BAB 3 PERANCANGAN

Pada bab ini akan dibahas mengenai model *phonetic similarity* yang akan dikembangkan, matriks substitusi dan peranannya dalam model, perancangan kuesioner, jumlah subjek yang dibutuhkan, penyajian kuesioner, pilot kuesioner yang dilakukan, dan penghitungan word distance.

3.1. Model phonetic similarity

Model *phonetic similarity* yang dikembangkan dalam penelitian ini ditujukan sebagai sebuah metrik untuk mengukur kemiripan antara pasangan bunyi kata. Model yang dibuat diharapkan bisa menjawab pertanyaan seperti: apakah bunyi “sapi” dan “sapu” lebih mirip daripada bunyi “sapu” dan “sepatu”?

Pada level yang lebih rendah, kemiripan bunyi antar kata terdiri dari kemiripan antar fonem-fonem penyusunnya. Misalnya kata "sapi" terdiri dari fonem /s/, /a/, /p/, dan /i/ sesuai dengan kamus fonetik Amalia Zahra [ZAR08] (lihat subbab 2.1) dan kata "sapu" terdiri dari fonem /s/, /a/, /p/, dan /u/. Oleh karena itu, kemiripan bunyi pasangan kata "sapi" dan "sapu" sangat ditentukan oleh kemiripan bunyi antar fonem /i/ dan fonem /u/ yang merupakan pasangan fonem yang berbeda dari pasangan kata tersebut. Untuk mengetahui nilai kemiripan antar kata "sapi" dan "sapu", kita perlu mengetahui nilai kemiripan antar fonem /i/ dan /u/. Secara umum, untuk mengetahui nilai kemiripan antar tiap kata, kita memerlukan nilai kemiripan antar tiap fonem. Matriks distance merupakan matriks yang berisi nilai-nilai jarak kemiripan antar tiap fonem. Dengan adanya matriks distance, diharapkan kita bisa mengetahui nilai kemiripan untuk tiap pasangan fonem dan pada akhirnya kita dapat membandingkan kemiripan antara pasangan-pasangan fonem tersebut.

Pada model *phonetic similarity* Stephen Rooney [STR05] dan riset STANDUP [RGH08] (lihat subbab 2.3 dan 2.4), juga telah dihasilkan matriks distance antar fonem. Namun matriks tersebut ditujukan bagi orang yang tidak berbahasa Indonesia dan model memiliki fonem yang berbeda dengan fonem dalam kamus fonetik Amalia Zahra sehingga penggunaan matriks tersebut pada konteks Bahasa Indonesia diragukan kesesuaiannya.

Pada konteks Levenshtein distance (lihat subbab 2.2), matriks distance juga dapat digunakan sebagai acuan nilai untuk operasi substitusi antar fonem, sehingga matriks distance dapat juga disebut sebagai matriks substitusi. Pembahasan lebih jauh mengenai penggunaan matriks substitusi dalam konteks Levenshtein distance akan diulas pada subbab 3.4.

3.1.1. Nilai kemiripan dan nilai jarak kemiripan

Istilah nilai kemiripan dan nilai jarak kemiripan akan banyak ditemui pada pembahasan mengenai perancangan ini. Keduanya merupakan istilah yang digunakan untuk mengukur kemiripan bunyi antar fonem atau kata. Nilai kemiripan fonem merupakan nilai yang menunjukkan seberapa mirip bunyi suatu pasangan fonem. Semakin besar nilai kemiripan, seharusnya semakin mirip bunyi pasangan fonem tersebut. Nilai jarak kemiripan fonem merupakan nilai yang menunjukkan seberapa jauh kemiripan suatu pasangan fonem. Semakin besar nilai jarak kemiripan, seharusnya semakin tidak mirip bunyi pasangan fonem tersebut. Nilai yang disimpan pada matriks distance/substitusi ialah nilai jarak kemiripan fonem. Notasi distance(f_1,f_2) menyatakan nilai jarak kemiripan antara fonem f_1 dan fonem f_2.

3.1.2. Rentang nilai jarak kemiripan

Matriks substitusi menyimpan nilai-nilai jarak kemiripan fonem. Diputuskan bahwa rentang nilai jarak kemiripan fonem ialah antara 0 sampai 1. Pasangan fonem yang memiliki nilai 0 merupakan pasangan yang memiliki bunyi yang sama persis. Pasangan fonem yang memiliki nilai 1 merupakan pasangan yang memiliki bunyi yang sama sekali tidak mirip. Sehubungan dengan nilai kemiripan, terdapat hubungan antara nilai kemiripan dengan nilai jarak kemiripan, yaitu:

Nilai kemiripan = 1 – Nilai jarak kemiripan

3.1.3. Nilai *default* pada matriks substitusi

Terdapat beberapa hubungan pada matriks substitusi sehingga tidak seluruh nilai jarak kemiripan pada matriks perlu didapatkan melalui kuesioner. Berikut hubungan-hubungan tersebut:

1. Nilai jarak kemiripan antar satu fonem dengan fonem itu sendiri ialah 0.

2. distance(f_1,f_2) = distance(f_2, f_1)

3. Nilai jarak kemiripan antara fonem vokal dan fonem konsonan ialah 1 (akan dibahas kemudian pada subbab 3.2.2.1).

3.2. Perancangan kuesioner

Matriks substitusi merupakan komponen penting dari model phonetic similarity. Mendapatkan nilai-nilai pada matriks substitusi yang sesuai dengan persepsi penutur bahasa Indonesia merupakan pekerjaan awal dari penelitian ini. Metode yang akan digunakan untuk memperoleh nilai-nilai pada matriks substitusi ialah metode empiris menggunakan kuesioner.

3.2.1. Metodologi kuesioner

Tujuan dari kuesioner ialah untuk memperoleh nilai-nilai pada matriks substitusi. Untuk mencapai tujuan tersebut, terdapat dua metode kuesioner yang dapat dipertimbangkan, yaitu metode nilai mutlak dan metode ranking.

3.2.1.1. Metode nilai mutlak

Metode pertama ialah dimana subjek diminta untuk memberikan nilai kemiripan bunyi antara pasangan fonem yang diberikan. Contoh pertanyaan:

Dari skala 0-10, tentukan nilai jarak kemiripan bunyi antara fonem b dan p.

Subjek akan menjawab pertanyaan tersebut dengan dibekali oleh kamus fonetik Amalia Zahra [ZAR08]. Keuntungan dari metode ini ialah bahwa hasil dari kuesioner mudah diekstrak menjadi matriks substitusi. Namun dari observasi terhadap penelitian serupa yang dilakukan oleh Stephen Rooney [STR05], metode ini dinilai kurang menguntungkan. Pada matriks substitusi yang dihasilkan banyak mengandung nilai 10, misalnya nilai 10 ditemui pada pasangan (B,Z),(CH,F),(CH,G),(CH,HH),(CH,K),(CH,L),(CH,M),(CH,N),(CH,NG),(CH,P) , dan (CH,R). Hal ini mengindikasikan bahwa matriks kurang memiliki kemampuan untuk membedakan kemiripan bunyi untuk banyak pasangan fonem. Hasil kuesioner ini disebabkan oleh subjek yang memberikan penilaian tanpa memikirkan nilai kemiripan untuk pasangan fonem lainnya sebagai pembanding. Penekanan pada metode kuesioner ini terletak pada nilai kemiripan pasangan

fonem itu sendiri, bukan perbandingannya dengan pasangan fonem lainnya. Hal ini tidak sesuai dengan tujuan pembentukan matriks substitusi ini yaitu memperoleh nilai kemiripan tiap pasangan fonem sebagai alat bantu untuk membandingkan kemiripan pasangan-pasangan fonem tersebut.

3.2.1.2. Metode ranking

Metode kedua ialah dimana subjek diminta untuk mengurutkan sejumlah fonem berdasarkan kemiripannya dengan fonem tertentu. Contoh pertanyaan:

Urutkan fonem-fonem berikut berdasarkan kemiripan dengan fonem a i aw ax o ay u e ey oy

Seperti pada metode pertama, subjek akan menjawab pertanyaan tersebut dengan dibekali oleh kamus fonetik Amalia Zahra. Keuntungan dari metode ini ialah dalam aktivitasnya menjawab pertanyaan, subjek akan melakukan perbandingan nilai kemiripan antara tiap pasangan fonem yang diberikan, yaitu (a,i),(a,aw),(a,ax),(a,o),(a,ay),(a,u),(a,e),(a,ey), dan (a,oy). Penekanan metode kuesioner ini terletak pada hubungan perbandingan nilai kemiripan pasangan-pasangan fonem yang diberikan, nilai kemiripan itu sendiri akan diekstrak dari hubungan perbandingan tersebut. Hal ini dinilai sesuai dengan tujuan pembuatan matriks substitusi yang telah disebutkan. Selain itu metode ini diperkirakan dapat lebih memaksa subjek untuk berpikir mengenai kemiripan antar bunyi. Hal ini disebabkan oleh adanya penyuguhan konteks perbandingan bunyi pada metode ini, bukan sekedar perintah untuk memberikan nilai tanpa skala perbandingan yang jelas.

Namun kekurangan dari metode ini juga perlu diperhatikan. Data yang diperoleh dari kuesioner ialah berupa pengurutan fonem, oleh karena itu hasil kuesioner tidak secara langsung membentuk sebuah nilai mutlak untuk distance(f_1,f_2) seperti pada metode nilai mutlak. Yang diperoleh ialah penilaian relatif antara distance(f_1,f_2) dan distance(f_1,f_3). Diperlukan perencanaan untuk mengekstrak data kuesioner menjadi matriks substitusi. Selain itu, pertanyaan kuesioner lebih sulit dan membebani subjek dibandingkan pertanyaan kuesioner pada metode pertama. Dengan karakteristik pertanyaan kuesioner yang seperti itu, dikhawatirkan munculnya kelelahan pada subjek sehingga mempengaruhi pengambilan

keputusan dalam menjawab pertanyaan. Misalnya subjek mengerjakan pertanyaan secara terburu-buru karena enggan mencurahkan pikiran lagi dalam menjawab pertanyaan. Perlu dipikirkan cara untuk mengatasi hal-hal tersebut.

Dengan menimbang kelebihan dan kekurangan kedua metode, diputuskan bahwa kuesioner akan dilakukan menggunakan metode ranking.

3.2.2. Penyusunan kuesioner

Secara umum pertanyaan kuesioner seperti dalam metode rangking akan memiliki format sebagai berikut:

Urutkan fonem-fonem berikut berdasarkan kemiripan dengan fonem f_x f_1 f_2 ... f_n

Dimana f_1 f_2 ... f_n merupakan set n fonem yang akan diurutkan berdasarkan kemiripannya dengan fonem f_x. Fonem f_x disebut sebagai fonem acuan. Subjek juga diberi instruksi mengenai format jawaban, yaitu fonem yang ditulis terlebih dahulu merupakan fonem yang paling mirip secara bunyi dengan fonem f_x. Pembahasan lebih jauh mengenai format jawaban akan dibahas pada subbab penyajian kuesioner.

