PEMBUATAN APLIKASI UNTUK MELATIH DAN MENGUJI KECEPATAN SERTA KETEPATAN PENGETIKAN DENGAN SISTEM *SCORE* BERBASIS SISTEM *FUZZY*

Rudy Adipranata[1], Rolly Intan[2] dan Yonathan Susanto
Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto 121-131, Surabaya
E-mail: rudya@petra.ac.id[1], rintan@petra.ac.id[2]

ABSTRAKSI

Pengetikan termasuk salah satu kemampuan dasar yang diperlukan dalam pekerjaan atau studi. Untuk melakukan pengetikan, biasanya orang menggunakan mesin ketik, tetapi sekarang ini dengan semakin banyaknya pengguna komputer, maka fungsi mesin ketik perlahan-lahan digantikan dengan komputer. Tetapi tidak semua orang mempunyai kemampuan mengetik dengan benar yaitu dengan menggunakan sepuluh jari.

Untuk membantu seseorang meningkatkan kemampuan mengetik dengan benar, pada penelitian ini dibuat sebuah aplikasi yang dapat melatih pengetikan serta menghitung kecepatan dan ketepatan pengetikan yang telah dilakukan seseorang. Hanya saja berbeda dengan aplikasi yang biasanya menghitung score akhir hasil pelatihan secara absolut, pada aplikasi ini digunakan sistem fuzzy untuk menghitung score akhir.

Untuk menentukan score digunakan penilaian secara absolut dan relatif untuk kecepatan dan ketepatan mengetik. Kemudian kedua buah score tersebut yaitu score absolut dan relatif digabungkan dengan menggunakan over weighted averanging operation. Setelah score ketepatan dan kecepatan didapat, digunakan generalized means untuk menggabungkan kedua score tersebut menjadi score akhir.

Kata kunci: *sistem fuzzy, latihan pengetikan, generalized means, over weighted averaging*

1. PENDAHULUAN

Pengetikan bukanlah hal sepele yang dapat diabaikan, sebagai contoh pada proses seleksi untuk mendapatkan pekerjaan di beberapa posisi, pelamar harus memiliki kemampuan yang bagus dalam mengetik. Kemampuan mengetik yang baik tidak hanya diperhatikan oleh perusahaan saja, pada suatu organisasi ada beberapa posisi yang anggotanya diharapkan memiliki kemampuan mengetik yang baik sehingga dapat mempercepat penyelesaian pekerjaannya. Dari hal-hal ini dapat dilihat bahwa kemampuan mengetik bukanlah suatu hal yang dapat diabaikan begitu saja.

Untuk melakukan pengetikan, biasanya digunakan mesin ketik, tetapi sejak komputer mulai dipakai secara umum, fungsi dari komputer mulai menggeser mesin ketik secara pelahan-lahan meskipun saat ini pun mesin ketik masih dipakai. Berbagai bentuk dan pengembangan dari komputer khususnya yang berkaitan dengan *keyboard* sebagai media input pengetikan terus terjadi sampai saat ini, sehingga muncul beberapa bentuk baru *keyboard* yang bertujuan untuk mempermudah pemakai dalam melakukan pengetikan dengan lebih santai dan lebih nyaman.

Tetapi meskipun perkembangan *keyboard* terus berkembang masih banyak orang yang belum mempunyai kemampuan mengetik secara benar dengan mengunakan kesepuluh jari. Karena itu dibutuhkan sebuah alat bantu untuk melatih serta menguji kemampuan mengetik seseorang, sehingga orang tersebut dapat mengetahui serta meningkatkan kemampuannya dalam mengetik. Dengan latar belakan kondisi tersebut di atas, pada penelitian ini dibuat sebuah aplikasi untuk membantu orang belajar mengetik dengan benar serta melatih kemampuan mengetik. Sebagai output, aplikasi ini memberi penilaian seberapa besar kesalahan dan kecepatan mengetik dari seseorang.

