No Makalah : 219

APLIKASI DTMC UNTUK POST-PROCESSING PENGENALAN CITRA DOKUMEN TEKS

Anastasia Rita Widiarti[1], Reza Pulungan[2]

[1] Jurusan Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Sanata Dharma
[2] Jurusan Ilmu Komputer dan Elektronika, Fakultas MIPA, Universitas Gadjah Mada
[1] rita_widiarti@usd.ac.id, [2] pulungan@ugm.ac.id

Abstrak

Pengenalan citra dokumen teks dapat menjadi salah satu cara untuk membantu pelestarian kekayaan budaya dan kesusastraan, misalnya untuk pengenalan citra dokumen teks sastra Jawa yang merupakan salah satu kekayaan kesusastraan di Yogyakarta. Salah satu persoalan yang muncul dalam tahap pengenalan adalah ketidakcocokan hasil pengenalan dokumen, yaitu terdapat kesalahan dalam pengenalan yang disebabkan antara lain karena kualitas dokumen yang sudah berkurang, kualitas *scanner* maupun kekurangtepatan dalam *preprosesing* dan pengenalan citra dokumen. Paper ini menyodorkan sebuah gagasan penggunaan *discrete-time Markov chain* (DTMC) dalam sistem pembangunan matrik suku kata yang dibangun dari kombinasi hasil pengenalan oleh sistem dan pembacaan secara manual. Hasil dari matriks suku kata, yang menyajikan informasi rangkaian suku kata terdekat dari suatu suku kata beserta probabilitas terjadinya rangkaian suku kata yang bersesuaian, akan dapat dipergunakan untuk membantu *post* proses pengenalan, yaitu untuk menebak suku kata yang seharusnya sehingga menjadi rangkaian dari suku kata yang lain. Dari pengujian dengan sistem pembangun matriks suku kata, yang dilakukan pada dua buah halaman buku sastra Jawa dengan banyak suku kata 587, diperoleh kesimpulan bahwa sistem telah dapat mengenali suku kata dasar dan rangkaian suku kata dasar pembentuk dokumen dengan sangat baik, dengan waktu proses yang relatif sangat cepat.

Kata kunci : *pengenalan citra dokumen, pengenalan aksara, rantai markov, DTMC.*

1. Pendahuluan

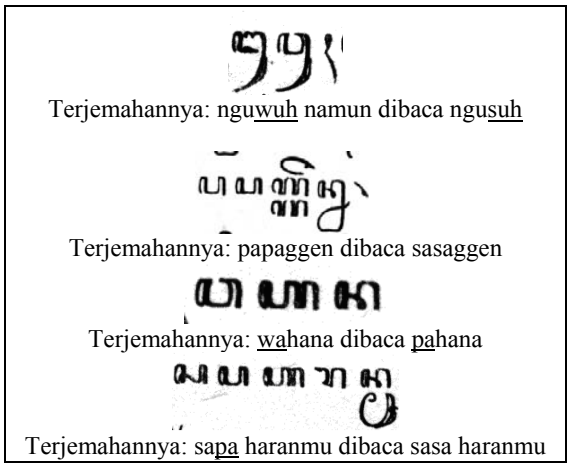
Yogyakarta sebagai salah satu daerah istimewa di Indonesia adalah daerah yang terkenal akan kekayaan sejarah budayanya, termasuk kesenian, dan kesusastraan. Salah satu kekayaan sejarah yang tidak ternilai harganya adalah naskah-naskah Jawa yang masih banyak dijumpai di Yogyakarta, yaitu di Kraton Kasultanan, dan Pura Pakualaman. Apabila naskah-naskah tersebut dapat dikonversikan ke dalam format digital akan banyak manfaat yang dapat diraih, antara lain kualitasnya dapat diperbaiki, keberadaannya dapat diperpanjang, serta dapat dimanipulasi untuk kepentingan penelitian, misalnya untuk diterjemahkan secara otomatis ke dalam tulisan latin tanpa mengubah makna naskah asli.

Persoalan muncul karena naskah kuno di Yogyakarta kebanyakan ditulis dengan menggunakan aksara Jawa, sementara komputer umumnya hanya mengenal dan merepresentasikan aksara Latin. Oleh karena itu, diperlukan suatu perangkat lunak yang mampu mengenali dokumen beraksara Jawa tersebut dan selanjutnya merepresentasikannya dalam komputer. Mengingat tidak semua orang mengenal aksara Jawa, maka akan lebih bermanfaat lagi apabila kemudian naskah beraksara Jawa tersebut dapat direpresentasikan pula dengan aksara Latin tanpa kehilangan maknanya.

