BAB 2
TINJAUAN PUSTAKA

2.1. Kesalahan Ejaan

Secara garis besar, kesalahan ejaan dapat dibagi menjadi dua:

1. Kesalahan ejaan *non-word*

Kesalahan *non-word* adalah kesalahan kata dimana kata yang salah tidak terdapat dalam kamus [4]. Contoh: Saya makan masi goreng. Di sini kata "masi" merupakan kesalahan dari kata "nasi", dan merupakan kesalahan *non-word* karena kata "masi" tidak ada dalam kamus.

Kesalahan *non-word* sudah banyak diteliti dan dibuatkan algoritmanya. Algoritma-algoritma ini biasanya dinamakan *spell-checker* atau *typo-checker*, dan banyak terdapat di perangkat lunak pengolah kata seperti *Microsoft Word*, *LibreOffice Writer*, *Ispell*, *Aspell*, dan lain-lain [2].

2. Kesalahan ejaan *real-word*

Kesalahan *real-word* adalah kesalahan kata dimana kata yang salah ada dalam kamus, tetapi bukan merupakan kata yang dimaksud [4]. Kesalahan-kesalahan *real-word* biasanya mengganggu sintaks dan semantik dari suatu kalimat, sehingga dibutuhkan bantuan manusia untuk mengkoreksinya. Contoh: Saya makan basi goreng. Di sini kata "basi" merupakan kesalahan *real-word*, karena walaupun kata ini terdapat dalam KBBI, kata ini bukanlah kata yang dimaksud, yaitu "nasi".

Program *spell-checker* biasa tidak dapat mendeteksi kesalahan *real-word*, karena program tersebut hanya dapat mendeteksi kesalahan kata jika kata tersebut tidak berada di dalam kamus. Bahkan kadangkala fitur *autocorrect* yang terdapat pada program malah mengubah kata yang harusnya benar menjadi kata yang salah [5].

Berdasarkan penelitian-penelitian yang disurvei oleh Kukich [6], kesalahan *real-word* mencakup sekitar 25% - 40% dari seluruh kesalahan ejaan yang sudah terdokumentasi.

2.2. Deteksi dan Koreksi Kesalahan *Real-Word*

Metode bigram dan trigram kata lokal ini mendeteksi kesalahan ejaan *real-word* dengan melihat bigram dan trigram dari kiri dan kanan kata kandidat, lalu memberikan sugesti koreksi kata berdasarkan perhitungan skor *confusion set* untuk kata tersebut.

Pada awalnya metode ini membuat *confusion set* untuk setiap kata kandidat menggunakan *Levenshtein distance* dari kata-kata kamus, lalu menghitung peringkat setiap elemen dalam *confusion set* tersebut. Berdasarkan itu, kesalahan dideteksi dan diberikan sugesti kata koreksi. [2]

2.2.1. *Confusion Set* Dengan *Levenshtein Distance*

Untuk membuat *confusion set*, dalam metode ini digunakan *Levenshtein Distance* [7], juga dikenal sebagai *minimum edit distance*, yaitu jumlah minimum operasi edit yang diperlukan untuk mengubah suatu kata menjadi kata yang lain.
Berikut adalah perhitungan *Levenshtein Distance*

$$d_j^i = min\begin{cases} d[i-1,j]+1 \\ d[i,j-1]+1 \\ d[i-1,j-1]+cost \end{cases}$$

(2.1)

Hasil penghitungan dapat dilihat pada kolom terakhir baris terakhir matriks. Kata-kata yang dijadikan anggota confusion set adalah kata-kata dalam daftar kata yang memiliki nilai *Levenshtein distance* maksimal 1.

Operasi edit merupakan pemasukan (*insertion*), penghapusan (*deletion*), atau penggantian (*substitution*) satu karakter dalam suatu kata. *Confusion set* dapat direpresentasikan sebagai

$$C(W_i) = \{W_1^i, W_2^i, \dots, W_j^i, \dots, W_{k_i}^i\}$$

(2.2)

dimana C adalah *count* disebut kemunculan dan W^i adalah kata ke-i di dalam kalimat tes dan k_i adalah jumlah elemen di dalam.
$C(W_i), C(W_i)$ adalah $count(W_i)$ dimana (W_i) memiliki $confusion\ set$ anggota i
.

2.2.2. Membuat Model N-gram

N-gram adalah rangkaian n karakter atau kata yang diekstrak dari suatu teks. Biasanya n-gram yang sering digunakan adalah bigram dan trigram, dengan nilai n masing-masing adalah 2 dan 3 [8].Contoh n-gram kata untuk kalimat "saya makan nasi goreng":

2-gram: "_ saya", "saya makan", "makan nasi", "nasi goreng", "goreng _"

3-gram: "_ saya makan", "saya makan nasi", "makan nasi goreng", "nasi goreng _".

Setelah *confusion set* dibuat, bangkitkan himpunan bigram kiri, bigram kanan, dan trigram untuk setiap anggota $C(W^i), C(W^i)$ adalah kemunculan kata dari W^i dimana W_i memiliki $confusion\ set$ anggota i. Ini dibuat dengan mengambil kata di kiri, kanan, dan keduanya (untuk trigram) dari kata tersebut. Untuk W^i, maka bigram dan trigram yang terbentuk adalah:

Bigram kiri: $W^{i-1}W_j^i$

(2.3)

$W^{i-1}W_j^i$ adalah kata W^i sebelum kata $confusion\ set\ W_j^i$

Bigram kanan: $W_j^i W^{i+1}$

(2.4)

$W_j^i W^{i+1}$ adalah kata W^i setelah kata $confusion\ set\ W_j^i$

Trigram: $W^{i-1}W_j^i W^{i+1}$

(2.5)

$W^{i-1}W_j^i W^{i+1}$ adalah kata W^i sebelum dan setelah kata $confusion\ set\ W_j^i$

dimana $0 \leq j \leq k_i$ (0 tidak lebih besar sama dengan dari anggota *confusion set* dan tidak lebih besar sama dengan dari banyaknya anggota *confusion set*)

