adalah jumlah kalimat yang sama dalam hasil ringkasan. C adalah jumlah kandidat kalimat ringkasan.