Perkembangan ilmu analisis citra dokumen, yaitu analisis pada representasi visual dokumen kertas seperti jurnal, hasil faksimili, surat-surat kantor, lembar isian, dan lain-lain [4], membuka peluang besar untuk dimanfaatkan bagi pelestarian naskah-naskah kuno yang banyak ditemukan di Yogyakarta. O'Gorman dan Kasturi [2] memberikan tahapan-tahapan proses analisis citra dokumen yang dapat dimodifikasi untuk pengenalan citra dokumen sastra Jawa. Dimulai dengan tahap pengambilan data di mana data dari dokumen kertas akan dibaca dengan alat scan optik dan hasilnya disimpan sebagai file citra. Dilanjutkan tahap pengolahan tingkat piksel yang bertujuan untuk menyiapkan dokumen citra, serta membuat fitur perantara untuk membantu mengenali citra. Tahap yang ketiga adalah tahap pengenalan karakter dengan tujuan untuk menerjemahkan sederetan karakter yang memiliki berbagai macam bentuk dan ukuran.

Dalam tahap pengambilan data mungkin sekali terdapat persoalan dalam membaca citra dokumen antara lain adalah kualitas kertas maupun tinta yang sudah berkurang pada dokumen yang akan didigitalisasi, pencahayaan yang kurang pada saat pembacaan dokumen dengan *scanner*, ataupun akibat dari proses persiapan data awal atau *preprocessing*; misalnya binarisasi yang berakibat putusnya citra aksara atau citra suku kata yang semula sambung,; maupun juga adanya kemiripan antara aksara yang satu dengan aksara yang lain. Untuk persoalan yang ditimbulkan dari kemiripan yang sangat besar antara citra yang satu dengan yang lain, sebenarnya dapat diatasi jika dapat ditemukan ciri dari setiap citra dengan baik, sehingga bisa membedakan dengan baik citra yang satu dengan citra lainnya. Namun pencarian ciri atau ekstraksi ciri juga bukan merupakan hal yang mudah.

Paper ini mencoba untuk menyodorkan sebuah gagasan tentang penggunaan *discrete-time Markov chain* (DTMC) dalam *post-processing* atau proses akhir pengenalan citra dokumen teks, yaitu saat di mana sistem sudah dapat mengenali citra dokumen yang diolah, namun hasil dari pengenalannya relatif kurang benar. Gambar 1 berikut ini menyajikan gambaran kesalahan pengenalan yang muncul [6].



Terjemahannya: nguwuh namun dibaca ngusuh

Terjemahannya: papaggen dibaca sasaggen

Terjemahannya: wahana dibaca pahana

Terjemahannya: sapa haranmu dibaca sasa haranmu

Gambar 1. Contoh-contoh kesalahan pengenalan

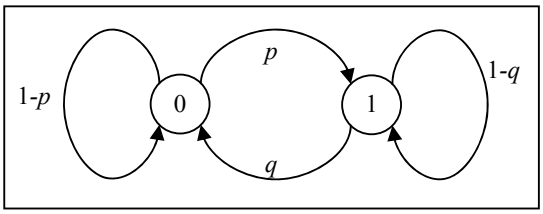
Untuk itu akan dibuat sebuah sistem yang akan dapat melakukan pemetaan suku kata dan rangkaian suku kata beserta probabilitas kemunculan suku-kata tersebut sehingga dapat menjadi alat bantu pemilihan suku kata ketika terjadi keragu-raguan atau kesalahan dalam pengenalan aksara yang ditemukan saat pengenalan citra dokumen teks berlangsung.

2. DTMC *(Discrete-Time Markov Chain)*

DTMC adalah salah satu kelas dalam proses stokastik, dengan sifat bahwa indeknya diskrit, dan ruang statenya juga diskrit. Ruang state adalah himpunan semua nilai-nilai yang mungkin muncul dari pengambilan variabel secara acak. Gambar 2 memberikan contoh representasi sebuah DTMC sederhana, di mana state dilambangkan dengan lingkaran yang dapat diberi label angka 0,1, 2, dst. Probabilitas terjadinya kejadian 1 dari keadaan 0 sebesar *p*, dan dari keadaan 0 ke 0 sebesar *1-p*, karena berlaku sifat bahwa:

$$\sum_j p_{ij} = 1 \qquad (1)$$

Keputusan perubahan dari state satu ke state yang lain pada slot waktu tertentu dibuat secara acak berdasarkan kondisi saat ini.