2.2.3. Menghitung Probabilitas N-gram

Dalam model ini probabilitas kalimat tidak dihitung, diambil asumsi lemah bahwa kemunculan kejadian (kata) bergantung pada kejadian (kata) sebelum dan sesudahnya terlepas dari kejadian-kejadian (kata-kata) lain dalam kalimat, dengan

menggunakan *Maximum Likelihood Estimation* (MLE)[2], didapatkan probabilitas bigram dan trigram sebagai:

Probabilitas unigram dari kata W_j^i = jumlah kemunculan dari kata *confusion set* c_j^i unigram dibagi dengan total kemunculan dari anggota *confusion set* c_r^i unigram, maka didapatkan rumus unigram:

$$P_0(W_j^i) = \frac{count(c_j^i)}{\sum_{r=1}^{k_i} count(c_r^i)} \quad (2.6)$$

$count(c_j^i)$ adalah kemunculan kata dari *confusion set* anggota c_j^i.

$\sum_{r=1}^{k_i} count(c_r^i)$ adalah total kemunculan kata dari *confusion set* anggota c_j^i sampai ke c_r^i.

Probabilitas bigram kiri dari kata W_j^i = jumlah kemunculan dari kata *confusion set* c_j^i bigram kiri dibagi dengan total kemunculan dari anggota *confusion set* c_r^i bigram kiri, maka didapatkan rumus bigram kiri:

$$P_1(W_j^i | W^{i-1}) = \frac{count(W^{i-1}c_j^i)}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i)} \quad (2.7)$$

$count(W^{i-1}c_j^i)$ adalah kemunculan kata dari *confusion set* anggota kata bigram kiri c_j^i.

$\sum_{r=1}^{k_i} count(W^{i-1}c_r^i)$ adalah total kemunculan kata dari *confusion set* anggota kata bigram kiri c_j^i sampai ke c_r^i.

Probabilitas bigram kanan dari kata W_j^i = jumlah kemunculan dari kata *confusion set* c_j^i bigram kiri dibagi dengan total kemunculan dari anggota *confusion set* c_r^i bigram kiri, maka didapatkan rumus bigram kanan:

$$P_2(W_j^i | W^{i+1}) = \frac{count(W^{i+1}c_j^i)}{\sum_{r=1}^{k_i} count(W^{i+1}c_r^i)} \quad (2.8)$$

$count(W^{i+1}c_j^i)$ adalah kemunculan kata dari *confusion set* anggota kata bigram kanan c_j^i.

$\sum_{r=1}^{k_i} count(W^{i-1+}c_r^i)$ adalah total kemunculan kata dari *confusion set* anggota kata bigram kanan c_j^i sampai ke c_r^i.

Probabilitas trigram dari kata $W_j^i =$ jumlah kemunculan dari kata *confusion set* c_j^i trigram dibagi dengan total kemunculan dari anggota *confusion set* c_r^i trigram, maka didapatkan rumus trugram:

$$P_3\left(W_j^i \middle| W^{i-1}, W^{i+1}\right) = \frac{count(W^{i-1}c_j^i W^{i+1})}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i W^{i+1})}$$

(2.9)

$count(W^{i-1}c_j^i W^{i+1})$adalah kemunculan kata dari *confusion set* anggota kata bigram kiri c_j^i.

$\sum_{r=1}^{k_i} count(W^{i-1}c_r^i W^{i+1})$ adalah total kemunculan kata dari *confusion set* anggota kata trigram c_j^i sampai ke c_r^i.

Karena pada rumus tersebut membutuhkan nilai probabilitas pada persamaan 2.5 untuk unigram, 2.6 untuk bigram kiri, 2.7 untuk bigram kanan dan 2.8 untuk trigram terdapat masalah pada rumus persamaan tersebut, masalah yang akan didapat banyak probabilitas yang tidak bernilai (*Nan),* jika menggunakan rumus *maximum likelihood*.

Dari analisis penelitian sebelumnya untuk mengatasinya digunakan suatu metode *smoothing* yang menambahkan *pseudocount* pada probabilitas, metode *smoothing* yang digunakan adalah *additive smoothing* [3], yang menambahkan suatu konstanta δ pada *count*, dalam kasus ini δ yang digunakan adalah 0.01, dengan pertimbangan agar nilai yang dihasilkan tidak terlalu mempengaruhi penghitungan skor pada tahap selanjutnya. Maka rumus-rumus probabilitas n-gram menjadi:

Probabilitas unigram dari kata $W_j^i =$ jumlah kemunculan dari kata *confusion set* c_j^i unigram dikalikan dengan *delta* dan dibagi dengan total kemunculan dari anggota *confusion set* c_r^i unigram ditambah *delta* dan dikalikan banyakya *confusion set*, maka didapatkan rumus unigram:

$$P_0'(c_j^i) = \frac{count(c_j^i) \times \delta}{\sum_{r=1}^{k_i} count(c_r^i) + \delta \times V_j}$$

(2.10)

$count(c_j^i) \times \delta$ adalah kemunculan kata dari *confusion set* anggota kata bigram kiri c_j^i.

$\delta = 0.01$

(2.11)

$V_j =$ jumlah $confusion\ set$ pada kata c_j^i

$\sum_{r=1}^{k_i} count(c_r^i) + \delta \times V_j$ adalah total kemunculan kata dari *confusion set* anggota kata trigram c_j^i sampai ke c_r^i.

Probabilitas bigram kiri dari kata $W_j^i =$ jumlah kemunculan dari kata *confusion set* c_j^i bigram kiri dikalikan dengan *delta* dan dibagi dengan total kemunculan dari anggota *confusion set* c_r^i bigram kiri ditambah *delta* dan dikalikan banyakya *confusion set*, maka didapatkan rumus bigram kiri:

$$P_1'(c_j^i | W^{i-1}) = \frac{count(W^{i-1}c_j^i) + \delta}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i) + \delta \times V_j}$$

(2.12)

$count(W^{i-1}c_j^i) + \delta$ adalah kemunculan kata dari *confusion set* anggota kata bigram kiri c_j^i.