2. TEORI PENUNJANG

Fuzzy System adalah sistem yang berfungsi untuk memberikan nilai atas sesuatu yang tidak pasti, selain itu juga berguna untuk meningkatkan ketepatan suatu nilai kesimpulan [2]. Dalam melakukan suatu penilaian tidak selalu hanya mengunakan nilai absolut (paten) karena penilaian absolut dibuat berdasarkan acuan kesempurnaan suatu hal. Perlu juga digunakan penilaian relatif yang ditentukan berdasarkan keadaan sesungguhnya. Konsep inilah yang digunakan pada aplikasi ini dimana perhitungan nilai hasil kecepatan dan ketepatan pengetikan digunakan nilai absolut serta relatif. Untuk menyatukan kedua nilai ini digunakan *Over Weighted Averanging Operation* (OWA). Dari hasil perhitungan OWA, dilakukan perhitungan kembali untuk mendapatkan nilai akhir.

Untuk menghitung nilai akhir biasanya digunakan fungsi rata-rata / *means* tetapi fungsi ini memiliki kelemahan yaitu muncul nilai akhir yang sama meskipun nilai yang dioperasikan berbeda. Sebagai contoh, *means* dari 2,3,4 sama dengan *means* dari 3,3,3 yaitu 3. Untuk mengatasi hal tersebut, digunakan *Generalized means* yang

merupakan salah satu fungsi untuk memberikan hasil akhir sedikit berbeda pada masalah tersebut.

2.1 Over Weighted Averanging Operation

Over Weighted Averanging Operation adalah cara untuk menghitung rata-rata berdasarkan bobot dari masing-masing nilai. Dimana jumlah total dari bobot yang akan dipakai haruslah satu.

$$\sum_{i=1}^{n} W = 1$$

$$H = B_1W_1 + B_2W_2 + B_3W_3 + \ldots + B_nW_n \qquad (1)$$

dimana:
W = bobot dari suatu nilai
B = nilai yang akan dirata-rata
n = banyak nilai yang akan dirata-rata
H = rata-rata

2.2 Generalized Means

Generalized means adalah fungsi rata-rata yang dapat menghasilkan nilai yang berbeda jika nilai-nilai yang membentuknya berbeda. Misalnya *means* yang didapat dari 2,3,4 sama dengan *means* dari 3,3,3 yaitu 3. Tetapi dengan *generalized means* dengan nilai α = 2 maka *means* dari 2,3,4 adalah 3,11 sedangkan *means* dari 3,3,3 adalah 3. Semakin besar nilai α maka semakin besar pula perbedaan yang terlihat.

$$H = \left(\frac{A_1^{\alpha} + A_2^{\alpha} + A_3^{\alpha} + \ldots + A_n^{\alpha}}{n} \right)^{1/\alpha} \qquad (2)$$

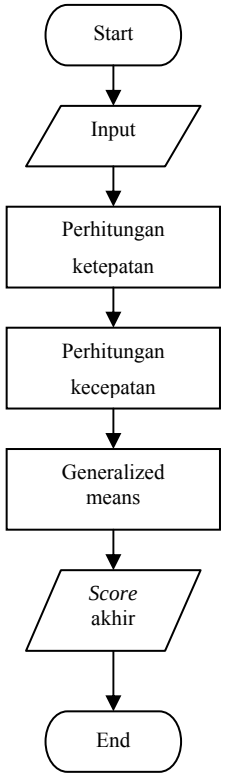
dimana:
A = nilai yang akan dirata-rata
n = banyak nilai yang akan dirata-rata
α = nilai untuk meningkatkan detil perhitungan
H = rata-rata

3. DESAIN APLIKASI

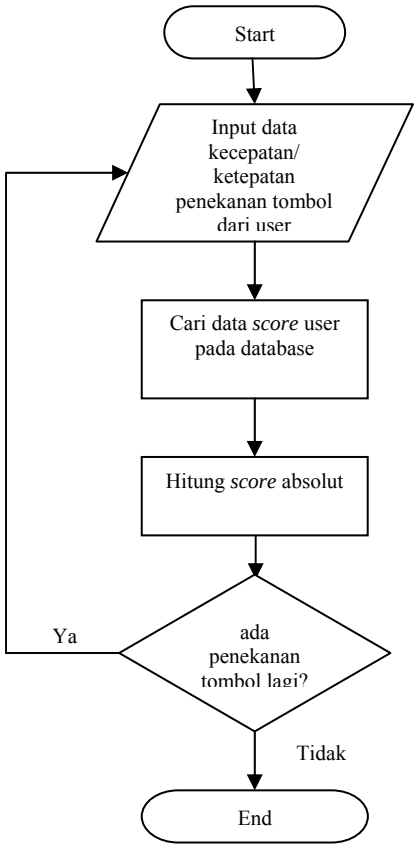
Sebelum melakukan implementasi aplikasi, terlebih dahulu dilakukan perancangan dengan menggunakan diagram alir. Diagram alir keseluruhan aplikasi terlihat pada Gambar 1.