Gambar 2. Contoh DTMC sederhana

Sebuah rantai Markov adalah suatu urutan dari variabel-variabel acak $\{X_n, n = 0,1,2,...\}$ dengan sifat Markov yaitu bahwa distribusi hasil dari X_{n+1} hanya tergantung pada X_n, atau dengan kata lain bentuk ke depan hanya tergantung pada keadaan sekarang, dan tidak bergantung pada bentuk sebelumnya. Secara formal hal tersebut di atas dinyatakan dengan:

$$P(X_n = i_n | X_0 = i_0, X_1 = i_1, ..., X_{n-1} = i_{n-1}) = P(X_n = i_n | X_{n-1} = i_{n-1}) \qquad (2)$$

di mana i_k adalah state yang dihasilkan dari proses stokastik untuk semua k = 0, 1, …, n [3,5].

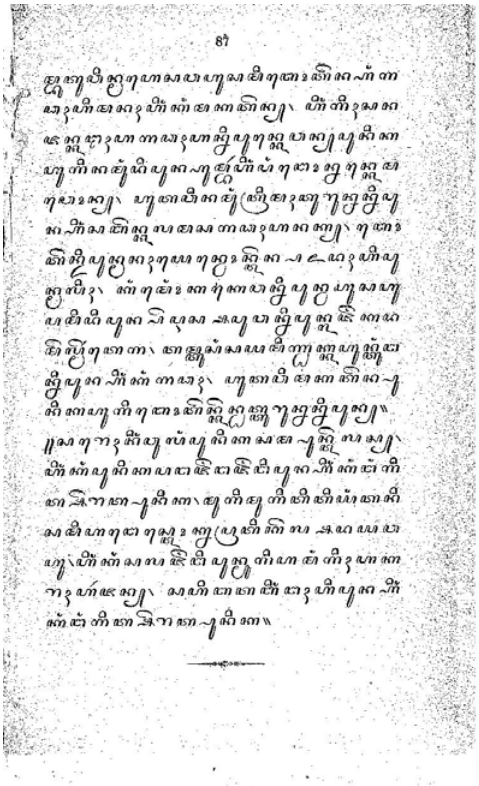
3. Analisis Kebutuhan Sistem

Sangat mungkin sekali, pada saat pembangunan sistem pengenalan dokumen menemui berbagai kendala yang diakibatkan baik oleh kualitas dokumen yang akan dikenali, proses pembacaan dokumen, maupun pada saat pengolahan data citra dokumen. Akibat dari kendala tersebut, maka akan terdapat kesalahan pengenalan. Menurut analisis kemungkinan munculnya kesalahan, maka perbaikan dalam kesalahan dapat dilakukan baik pada level *preprocessing*, dan pengenalan aksara, namun juga dapat dilakukan setelah proses pengenalan selesai.

Setelah proses pengenalan dokumen selesai, maka semestinya akan diperoleh deretan suku-suku kata yang akan membentuk kata-kata penyusun dokumen masukan. Dengan pengandaian tersebut, maka terdapat peluang untuk memperbaiki hasil pengenalan, yaitu dengan cara mencocokkan apakah suku-suku kata yang ditemukan membentuk kata yang benar sesuai kamus. Sebagai contoh studi kasus, dalam penelitian ini dipergunakan data dari hasil *scanner* pada satu halaman dari buku Hamong

Tani [7], yaitu dari halaman 87 seperti terlihat dalam Gambar 3.

Dengan mendasarkan pada hasil terjemahan seorang ahli sastra Jawa di buku Hamong Tani halaman 87, diperoleh himpunan suku kata yang muncul dari buku di halaman tersebut, seperti terlihat dalam Tabel 1.