δ didapat dari persamaan 2.11.

$V_j =$ jumlah $confusion\ set$ pada kata c_j^i

$\sum_{r=1}^{k_i} count(W^{i-1}c_r^i) + \delta \times V_j$ adalah total kemunculan kata dari *confusion set* anggota kata trigram c_j^i sampai ke c_r^i.

Probabilitas bigram kanan dari kata $W_j^i =$ jumlah kemunculan dari kata *confusion set* c_j^i bigram kanan dikalikan dengan *delta* dan dibagi dengan total kemunculan dari anggota *confusion set* c_r^i bigram kanan ditambah *delta* dan dikalikan banyakya *confusion set*, maka didapatkan rumus bigram kanan:

$$P_2{}'\left(c_j^i \middle| W^{i+1}\right) = \frac{count(W^{i+1}c_j^i)+\delta}{\sum_{r=1}^{k_i} count(W^{i+1}c_r^i)+\delta \times V_j}$$

(2.13)

$count(W^{i+1}c_j^i) + \delta$ adalah kemunculan kata dari *confusion set* anggota kata bigram kiri c_j^i.

δ didapat dari persamaan 2.11.

$V_j =$ jumlah $confusion\ set$ pada kata c_j^i

$\sum_{r=1}^{k_i} count(W^{i+1}c_r^i) + \delta \times V_j$ adalah total kemunculan kata dari *confusion set* anggota kata trigram c_j^i sampai ke c_r^i.

Probabilitas trigram dari kata $W_j^i =$ jumlah kemunculan dari kata *confusion set* c_j^i trigram dikalikan dengan *delta* dan dibagi dengan total kemunculan dari anggota *confusion set* c_r^i trigram ditambah *delta* dan dikalikan banyakya *confusion set*, maka didapatkan rumus trigram:

$$P_3{}'\left(c_j^i \middle| W^{i-1}, W^{i+1}\right) = \frac{count\left(W^{i-1}c_j^i W^{i+1}\right)+\delta}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i W^{i+1})+\delta \times V_j}$$

(2.14)

$count(W^{i-1}c_j^i W^{i+1})$adalah kemunculan kata dari *confusion set* anggota kata bigram kiri c_j^i.

δ didapat dari persamaan 2.11.

$V_j =$ jumlah $confusion\ set$ pada kata c_j^i

$\sum_{r=1}^{k_i} count(W^{i-1}c_r^i W^{i+1})$ adalah total kemunculan kata dari *confusion set* anggota kata trigram c_j^i sampai ke c_r^i.

Dalam persamaan 2.10, hitung P_1 untuk setiap elemen *confusion set* untuk setiap kata menggunakan jumlah bigram kiri. Penyebut di sini mewakili jumlah semua bigram dari kata sebelumnya dan satu kata dari *confusion set*. Dengan cara yang sama, hitung P_2 dengan persamaan 2.11. Hitung persamaannya untuk setiap elemen di setiap *confusion set* sehingga memuaskan kondisi berikut:

Probabilitas bigram kiri untuk setiap elemen *confusion set* untuk setiap kata menggunakan jumlah bigram kiri:

$$\sum_{r=1}^{k_i} P_1\left(W_j^i \middle| W^{i-1}\right) = 1 \quad (2.15)$$

Probabilitas bigram kanan untuk setiap elemen *confusion set* untuk setiap kata menggunakan jumlah bigram kanan:

$$\sum_{r=1}^{k_i} P_2\left(W_j^i \middle| W^{i+1}\right) = 1 \quad (2.16)$$

Untuk persamaan 2.12 lakukan langkah yang sama seperti persamaan 2.10 dan 2.11, bedanya dengan trigram yaitu dari kata sebelum, satu kata dari *confusion set*, dan kata sesudah. Lakukan untuk setiap elemen di setiap *confusion set* sehingga memastikan kondisi berikut:

$$\sum_{r=0}^{k_i} P_3\left(W_j^i \middle| W^{i-1}, W^{i+1}\right) = 1 \quad (2.17)$$

Gabungkan ketiga hasil persamaan (2.13), (2.14), (2.15) untuk menghasilkan skor bukti bahwa kata c_j^i adalah alternatif yang benar untuk W^i. Skor dihitung dengan menggunakan *weighted combination score*.

2.2.4. Metode *Smoothing* Yang Digunakan

Ada beberapa metode *smoothing* yang menggunakan *interpolation*, yaitu yang berhubungan dengan probabilitas n-gram di bawah tingkatannya, metode *smoothing* yang menggunakan *interpolation* adalah *jelinek mercer, katz smoothing, absolute discounting*[9]. Contoh *interpolation* pada metode *jelinek mercer* seperti pada persamaan 2.17 seperti dibawah ini:

Bigram $P_1'(c_j^i|W^{i-1}) = \lambda\, \boldsymbol{P_1}(\boldsymbol{c_j^i|W^{i-1}}) + (1-\lambda)\, \boldsymbol{P_0}(\boldsymbol{c_j^i})$

Yang dihitamkan pada $\boldsymbol{P_1}$ adalah perhitungan probabilitas bigram kiri, dan $\boldsymbol{P_0}$ adalah perhitungan probabilitas unigram, bigram kiri tersebut mengkalikan probabilitas bigram kiri dengan probabilitas unigram.

