Penerapan *Rule Based* dalam Membangun Transliterator Jawatex

Ema Utami[1], Jazi Eko Istiyanto[2], Sri Hartati[3], Marsono[4], Ahmad Ashari[5]

[1]Jurusan Sistem Informasi STMIK AMIKOM Yogyakarta
E-mail : *emma@nrar.net*
[2,3,5]Program Studi Magister Ilmu Komputer Sekolah Pascasarjana UGM
E-mail : *jazi@ugm.ac.id*, *shartati@ugm.ac.id*, *ashari@ugm.ac.id*
[4] Jurusan Sastra Nusantara Fakultas Ilmu Budaya UGM
E-mail : *nusantara@ugm.ac.id*

Intisari

Model transliterasi yang dibutuhkan saat ini tidak berfokus pada pembuatan font tetapi lebih pada transliterasi dokumen teks Latin. Untuk itu perlu dibangun model transliterasi yang diberi nama JawaTeX untuk mengalihaksarakan dokumen teks Latin ke aksara Jawa. Dokumen teks Latin diparsing untuk menentukan daftar pola pemenggalan string Latin sebagai token. Metode parser yang digunakan pada penelitian ini adalah *The Context Free Recursive Descent Parser*. Pengolahan dokumen teks Latin menjadi daftar pola pemenggalan string Latin menggunakan metode *rule based*, sedangkan proses pemadanan masing-masing pola pemenggalan string Latin dalam bentuk pemetaan dalam format LaTeX menggunakan metode *Pattern Matching*. Dengan metode *rule based*, maka masalah-masalah yang belum tertangani dalam penelitian sebelumnya dapat diatasi dengan menggunakan aturan-aturan tertentu. Model transliterasi yang dibangun didukung oleh aturan produksi penelusuran pola pemenggalan string Latin, model-model pola pemenggalan string Latin, aturan produksi untuk pemetaan aksara Latin-Jawa, model-model pola pemetaan, *style* atau makro LaTeX, Metafont aksara Jawa, serta paket program JawaTeX yang terdiri dari program parsing dan *style* LaTeX yang digunakan untuk mengkodekan format LaTeX. Model transliterasi yang dibangun dilengkapi pengkoreksian kesalahan pengetikan tulisan (*spell checker*). Paket program JawaTeX terdiri dari program pengecekan dan pemenggalan string Latin untuk penelusuran pola string Latin dan *style* LaTeX yang digunakan untuk mengkodekan format LaTeX.

Beberapa hasil pengujian membuktikan bahwa jika pengguna bisa menulis dengan benar setiap kata atau istilah termasuk kata serapan sesuai dengan pengucapan aslinya dan menuliskan atau menata kembali ejaan Latin dalam teks sumber maka terlihat bahwa model transliterasi dokumen teks Latin ke aksara Jawa yang terbentuk bisa digunakan untuk mengalihaksarakan dokumen teks Latin ke tulisan aksara Jawa. Konsep pemenggalan dokumen teks dan pengalihaksaraan yang dibangun dalam tulisan ini dapat dijadikan dasar untuk dikembangkan dalam kasus yang lain. Untuk penelitian berikutnya, pemecahan penulisan aksara Jawa secara baik masih perlu dipikirkan. Penulisan aksara Jawa terkadang melewati batas kanan dari dokumen karena dalam penulisan aksara Jawa tidak mengenal spasi.

***Kata kunci**: pemenggalan string Latin, rule based, Pattern Matching, The Context-Free Recursive-Descent Parser, JawaTeX, LaTeX*

Abstract

The transliteration model needed recently does not focus on the font making, yet more to the transliteration of Latin text document. Therefore it is necessary to build the transliteration model named JawaTeX to transliterate the Latin text document to Javanese characters. The Latin text document is parsed to determine the list of Latin string split patterns as token. The parser method used in this research is The Context Free Recursive Descent Parser. The Latin text document processing becomes the list of the Latin string split pattern by using rule-based method, whereas the matching process of each Latin string split pattern in maping form of LaTeX uses Pattern Matching method. With the rule-based method, the unsolved problems of

the previous researhes can be overcomed by using certain methods. The established transliteration model is suppported by the production rule of browsing the Latin string split pattern, the models of the Latin string split pattern, the production rule for the Latin-Javanese character mapping, the models of syntax coding pattern, style or macro LaTeX, Javanese character metafont, and JawaTeX program consisting of parsing program and LaTeX style used to code LaTeX syntax. The established transliteration model is completed with the spelling checker to correct the mistake of letter typing. JawaTeX program consists of checking program and Latin string split to browse the Latin string pattern and LaTeX style which are used to code LaTeX syntax.

Several testing results prove that if the user can write every word correctly including absorption suitable with the original pronunciation and write or re-arrange the Latin spelling in the source text, so the transliteration model of the Latin text document to Javanese character formed can be used to transliterate the Latin text document to Javanese character writing. The concept of the text document split and the established transliteration in this article can be used as a basis to develop other cases. For the next research, the Javanese character split writing in good form still needs to be developed. The Javanese character writing sometimes cannot be justified allignment since the Javanese character writing does not recognize space between words.