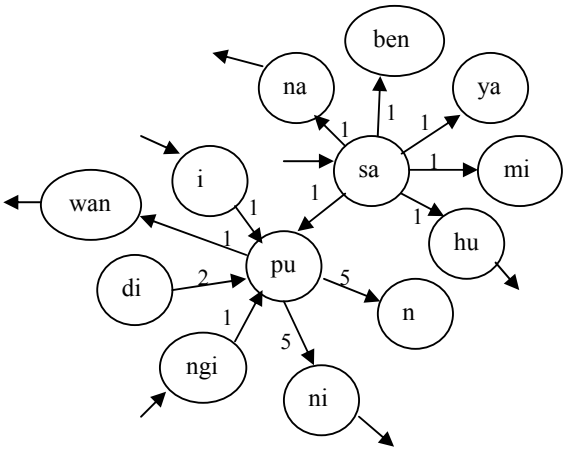


Gambar 3. Dokumen teks Hamong Tani halaman 87.

Tabel 1. Suku kata dasar dari buku Hamong Tani halaman 87

Ba	h	kang	mung	pang	tha
Bar	ha	ke	n	pe	ti
Be	hang	kt	na	pra	to
Bing	har	la	ndha	pu	tri
Bo	he	lang	ndi	ra	tu
Bu	he	li	nga	re	tung
Da	hi	ma	nga	ru	wa
Dha	hing	mang	ngang	s	wi
Di	hu	me	nge	sa	wo
Ga	ja	mer	ngi	sang	wu
Ge	jeng	mi	ni	se	ya
Gi	k	mong	ning	ta	yang
Go	ka	mu	pa	te	ye

Dengan mempergunakan suku kata dasar yang telah ditemukan dalam Tabel 1, maka dapat disusun diagram yang menggambarkan suku kata yang terdapat pada dokumen, hubungan antar suku kata, dan berapa kali terjadi perpindahan dari satu suku kata ke suku kata yang lain. Gambar 4 memperlihatkan sebagian suku kata yang ditemukan, hubungan antar suku kata yang ditemukan dan berapa kali terjadi urutan suku kata sumber ke suku kata tujuan. Sebagai contohnya, untuk suku kata *pu*, ditemukan bahwa suku kata tersebut berhubungan dengan suku kata *i, wan, di, ngi, di, n,* dan *sa.* Dari hasil pembacaan dokumen tesebut juga diperoleh informasi bahwa banyaknya kejadian munculnya suku kata *ni* dari suku kata *pu* sebanyak 5 kali kejadian. Dari informasi ini pula diperoleh probabilitas kemunculan suku kata *ni* setelah suku kata *pu* sebesar 5/11.



Gambar 4. Diagram DTMC suku kata

Dengan melakukan pengamatan model diagram suku kata seperti pada gambar 4, dapat dilihat bahwa model tersebut adalah sebuah model diagram DTMC, di mana satu suku kata yang dituliskan dalam lingkaran kecil adalah state, dan garis panah yang menunjukkan hubungan dari satu suku kata ke suku kata yang lain menunjukkan transisi dari satu state ke state yang lain. Selain itu apabila melihat sifat dari persoalan, yaitu bahwa proses selanjutnya hanya tergantung pada saat ini, waktu proses dan populasinya diskrit, dan relatif homogen dalam prosesnya, maka jelas perkiraan suku kata dan rangkaiannya dapat diselesaikan dengan DTMC.

4. Desain dan Implementasi Sistem

Setelah diagram suku kata seperti terlihat pada Gambar 4 diperoleh, maka langkah awal dalam desain sistem adalah membuat model penyimpanan data suku kata, rangkaian suku kata yang muncul, dan jumlah terjadinya perpindahan dari satu suku kata ke suku kata yang lain. Salah satu cara yang

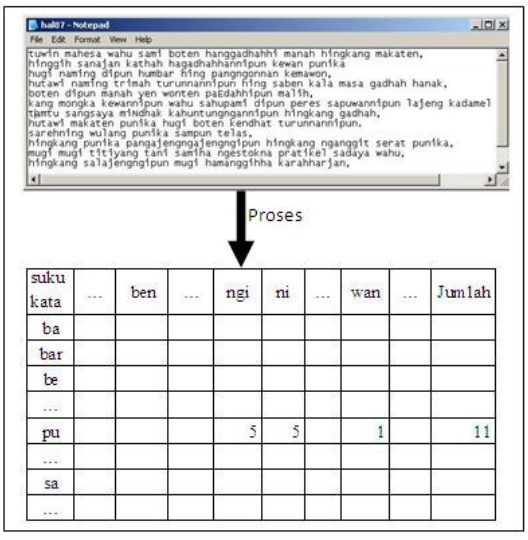
paling mudah dilakukan untuk tahap implementasi adalah dengan membuat matriks dua dimensi, yang menggambarkan suku kata dan hubungan suku kata tersebut dengan suku kata yang lain beserta kemungkinan probabilitas kemunculan suku kata yang menyertai tersebut seperti digambarkan dalam Tabel 2. Matriks dua dimensi seperti digambarkan dalam tabel 2 ini cukup mudah diimplementasikan dengan mempergunakan struktur data array dua dimensi [1].