Untuk perhitungan trigram, disamakan dengan saat perhitungan bigram, karna bigram ada dua yaitu bigram kiri dan kanan maka bigram kiri dan kanan akan dijumlahkan lalu dibagi 2. Berikut contoh trigram seperti pada persamaan 2.19:

$P_3'(c_j^i|W^{i-1}, W^{i+1}) =$

$\lambda\, \boldsymbol{P_3}(\boldsymbol{c_j^i}|\boldsymbol{W^{i-1}}, \boldsymbol{W^{i+1}}) + (1-\lambda)\, \boldsymbol{P_1}(\boldsymbol{c_j^i}|\boldsymbol{W^{i-1}}) + \boldsymbol{P_2}(\boldsymbol{c_j^i}|\boldsymbol{W^{i+1}})/\boldsymbol{2}$

Yang dihitamkan pada $\boldsymbol{P_3}$ adalah perhitungan probabilitas trigram, $\boldsymbol{P_1}$dan $\boldsymbol{P_2}$ adalah perhitungan probabilitas bigram kiri dan kanan, trigram tersebut mengkalikan probabilitas trigram dengan probabilitas bigram kiri dan kanan.

Metode yang akan digunakan pada metode *smoothing* ada 5 metode yaitu:

1. *Good-Turing estimate* [10]

   $\frac{1}{N}(r+1)\frac{N_{r+1}}{N_r}$,

   (2.18)

   N adalah $N = \sum_r rN_r$.

   (2.19)

   r adalah frekuensi kemunculan

   N_r = jumlah frekuensi r yang muncul
2. *Jelinek-Mercer* [9]

   Bigram Kiri $P_1'(c_j^i|W^{i-1}) = \lambda\, P_1(c_j^i|W^{i-1}) + (1-\lambda)\, P_0(c_j^i)$

   (2.20)

   Bigram Kanan $P_2'\,(c_j^i|W^{i+1}) = \lambda\, P_2(c_j^i|W^{i+1}) + (1-\lambda)\, P_0(c_j^i)$

   (2.21)

   Trigram $P_3' = (c_j^i|W^{i-1}, W^{i+1}) = \lambda\, P_3(c_j^i|W^{i-1}, W^{i+1}) + (1-\lambda)(P_1(c_j^i|W^{i-1}) + P_2(c_j^i|W^{i+1})/2)$

   (2.22)

   $P_0(c_j^i)$ adalah model unigram, rumus unigram didapat dari persamaan 2.10.

   $P_2'\,(c_j^i|W^{i+1})$ adalah model bigram kiri, rumus bigram kiri didapat dari persamaan 2.12.

$P_2'(c_j^i|W^{i+1})$ adalah model bigram kanan, rumus bigram kanan didapat dari persamaan 2.13.

$P_3' = (c_j^i|W^{i-1}, W^{i+1})$ adalah model bigram kanan, rumus bigram kanan didapat dari persamaan 2.14.

λ = 0.1.

(2.23)

3. *Katz smoothing* [9]

   Ada 2 aturan untuk metode *katz smoothing*:

   1. Jika $r > 0$ gunakan rumus $P_1(c_j^i|W^{i-1}) = dr$

      $$dr = \frac{r^*}{r}$$

      (2.24)

      $$r^* = (r+1)\frac{N_{r+1}}{Nr}$$

      (2.25)

      r = frekuensi dari kemunculan kata.

      Nr adalah banyalnya frekuensi kemunculan kata.

   2. Jika $r = 0$ gunakan rumus

      Bigram Kiri $\alpha(c_j^i|W^{i-1})P_0(c_j^i)$

      (2.26)

      $$\alpha(c_j^i|W^{i-1}) = \frac{1-\sum_{w_i:count(W^{i-1}c_r^i)>0} P_1(c_j^i|W^{i-1})}{1-\sum_{w_i:count(c_r^i)>0} P_0(c_j^i)}$$

      (2.27)

      Bigram Kanan $\alpha(c_j^i|W^{i+1})P_0(c_j^i) = \frac{1-\sum_{w_i:count(W^{i+1}c_r^i)>0} P_2(c_j^i|W^{i+1})}{1-\sum_{w_i:count(c_r^i)>0} P_0(c_j^i)}$

      (2.28)

      Trigram $\alpha(c_j^i|W^{i-1}, W^{i+1})\ (P_1(c_j^i|W^{i-1}) + P_2(c_j^i|W^{i+1})/2)$

$$\alpha\left(c_j^i|W^{i-1},W^{i+1}\right)=$$

$$\frac{1-\sum_{w_i:count\left(W^{i-1},W^{i+1}c_r^i\right)>0}P_3(c_j^i|W^{i-1},W^{i+1})}{1-\sum_{w_i:count\left(W^{i-1}c_r^i\right),count\left(W^{i+1}c_r^i\right)>0}(P_1(c_j^i|W^{i-1})+P_2(c_j^i|W^{i-1}))/2}$$

(2.29)

$\sum_{w_i:count\left(c_j^i\right)>0}P_0\left(c_j^i|W^{i-1}\right)$ adalah jumlah W_i yang memiliki jumlah kemunculan unigram yang lebih dari 0.

$\sum_{w_i:count\left(c_j^iw_{i-1}\right)>0}P_1\left(c_j^i|W^{i-1}\right)$ adalah jumlah W_i yang memiliki jumlah kemunculan bigram kanan yang lebih dari 0.

$\sum_{w_i:count\left(c_j^iw_{i+1}\right)>0}P_2\left(c_j^i|W^{i+1}\right)$ adalah jumlah W_i yang memiliki jumlah kemunculan bigram kanan yang lebih dari 0.

$\sum_{w_i:count\left(c_j^iw_{i-1}\right)>0}P_3\left(c_j^i|W^{i-1},W^{i+1}\right)$ adalah jumlah $W_i$yang memiliki jumlah kemunculan trigram yang lebih dari 0.

$\sum_{w_i:count(W^{i-1}c_r^i),count(W^{i+1}c_r^i)>0}(P_1(c_j^i|W^{i-1})+P_2\left(c_j^i|W^{i-1}\right))/2$ adalah jumlah W_i yang memiliki jumlah kemunculan bigram kiri dan bigram kanan yang lebih dari 0.