Terdapat beberapa isu yang harus dipikirkan sebelum menyajikan pertanyaan dengan format seperti ini kepada subjek. Isu-isu tersebut yaitu hubungan antara fonem konsonan dan fonem vokal, jumlah fonem yang tepat dalam set fonem yang akan diurutkan, distribusi fonem dalam set-set fonem tersebut, dan jumlah pertanyaan untuk tiap subjek.

3.2.2.1. Fonem konsonan dan fonem vokal

Pada kamus fonetik Amalia Zahra terdapat 33 fonem yang terdiri dari 23 fonem konsonan dan 10 fonem vokal. Terdapat dua pilihan mengenai matriks substitusi yang akan dihasilkan oleh kuesioner menyangkut isu fonem konsonan dan fonem vokal. Pilihan pertama ialah bahwa kuesioner akan menghasilkan satu buah matriks 33 x 33 berisi tiap nilai jarak kemiripan antar fonem, meliputi nilai jarak kemiripan antar fonem konsonan, nilai jarak kemiripan antar fonem vokal, dan nilai jarak kemiripan antara fonem vokal dan fonem konsonan.

Pilihan kedua ialah bahwa kuesioner akan menghasilkan dua buah matriks, yaitu matriks 23 x 23 yang berisi nilai jarak kemiripan antar fonem konsonan dan matriks 10 x 10 yang berisi nilai jarak kemiripan antar fonem vokal. Nilai jarak kemiripan antara fonem vokal dan fonem konsonan merupakan konstanta bernilai 1 (nilai maksimum).

Diputuskan bahwa model ini akan menggunakan dua buah matriks sesuai dengan pilihan kedua. Dengan dua buah matriks, data yang dibutuhkan lebih sedikit sehingga dapat mengurangi kebutuhan subjek untuk mengisi kuesioner. Selain itu, konstanta bernilai 1 untuk nilai jarak kemiripan antara fonem vokal dan fonem konsonan ini sesuai dengan model *phonetic similarity* untuk riset STANDUP [RGH08] (lihat bab 2).

3.2.2.2. Hubungan distance relatif

Misal untuk pertanyaan:

Urutkan fonem-fonem berikut berdasarkan kemiripan dengan fonem f_x f_1 f_2 f_3 f_4 f_5

Subjek menjawab:

f_4 f_3 f_2 f_1 f_5

Sesuai dengan format jawaban yang diinstruksikan, jawaban diatas diinterpretasikan sebagai subjek menganggap bahwa fonem f_4 paling mirip dengan fonem f_x secara bunyi dan fonem f_5 paling tidak mirip dengan fonem f_x secara bunyi. Dari jawaban diatas diperoleh hubungan distance relatif sebagai berikut:

$distance(f_x, f_4) < distance(f_x, f_3) < distance(f_x, f_{z2}) < distance(f_x, f_1) < distance(f_x, f_5)$

Fonem vokal terdiri dari 10 fonem. Misal v_1, v_2, ... v_{10} merupakan fonem vokal, maka untuk mendapatkan hubungan perbandingan yang memadai pada matriks substitusi untuk fonem vokal, pertanyaan kuesioner harus meliputi tiap hubungan relatif antara $distance(v_r,v_p)$ dan $distance(v_r,v_q)$ dimana $1 \leq p,q,r \leq 10$, untuk tiap triplet p, q, dan r dimana $p \neq q \neq r$.

Demikian juga untuk fonem konsonan. Fonem konsonan terdiri dari 23 fonem. Misal k_1, k_2, ... k_{23} merupakan fonem konsonan, maka untuk mendapatkan hubungan perbandingan yang memasai pada matriks substitusi untuk fonem konsonan, pertanyaan kuesioner harus meliputi tiap hubungan relatif antara distance(k_r,k_p) dan distance(k_r,k_q) dimana $1 \leq p,q,r \leq 23$, untuk tiap triplet p, q, dan r dimana $p \neq q \neq r$.

3.2.2.3. Jumlah fonem dalam sebuah pertanyaan

Jumlah fonem yang perlu diurutkan terhadap fonem acuan dalam sebuah pertanyaan perlu dipertimbangkan. Secara umum, semakin banyak fonem yang perlu diurutkan dalam sebuah pertanyaan akan semakin menyulitkan subjek untuk melakukan pengurutan. Namun, semakin sedikit fonem dalam sebuah pertanyaan berarti semakin sedikit pula data yang dapat diperoleh dalam satu pertanyaan, sehingga diperlukan lebih banyak pertanyaan untuk mendapatkan data yang dibutuhkan dalam pembentukan matriks substitusi. Selain itu, jumlah fonem dalam satu set fonem erat kaitannya dengan isu distribusi fonem dalam set-set fonem yang akan dibahas pada subbab berikut.

3.2.2.4. Distribusi fonem dalam set-set fonem

Seperti dengan yang telah ditulis pada subbab 3.2.2, secara umum pertanyaan kuesioner seperti dalam metode rangking akan memiliki format sebagai berikut:

Urutkan fonem-fonem berikut berdasarkan kemiripan dengan fonem f_x f_1 f_2 ... f_n

Dimana f_1 f_2 ... f_n merupakan set n fonem yang akan diurutkan berdasarkan kemiripannya dengan fonem fx. Sesuai dengan keputusan bahwa kita tidak akan mempertanyakan kemiripan antara fonem vokal dan fonem konsonan, jika fx merupakan fonem konsonan, maka f1 sampai dengan fn juga merupakan fonem konsonan, demikian juga sebaliknya.

Misal f_x merupakan fonem konsonan. Yang dimaksud dengan distribusi fonem disini ialah dari total 23 fonem konsonan, fonem-fonem konsonan mana saja yang akan disertakan dalam tiap pertanyaan dan bagaimana pembagiannya hingga jawaban kuesioner bisa menghasilkan matriks substitusi. Isu-isu yang harus

diperhatikan disini ialah jumlah fonem dalam set fonem dan kemudahan untuk melakukan ekstraksi.Pembahasan dibagi menjadi untuk menjadi distribusi untuk pertanyaan fonem vokal dan distribusi untuk pertanyaan fonem konsonan.

1. Distribusi untuk pertanyaan fonem vokal

Terdapat 10 fonem vokal dalam kamus fonetik Amalia Zahra. Distribusi pertanyaan bersifat trivial yaitu setiap pertanyaan fonem vokal akan terdiri dari satu fonem vokal acuan dan sembilan fonem yang akan diurutkan. Misal v_1, v_2, ... v_{10} merupakan fonem vokal, maka pertanyaan-pertanyaan kuesionernya adalah:

Urutkan fonem-fonem berikut berdasarkan kemiripan dengan fonem v_1

v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10}

Urutkan fonem-fonem berikut berdasarkan kemiripan dengan fonem v_2

v_1 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10}

…

Urutkan fonem-fonem berikut berdasarkan kemiripan dengan fonem v_{10}

v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9

Terdapat total sepuluh pertanyaan untuk pertanyaan fonem vokal. Dengan distribusi pertanyaan yang menyertakan tiap fonem dalam pertanyaannya, setiap hubungan distance relatif bisa didapatkan. Diasumsikan bahwa jumlah 9 fonem untuk diurutkan bukanlah jumlah yang terlalu banyak untuk dikerjakan oleh subjek.

Jika sepuluh pertanyaan tersebut dijawab oleh para subjek, maka tiap sel pada matriks substitusi akan terisi oleh data dari 2 pertanyaan. Hal ini karena terdapat hubungan distance(f_1,f_2) = distance(f_2, f_1).

Misalnya pada pertanyaan:

Urutkan fonem-fonem berikut berdasarkan kemiripan dengan fonem v_1

v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10}

Akan didapat data untuk nilai pasangan v_1 dan v_2. Dan pada pertanyaan

Urutkan fonem-fonem berikut berdasarkan kemiripan dengan fonem v_2 v_1 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10}

Akan didapat data untuk nilai pasangan v_2 dan v_1.

Karena distance(v_1,v_2) = distance(v_2, v_1), maka nilai distance (v_1,v_2) pada matriks substitusi akan diperoleh melalui data 2 pertanyaan dari 10 pertanyaan.

2. Distribusi untuk pertanyaan fonem konsonan

Fonem konsonan berjumlah 23, jumlah fonem terlalu banyak untuk mendapat perlakuan seperti fonem vokal dimana setiap fonem disertakan pada satu pertanyaan. Perlu dilakukan pembagian grup konsonan sehingga kita mendapati jumlah fonem yang lebih manusiawi bagi subjek kuesioner untuk dilakukan pengurutan. Pertimbangkan pembagian grup berikut:

Untuk setiap fonem konsonan k, misal k_1 merupakan fonem konsonan pertama setelah k dan k_{22} merupakan fonem konsonan ke-22 setelah k (atau fonem sebelum k), maka pembagian grup terhadap fonem konsonan k tersebut bisa dilakukan sebagai berikut.

Grup A terdiri dari fonem $k_1 – k_6$ Grup B terdiri dari fonem k_7 - k_{11} Grup C terdiri dari fonem k_{12} - k_{16} Grup D terdiri dari fonem $k_{17} – k_{22}$

Perlu diperhatikan bahwa pembagian grup tersebut merupakan pembagian terhadap fonem tertentu. Setiap fonem memiliki grup A, B, C, D masing-masing yang terdiri dari fonem yang berbeda.

Untuk setiap fonem konsonan k akan dirancang pertanyaan untuk mendapatkan distance relatif antara fonem k dengan setiap fonem bukan k dengan formula sebagai berikut:

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem kx: Grup A + Grup B

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem kx: Grup A + Grup C

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem kx: Grup A + Grup D

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem kx: Grup B + Grup C

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem kx: Grup B + Grup D

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem kx: Grup C + Grup D

Sebagai contoh, fonem konsonan pada kamus fonetik Amalia Zahra terdiri dari fonem p, b, t, d, c, j, k, g, f, s, z, sy, kh, h, m, n, ng, ny, l, r, w, y, dan kk. Maka pembagian grup terhadap fonem s ialah sebagai berikut:

- Grup A: z, sy, kh, h, m, dan n (z merupakan k_1 terhadap fonem s, sy merupakan k_2, kh merupakan k_3 dan seterusnya)
- Grup B: ng, ny, l, r, dan w
- Grup C: y, kk, p, b, dan t
- Grup D: d, c, j, k, g, dan f

Pertanyaan-pertanyaan terhadap fonem s ialah sebagai berikut:

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s z sy kh h m n ng ny l r w (Grup A + Grup B)

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s: z sy kh h m n y kk p b t (Grup A + Grup C)

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s: z sy kh h m n d c j k g f (Grup A + Grup D)

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s: ng ny l r w y kk p b t (Grup B + Grup C)

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s: ng ny l r w d c j k g f (Grup B + Grup D)

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s: y kk p b t d c j k g f (Grup C + Grup D)

Pembagian grup dan distribusi pertanyaan untuk subjek

Seperti yang telah disebutkan bahwa untuk mendapatkan hubungan perbandingan pada matriks substitusi bagi fonem konsonan, pertanyaan kuesioner harus meliputi tiap hubungan distance relatif. Pada distribusi pertanyaan seperti diatas, dapat dilihat bahwa setiap grup pernah berpasangan dengan tiap grup lainnya dalam set fonem yang akan diurutkan. Sehingga setiap fonem akan mendapat giliran untuk berada dalam set yang sama dengan tiap fonem lainnya.