Dalam proses pengetikan dilakukan tiga tahap perhitungan yaitu perhitungan ketepatan, perhitungan kecepatan dan perhitungan *score* total. Pada tahap perhitungan ketepatan dan perhitungan kecepatan dilakukan dua macam perhitungan *score* yaitu: *score* absolut dan *score* relatif. Untuk mengabungkan *score* absolut dan *score* relatif digunakan operasi *Over Weighted Averanging*. Sedangkan untuk perhitungan *score* total dilakukan perhitungan dengan menggunakan *generalized means.*

Yang pertama kali dilakukan dalam perhitungan *score* absolut adalah menyimpan point-point data yang didapat dari user untuk dibandingkan dengan data yang telah disimpan didalam *database*. Dari perbandingan tersebut dapat diambil *score-score* yang akan dijadikan sebagai *score* akhir dari nilai absolut. Diagram alir untuk proses perhitungan *score* absolut adalah seperti terlihat pada Gambar 2.



Gambar 1. Diagram Alir Aplikasi



Gambar 2. Diagram Alir Proses Perhitungan *Score Absolut*

Pada perhitungan *score* relatif, data yang diperoleh dari *database* diproses terlebih dahulu untuk menentukan *range* guna penentuan *score*. Setelah didapatkan *range* maka data yang didapat dari user dibandingkan masuk ke dalam *range score* yang bersangkutan. Langkah terakhir adalah menjumlahkan semua *score*-*score* yang didapatkan menjadi *score* relatif. Terdapat sedikit perbedaan pada proses pengolahan *score* relatif proses perhitungan kecepatan dan proses perhitungan ketepatan. Pada proses perhitungan ketepatan, data yang didapat dari *user* disimpan terlebih dahulu pada *array*, jika terjadi kesalahan yang sama pada pengetikan maka jumlah kesalahan bertambah satu.

Setelah proses pengetikan selesai kemudian akan dicari pada *database* sesuai dengan jumlah kesalahan yang telah terjadi. Kemudian jumlah kesalahan yang ada di database dijumlahkan dengan yang ada di *array*. Sedangkan pada proses perhitungan kecepatan, perhitungan *score* relatif dilakukan dengan cara mengolah semua waktu pengetikan *user*. Jika terdapat kombinasi tombol yang sama maka waktu tersebut akan dijumlahkan. Setelah selesai mengetik untuk masing-masing kombinasi tombol akan dicari rata-rata waktunya. Kemudian catatan rata-rata waktu mengetik untuk masing-masing kombinasi tombol dicari pada *database* dan diurutkan terlebih dahulu sebelum data tersebut di-*update* kembali ke dalam database.

3.1 Absolut Scoring

Absolut scoring adalah perhitungan dengan menjumlahkan semua *score* yang didapat pada waktu proses pengetikan berlangsung.

Untuk proses perhitungan ketepatan, penentuan *score* ditentukan dengan melihat dari letak posisi jari yang benar saat mengetik apakah kesalahan yang dilakukan adalah salah jari atau salah mengingat posisi tombol *keyboard.*

Untuk proses perhitungan kecepatan, penentuan *score* ditentukan dengan melihat kecepatan mengetik yang paling lambat (sekitar 4 detik). Keputusan mengambil 4 detik sebagai jangka waktu terlama dalam mengetik karena jika jarak antara penekanan huruf dengan huruf selanjutnya lebih dari 4 detik hal ini menandakan *user* mencari terlebih dahulu huruf yang ingin dia tekan. Jika kecepatan mengetik antara dua huruf kurang dari 0,4 detik maka nilai yang didapat adalah 1 (kecepatan 0,4 detik menandakan user sudah tahu dimana posisi huruf yang seharusnya dia ketik). Kemudian untuk selanjutnya digunakan kelipatan 0,4 untuk *range score* antara 0,9 hingga 0

3.2 Relatif Scoring

Relatif scoring adalah perhitungan yang dilakukan berdasarkan data yang telah disimpan selama pengetikan dilakukan. Sehingga semakin banyak pengetikan yang pernah dilakukan maka *score* yang dihasilkan akan semakin bervariatif.