4. *Witten-Bell* [3]

Bigram kiri $P_1'(c_j^i|W^{i-1})=$

$$\frac{count\left(W^{i-1}C_j^i\right)+N_{1+}\left(c_j^i|W^{i-1}\cdot\right)P_1(c_j^i|W^{i-1})}{\sum_{r=1}^{k_i}count(W^{i-1}c_r^i)+0.01\times V_j+N_{1+}\left(c_j^i|W^{i-1}\cdot\right)} \qquad (2.30)$$

$$N_{1+}\left(c_j^i|W^{i-1}\cdot\right)=|\{W^i:count(c_j^iW^{i-1})>0\}|$$

(2.31)

$N_{1+}\left(c_j^i|W^{i-1}\cdot\right)$ adalah nilai dari kata W^i dimana kemunculan kata dari bigram kiri yang memiliki nilai lebih besar dari 0.

Bigram kanan $P_1'(c_j^i|W^{i+1})=$

$$\frac{count\left(W^{i+1}C_j^i\right)+N_{1+}\left(c_j^i|W^{i+1}\cdot\right)P_2(c_j^i|W^{i+1})}{\sum_{r=1}^{k_i}count(W^{i+1}c_r^i)+0.01\times V_j+N_{1+}\left(c_j^i|W^{i+1}\cdot\right)}$$

(2.32)

$N_{1+}(c_j^i|W^{i+1}\ \cdot) = |\{W^i: count(c_j^i W^{i+1}) > 0\}|$

(2.33)

$N_{1+}(c_j^i|W^{i+1}\ \cdot)$ adalah nilai dari kata W^i dimana kemunculan kata dari bigram kanan yang memiliki nilai lebih besar dari 0.

Trigram $P_1'(c_j^i|W^{i-1}, W^{i+1}) =$

$$\frac{count(W^{i-1}, W^{i+1}c_j^i) + N_{1+}(c_j^i|W^{i-1}, W^{i+1}\ \cdot)\, P_3(c_j^i|W^{i-1}, W^{i+1})}{\sum_{r=1}^{k_i} count(W^{i-1}, W^{i+1}c_r^i) + 0.01 \times V_j + N_{1+}(c_j^i|W^{i-1}, W^{i+1}\ \cdot)} \quad (2.34)$$

$N_{1+}(c_j^i|W^{i-1}, W^{i+1}\ \cdot) = |\{W^i: count(c_j^i W^{i-1}, W^{i+1}) > 0\}|$

(2.35)

$N_{1+}(c_j^i|W^{i-1}, W^{i+1}\ \cdot)$ adalah nilai dari kata W^i dimana kemunculan kata dari trigram yang memiliki nilai lebih besar dari 0.

5. *Absolute discounting* [9]

Bigram kiri $=$

$$\frac{\max\{count(c_j^i W^{i-1}) - \delta, 0\}}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i) + 0.01 \times V_j} + (1-\lambda)\, P_0(c_j^i)$$

(2.36)

Untuk mencari nilai $(1-\lambda) =$

$$\frac{\delta}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i) + 0.01 \times V_j}\ N_{1+}(c_j^i W^{i-1}\ \cdot)$$

(2.37)

Untuk rumus $N_{1+}(c_j^i W^{i-1}\ \cdot)$ bisa dilihat dari persamaan 2.31.

δ didapat dari persamaan 2.11.

Bigram Kanan $=$

$$\frac{\max\{count(c_j^i W^{i+1}) - \delta, 0\}}{\sum_{r=1}^{k_i} count(W^{i+1}c_r^i) + 0.01 \times V_j} + \quad (1-\lambda)\, P_0(c_j^i)$$

(2.38)

$$\frac{\delta}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i) + 0.01 \times V_j}\ N_{1+}(c_j^i W^{i+1}\ \cdot)$$

(2.39)

Untuk rumus $N_{1+}(c_j^i W^{i+1}\ \cdot)$ bisa dilihat dari persamaan 2.33.

δ didapat dari persamaan 2.11.

Trigram =

$$\frac{\max\{count(c_j^i W^{i-1}, W^{i+1}) - \delta, 0\}}{\sum_{r=1}^{k_i} count(W^{i-1}, W^{i+1} c_r^i) + 0.01 \times V_j} + (1-\lambda)(\, P_1(c_j^i | W^{i-1}) + \, P_2(c_j^i | W^{i+1})/2)$$

(2.40)

Untuk mencari nilai $(1-\lambda) =$

$$\frac{\delta}{\sum_{r=1}^{k_i} count(W^{i-1}, W^{i+1} c_r^i) + 0.01 \times V_j} \; N_{1+}(c_j^i W^{i-1}, W^{i+1} \cdot)$$

(2.41)

Nilai dari $\delta = 0.01$ didapat dari persamaan

Untuk rumus $N_{1+}(c_j^i W^{i-1}, W^{i+1} \cdot)$ bisa dilihat pada persamaan 2.35.

δ didapat dari persamaan 2.11.

2.2.5. *Weighted Combination Score*

Model-model n-gram berorde tinggi maupun rendah memiliki kelebihan dan kekurangan masing-masing. N-gram berorde lebih tinggi lebih sensitif terhadap konteks, namun memiliki jumlah hitungan sedikit. N-gram berorde lebih rendah lebih terbatas mengenal konteks, tetapi jumlah hitungannya lebih banyak. Oleh karena itu model ini menggabungkan bigram dan trigram agar tidak terlalu bergantung pada suatu n-gram. [2]

Dengan menggunakan *weighted combination score*, jumlahkan ketiga hasil persamaan (2.5), (2.6), (2.7) yang masing-masing telah diberi bobot, menjadi:

$$Score(W_j^i) = \lambda_1 P_1(W_j^i | W^{i-1}) + \lambda_2 P_2(W_j^i | W^{i+1}) + \lambda_3 P_3(W_j^i | W^{i-1}, W^{i+1})$$

(2.42)

Nilai $\lambda_1, \lambda_2, \lambda_3$ dapat dihitung dengan mengoptimasi ketepatan training set. Dari penelitian [2], nilai terbaik adalah $\lambda_1 = \lambda_2 = 0.25$ dan $\lambda_3 = 0.5$. Juga, W_j^i terbatas pada $0 \leq Score(W_j^i) \leq 1$.