Dengan pembagian 22 fonem menjadi 4 grup dapat dilihat pada pembagian grup bahwa terdapat ketidakseragaman jumlah fonem untuk tiap grup. Hal ini mengakibatkan jumlah fonem dalam set fonem untuk tiap pertanyaan ialah 10 hingga 12 fonem. Diasumsikan bahwa jumlah 12 fonem untuk diurutkan bukanlah jumlah yang terlalu banyak untuk dikerjakan oleh subjek. Pada subjek, ketidakseragaman jumlah fonem dalam tiap pertanyaan dinilai tidak menimbulkan masalah selama asumsi diatas berlaku. Sedangkan pada pembentukan matriks substitusi, hal yang penting dalam melakukan pembagian grup fonem ialah memastikan bahwa setiap hubungan distance relatif antar tiap fonem telah dicakupkan.

Untuk setiap fonem acuan terdapat 6 pertanyaan dengan set fonem yang berbeda terhadapnya. Sehingga untuk seluruh fonem konsonan terdapat total 23 x 6 = 138

pertanyaan. Dalam hubungannya dengan pembentukan matriks substitusi, jika 138 pertanyaan fonem konsonan tersebut terjawab oleh para subjek maka tiap sel pada matriks substitusi akan terisi oleh data dari 6 pertanyaan. Hal ini karena terhadap fonem acuan k_x, tiap fonem akan disertakan dalam 3 pertanyaan yang berbeda (dalam set fonem yang berbeda) dan terdapat hubungan distance(f_1,f_2) = distance(f_2,f_1).

Pembagian grup yang telah dibahas ialah pembagian fonem konsonan menjadi 4 grup yang setiap grup terdiri dari 5 atau 6 fonem. Terdapat alternatif pembagian grup dan distribusi pertanyaan lain yang dipertimbangkan. Misalnya pembagian menjadi 5 grup. Pada pembagian menjadi 5 grup, tiap grup beranggotakan 4 sampai 5 fonem. sehingga jumlah fonem pada set fonem untuk tiap pertanyaan ialah 8 hingga 10 pertanyaan. Jumlah fonem yang lebih sedikit membuat subjek lebih tidak terbebani dibanding pembagian sebelumnya yang memiliki jumlah fonem 10 - 12 untuk tiap pertanyaan.

Namun dari sudut pandang jumlah pertanyaan yang dihasilkan, pembagian menjadi 5 grup menghasilkan lebih banyak pertanyaan. Dengan pertanyaan yang lebih banyak, subjek yang dibutuhkan juga lebih banyak, sehingga dengan adanya keterbatasan waktu diputuskan bahwa kuesioner dilakukan dengan pembagian fonem menjadi 4 grup. Jumlah 12 fonem dinilai bukanlah jumlah yang terlalu banyak untuk dikerjakan oleh subjek.

3.2.2.5. Randomisasi set fonem

Misal pembagian grup untuk fonem s ialah sebagai berikut:

Grup A: z, sy, kh, h, m, dan n
Grup B: ng, ny, l, r, dan w
Grup C: y, kk, p, b, dan t
Grup D: d, c, j, k, g, dan f

Beberapa pertanyaan terhadap fonem s ialah sebagai berikut:

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s
z sy kh h m n ng ny l r w (Grup A + Grup B)

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s: z sy kh h m n y kk p b t (Grup A + Grup C)

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s: z sy kh h m n d c j k g f (Grup A + Grup D)

Tiga pertanyaan diatas melibatkan grup A dari fonem s. Set fonem pada tiga pertanyaan diatas disajikan dalam urutan yang tidak jauh berbeda. Dimulai dengan fonem z, sy, kh, h, m, dan n (fonem-fonem grup A dari fonem s). Dikhawatirkan bahwa penyajian set fonem dalam urutan yang sama seperti ini dapat menimbulkan bias bagi subjek. Untuk menghindari hal tersebut set fonem yang akan disajikan dalam pertanyaan perlu dilakukan randomisasi terlebih dahulu. Setelah dilakukan randomisasi, tiga pertanyaan diatas akan menjadi:

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s z m sy kh r w n ng h l ny (set fonem Grup A + Grup B setelah dilakukan randomisasi)

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s: n y kk p z kh h m b sy t (set fonem Grup A + Grup C setelah dilakukan randomisasi)

dan

Urutkan fonem-fonem berikut berdasarkan kemiripan bunyi dengan fonem s: d g z j k kh h m n f sy c (Grup A + Grup D)

3.2.2.6. Jumlah pertanyaan dan pembagiannya untuk subjek

Seperti yang telah dibahas pada subbab 3.2.2.5, jumlah pertanyaan untuk pertanyaan fonem vokal ialah 10, sedangkan jumlah pertanyaan untuk pertanyaan fonem konsonan ialah 138. Total pertanyaan ialah 148. Namun dengan 148 pertanyaan tersebut akan diperoleh matriks substitusi fonem vokal yang nilai tiap selnya terisi oleh data dari 2 pertanyaan dan matriks substitusi fonem konsonan yang nilai tiap selnya terisi oleh data dari 6 pertanyaan. Agar diperoleh kesamaan kualitas data matriks yang dihasilkan, matriks substitusi fonem vokal juga akan menjadikan nilai tiap selnya terisi oleh data dari 6 pertanyaan. Sehingga untuk

pertanyaan fonem vokal terdapat 30 pertanyaan dan total pertanyaaan sekarang adalah 138 + 30 = 168 pertanyaan.

Jumlah pertanyaan yang ideal untuk diberikan ke tiap subjek memerlukan pertimbangan. Mula-mula diputuskan bahwa jumlah pertanyaan ialah 7, yang terdiri dari 6 pertanyaan untiuk fonetik konsonan dan 1 pertanyaan fonetik vokal atau 5 pertanyaan fonetik konsonan dan 2 pertanyaan fonetik vokal. Keputusan terakhir mengenai jumlah pertanyaan yang ideal akan ditentukan setelah dilakukan pilot kuesioner.

Selain itu yang juga perlu diperhatikan ialah bahwa setiap pertanyaan yang diterima subjek tidak boleh mengandung fonem acuan yang sama.

3.2.2.7. Penyajian kuesioner

Setelah perencanaan teknis menyangkut kuesioner selesai dilakukan, yang harus dipikirkan berikutnya ialah penyajian kuesioner. Penyajian kuesioner sehingga subjek dapat mengerjakan kuesioner sesuai dengan tujuan pengerjaan kuesioner merupakan salah satu komponen keberhasilan penelitian ini.

Diputuskan bahwa penyajian kuesioner akan terdiri dari hal-hal berikut:

- Penjelasan singkat mengenai penelitian
- Instruksi pengerjaan
- Lembar pertanyaan/jawaban yang turut disertai oleh kamus fonetik Amalia Zahra
- Lembar saran dan masukan

Contoh lembar kuesioner untuk dibagikan kepada subjek dapat ditemui pada lampiran D.

3.2.2.8. Pilot kuesioner

Sebelum dilakukan penyebaran kuesioner, perlu diadakan eksperimen terhadap rancangan kuesioner. Pilot kuesioner dilakukan terhadap tiga subjek yang terdiri dari mahasiswa Fasilkom angkatan 2005, mahasiswa Fasilkom angkatan 2006, dan mahasiswa Fasilkom angkatan 2007. Tiap subjek diberikan kuesioner sesuai dengan rancangan kuesioner dan pengerjaan kuesioner tidak dibatasi oleh waktu.

Kuesioner yang dibagikan sesuai dengan perancangan kuesioner yang telah dilakukan. Terdapat penjelasan singkat mengenai penelitian, instruksi pengerjaan, lembar pertanyaan dilengkapi dengan kamus fonetik Amalia Zahra, dan lembar saran dan masukan. Pada setiap kuesioner terdapat tujuh pertanyaan yang terdiri dari enam pertanyaan fonem konsonan dan satu pertanyaan fonem vokal. Untuk pertanyaan fonem konsonan, terdapat sepuluh hingga dua belas fonem dalam set fonem untuk tiap pertanyaan. Sedangkan untuk pertanyaan fonem vokal, terdapat sembilan fonem dalam set fonem untuk tiap pertanyaan.

Penjelasan singkat dan instruksi pengerjaan pada kuesioner ditemui tidak menimbulkan masalah pada subjek. Penjelasan penelitian cukup deskriptif dan instruksi pengerjaan berperan baik dalam membantu pengerjaan kuesioner.

Beberapa saat setelah subjek mengerjakan pertanyaan kuesioner, keluhan mengenai tingkat kesulitan kuesioner diutarakan. Hal ini terjadi pada tiga subjek pilot kuesioner. Dari observasi, terlihat bahwa kuesioner sangat membebankan subjek dan ada kekhawatiran bahwa hal ini dapat menjauhkan subjek dari pengisian kuesioner secara optimal. Waktu pengerjaan untuk tiap subjek bervariasi namun ketiga subjek menghabiskan waktu lebih dari 30 menit.

3.2.2.9. Perubahan kuesioner

Dari hasil yang diperoleh pada kuesioner, perubahan pada kuesioner perlu dilakukan. Akan dilakukan salah satu dari alternatif perubahan berikut:

1. Pengurangan jumlah pertanyaan
2. Pengurangan jumlah fonem pada set fonem dalam tiap pertanyaan.

Pendapat dari para subjek pilot kuesioner mengenai hal ini ialah dari sudut pandang mereka pengurangan jumlah pertanyaan akan lebih memudahkan. Diakui bahwa salah satu hambatan terbesar mereka dalam mengerjakan kuesioner ialah bahwa tiap memulai pengerjaan pertanyaan baru, muncul perasaan enggan yang membebani mereka. Selain karena beban pengerjaan untuk tiap soal sulit, tiap pertanyaan baru bagi subjek berarti proses pengurutan terhadap fonem acuan yang baru.

Dengan pertimbangan diatas maka diputuskan bahwa perubahan yang dilakukan ialah pengurangan jumlah pertanyaan. Jumlah pertanyaan untuk tiap subjek dikurangi dari 7 menjadi 5.