Untuk proses perhitungan ketepatan, bila tombol yang ditekan sudah tepat maka diberi nilai sepuluh (nilai maksimal), apabila tombol yang ditekan tidak tepat maka kesalahan tersebut akan disimpan. Yang disimpan adalah tombol yang seharusnya ditekan, tombol yang ditekan oleh user dan kesalahan ini dilakukan berapa kali. Kemudian dari semua kesalahan yang pernah terjadi dicari yang paling sering terjadi dan dari jumlah tersebut ditentukan range kesalahan. Untuk menentukan *score* dilihat dari kesalahan yang pernah terjadi (sesuai dengan yang tercatat). Secara matematis dapat dilihat sebagai berikut:

$$R = N/10$$
$$N - R \times i \le n < N - R \times (i+1) \quad (3)$$

dengan:

$i : \{0,1,2,\ldots,9\}$

R = range.

N = jumlah kesalahan paling sering.

N = jumlah kesalahan yang dilakukan.

Score 0,9 didapat jika i=0, sehingga kesalahan berada antara $N \le n < N - R$, untuk *score* 0,8 didapat jika i=1, sehingga kesalahan berada antara $N - R \le n < N - R \times 2$, untuk *score* 0,7 didapat jika i=2, sehingga kesalahan berada antara $N - R \times 2 \le n < N - R \times 3$ dan seterusnya.

Sebagai contoh, kesalahan terbanyak yang ada pada database adalah 'a' dengan kesalahan tekan menjadi 's' dan kesalahan ini telah terjadi sebanyak 20 kali. Pada proses pengetikan user seharusnya menekan '-' tetapi yang ditekan '=' dan kesalahan sebelum user mengetik telah terjadi 15 kali. Dengan data diatas dapat diketahui *range score* dengan cara:

$$R = 20/10$$
$$R = 2$$

Range didapat dengan membagi kesalahan terbanyak dengan 10, sehingga didapat 2. Setelah mendapatkan *range* kemudian dicari kesalahan antara '-' dengan '=' berada diantara range *score* berapa.

Score 0,9 : $20 \le n < 20 - 2$

Score 0,8 : $20 - 2 \le n < 20 - 2 \times 2$

Score 0,7 : $20 - 2 \times 2 \le n < 20 - 2 \times 3$

$16 \le n < 14$

Jadi jika telah terjadi kesalahan sebanyak 15 kali sebelum *user* yang bersangkutan, maka dengan menggunakan *score* relatif, *score* yang diperoleh oleh *user* adalah 0,7.

Untuk proses perhitungan kecepatan, semua *score* kecepatan dihitung dari data yang telah didapat. Dari waktu mengetik yang telah diambil, dihitung rata-rata kecepatan setiap kombinasi (tombol pertama, tombol kedua dan detik). Kemudian data tersebut disimpan beserta data yang lain yaitu : tombol pertama, tombol kedua, rata-rata

serta berapa orang yang memiliki rata-rata waktu tersebut. Dari setiap kombinasi tombol diambil jumlah total orang yang melakukan kombinasi tersebut, kemudian jumlah dibagi menjadi sepuluh (sesuai dengan point yang akan dipakai dalam kasus ini satu sampai sepuluh). Hal ini digunakan untuk menggolongkan orang sesuai dengan kecepatannya. Untuk setiap kombinasi tombol dicari waktu tercepat, kemudian diambil rata-rata waktu dan berapa orang yang telah melakukannya. Dengan menggunakan rata-rata orang, rata-rata waktu dan jumlah orang yang melakukan setiap kombinasi dicari range waktu dengan *score* antara satu sampai sepuluh.

$$M = \sum P \Big/ 10 \qquad (4)$$

dimana:
S_n= rata-rata waktu ke-n
P_n= jumlah orang dengan rata-rata waktu sama ke-n
M= rata-rata orang

Score 1 didapat jika waktu antara 0 sampai X1 dimana

$$X1 = \frac{M \times S_n}{M} = S_n$$

jika Pn < rP maka

$$X1 = \frac{P_n \times S_n + (M - P_n) \times S_{n+1}}{M}$$

jika $P_{n+1} < (P_n\text{-}rP)$ maka

$$X1 = \frac{P_n \times S_n + P_{n+1} \times S_{n+1} + (M - P_n - P_{n+1}) \times S_{n+2}}{M}$$

dan seterusnya.