Untuk perhitungan skor pada koreksi dan koreksi kata jika kata salah, diberikan rekomendasi kata koreksi yang didapat dari urutan peringkat *confusion*

set kata tersebut berdasarkan urutan peringkat *confusion set*, peringkat teratas berarti rekomendasi kata yang paling cocok secara konteks. Jika anggota itu skornya 0 atau kurang dari 1% skor anggota peringkat tertinggi *confusion set* , maka kata yang diuji dianggap salah.

2.2.6. Deteksi Kesalahan dan Pemilihan Kata Sugesti

Untuk mengkonfirmasi suatu kata sebagai kesalahan *real-word*, diberikan beberapa aturan. Pertama, urutkan elemen-elemen *confusion set* berdasarkan skornya dari yang terbesar sampai yang terkecil. Gunakan kepercayaan apriori bahwa kata uji yang diamati bukanlah suatu kesalahan *real-word*. Dalam penelitiannya Mays et al. mendapatkan nilai optimum kepercayaan ini sebagai 0.99 [11] yang akan digunakan dalam metode ini juga. Dengan kata lain, dipercaya bahwa kata yang diuji dapat menjadi kesalahan kata *real-word* dalam 1% kasus. Nilai ini digunakan untuk normalisasi skor dalam algoritma deteksi kesalahan *real-word* yang akan dijelaskan di bawah ini.

Misalkan W^i adalah kata uji dalam suatu kalimat. Berdasarkan hasil skor yang didapat, buat keputusan berdasarkan *pseudocode* berikut:

```
Begin
if Score(W_i) = 0
     nyatakan W^i sebagai kesalahan real-word
else
if Score(W_i) < 0.01 *Skor tertinggi elemen confusion set
     nyatakan W_i sebagai kesalahan real-word
else
     W_i benar
end if
end if
End
```

2.3. *Preprocessing*

Preprocessing digunakan untuk pembuatan korpus n-gram dan mencari *confusion set* untuk deteksi dan koreksi kesalah ejaan *real-word*. Isi dari *preprocessing*[12] adalah:

1. *Case folding*

   Case folding adalah proses mengubah semua huruf kapital menjadi huruf kecil.

2. Tokenisasi kalimat

   Tokenisasi kalimat yaitu proses memisahkan tiap-tiap kalimat.

3. *Filtering* tanda baca

   Filtering tanda baca yaitu proses yang menghapus tanda baca karakter selain karakter huruf, angka atau spasi.

4. Tokenisasi kata

   Tokenisasi kata yaitu proses kalimat yang telah dihilangkan tanda bacanya dipisah-

   pisah per kata menjadi deretan kata tunggal.

2.4. Pemodelan

Skema pemodelan data adalah metode yang memungkinkan untuk memodelkan atau menggambarkan *database* pada aplikasi pengkoreksi kesalahan ejaan *real-word* yang akan dibangun.

2.4.1. *Entity Relationship Diagram*

ERD merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak. Tujuan dari *entity relationship* adalah untuk menunjukkan objek data dan relationship yang ada pada objek tersebut. Di samping itu model ERD ini merupakan salah satu alat untuk perancangan dalam basis data.[13]

Komponen-komponen ERD adalah sebagai berikut:

1. Entitas

Entitas (*entity*) menunjukkan objek-objek dasar yang terkait di dalam sistem. Objek dasar dapat berupa orang, benda atau hal lain yang keterangannya perlu disimpan dalam basis data.

2. Atribut

   Atribut sering juga disebut sebagai properti merupakan keterangan-keterangan yang terkait pada sebuah entitas yang perlu disimpan sebagai basis data. Atribut berfungsi sebagai penjelas sebuah entitas.

   Jenis-jenis Kardinalitas:

   a. Satu ke Satu (1:1)

      Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding satu berbanding satu.

   b. Satu ke Banyak (1:N)

      Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding satu berbanding banyak.

   c. Banyak ke Satu (N:1)

      Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding banyak berbanding satu.

   d. Banyak ke Banyak (N:N)

      Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding banyak berbanding banyak.

3. Relasi

Adalah karakteristik dari *entity* atau *relationship* yang menyediakan penjelasan detail tentang *entity* atau relationship tersebut. Relasi atau hubungan adalah kejadian atau transaksi yang terjadi di antara dua entitas yang keterangannya perlu disimpan dalam basis data.

4. Derajat Relasi (Kardinalitas)

Kardinalitas relasi menunjukkan maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi yang terjadi di antara dua himpunan entitas (misalkan A dan B) dapat berupa satu ke satu (*one to one*), satu ke banyak (*one to many*), banyak ke satu (*many to one*), dan banyak ke banyak (*many to many*).

2.4.2. *Flowchart*

Flowchart adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah penyelesaian suatu masalah. *Flowchart* merupakan cara penyajian suatu algoritma. *Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program. *Flowchart* biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut [13].

Tujuan membuat *Flowchart* adalah menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai, rapi dan jelas menggunakan simbol-simbol standar.

2.4.3. Diagram Konteks

Diagram Konteks adalah sebuah diagram sederhana yang menggambarkan hubungan antara *entity* luar, masukan dan keluaran dari sistem. Diagram konteks direpresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem [13].

Diagram konteks dimulai dengan penggambaran terminator, aliran data, aliran kontrol penyimpanan, dan proses tunggal yang menunjukan keseluruhan sistem. Ada beberapa aturan dalam menggambarkan diagram konteks, berikut di bawah ini menunjukan aturan tersebut.