3.2.2.10. Jumlah subjek yang dibutuhkan dan pembagian pertanyaan pada lembar pertanyaan

Dengan jumlah pertanyaan untuk tiap subjek yang menjadi 5 dibutuhkan 168/5=34 lembar pertanyaan yang berbeda untuk dikerjakan subjek. Tiap pertanyaan akan terdiri dari 1 pertanyaan fonetik vokal dan 4 pertanyaan fonetik konsonan, atau 5 pertanyaan fonetik konsonan. Tiga puluh set pertanyaan pertama termasuk golongan 1 pertanyaan fonetik vokal dan 4 pertanyaan fonetik konsonan. Set pertanyaan ke-31 dan ke-32 merupakan golongan 5 pertanyaan fonetik konsonan. Dua set pertanyaan terakhir terdiri dari 1 pertanyaan fonetik vokal dan 4 pertanyaan fonetik konsonan.

3.3. Ekstraksi nilai kuesioner

Setelah penyusunan kuesioner selesai dilakukan, tahap berikutnya ialah memikirkan cara untuk melakukan ekstraksi sehingga jawaban kuesioner yang diperoleh dapat dibuat menjadi matriks substitusi.

Perhatikan kompilasi jawaban berikut.

$f_{(1,0)} \rightarrow f_{(1,1)}\ f_{(1,2)}\ f_{(1,3)} \ldots f_{(1,n)}$

$f_{(2,0)} \rightarrow f_{(2,1)}\ f_{(2,2)}\ f_{(2,3)} \ldots f_{(2,n)}$

$f_{(3,0)} \rightarrow f_{(3,1)}\ f_{(3,2)}\ f_{(3,3)} \ldots f_{(3,n)}$

…

$f_{(m,0)} \rightarrow f_{(m,1)}\ f_{(m,2)}\ f_{(m,3)} \ldots f_{(m,n)}$

Contoh kompilasi jawaban diatas menyatakan kumpulan jawaban dari para subjek untuk setiap pertanyaan. Setelah seluruh jawaban kuesioner dikumpulkan akan didapati 2 kompilasi jawaban seperti ini. Kompilasi jawaban untuk fonem vokal dan kompilasi jawaban untuk fonem konsonan. Nilai m ditentukan oleh jumlah fonem dalam matriks dan nilai n ditentukan oleh jumlah fonem dalam set fonem. Untuk pertanyaan fonem vokal nilai m ialah 3 x 10 = 30 dan nilai n ialah 9

sedangkan untuk pertanyaan fonem konsonan nilai m ialah 6 x 23 = 138 dan nilai n ialah 10 hingga 12.

Untuk setiap $i \geq 1$ dan $i \leq m$, $f_{(i,0)}$ merupakan fonem acuan bagi $f_{(i,1)}$ hingga $f_{(i,n)}$. Dan untuk $j \geq 1$ dan $j \leq n$, $f_{(i,j)}$ merupakan anggota set fonem yang memiliki fonem acuan $f_{(i,0)}$.

Dengan data kompilasi jawaban diatas akan dihasilkan nilai distance($f_{(a,b)}$, $f_{(c,d)}$) untuk setiap $1 \leq a,c \leq m$ dan $0 \leq b,d \leq n$ dalam rentang nilai 0 sampai 1. Nilai distance tersebut diperoleh dengan melakukan suatu mekanisme pemberian nilai untuk setiap pasangan fonem yang ditemui pada kompilasi jawaban dan penjumlahan nilai tersebut untuk pasangan fonem yang sama. Nilai yang diberikan akan dinotasikan sebagai score(a,b), dimana a dan b merupakan fonem pada kompilasi jawaban dan score(a,b) = score(b,a). Sesuai dengan instruksi pengerjaan kuesioner, kompilasi jawaban diatas diinterpretasikan sebagai untuk tiap nilai $i \geq 1$ dan $i \leq m$, f_{i1} merupakan fonem yang dinilai paling mirip terhadap fonem $f_{(i,0)}$ dibanding fonem $f_{(i,2)}$, $f_{(i,3)}$, hingga $f_{(i,n)}$. Dan $f_{(i,n)}$ merupakan fonem yang dinilai paling tidak mirip terhadap fonem $f_{(i,0)}$ dibanding fonem $f_{(i,1)}$ hingga $f_{(i,n-1)}$. Maka dilakukan pemberian nilai score($f_{(i,0)}$, $f_{(i,1)}$) = 1 dan untuk tiap peringkat distance berikutnya akan diberi nilai increment 2. Yaitu score($f_{(i,0)}$, $f_{(i,2)}$) = 3, score($f_{(i,0)}$, $f_{(i,3)}$) = 5, hingga score($f_{(i,0)}$, $f_{(i,n)}$) = 2n-1. Setelah pemberian score selesai dilakukan untuk setiap baris, akan dilakukan penghitungan sum($f_{(a,b)}$, $f_{(c,d)}$) untuk setiap $1 \leq a,c \leq m$ dan $0 \leq b,d \leq n$. Nilai sum(a,b) didefinisikan sebagai penjumlahan seluruh score(a,b) dimana:

- Fonem a merupakan fonem acuan dan fonem b merupakan fonem jawaban terhadap fonem acuan a.
- Fonem b merupakan fonem acuan dan fonem a merupakan fonem jawaban terhadap fonem acuan b.

Nilai sum(a,b) pada dasarnya merupakan nilai distance(a,b) sebelum dilakukan normalisasi. Terhadap setiap nilai sum yang diperoleh, akan dilakukan normalisasi. Normalisasi dilakukan agar nilai distance memiliki rentang nilai 0 sampai 1. Untuk fonem konsonan, nilai sum maksimum yang dapat diperoleh ialah 19 + 21 + 21 + 21 + 21 + 19 = 126, sehingga nilai sum yang diperoleh

dinormalisasikan dengan dibagi 126. Sedangkan untuk fonem vokal, nilai sum maksimum yang dapat diperoleh ialah 17 + 17 + 17 + 17 + 17 + 17 = 102, sehingga nilai sum yang diperoleh dinormalisasikan dengan dibagi 102.

3.4. Penghitungan word distance

Seperti yang telah diulas pada subbab 3.1, model phonetic similarity yang dibuat ditujukan sebagai metrik untuk mengukur kemiripan antara pasangan bunyi kata. Kemiripan antara pasangan bunyi kata ini diistilahkan sebagai word distance.

Sebelum dilakukan penghitungan, pasangan kata yang akan menjadi masukan dipetakan ke bentuk fonemnya. Proses pemetaan kata dilakukan dengan mengacu pada daftar leksikon Amalia Zahra. Daftar leksikon Amalia Zahra berisikan 8285 kata dan transkripsi fonetiknya.

Metode penghitungan word distance pada dasarnya merupakan varian dari Levenshtein Distance (lihat bab 2). Secara umum, Levenshtein Distance merupakan metrik untuk mengukur jarak antara dua sequence. Pada penghitungan word distance, yang berperan sebagai sequence adalah fonem. Seperti pada Levenshtein Distance, penghitungan word distance juga berdasarkan nilai operasi yang dibutuhkan untuk merubah satu kata (barisan fonem) ke kata yang lain. Untuk mendapatkan esensi dari kemiripan bunyi, diperlukan pemikiran mengenai pembobotan nilai yang tepat untuk tiap operasi. Nilai operasi substitusi fonem diperoleh dari ekstraksi kuesioner yang telah dibahas pada subbab 3.3. Nilai operasi insertion dan deletion ditentukan dengan landasan bahwa operasi substitusi dapat dipandang sebagai dua langkah operasi, yaitu deletion yang diikuti dengan insertion. Oleh karena nilai maksimal operasi substitusi ialah 1, maka nilai operasi deletion dan substitution ialah 1:2 = 0,5.

3.3.1. Post-normalized word distance

Pada penghitungan word distance, karena nilai diperoleh dari nilai operasi yang dibutuhkan, semakin panjang suatu kata, semakin besar kecenderungan kata tersebut membutuhkan lebih banyak nilai operasi untuk berubah ke kata yang lain. Dengan kata lain, pasangan kata yang memiliki banyak fonetik cenderung untuk memiliki nilai word distance yang lebih besar. Idealnya, pasangan kata yang panjang namun memiliki bunyi yang mirip tetap mendapatkan nilai word distance

yang kecil. Oleh karena itu, perlu dilakukan proses normalisasi setelah penghitungan word distance dilakukan.

Misal w_1 dan w_2 adalah pasangan kata yang akan menjadi input untuk penghitungan word distance, notasi $wd(w_1,w_2)$ menyatakan nilai word distance antara w1 dan w2 dan notasi $pwd(w_1,w_2)$ menyatakan nilai post normalized word distance.

Jika w1length merupakan jumlah fonem dari w1 dan w2length merupakan jumlah fonem dari w2, maka nilai word distance setelah dilakukan post-normalisasi adalah:

$$pwd(w_1,w_2) = wd(w_1,w_2) / (w1length+w2length)$$

3.4. Rangkuman

Pada bab ini telah dibahas model *phonetic similarity* yang akan dikembangkan serta hal-hal yang harus dilakukan untuk membentuknya hingga dapat menghasilkan cara untuk mendapatkan kemiripan antar bunyi fonem atau kata. Kuesioner berperan penting dalam pembentukan model. Perencanaan kuesioner yang baik merupakan modal untuk memperoleh data yang baik dalam membentuk model. Pada bab selanjutnya akan dibahas mengenai implementasi yang dilakukan pada penelitian ini.

BAB 4 IMPLEMENTASI

Dalam bab ini akan dijelaskan mengenai implementasi word distance yang dilakukan berdasarkan perancangan yang sudah dilakukan dalam Bab 3 yang meliputi implementasi Levenshtein *distance*, modifikasi umum, modifikasi nilai operasi Levenshtein *distance*, modifikasi *post-normalized word distance*, modifikasi masukan, dan aplikasi *word distance*. Pembahasan mengenai implementasi ekstraksi kuesioner juga akan dilakukan pada bab ini.

4.1. Implementasi word distance

Untuk memperoleh implementasi word distance sesuai dengan perancangan yang telah dibahas pada bab 3, dilakukan beberapa tahapan, yaitu mendapatkan implementasi Levenshtein distance dan melakukan beberapa modifikasi pada implementasi tersebut.

4.1.1. Implementasi Levenshtein Distance

Implementasi Levenshtein diperoleh dari Merriam Park [MEP]. Gambar 4.1 merupakan *source code* dari program.

Pada bab 2 Landasan teori disebutkan bahwa umumnya Algoritma Levenshtein *distance* menggunakan metode *dynamic programming* yang melibatkan matriks berukuran (n+1) x (m+1) dimana n dan m merupakan panjang kedua string masukan.

Pada dasarnya program ini juga menggunakan metode *dynamic programming* namun memiliki perbedaan struktur data penyimpanan nilai, yaitu dua buah array satu dimensi berukuran n+1 dimana n adalah panjang string pertama. Penggunaan dua buah array satu dimensi sebagai ganti penggunaan matriks (array dua dimensi), dapat mengatasi masalah kebutuhan *memory* untuk kasus masukan string yang panjang.