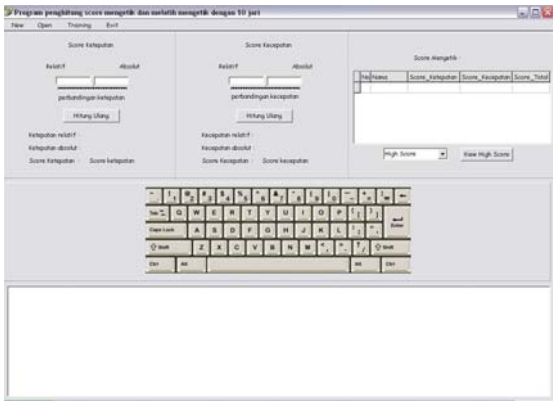
Score 0,9 didapat jika waktu antara X1 sampai X2 dimana:
jika Pn<rP maka mengunakan metode seperti diatas
jika Pn >rP maka $X2 = \frac{(P_{n+1} - M) \times S_{n+1}}{M}$

Sebagai contoh: untuk satu kombinasi tombol, orang yang mengetik dengan kecepatan rata-rata 0,5 detik adalah 1 orang, rata-rata 1 detik dilakukan oleh 3 orang, rata-rata 2 detik dilakukan oleh 2 orang dan rata-rata 2,5 detik dilakukan oleh 9 orang.

4. IMPLEMENTASI DAN HASIL

Setelah melakukan desain, aplikasi diimplementasikan dengan menggunakan bahasa pemrograman Delphi. Adapun hasil aplikasi seperti terlihat pada Gambar 3.

Saat menggunakan aplikasi, *user* dapat memilih untuk melakukan pelatihan dengan menggunakan teks yang sudah ada ataupun menggunakan huruf-huruf yang ditampilkan secara random. Di samping itu, *user* juga dapat melakukan pelatihan terhadap jari tertentu saja dengan menggunakan pilihan seperti terlihat pada Gambar 4.



Gambar 3. Tampilan Aplikasi

Gambar 4. Pilihan Penggunaan Jari Tertentu

Setelah user selesai melakukan pelatihan pengetikan, maka aplikasi akan melakukan perhitungan hasil ketepatan serta hasil kecepatan pengetikan yang telah dilakukan serta juga menghitung *score* akhir user tersebut.

5. KESIMPULAN

Dari hasil yang didapat, dapat diambil kesimpulan bahwa aplikasi telah dapat berjalan dengan baik sesuai dengan tujuan yang diharapkan yaitu untuk membuat sebuah aplikasi yang membantu seseorang untuk melakukan pengetikan dengan menggunakan sepuluh jari dan memberikan penilaian.

Hasil penilaian yang menggunakan sistem *fuzzy* mempunyai keunggulan bahwa nilai yang diperoleh tidak dibandingkan dengan hasil absolut tetapi juga dibandingkan dengan *user* lain yang telah melakukan pelatihan pengetikan dengan aplikasi ini sehingga *user* dapat mengetahui level dimana dia berada dibandingkan dengan *user* lain.

DAFTAR PUSTAKA

[1] Berg, C. *Typing with All Ten Fingers.* http://www.df7cb.de/projects/10finger/ (akses: 13 Desember 2005)
[2] Klir, Yuan. *Fuzzy Set and Fuzzy Logic*. Prentice Hall. 1995
[3] Nugroho, N. *Tip dan Trik Pemrograman Delphi*. Jakarta: PT. Elex Media Komputindo. 2002
[4] Pranata, A. *Pemrograman Borland Delphi* (3rd ed.). Yogyakarta: ANDI. 2000.