1. Menggunakan hanya satu simbol proses.
2. Memberi label simbol proses tersebut untuk menggambar seluruh sistem, biasanya berupa kata kerja di tambah objek.
3. Tidak memberi nomor pada simbol proses.
4. Menyertakan semua terminator dari system.
5. Menunjukan semua arus data antara terminator dan system.

2.4.4. DFD (*Data Flow Diagram*)

DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan darimana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan

interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut [13].

1. Arus Data (*Data Flow*)

   Menunjukan arus dari data yang dapat berupa masukan untuk sistem atau dari proses sistem.

2. Proses

   Proses adalah kegiatan yang dilakukan oleh orang, mesin atau komputer dari hasil arus data yang masuk ke dalam proses untuk dihasilkan arus data yang akan keluar dari proses.

3. Kesatuan Luar (*External Entity*)

   Kesatuan luar merupakan kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lain yang akan memberikan masukan atau menerima keluaran dari sistem.

4. File

   Kumpulan data yang disimpan dengan cara tertentu. Data yang mengalir disimpan dalam file. Aliran data di-update atau ditambahkan ke dalam file.

2.4.5. Kamus Data

Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan (*input*) dan keluaran (*output*) dapat dipahami secara umum [13].

Kamus data mendefinisikan elemen data dengan menjelaskan arti aliran data dan penyimpanan data dalam DFD, mendeskripsikan komposisi paket data yang bergerak melalui aliran, mendeskripsikan komposisi penyimpanan data, menspesifikasikan nilai dan satuan yang relevan bagi penyimpanan dan aliran, mendeskripsikan hubungan detil antar penyimpanan.

Kamus data berfungsi untuk membantu pelaku sistem untuk mengartikan alokasi secara detail dan mengorganisasikan semua elemen data yang digunakan dalam sistem secara persis sehingga baik pemakai atau penganalisis sistem mempunyai dasar pengertian yang sama tentang masukan, keluaran, penyimpanan dan proses.

Kamus data biasanya berisi:

1. Nama-nama dari data
2. Digunakan pada – merupakan proses-proses yang terkait data
3. Deskripsi – merupakan deskripsi data
4. Informasi tambahan – seperti tipe data, nilai data, batas nilai data, dan komponen yang membentuk data.

Kamus data memiliki simbol untuk menjelaskan informasi tambahan seperti pada tabel 2.3 berikut.

Tabel 2.1 Kamus Data

Simbol	Keterangan
=	disusun atau terdiri dari
+	dan
[|]	baik ... atau....
$\{\}^n$	n kali diulang/ bernilai banyak
()	data opsional
...	batas komentar

2.5. Bahasa Pemrograman

Bahasa pemrograman merupakan kumpulan aturan yang disusun sedemikian rupa sehingga memungkinkan pengguna komputer membuat program yang dapat dijalankan dengan aturan tersebut [14]. Beberapa contoh bahasa pemrograman yang banyak dipakai, yaitu C, C++, C#, Pascal, Java, JavaScript, PHP, SQL, dan Python. Bahasa pemrograman yang digunakan pada pembangunan aplikasi pengkoreksi kesalahan ejaan *real-word* ini adalah PHP, dan JavaScript.

2.5.1. PHP

Hypertext Prepocessor (PHP) adalah skrip yang menyatu dengan kode HTML dan berjalan pada sisi server (server-side). Artinya semua sintaks yang kita berikan akan sepenuhnya dijalankan pada server secangkan yang dikirimkan ke browser hanya hasilnya [15].

Keuntungan menggunakan PHP adalah tidak diedarkannya kode sumber yang membangun PHP ke sisi klien sehingga kerahasiaan kode dapat dilindungi. PHP berjalan pada sisi server, sehingga untuk dapat menggunakannya kita harus mengaktifkan web server terlebih dahulu baik offline maupun online.

Deklarasi variabel dengan tipe data tidak diperlukan dalam PHP seperti bahasa pemrograman pascal. Setiap variabel yang terbentuk dalam program dianggap bertipe variant, dengan kata lain dapat menampung tipe data dengan jenis apapun (*dynamic variable*). Sistem yang dibangun menggunakan PHP versi 5.6.19.

2.5.2. *JavaScript* (*AJAX & jQuery*)

HTTP (*Hypertext Transfer protocol*) merupakan protokol utama dibalik WWW (*World Wide Web*). Protokol ini yang menangani semua permintaan halaman website, gambar, link, dan form. Dengan adanya HTTP memungkinkan sebuah cara yang terstandarisasi untuk komputer berkomunikasi dengan komputer lainnya. HTTP akan menspesifikasikan bagaimana klien meminta data, dan bagaimana server memproses permintaan tersebut. HTTP memiliki tiga tipe permintaan dasar: POST, GET, dan HEAD [18]. Permintaan HEAD digunakan untuk memverifikasi bahwa sebuah halaman web yang diminta masih tersedia. Sebuah mesin pencari biasanya menggunakan permintaan HEAD untuk memastikan bahwa hasil pencariannya masih aktif. Jika permintaan HEAD gagal, maka mesin pencari tersebut bisa langsung menghapus hasil pencarian dari *database* yang dimiliki oleh mesin pencari tersebut.

Javascript merupakan salah satu bahasa pemrograman yang pertama kali dikembangkan oleh Netscape yang digunakan dalam sebuah browser atau perambah *web* [16]. *JavaScript* dibuat agar mudah diintegrasikan ke dalam program dan aplikasi berbasiskan web. AJAX merupakan singkatan dari *Asynchronous JavaScript* danXML merupakan bagian dari JavaScript yang bertugas atau berfungsi untuk memodifikasi sebagian atau semua dari sebuah halaman web secara dinamis pada saat dibutuhkan saja [17].

2.6. Pengelola *Database*

Database merupakan kumpulan file-file yang berhubungan secara logis dan digunakan secara rutin pada operasi-operasi sistem informasi manajemen. Semua *database* umumnya berisi elemen-elemen data yang disusun ke dalam file-file yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di hardware komputer dan dengan software untuk melakukan manipulasi data untuk kegunaan tertentu. Jadi, suatu *database* adalah menunjukkan suatu kumpulan tabel yang dipakai dalam suatu lingkup perusahaan atau instansi untuk tujuan tertentu [18].