Pada program yang menggunakan matriks $M_{(m+1)\times(n+1)}$, untuk $1 \le i \le m$ dan $1 \le j \le n$ ketika iterasi program berada pada baris i untuk mengisi satu sel $M_{(i,j)}$ sebenarnya yang dibutuhkan ialah nilai pada sel $M_{(i,j-1)}$, $M_{(i-1,j)}$ dan $M_{(i-1,j-1)}$, sehingga pada matriks $M_{m\times n}$ hanya nilai-nilai pada baris yang sedang berada dalam iterasi dan baris sebelumnya yang perlu disimpan pada program. Oleh karena itu,

penggunaan dua buah array satu dimensi berukuran (n+1) cukup untuk menjalankan program.

Program menerima masukan dua buah *string* dan mengembalikan nilai *integer* yang merupakan nilai Levenshtein distance antara dua *string* tersebut. Terhadap dua *string*

```
public static double getLevenshteinDistance (String s, String t) {
  if (s == null || t == null) {
    throw new IllegalArgumentException("Strings must not be null");
  }

  int n = s.length(); // length of s
  int m = t.length(); // length of t

  if (n == 0) {
    return m;
  } else if (m == 0) {
    return n;
  }

  int p[] = new int[n+1]; //'previous' cost array, horizontally
  int d[] = new int[n+1]; // cost array, horizontally
  int _d[]; //placeholder to assist in swapping p and d

  // indexes into strings s and t
  int i; // iterates through s
  int j; // iterates through t

  char t_j; // jth character of t

  double cost; // cost

  for (i = 0; i<=n; i++) {
    p[i] = i;
  }

  for (j = 1; j<=m; j++) {
    t_j = t.charAt(j-1);
    d[0] = j;

    for (i=1; i<=n; i++) {
      cost = s.charAt(i-1)==t_j ? 0 : 1;
      // minimum of cell to the left+1, to the top+1, diagonally left and up +cost
      d[i] = Math.min(Math.min(d[i-1]+1, p[i]+1),  p[i-1]+cost);
    }

    // copy current distance counts to 'previous row' distance counts
    _d = p;
    p = d;
    d = _d;
  }

  // our last action in the above loop was to switch d and p, so p now
  // actually has the most recent cost counts
  return p[n];
}
```

Gambar 4. 1 Source code Levenshtein distance [MEP]

yang menjadi masukan, program melakukan iterasi terhadap setiap karakter dan evaluasi terhadap operasi-operasi yang dapat dilakukan. Analogi seperti pada

implementasi dengan menggunakan matriks (n+1) x (m+1), program memulai pengisian nilai matriks pada kolom 0 dari baris 0 sampai n. Kemudian program mengisi kolom 1, kolom 2, hingga kolom m yang artinya hingga tiap karakter selesai dilakukan proses iterasi. Setelah iterasi terhadap dua *string* masukan selesai dilakukan, program mengembalikan nilai dari array pada indeks terakhir.

4.1.2. Modifikasi umum

Setelah mendapatkan implementasi Levenshtein *distance* dan memahami alur dari program, langkah berikutnya ialah melakukan modifikasi program agar sesuai dengan kebutuhan penggunaan. Modifikasi secara umum melibatkan data masukan dan keluaran. Modifikasi tersebut yaitu:

1. Implementasi Levenshtein *distance* mengembalikan nilai *integer* sedangkan *word distance* diharapkan untuk mengembalikan nilai *double*. Perubahan tipe data *integer* menjadi *double* perlu dilakukan pada deklarasi fungsi, pada array p, d, dan _d, juga pada variable cost.

   Dilakukan perubahan pada baris:

```
public static int getLevenshteinDistance (String s, String t) {
  if (s == null || t == null) {
    throw new IllegalArgumentException("Strings must not be null");
  }
```

   Sehingga menjadi:

```
public static double getLevenshteinDistance (String s, String t) {
  if (s == null || t == null) {
    throw new IllegalArgumentException("Strings must not be null");
  }
```

   Dilakukan juga perubahan pada baris:

```
  int p[] = new int[n+1]; //'previous' cost array, horizontally
  int d[] = new int[n+1]; // cost array, horizontally
  int _d[]; //placeholder to assist in swapping p and d
```

   Sehingga menjadi:

```
  double p[] = new double[n+1]; //'previous' cost array, horizontally
  double d[] = new double[n+1]; // cost array, horizontally
  double _d[]; //placeholder to assist in swapping p and d
```

Dilakukan juga perubahan pada baris:

```
int cost; // cost
```

Sehingga menjadi:

```
double cost; // cost
```

2. Implementasi Levenshtein *distance* dilakukan terhadap dua buah kata, sedangkan *word distance* dilakukan terhadap dua buah kumpulan fonem. Masukan data masih berupa string, namun jika pada implementasi Levenshtein distance contoh masukan misalnya “landasan” dan “teori”, pada word distance contoh masukan misalnya “l a n d a s a n” dan “t e o r i”. Kata yang menjadi masukan untuk *word distance* harus sudah diubah kedalam bentuk fonetiknya terlebih dahulu.

Dilakukan perbuahan pada baris:

```
public static double getLevenshteinDistance (String s, String t) {
  if (s == null || t == null) {
    throw new IllegalArgumentException("Strings must not be null");
  }

  int n = s.length(); // length of s
  int m = t.length(); // length of t
```

Sehingga menjadi:

```
public static double getLevenshteinDistance (String s, String t) {
  if (s == null || t == null) {
    throw new IllegalArgumentException("Strings must not be null");
  }

  String[] fonetiklist1;
  String[] fonetiklist2;
  String delimeter = "\\s+";
  fonetiklist1 = s.split(delimeter);
  fonetiklist2 = t.split(delimeter);

  int n = fonetiklist1.length(); // length of s
  int m = fonetiklist2.length(); // length of t
```

Dilakukan juga perubahan pada baris:

```
char t_j; // jth character of t

double cost; // cost

for (i = 0; i<=n; i++) {
    p[i] = i;
}

for (j = 1; j<=m; j++) {
    t_j = t.charAt(j-1);
    d[0] = j;

    for (i=1; i<=n; i++) {
        cost = s.charAt(i-1)==t_j ? 0 : 1;
```

Sehingga menjadi:

```
String t_j; // jth character of t

double cost; // cost

for (i = 0; i<=n; i++) {
    p[i] = i;
}

for (j = 1; j<=m; j++) {
    t_j = fonetiklist[j-1];
    d[0] = j;

    for (i=1; i<=n; i++) {
        cost = fonetiklist1[i-1]==t_j ? 0 : 1;
```

Setelah dilakukan perubahan-perubahan tersebut, kini program menjadi seperti pada gambar 4.2.

4.1.3. Modifikasi nilai operasi Levenshtein distance

Umumnya pada implementasi Levenshtein distance setiap operasi memiliki nilai 1. Sedangkan pada word distance, terdapat perbedaan nilai operasi. Seperti yang telah disebutkan pada tahap perencanaan, operasi *insertion* dan *deletion* memiliki nilai 0,5. Untuk operasi substitusi nilai diperoleh dari matriks substitusi sesuai dengan pasangan fonem yang tersubstitusi.

Untuk mengubah nilai operasi *insertion* dan *deletion* dilakukan perubahan pada baris:

```
// minimum of cell to the left+1, to the top+1, diagonally left and up +cost
d[i] = Math.min(Math.min(d[i-1]+1, p[i]+1),  p[i-1]+cost);
```

Sehingga menjadi:

```
// minimum of cell to the left+1, to the top+1, diagonally left and up +cost
d[i] = Math.min(Math.min(d[i-1]+0.5, p[i]+0.5),  p[i-1]+cost);
```

Untuk mengubah nilai operasi substitusi, perlu dibuat matriks substitusi yang merupakan komponen utama dari penilaian operasi substitusi. Matriks tersebut direpresentasikan sebagai array dua dimensi berukuran 33 x 33.

```
distancematrix = new double[33][33];
```

Untuk mengakses nilai pada matriks, setiap fonem memiliki pemetaan dengan nilai *integer*. Pemetaan tersebut dilakukan pada method fonemToInt pada program. Misalnya fonem "kk" memiliki nilai pemetaan 9 dan fonem "y" memiliki nilai pemetaan 21, sehingga nilai operasi substitusi antara fonem "kk" dan fonem "y" disimpan pada distancematrix[21][9] atau distancematrix[9][21]. Pada program nilai distance diperoleh dengan cara:

```
if(idx1==100||idx2==100)
{
  cost = 0;
}
else
{
  cost = distancematrix[idx1][idx2];
}
```

Nilai variabel idx1 dan idx2 bernilai 100 jika fonem yang akan dipetakan menjadi nilai integer tidak terdapat dalam kamus fonetik Amalia Zahra.

4.1.4. Modifikasi post-normalized word distance

Seperti yang telah dibahas pada subbab 3.3.1, penghitungan word distance tanpa normalisasi cenderung lebih menganggap mirip pasangan kata yang pendek daripada yang panjang. Oleh karena itu, modifikasi dilakukan pada program agar proses normalisasi berjalan setelah penghitungan nilai word distance.

Modifikasi dilakukan pada baris program berikut:

```
  return p[n];
}
```

Sehingga menjadi:

```
double result = p[n]/(m+n);
return result;
```

Sebelum mengembalikan nilai, program melakukan proses normalisasi terlebih dahulu seperti yang telah dibahas pada subbab 3.3.1.

4.1.5. Modifikasi masukan dan keluaran

Sejauh ini, modifikasi yang dilakukan sudah sesuai dengan perancangan yang dilakukan pada bab 3. Method untuk menghitung nilai word distance telah dihasilkan. Namun untuk keperluan eksperimen dimana akan dilakukan penghitungan terhadap sejumlah besar pasangan kata, maka perlu dilakukan perubahan pada program. Program diharapkan untuk mampu menerima masukan berupa berkas teks yang berisi pasangan-pasangan kata beserta bentuk fonemnya. Berikut format masukan program:

```
kata-1 [tab] fonem kata-1 [tab] kata-2 [tab] fonem kata-2
[baris kosong]
kata-1 [tab] fonem kata-2 [tab] kata-3 [tab] fonem kata-3
…
```

Setiap baris berisi pasangan kata yang akan dihitung nilai jarak kemiripan bunyinya. Misal pada baris pertama pasangan tersebut ialah kata-1 dan kata-2. Fonem kata-1 merupakan bentuk fonem dari kata-1 dan fonem kata-2 merupakan bentuk fonem dari kata-2. Antara kata dan fonem pada satu baris dipisahkan oleh tab dan antar baris dipisahkan oleh baris baru. Yang dijadikan masukan untuk menghitung jarak kemiripan bunyi antara kata-1 dan kata-2 melalui method word distance ialah fonem-1 dan fonem-2. Gambar 4.3 beberapa tambahan pada program agar bisa menerima masukan program seperti yang dibahas diatas.