Tabel 2. Model matriks keterhubungan suku kata

suku kata	ba	be	...	ni	...	yang	ye
Ba	0	0	...	0	...	0	0
Be	0	0	...	0	...	0	0
...	...	...	...	...	...	...	...
Pu	0	0	...	5	...	0	0
...	...	...	...	...	...	...	...
Yang	0	0	...	0	...	0	0
Ye	0	0	...	0	...	0	0

Dengan bermodalkan pada hasil penyelidikan tersebut, maka akan diperoleh semacam panduan ketika terjadi kebingungan dalam mengenali suku kata yang disebabkan karena kerusakan data citra dari suku kata tersebut. Cara yang dapat ditempuh adalah dengan menebak suku kata apa yang dilambangkan oleh suatu citra suku kata, dengan cara melihat probabilitas yang tertinggi yang muncul dari suku kata sebelumnya, karena dari Tabel 2 tersebut, dapat ditelusuri suku-suku kata yang menjadi pasangan suatu suku kata.

Untuk mencapai tujuan dari sistem yang akan dibangun, maka dibuat gambaran umum sistem seperti terlihat pada Gambar 5. Masukan untuk sistem adalah dokumen teks dalam format txt, dan keluarannya akan berupa matriks suku kata yang dihasilkan dari dokumen teks masukan tersebut.



Gambar 5. Diagram konteks sistem

Untuk membangun sistem Pembangun Matriks Suku Kata, maka dibuatlah 5 buah metode sebagai berikut:

4.1. Metode **baca**(File file)
Fungsi: membaca file dokumen masukan dan memecah kalimat menjadi kata-kata.
Input: variabel File bertipe file yang menampung data masukan
Output: -
Algoritma:

1. Baca file data dengan mempergunakan klas *Scanner* dan simpan dalam variabel scan
2. Selama (scan.hasNext()) lakukan langkah 3 sampai dengan 4
3. simpan hasil perintah scan.next().split("[^a-zA-Z]+") dalam variabel kata
4. mulai dari i=0 sampai panjang kata, panggil metode getSukuKata(kata[i]) untuk mendapatkan suku-suku kata dari kata yang telah ditemukan.

4.2. Metode **getSukuKata**(String kata)
Fungsi: memecah sebuah kata menjadi suku-suku kata
Input: variabel string yang memuat kata yang akan dipecahkan.
Output: suku-suku kata.
Algoritma:

1. Inisialisasi variabel awal = 0 untuk menampung indek suku kata awal
2. Buat variabel akhir untuk menampung indek suku kata terakhir dari suku kata yang ditemukan.
3. Buat variabel string sou, dan dest masing-masing untuk menampung suku kata yang menjadi state sumber, dan state tujuan.
4. Panggil metode searchEnd(awal, kata) untuk menari indeks huruf yang terakhir dari suku kata pertama yang ditemukan, tampung dalam akhir.
5. sou = kata.substring(awal, akhir + 1)
6. Set awal dari suku kedua = akhir + 1
7. Selama awal kurang dari panjang kata lakukan langkah 8, jika tidak selesai
8. Cari indeks huruf yang terakhir dari suku kata pertama yang ditemukan, tampung dalam akhir.
9. dest = kata.substring(awal, akhir + 1)
10. panggil metode checkdanMasukSuku(sou, dest) untuk ...
11. awal = akhir + 1
12. set sou yang baru = dest

4.3. Metode **searchEnd**(int awal, String kata)
Fungsi: mencari indek dari akhir sebuah suku kata.
Catatan: macam kombinasi suku misalnya *tu, win, mung*, yang masing-masing indeknya

berakhir di huruf vokal, atau huruf vokal ditambah 1, atau huruf vokal ditambah 2.