Software yang digunakan untuk mengelola dan permintaan panggilan (*query*) basis data yang disebut sistem manajemen *database* (*database management system*, DBMS).

Dalam pengelolaan *database* menggunakan MySQL dengan menggunakan bahasa SQL. Penjabaran MySQL dan SQL akan dijelaskan seperti berikut.

2.6.1. SQL (*Structured Query Language*)

Structured Query Language adalah bahasa yang digunakan untuk mengelola data pada RDBMS. SQL digunakan untuk mengambil, menambahkan, dan memanipulasi data dari sebuah basis data relasional [15].

Bahasa SQL terbagi menjadi dua kategori yaitu kategori pertama *schema statements* kategori pernyataan *(statements)* untuk membuat objek basis data berupa tabel, index, constraints, dan sebagainya. Kategori kedua (SQL *data statements*) yaitu pernyataan yang digunakan untuk memanipulasi dan mengambil data yang tersimpan dalam basis data.

2.6.2. MySQL

MySQL merupakan *database server* yang bersifat *multiuser* dan *multi-threaded*. *SQL* adalah bahasa *database* standar yang memudahkan penyimpanan, pengubahan dan akses informasi.

Pada MySQL dikenal dengan istilah *database* dan tabel. Tabel sendiri merupakan sebuah struktur data dua dimensi yang terdiri dari baris-baris record dan kolom [19].

MySQL termasuk salah satu *database* open source yang paling banyak digunakan karena selain gratis MySQL pun mempunyai banyak dukungan bahasa pemrograman dan aplikasi sebagai solusi *database*.

Dalam penelitian ini MySQL digunakan untuk membuat dan mengola *database* korpus kamus dan korpus n-gram beserta isinya.

2.7. Perangkat Lunak Pembangunan

Perangkat lunak adalah istilah khusus untuk data yang diformat, dan disimpan secara digital, termasuk program komputer, dokumentasinya, dan berbagai informasi yang bisa dibaca, dan ditulis oleh komputer. Dengan kata lain, bagian sistem komputer yang tidak berwujud.

Dalam pembangunan dan pengujian aplikasi pengkoreksi kesalahan ejaan *real-word*, peneliti menggunakan XAMPP sebagai *web server* untuk pembangunan dan pengujian aplikasi pengkoreksi kesalahan ejaan *real-word* yang dibangun, *Web Browser*, dengan Sublime Text 3 sebagai penyunting kode bahasa pemrograman.

2.7.1. XAMPP

XAMPP adalah aplikasi *web server* yang berdiri sendiri terdiri dari Apache HTTP Server, MySQL *database* dan PHP. XAMPP juga dilengkapi dengan fitur manajemen *database* PHPMyAdmin seperti pada server hosting yang terhubung internet [19].

Untuk pembangunan sumber daya jenis kata menggunakan XAMPP sebagai *webserver*. XAMPP memiliki kelebihan untuk bisa berperan sebagai server web Apache dalam melakukan simulasi pengembangan web. Tool pengembangan web ini mendukung teknologi web populer seperti PHP, MySQL dan Perl.

Dalam penelitian ini XAMPP digunakan sebagai server *localhost* untuk membuka aplikasi koreksi kesalahan ejaan *real-word* secara *offline.*

2.7.2. *Web Browser*

Web Browser adalah layanan internet untuk menjelajah atau berselancar di dunia maya dengan menggunakan jaringan internet[20]. Fungsi dari *Web Browser* sendiri adalah untuk menampilkan halaman web atau melakukan interaksi dengan dokumen yang disediakan server.

Penelitian ini menggunakan Google Chrome versi 62.0.3202.94 sebagai *Web Browser* untuk pembangunan dan pengujian aplikasi pengkoreksi kesalahan ejaan *real-word* yang dibangun.

2.7.3. *Notepad++*

Notepad++ adalah sebuah penyunting teks dan penyunting kode sumber yang berjalan di sistem operasi Windows. Notepad++ menggunakan komponen Scintilla untuk dapat menampilkan dan menyuntingan teks dan berkas kode sumber berbagai bahasa pemrograman. Berikut sejumlah bahasa pemrograman yang didukung oleh Notepad++ sejak versi 5.9.3 – 7.5.1 yaitu ActionScript, Ada, ASP, Assembler, AutoIt, Batch, C, C++, C#, Caml, Cmake, COBOL, CSS, D. Diff, Flash ActionScript, Fortran, Gui4CLI, Haskell, HTML, Berkas INI, InnoSetup, Java, Javascript, JSP, KiXtart, LISP, Lua Makefile, Matlab, MS-DOS, NSIS, Objective-C, Pascal, Perl, PHP, Postscript, PowerShell, Properties file, Python, R, Resource file, Ruby, hell, Scheme, Smalltalk, SQL, TCL, TeX, Pascal, Perl, PHP, Postscript, PowerShell, Properties file, Python, XML, YAML[21].

2.8. Rencana Pengujian

Kinerja dari metode deteksi dan koreksi kesalahan ejaan *real-word* ini diukur dengan melakukan pengukuran akurasi deteksi dan akurasi koreksi[1]. Pengukuran akurasi deteksi dilakukan dengan persamaan

$$\text{Akurasi deteksi} = \frac{ds}{dm} \times 100\% \tag{2.43}$$

dimana

ds : jumlah deteksi yang dihasilkan sistem

dm : jumlah deteksi yang dihasilkan secara manual

Sedangkaran pengukuran akurasi koreksi dilakukan dengan persamaan

$$\text{Akurasi koreksi} = \frac{kb}{kb+ks} \times 100\% \quad (2.44)$$

dimana

kb : jumlah koreksi benar yang dihasilkan sistem

ks : jumlah koreksi salah yang dihasilkan sistem