Program membaca masukan baris demi baris. Terhadap setiap baris tersebut program melakukan split terhadap string yang dibaca sehingga pada satu baris didapatkan 4 string yang terdiri dari kata-1, kata-2, fonem-1, dan fonem-2. Fonem-1 dan fonem-2 dijadikan masukan bagi method bernama getLevenshteinDistance yang merupakan method untuk menghitung nilai word distance. Hasil dari penghitungan kemudian ditulis dalam satu berkas teks.

Implementasi Levenshtein distance yang digunakan semula hanya merupakan sebuah *method* yang mengembalikan nilai hasil penghitungan. Untuk keperluan

```
public static double getLevenshteinDistance (String s, String t) {
  if (s == null || t == null) {
    throw new IllegalArgumentException("Strings must not be null");
  }

  String[] fonetiklist1;
  String[] fonetiklist2;
  String delimeter = "\\s+";
  fonetiklist1 = s.split(delimeter);
  fonetiklist2 = t.split(delimeter);

  int n = fonetiklist1.length(); // length of s
  int m = fonetiklist2.length(); // length of t

  if (n == 0) {
    return m;
  } else if (m == 0) {
    return n;
  }

  double p[] = new double[n+1]; //'previous' cost array, horizontally
  double d[] = new double[n+1]; // cost array, horizontally
  double _d[]; //placeholder to assist in swapping p and d

  // indexes into strings s and t
  int i; // iterates through s
  int j; // iterates through t

  String t_j; // jth character of t

  double cost; // cost

  for (i = 0; i<=n; i++) {
    p[i] = i;
  }

  for (j = 1; j<=m; j++) {
    t_j = fonetiklist[j-1];
    d[0] = j;

    for (i=1; i<=n; i++) {
      cost = fonetiklist1[i-1]==t_j ? 0 : 1;
      // minimum of cell to the left+1, to the top+1, diagonally left and up +cost
      d[i] = Math.min(Math.min(d[i-1]+1, p[i]+1),  p[i-1]+cost);
    }

    // copy current distance counts to 'previous row' distance counts
    _d = p;
    p = d;
    d = _d;
  }

  // our last action in the above loop was to switch d and p, so p now
  // actually has the most recent cost counts
  return p[n];
}
```

Gambar 4. 2 Implementasi program setelah modifikasi umum

```
try
{
  FileReader pembaca = new FileReader("pairs_56_3.txt");
  BufferedReader br = new BufferedReader(pembaca);
  FileWriter penulis = new FileWriter("result56tes.txt",true);

  String s;
  int oddeven = 0;
  while((s = br.readLine()) != null) {
    //split string
    String [] slist = s.split("\\t") ;

    if((oddeven % 2) == 0)
    {
      double distance = getLevenshteinDistance(slist[1],slist[3]);

      if(distance<THRESHOLD)
      {
        if((slist[0]!=null) || (slist[2]!=null))
        {
          penulis.write(slist[0] +","+slist[2]+"\t" + distance + "\n");
          penulis.flush();
        }
      }
    }
    oddeven++;
  }
  pembaca.close();
  penulis.close();
}
catch(Exception ex)
{

}
```

Gambar 4. 3 Tambahan pada program untuk modifikasi masukan

penelitian, hasil penghitungan akan disimpan dalam berkas teks. Oleh karena itu, pada program ditambahkan baris berikut untuk menangani masalah masukan dan keluaran.

```
FileReader pembaca = new FileReader("pairs_36.txt");
BufferedReader br = new BufferedReader(pembaca);
FileWriter penulis = new FileWriter("result36_good.txt",true);
FileWriter penulis2 = new FileWriter("result36_bad.txt",true);
```

4.1.6. Nilai *threshold*

Setelah program melakukan penghitungan word distance terhadap pasangan kata yang diberikan, untuk menyimpan data penghitungan dalam rentang nilai tertentu diperlukan adanya nilai *threshold*. Terdapat dua nilai *threshold* pada program,

yaitu 0,05 dan 0,48. Pembahasan lebih lanjut mengenai penggunaan nilai *threshold* akan diulas pada subbab 5.2.1.

4.1.7. Aplikasi word distance

Setelah modifikasi-modifikasi dilakukan terhadap implementasi Levenshtein distance, maka aplikasi word distance yang sesuai dengan tujuan penelitian ini siap digunakan. Method word distance menerima masukan dua buah kata yang telah diubah menjadi bentuk fonemnya masing-masing dan mengembalikan nilai word distance dengan nilai operasi sesuai dengan modifikasi dan proses normalisasi pada akhir penghitungan. Program juga telah dapat menerima masukan berupa berkas teks yang berisi pasangan-pasangan kata beserta bentuk fonemnya.

Source code lengkap aplikasi word distance yang dihasilkan dapat dilihat pada lampiran A.

4.2. Implementasi ekstraksi kuesioner

Data hasil kuesioner memiliki jumlah yang cukup banyak, sehingga untuk memudahkan proses ekstraksi kuesioner dibuat program yang dapat memproses data hasil kuesioner dengan cara yang telah dibahas pada subbab 3.3. Program ekstraksi dibuat dalam bahasa pemrograman Perl.

4.2.1. Format masukan program

Data hasil kuesioner yang diperoleh mula-mula dicatat dalam berkas teks dengan format sebagai berikut:

a_1
Set fonem yang diurutkan
[baris kosong]
a_2
Set fonem yang diurutkan
[baris kosong]
…
a_n
Set fonem yang diurutkan

Fonem a_1 hingga a_n merupakan fonem acuan pertanyaan tersebut. Pada baris setelah fonem acuan merupakan jawaban dari pertanyaan, yaitu fonem-fonem hasil pengurutan terhadap fonem acuan yang dipisahkan dengan spasi diantara

fonem. Fonem yang berada paling kiri pada jawaban merupakan fonem yang paling mirip dengan fonem acuan pertanyaan tersebut. Tiap pertanyaan dipisahkan dengan baris baru. Berikut merupakan contoh jawaban dari salah satu subjek kuesioner sesuai dengan format yang dijelaskan diatas.

```
a
aw ay ey e ax o oy u i

p
b g d c j t z f s sy k

b
d c g j t n m ny ng h k

t
c d b p g w j y f r k kk

d
z n m r l h ny sy kh ng
```

4.2.2. Implementasi yang dilakukan

Setelah menetapkan format masukan, selanjutnya akan dibuat program untuk mengekstraksi jawaban dengan format tersebut. Dalam format tersebut, jawaban untuk setiap subjek disimpan dalam satu berkas teks. Terdapat 34 berkas teks untuk setiap jawaban. Masukan untuk program ekstraksi kuesioner ialah satu berkas teks yang merupakan kompilasi dari 34 berkas teks jawaban tersebut. Gambar 4.4 dan gambar 4.5 merupakan source code program ekstraksi kuesioner.

Berkas masukan ialah “subjekall3.txt” dan berkas keluaran ialah “hasilekstraksi_tes2.txt”. Program membaca berkas masukan baris demi baris. Variabel linenumber ialah bilangan integer yang menyatakan baris dari berkas masukan yang sedang dibaca oleh program. Sedangkan variabel fonem1 merupakan

fonem acuan. Jika nilai linenumber modulo 3 ialah 1, maka program berada pada baris fonem acuan. Fonem acuan yang dibaca disimpan pada variable fonem1. Kemudian program membaca baris berikutnya yang berisi fonem-fonem hasil pengurutan terhadap fonem acuan. Program lalu membentuk pasangan-pasangan fonem yang dihasilkan antara fonem acuan dengan fonem yang diurutkan lalu menambahkan nilai terhadap pasangan tersebut sesuai dengan posisi pengurutan

pada jawaban. Hasil penambahan nilai tersebut disimpan dalam associative array dengan keyword pasangan fonem tersebut. Setelah seluruh baris selesai dibaca oleh program, maka akan diperoleh hasil penjumlahan nilai untuk setiap pasangan fonem.

Selanjutnya ialah melakukan normalisasi nilai penjumlahan tersebut agar nilai jarak kemiripan antar pasangan fonem berada pada jangkauan 0 sampai 1. Setelah itu program menulis hasil ekstraksi pada berkas “hasilekstraksi_tes2.txt”.

4.3. Rangkuman

Implementasi program-program yang dibutuhkan untuk eksperimen telah dilakukan. Eksperimen kini siap untuk dilakukan. Pembahasan mengenai eksperimen yang dilakukan dibahas pada bab selanjutnya.

```
open(IN, "<subjekall3.txt");
open(OUT, ">hasilekstraksi_tes2.txt");

$fonem1 = "";
$linenumber = 0;
$tobeprinted = "";
$index = "";

%distance = ();
%frekuensi = ();

while ($line=<IN>)
{
	$line =~ s/^\s*//;
	$line =~ s/\s*$//;

	if(($linenumber % 3) == 0)
	{
		$fonem1 = $line;
	}

	elsif(($linenumber % 3) == 1)
	{
		@list = split(/\s+/,$line);
		@reverselist = reverse(@list);

		$score = 1;

		while($kata=pop@reverselist)
		{
			$index = $fonem1 . "_" . $kata;
			$indexreverse = $kata . "_" . $fonem1;

			if(!($distance{$indexreverse}))
			{
				$distance{$index} = $distance{$index} + $score;
				$frekuensi{$index} ++;
			}
```

Gambar 4. 4 Source code program ekstraksi kuesioner

```
            else
            {
                $distance{$indexreverse} = $distance{$indexreverse} + $score;
                $frekuensi{$indexreverse} ++;
            }

            $score = $score + 2;
        }
    }
    $linenumber ++;
}


foreach $fon_idx (sort keys %distance)
{
    @list2 = split(//,$fon_idx);
    @reverselist2 = reverse(@list2);

    $fnm=pop@reverselist2;

    if( ($fnm eq "a") || ($fnm eq "e") || ($fon_idx eq "i") || ($fon_idx eq "o") || ($fon_idx eq "u"))
    {
        $normalized = $distance{$fon_idx}/102;
    }
    else
    {
        $normalized = $distance{$fon_idx}/126;
    }
    $tobeprinted .= "distance(" . $fon_idx . ") = " . $normalized . "\n\n";

}

print OUT "$tobeprinted";
```

Gambar 4. 5 Terusan source code program ekstraksi kuesioner

BAB 5 EKSPERIMEN

5.1. Pelaksanaan kuesioner

Setelah perencanaan kuesioner dinilai sudah baik dan format kuesioner sudah final, kuesioner mulai disebarkan. Subjek kuesioner ialah mahasiswa Fasilkom dari angkatan 2004 hingga 2008. Terdapat 34 subjek yang berpartisipasi dalam pengisian kuesioner sesuai dengan yang direncanakan.

Proses pengerjaan kuesioner dilakukan dengan subjek diberikan lembar kuesioner yang terdiri dari sedikit penjelasan mengenai penelitian yang dilakukan, instruksi pengerjaan, dan lembar pengerjaan seperti yang telah dibahas pada subbab 3.2.2.8. Penulis memberikan sedikit penjelasan terhadap lembar kuesioner dan subjek kuesioner dapat menanyakan pertanyaan terhadap penulis. Pengerjaan kuesioner tidak diberikan batas waktu.