Input: indek dari awal suku kata dan kata yang sedang diproses untuk dipecah suku katanya

Output: indek dari akhir suku kata dengan awalan indek di variabel awal

Algoritma:

1. Inisialisasi variabel vokal = -1 untuk menampung jumlah indek dari huruf vokal
2. Mulai dari awal sampai dengan panjang kata, temukan huruf vokal [aAiIuUeEoO], simpan indek dari huruf vokal dalam variabel vokal
3. Jika vokal < 0, maka return kata.length() – 1 sebagai nilai akhir, jika tidak lanjutkan langkah 4
4. Cek apakah ((kata.charAt(vokal + 2) + "").matches("[aAiIuUeEoO]")), Jika ya, maka return vokal sebagai nilai akhir, program selesai
5. Jika tidak, cek apakah ((kata.charAt(vokal + 3) + "").matches("[aAiIuUeEoO]")).
6. Jika ya, panggil metode (checkHubungan(kata.substring(vokal + 1, vokal + 3))) untuk mengetahui apakah dua huruf mati yang berdampingan setelah huruf vokal adalah benar.
    a. Jika ya, maka return vokal sebagai nilai akhir
    b. Jika tidak, maka return vokal+1 sebagai nilai akhir
7. Jika ((kata.charAt(vokal + 3) + "").matches("[aAiIuUeEoO]")) = false, panggil metode (checkHubungan(kata.substring(vokal + 1, vokal + 3))) untuk melihat keabsahan dua huruf mati yang berdampingan. Jika ya, maka return vokal+2 sebagai nilai akhir, tetapi jika tidak maka return vokal+1 sebagai nilai akhir.

4.4. Metode **checkHubungan**(String dou)

Fungsi: melakukan pengujian apakah dua buah huruf mati yang berdampingan dibenarkan atau tidak.

Input: variabel string yang memuat dua huruf mati berdampingan (dou).

Output: true atau false.

Algoritma:

1. Buat variabel doubleHurufMati untuk menampung kombinasi huruf mati yang diperbolehkan. Dalam program ini doubleHurufMati = {"tr", "pr", "dh", "kh","dy", "ny", "ng"}.
2. Untuk setiap dua huruf mati berdampingan yang ditemukan, jika sesuai dengan yang ada dalam doubleHurufMati, return true, jika tidak return false.

4.5. Metode **checkdanMasukSuku**(String sou, String dest)

Fungsi: melakukan pengujian apakah dua suku kata masukan adalah suku kata baru dan menambahkan jumlah hubungan dari dua suku kata yang berhubungan

Input: dua suku kata sou yang merupakan suku kata di state sumber, dan dest yang merupakan suku kata di state tujuan

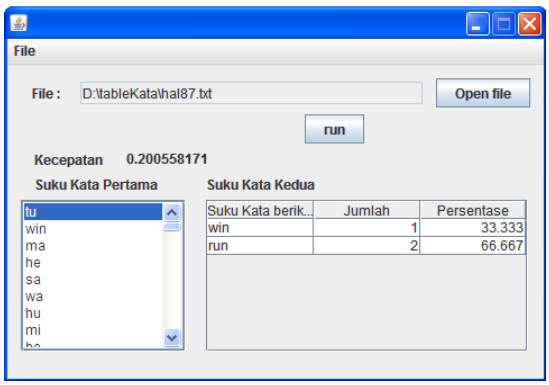
Output: update matrik penyimpan suku kata

Algoritma:

1. cek apakah sou dan dest sudah terdapat di dalam matriks suku kata penyimpan.
2. Jika tidak, maka tambahkan sou atau dest ke dalam matriks suku kata.
3. Tambahkan jumlahan dalam matriks di indeks [sou][dest] dengan 1 karena terdapat tambahan transisi state dari sou ke dest

5. Analisis hasil pengujian sistem

Setelah sistem selesai dikembangkan, maka dilakukan pengujian sistem, untuk menganalisis kebenaran hasil maupun kinerja sistem. Dengan mempergunakan masukan berupa teks yang sudah merupakan terjemahan manual dari buku Hamong Tani halaman 87, diperoleh tampilan keluaran seperti terlihat pada Gambar 5.



Gambar 5. Contoh tampilan keluaran dari sistem

Dalam contoh pada Gambar 5 di atas, dapat dilihat bahwa lama proses penyusunan matriks suku kata untuk dokumen dari file hal87.txt sebesar 0.20 detik. Pada gambar 5 tersebut juga terlihat suku kata berikutnya dari suku kata tu, yaitu suku kata win dengan probabilitas kemunculan 33.33%, dan suku kata run dengan probabilitas 66.67%, yang artinya dari dokumen masukan yang diproses dalam sistem, suku kata yang mempunyai kemunculan terbesar setelah suku tu, adalah suku kata run.