Contoh jawaban subjek kuesioner dapat dilihat pada lampiran E.

5.1.1. Hasil kuesioner

Data pada kuesioner yang diperoleh diekstrak untuk mendapatkan nilai matriks substitusi. Ekstraksi kuesioner dilakukan sesuai dengan yang telah direncanakan pada subbab 3.3. Hasil ekstraksi yang diperoleh dapat dilihat pada tabel 5.1 dan tabel 5.2.

Setelah ekstraksi selesai dilakukan dan matriks substitusi diperoleh, dilakukan proses observasi. Berdasar observasi tersebut, nilai-nilai yang terdapat pada matriks substitusi cukup sesuai dengan intuisi mengenai kemiripan bunyi. Seperti misalnya distance antara fonetik p dan b yang bernilai kecil.

Selain itu, juga dilakukan penghitungan nilai rata-rata dan standar deviasi terhadap data yang diperoleh. Penghitungan nilai rata-rata dilakukan untuk setiap pasangan fonem dan diperoleh dengan cara membagi nilai pada matriks dengan 6, hal ini disebabkan oleh nilai pada matriks substitusi pada dasarnya merupakan penjumlahan dari nilai-nilai pada data kuesioner. Nilai standar deviasi juga dilakukan untuk tiap pasangan fonem pada data kuesioner. Hasil penghitungan nilai rata-rata dan standar deviasi dapat dilihat pada lampiran B.

Tabel 5. 1 Matriks substitusi hasil ekstraksi kuesioner untuk fonem konsonan

	p	b	t	d	c	j	k	g	f	s	z	sy	kh	h	m	n	ng	ny	l	r	w	y	kk
p	0	**0.05**	0.25	0.29	0.48	0.57	0.47	0.25	0.25	0.75	0.67	0.79	0.6	0.46	0.43	0.45	0.59	0.73	0.48	0.4	0.6	0.63	0.35
b	**0.05**	0	0.4	0.25	0.65	0.57	0.57	0.25	0.43	0.63	0.41	0.83	0.34	0.44	0.35	0.67	0.65	0.71	0.83	0.63	0.41	0.65	0.59
t	0.25	0.4	0	**0.11**	0.25	0.46	0.63	0.51	0.49	0.71	0.57	0.85	0.39	0.48	0.71	0.35	0.43	0.56	0.48	0.49	0.51	0.68	0.51
d	0.29	0.25	**0.11**	0	0.57	0.4	0.49	0.44	0.54	0.64	0.61	0.76	0.63	0.69	0.37	0.58	0.71	0.75	0.49	0.54	0.67	0.54	0.59
c	0.48	0.65	0.25	0.57	0	0.28	0.24	0.4	0.54	0.29	**0.19**	0.27	0.28	0.6	0.57	0.67	0.67	0.57	0.71	0.65	0.38	0.46	0.87
j	0.57	0.57	0.46	0.4	0.28	0	0.54	**0.14**	0.67	0.47	0.24	0.62	0.59	0.51	0.68	0.51	0.52	0.45	0.56	0.65	0.37	0.29	0.76
k	0.47	0.57	0.63	0.49	0.24	0.54	0	0.54	0.62	0.68	0.86	0.7	0.18	0.37	0.79	0.62	0.65	0.79	0.67	0.44	0.7	0.67	**0.1**
g	0.25	0.25	0.51	0.44	0.4	**0.14**	0.54	0	0.48	0.65	0.38	0.79	0.17	0.57	0.46	0.51	0.37	0.59	0.64	0.57	0.52	0.49	0.25
f	**0.25**	0.43	0.49	0.54	0.54	0.67	0.62	0.48	0	0.38	0.79	0.65	0.4	0.29	0.57	0.67	0.7	0.52	0.44	0.57	0.4	0.63	0.78
s	0.75	0.63	0.71	0.64	0.29	0.47	0.68	0.65	0.38	0	0.13	**0.1**	0.93	0.87	0.7	0.57	0.6	0.49	0.42	0.51	0.51	0.41	0.65
z	0.67	0.41	0.57	0.61	0.19	0.24	0.86	0.38	0.79	**0.13**	0	0.29	0.67	0.71	0.73	0.65	0.67	0.81	0.55	0.35	0.52	0.65	0.56
sy	0.79	0.83	0.85	0.76	0.27	0.62	0.7	0.79	0.65	**0.1**	0.29	0	0.64	0.7	0.68	0.65	0.6	0.47	0.38	0.45	0.33	0.17	0.57
kh	0.6	0.34	0.39	0.63	0.28	0.59	0.18	0.17	0.4	0.93	0.67	0.64	0	0.17	0.62	0.59	0.6	0.71	0.35	0.67	0.63	0.65	**0.08**
h	0.46	0.44	0.48	0.69	0.6	0.51	0.37	0.57	0.29	0.87	0.71	0.7	**0.17**	0	0.57	0.71	0.89	0.62	0.47	0.47	0.38	0.43	0.6
m	0.43	0.35	0.71	0.37	0.57	0.68	0.79	0.46	0.57	0.7	0.73	0.68	0.62	0.57	0	**0.08**	0.17	0.25	0.52	0.88	0.4	0.7	0.46
n	0.45	0.67	0.35	0.58	0.67	0.51	0.62	0.51	0.67	0.57	0.65	0.65	0.59	0.71	**0.08**	0	0.1	0.08	0.46	0.57	0.6	0.61	0.68
ng	0.59	0.65	0.43	0.71	0.67	0.52	0.65	0.37	0.7	0.6	0.67	0.6	0.6	0.89	0.17	**0.1**	0	0.14	0.5	0.52	0.63	0.56	0.55
ny	0.73	0.71	0.56	0.75	0.57	0.45	0.79	0.59	0.52	0.49	0.81	0.47	0.71	0.62	0.25	0.08	0.14	0	0.37	0.6	0.54	**0.06**	0.87
l	0.48	0.83	0.48	0.49	0.71	0.56	0.67	0.64	0.44	0.42	0.55	0.38	0.35	0.47	0.52	0.46	0.5	0.37	0	**0.05**	0.52	0.46	0.75
r	0.4	0.63	0.49	0.54	0.65	0.65	0.44	0.57	0.57	0.51	0.35	0.45	0.67	0.47	0.88	0.57	0.52	0.6	**0.05**	0	0.29	0.56	0.68
w	0.6	0.41	0.51	0.67	0.38	0.37	0.7	0.52	0.4	0.51	0.52	0.33	0.63	0.38	0.4	0.6	0.63	0.54	0.52	**0.29**	0	0.4	0.76
y	0.63	0.65	0.68	0.54	0.46	0.29	0.67	0.49	0.63	0.41	0.65	0.17	0.65	0.43	0.7	0.61	0.56	**0.06**	0.46	0.56	0.4	0	0.72
kk	0.35	0.59	0.51	0.59	0.87	0.76	0.1	0.25	0.78	0.65	0.56	0.57	**0.08**	0.6	0.46	0.68	0.55	0.87	0.75	0.68	0.76	0.72	0

Tabel 5. 2 substitusi hasil ekstraksi kuesioner untuk fonem vokal

	a	i	u	e	ax	o	ay	oy	aw	ey
a	0	0.666	0.627	0.401	0.568	0.529	**0.313**	0.617	0.333	0.696
i	0.666	0	0.745	**0.392**	0.843	0.676	0.529	0.794	0.666	0.45
u	0.627	0.745	0	0.754	0.764	0.48	0.852	0.45	**0.254**	0.764
e	0.401	0.392	0.754	0	0.431	0.686	0.45	0.588	0.647	**0.058**
ax	0.568	0.843	0.764	0.431	0	0.901	0.382	0.764	0.607	**0.352**
o	0.529	0.676	0.48	0.686	0.901	0	0.558	**0.098**	0.235	0.647
ay	0.313	0.529	0.852	0.45	0.382	0.558	0	0.254	0.47	**0.156**
oy	0.617	0.794	0.45	0.588	0.764	0.098	0.254	0	0.392	**0.274**
aw	0.333	0.666	0.254	0.647	0.607	**0.235**	0.47	0.392	0	0.686
ey	0.696	0.45	0.764	**0.058**	0.352	0.647	0.156	0.274	0.686	0

5.1.2. Perbandingan dengan model phonetic similarity STANDUP

Perbandingan dengan model phonetic similarity STANDUP [RGH08] (lihat bab 2) akan dilakukan terhadap matriks substitusi yang telah dihasilkan. Setiap nilai jarak kemiripan pasangan fonem pada model phonetic similarity STANDUP akan dibandingkan dengan nilai matriks substitusi untuk pasangan fonem yang sama. Fonem pada model STANDUP diperoleh dari Unisyn Phonetic Alphabet. Fonem pada model STANDUP tidak memiliki fonem yang sama dengan fonem pada kamus fonetik Amalia Zahra, sehingga mula-mula perlu dilakukan pemetaan antara fonem pada model STANDUP dengan kamus fonetik Amalia Zahra. Tabel 5.3 merupakan tabel fonem yang berhasil dipetakan untuk fonem konsonan. Fonem-fonem yang tidak berhasil dipetakan ialah simbol ?, dh, hw, l!, m!, n!, t^, th, dan zh.

Setelah dilakukan pemetaan, terhadap fonem yang berhasil dipetakan dilakukan penghitungan nilai jarak kemiripan pasangan fonem sesuai dengan yang dibahas pada bab 2. Tabel 5.4 merupakan matriks yang dihasilkan dari penghitungan nilai jarak kemiripan.

Setelah nilai jarak kemiripan model STANDUP diperoleh, dilakukan proses pembandingan. Proses pembandingan dilakukan setelah melakukan penghitungan nilai rata-rata dan nilai standar deviasi terhadap data hasil kuesioner.