Dari analisis pada perolehan suku kata dasar yang dihasilkan oleh sistem, ternyata dari 95 suku kata yang seharusnya ada, hanya diperoleh 94 suku kata dasar dengan dua suku kata yang tidak benar,

sehingga dapat diperoleh prosentase kebenaran suku kata sebesar 96.84%. Sedangkan dari analisis pada rangkaian suku kata terdekat yang ditemukan oleh sistem, ternyata diperoleh sebanyak 155 rangkaian, dan jumlah tersebut sesuai dengan rangkaian suku kata pada dokumen aslinya. Dari sini dapat disimpulkan prosentase kebenaran dalam penemuan rangkaian suku kata adalah 100%.

Hal yang sama kemudian dilakukan untuk dokumen yang lain, yaitu dari halaman 86 buku Hamong Tani. Dari hasil pengujian pada dua buah file dokumen dengan mempergunakan sistem yang dibangun, diperoleh rangkuman hasil analisis seperti terlihat dalam Tabel 3.

Tabel 3. Hasil pengamatan pengujian

Nama File (*.txt)	Jumlah Suku Kata	Prosentase Kebenaran		Lama Waktu Proses (detik)
		Pengenalan Suku Kata Dasar	Rangkaian Suku Kata	
hal87	290	96.84	100	0.20
hal86	297	98.91	100	0.05
Rata-rata		97.88	100	0.125

6. Kesimpulan Dan Saran

Telah dapat dihasilkan sebuah matriks suku kata dari dokumen teks dalam sastra Jawa, yang merupakan implementasi dari konsep DTMC. Sistem aplikasi yang dibangun mempergunakan bahasa Java dalam implementasinya. Dengan mempergunakan data teks dari dua buah halaman buku Hamong Tani, diperoleh hasil bahwa sistem telah dapat mengenal suku kata dasar yang membentuk kedua dokumen dan mengenal rangkaian suku kata dari dokumen dengan sangat baik. Sedangkan dari sisi waktu yang diperlukan untuk membuat matriks suku kata dari dua buah file yang diuji yang berisikan rata-rata jumlah suku kata 293, diperoleh rata-rata lama proses sebesar 0.125 detik, atau dengan kata lain relatif sangat cepat.

Matriks suku kata yang telah terbentuk dari sistem, dapat dipadukan dengan sistem pengenalan citra dokumen, yaitu pada saat terdapat keraguan dalam mengenali citra suku kata, lakukan proses pencarian suku kata berikutnya dari matriks suku kata. Dari matriks akan diperoleh data suku kata yang menjadi pasangan dari suku kata sebelumnya yang sudah dikenali, dengan memilih suku kata yang mempunyai probabilitas kemunculan terbesar.

7. Ucapan Terimakasih

Penulis mengucapkan terimakasih kepada Audris Evan Utomo yang telah membantu dalam implementasi sistem, dan kepada Bapak Ir. Ig. Aris Dwiatmaka, M.Sc., atas waktu yang disediakan kepada penulis untuk berkonsultasi mengenai kaidah matematika dari DTMC.

Daftar Pustaka:

[1] Lafore, R., 2003, *Data Structures & Algorithms in Java. Second Edition*, USA, Sams Publishing.

[2] O'Gorman, L., dan Kasturi, R., 1997, *Executive briefing: documen image analysis*, USA, IEEE Computer Society Press.

[3] Ross, SM., 2003, *Introduction to Probability Models. Eight Edition*, USA, Academic Press.

[4] Srihari, S.N., Lam, S.W., Govindaraju, V., Srihari, R.K., dan Hull, J.J., 1986, *Document Image Understanding,* New York, CEDAR.

[5] Taylor, HM., dan Karlin, S., 1998, *An Introduction to Stochastic Modeling, 3RD Edition*, USA, Academic Press.

[6] Widiarti, AR. dan Harjoko, A., 2006, *Pengenalan Citra Dokumen Sastra Jawa (Studi Kasus Pada Buku Sastra Jawa: Menak Sorangan I dan Panji Sekar). SAINS dan SIBERNETIKA.,* Vol**:** 19, No.: 3, Hal : 269-282.

[7] Winter, 1876, *Hamong Tani,* Batavia.