Tabel 5. 3 Pemetaan fonem STANDUP dan fonem Amalia Zahra

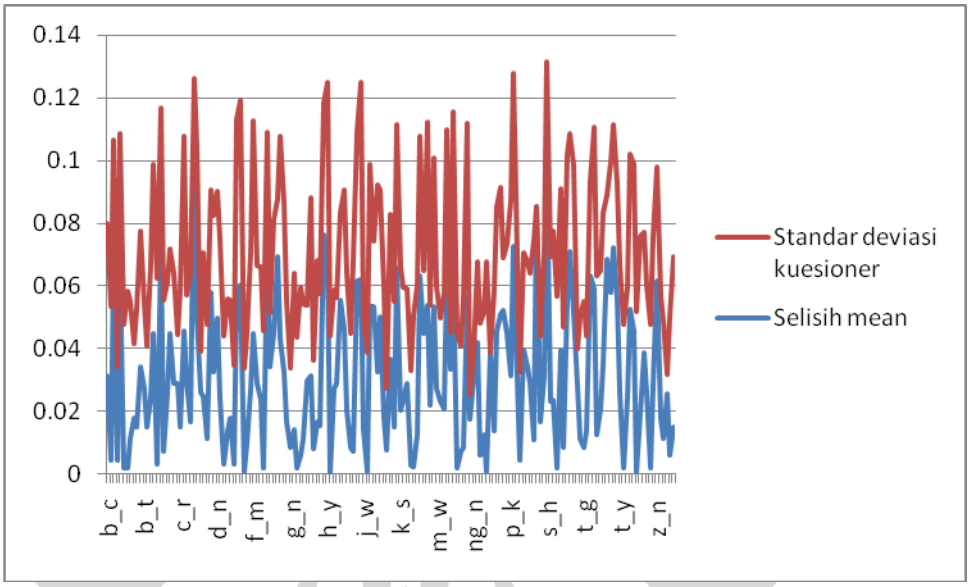
Fonem STANDUP	Fonem Amalia Zahra
b	b
ch	c
d	d
f	f
g	g
h	h
jh	j
k	k
l	l
m	m
n	n
ng	ng
p	p
r	r
s	s
sh	sy
t	t
w	w
y	y
z	z

Tabel 5. 4 Nilai jarak kemiripan untuk fonem konsonan pada model phonetic similarity STANDUP

	b	c	d	f	g	h	j	k	l	m	n	ng	p	r	s	sy	t	w	y	z
b	0	0.84	0.28	0.84	0.28	0.84	0.56	0.56	0.84	0.28	0.56	0.56	0.28	0.56	0.84	0.84	0.56	0.56	0.56	0.56
c	0.84	0	0.84	0.56	0.84	0.56	0.28	0.56	0.56	0.84	0.84	0.84	0.56	0.56	0.56	0.28	0.56	0.56	0.56	0.84
d	0.28	0.84	0	0.84	0.28	0.84	0.56	0.56	0.84	0.56	0.28	0.56	0.56	0.56	0.56	0.84	0.28	0.56	0.56	0.28
f	0.84	0.56	0.84	0	0.84	0.28	0.84	0.56	0.28	0.84	0.84	0.84	0.56	0.56	0.28	0.28	0.56	0.84	0.84	0.56
g	0.28	0.84	0.28	0.84	0	0.84	0.56	0.28	0.84	0.56	0.56	0.28	0.56	0.56	0.84	0.84	0.56	0.56	0.56	0.56
h	0.84	0.56	0.84	0.28	0.84	0	0.84	0.56	0.28	0.84	0.84	0.84	0.56	0.56	0.28	0.28	0.56	0.84	0.84	0.56
j	0.56	0.28	0.56	0.84	0.56	0.84	0	0.84	0.84	0.56	0.56	0.56	0.84	0.28	0.84	0.56	0.84	0.28	0.28	0.56
k	0.56	0.56	0.56	0.56	0.28	0.56	0.84	0	0.56	0.84	0.84	0.56	0.28	0.84	0.56	0.56	0.28	0.84	0.84	0.84
l	0.84	0.56	0.84	0.28	0.84	0.28	0.84	0.56	0	0.84	0.84	0.84	0.56	0.56	0.28	0.28	0.56	0.84	0.84	0.56
m	0.28	0.84	0.56	0.84	0.56	0.84	0.56	0.84	0.84	0	0.28	0.28	0.56	0.56	0.84	0.84	0.84	0.56	0.56	0.56
n	0.56	0.84	0.28	0.84	0.56	0.84	0.56	0.84	0.84	0.28	0	0.28	0.84	0.56	0.56	0.84	0.56	0.56	0.56	0.28
ng	0.56	0.84	0.56	0.84	0.28	0.84	0.56	0.56	0.84	0.28	0.28	0	0.84	0.56	0.84	0.84	0.84	0.56	0.56	0.56
p	0.28	0.56	0.56	0.56	0.56	0.56	0.84	0.28	0.56	0.56	0.84	0.84	0	0.84	0.56	0.56	0.28	0.84	0.84	0.84
r	0.56	0.56	0.56	0.56	0.56	0.56	0.28	0.84	0.56	0.56	0.56	0.56	0.84	0	0.56	0.28	0.84	0.56	0.56	0.28
s	0.84	0.56	0.56	0.28	0.84	0.28	0.84	0.56	0.28	0.84	0.56	0.84	0.56	0.56	0	0.28	0.28	0.84	0.84	0.28
sy	0.84	0.28	0.84	0.28	0.84	0.28	0.56	0.56	0.28	0.84	0.84	0.84	0.56	0.28	0.28	0	0.56	0.84	0.84	0.56
t	0.56	0.56	0.28	0.56	0.56	0.56	0.84	0.28	0.56	0.84	0.56	0.84	0.28	0.84	0.28	0.56	0	0.84	0.84	0.56
w	0.56	0.56	0.56	0.84	0.56	0.84	0.28	0.84	0.84	0.56	0.56	0.56	0.84	0.56	0.84	0.84	0.84	0	0.28	0.56
y	0.56	0.56	0.56	0.84	0.56	0.84	0.28	0.84	0.84	0.56	0.56	0.56	0.84	0.56	0.84	0.84	0.84	0.28	0	0.56
z	0.56	0.84	0.28	0.56	0.56	0.56	0.56	0.84	0.56	0.56	0.28	0.56	0.84	0.28	0.28	0.56	0.56	0.56	0.56	0

Nilai yang dibandingkan ialah nilai selisih rata-rata data kuesioner dengan nilai jarak kemiripan model STANDUP dan nilai standar deviasi data kuesioner. Nilai jarak kemiripan model STANDUP yang dibandingkan dibagi 6 agar bersifat comparable dengan nilai rata-rata jarak kemiripan hasil kuesioner.

Berikut hasil perbandingan nilai yang dilakukan disajikan dalam bentuk grafik.



Gambar 5. 1 Perbandingan standar deviasi data kuesioner dengan selisih nilai data kuesioner dan model STANDUP

Perbandingan yang dilakukan memperoleh hasil yang baik. Dapat dilihat bahwa nilai selisih antara rata-rata hasil kuesioner dan nilai phonetic distance model STANDUP yang dibagi 6 pada kebanyakan pasangan masih lebih kecil dari nilai standar deviasi data hasil kuesioner. Hal ini dapat diinterpretasikan bahwa perbedaan nilai phonetic distance yang ada dapat dibandingkan dengan perbedaan pendapat antara tiap subjek terhadap kemiripan bunyi. Hasil lengkap perbandingan dalam berupa tabel dapat dilihat pada lampiran B.

5.2. Eksperimen word distance

Setelah matriks substitusi diperoleh, dilakukan penghitungan word distance terhadap daftar leksikon Amalia Zahra yang terdiri dari 8285 kata dan bentuk fonemnya. Mula-mula dengan input daftar leksikon, dibuat daftar pasangan

leksikon. Program untuk membuat pasangan leksikon dibuat dalam bahasa pemrograman Perl.

Misalnya untuk daftar kata:

adegan	[adegan]	a d e g a n
adegan	[adegan]	a d ax g a n
adeline	[adeline]	a d e l ey n

setelah program pembuat pasangan leksikon dijalankan, akan dihasilkan output:

adegan	a d e g a n	adegan	a d ax g a n
adegan	a d e g a n	adeline	a d e l ey n
adegan	a d ax g a n	adeline	a d e l ey n

Setelah mendapat pasangan-pasangan kata untuk dijadikan input, penghitungan word distance dilakukan terhadap pasangan-pasangan tersebut. Penghitungan dilakukan dengan menjalankan program word distance yang telah dibuat (lihat subbab 4.1).

5.2.1. Hasil eksperimen word distance

Penghitungan nilai word distance dilakukan terhadap setiap pasangan kata, terdapat total kurang lebih 34.316.470 pasangan kata yang dilakukan penghitungan nilai word distance terhadapnya. Hasil dari penghitungan terhadap seluruh pasangan kata tersebut disimpan dalam dua buah berkas teks. Berkas teks pertama menyimpan data dari pasangan-pasangan kata yang memiliki nilai word distance kurang dari *threshold* yang ditentukan yaitu 0,05. Berkas teks kedua menyimpan data dari pasangan-pasangan kata yang memiliki nilai word distance lebih dari 0,48. Berkas teks pertama ditujukan untuk menyimpan daftar pasangan kata yang mirip secara bunyi sedangkan berkas teks kedua ditujukan untuk menyimpan daftar pasangan kata yang tidak mirip secara bunyi. Selain data pada dua berkas teks tersebut disimpan juga data beberapa pasangan-pasangan kata yang memiliki nilai diantara 0,05 dan 0,48 untuk melihat hasil penghitungan secara umum. Namun data tersebut diperoleh bukan dari penghitungan word distance seluruh pasangan kata.

Setelah penghitungan selesai, dilakukan observasi terhadap hasil penghitungan tersebut. Tujuan dari observasi ialah untuk melihat apakah pasangan kata yang memiliki kemiripan bunyi mendapat nilai lebih kecil dari pasangan kata yang tidak memiliki kemiripan bunyi. Perlu diperhatikan bahwa pada dasarnya kemiripan bunyi merupakan hal yang subjektif dan tidak ada definisi mutlak mengenai kemiripan bunyi sehingga kemiripan bunyi yang dimaksud diatas ialah kemiripan bunyi secara intuitif.

Dari hasil observasi penulis terhadap data hasil penghitungan word distance untuk setiap pasangan fonem, kata-kata yang memiliki nilai word distance kecil cenderung lebih mirip secara bunyi dari kata-kata yang memiliki nilai word distance besar. Berikut tabel-tabel beberapa pasangan kata dan nilai word distance pasangan kata tersebut sebagai contoh.

Terlihat dari contoh pasangan yang diberikan pada tabel 5.5 bahwa pasangan kata yang memiliki nilai word distance lebih kecil lebih mirip secara bunyi daripada pasangan kata yang memiliki nilai lebih besar. Secara umum, model penghitungan word distance cenderung setuju dengan intuisi manusia mengenai kemiripan bunyi.

Tabel 5. 5 Contoh pasangan kata dan nilai word distance pasangan kata tersebut

No	Pasangan kata	Nilai word distance
1	Bersalah,berdarah	0,043
2	bidang,pinang	0,063
3	acuan,anjuran	0,082
4	binatang,penataan	0,099
5	Fakta,pabrik	0,226
6	Persegi,fenomena	0,271
7	Kepala,esensi	0,331
8	Era,kritis	0,352

Rangkuman hasil dari penghitungan word distance dapat dilihat pada lampiran C Pada lampiran tersebut dicantumkan data beberapa pasangan yang terdapat pada berkas data pertama dan kedua yang telah dibahas sebelumnya. Terhadap data

pada berkas pertama, yaitu data dengan nilai word distance kurang dari 0,05, akan ditampilkan 70 pasangan dengan nilai word distance terkecil, 70 pasangan dengan nilai terkecil namun lebih dari 0 dan 70 pasangan dengan nilai word distance terbesar. Terhadap data pada berkas kedua, yaitu data dengan nilai word distance lebih dari 0,48, akan ditampilkan 70 pasangan dengan nilai word distance terbesar.