***Keywords**: Latin string split pattern, rule based, Pattern Matching, The Context-Free Recursive-Descent Parser, JawaTeX, LaTeX*

1. Pendahuluan

Di beberapa negara yang memiliki keragaman budaya, khususnya yang berhubungan dengan aksara telah terdapat penelitian untuk mengembangkan budaya tersebut melalui komputerisasi aksara. Mesin-mesin alihaksara dari karakter latin ke aksara Jawa merupakan salah satu bidang riset dari komputasi linguistik. Penelitian mengenai bidang komputasi linguistik itu sendiri di Indonesia masih kurang, dengan kata lain masih tertinggal dari penelitian-penelitian sejenis di luar negeri (Utami dkk, 2008b). Pada saat ini, telah terdapat dua buah font berbentuk *true type font* untuk aksara Jawa. Fokus penelitian-penelitian yang sudah ada adalah membuat font yang digunakan untuk perangkat lunak pengolah kata bukan teks editor. Tools yang digunakan untuk membuat font dalam format digital merupakan tools yang bersifat komersial. Pada penelitian-penelitian sebelumnya untuk menuliskan aksara Jawa menggunakan font tersebut pengguna harus menguasai cara penulisan aksara Jawa. Pada penelitian-penelitian sebelumnya penulisan aksara Jawa dihasilkan setelah mengetikkan simbol-simbol tertentu yang mewakili setiap font yang dikehendaki, sehingga pengguna dituntut untuk menghapal simbol-simbol latin yang rumit (Utami dkk, 2008a). Misal untuk menampilkan tulisan dalam aksara Jawa *aksara jawi punika teksih kathah kekiranganipun* maka penulisan huruf latinnya *?aksrjwipunika [tkSih kqh kekirqnNipun*\.

Font yang dibuat pada penelitian-penelitian sebelumnya lebih fokus pada penulisan yang dapat dialihaksaran menggunakan pedoman penulisan aksara Jawa yang selama ini telah ada. Dalam kenyataannya tidak mungkin membatasi string latin yang akan dialihaksarakan menjadi aksara Jawa. Misalnya masyarakat mancanegara yang perbendaharaan katanya lebih kompleks, diantaranya tidak ada pembatasan banyaknya konsonan bertingkat (berurutan), sehingga setiap karakter input bisa bertemu atau diikuti dengan sembarang karakter yang bahkan tidak ada padanan abjadnya dalam aksara Jawa. Beberapa masalah belum dapat diselesaikan dengan penelitian-penelitian sebelumnya (Utami dkk, 2008b), diantaranya: penulisan karakter yang belum mempunyai padanan dalam aksara Jawa, penentuan pola suku kata dari string latin yang memuat kombinasi karakter yang bisa jadi tidak mungkin terjadi pada suku kata dalam aksara Jawa, penggunaan diftong, penulisan konsonan bertingkat lebih

dari tiga, pengunaan bilangan romawi, serta masalah ambiguitas yang diakibatkan penggunaan spasi, tanda penghubung dan titik

Kajian ilmiah mengenai transliterasi karakter latin ke aksara Jawa dengan Metafont dan LaTeX belum pernah dijumpai (Utami dkk, 2008b). Hal ini tampak dengan belum diperolehnya naskah publikasi yang khusus membahas topik tersebut. Untuk itu, penelitian ini merupakan penelitian awal dalam menuliskan aksara Jawa dalam media digital dengan Metafont dan LaTeX. Dengan demikian, penelitian ini benar-benar merupakan penelitian baru di bidang transliterasi karakter latin ke dalam aksara Jawa menggunakan LaTeX (Utami dkk, 2008b). Dengan mengetahui latar belakang permasalahan dan kekurangan penelitian sebelumnya, maka perlu dilakukan penelitian yang tidak berfokus pada penulisan font Jawa tetapi lebih pada transliterasi dokumen teks Latin yang dapat digunakan tanpa harus memahami aturan penulisan aksara Jawa dan ruang lingkupnya tidak terbatas dalam masyarakat Jawa tetapi masyarakat lebih luas yang menggunakan karakter latin dalam komunikasinya.

2. Tinjauan teori

Inti dari sistem Pemrosesan Bahasa Alami adalah parser. Ada 3 macam parser, yaitu *The State-Machine-Parse*r, *The Context-Free Recursive-Descent Parser* dan *The Noise Diposal Parser*. Pada *The State-Machine-Parser*, proses dilakukan dengan mengikuti suatu aturan tertentu yang mengacu pada keadaan terkini. Pada metode *The Context-Free Recursive-Descent Parser*, suatu kalimat disusun oleh beberapa item, dimana item-item tersebut juga disusun oleh item-item yang lain, sedemikian sehingga dapat dipecah dalam bentuk item atomik (Schildt, 1987). Aturan yang mengatur bagaimana setiap bagian dapat dibangun disebut sebagai aturan produksi dari grammar. Sedangkan dalam *The Noise Diposal Parser* dilakukan dengan menghilangkan kata-kata yang tidak perlu, semacam transkripsi.

Metode parsing yang digunakan pada penelitian ini adalah *The Context-Free Recursive-Descent Parser* (Utami dkk, 2008a), (Utami dkk, 2008b). Dalam penelitian ini mengikuti aturan produksi untuk pengecekan dan pemecahan string latin dengan membagi string latin sampai dengan nilai terkecil (token) berupa pola pemenggalan suku kata. Untuk memenggal suatu string latin, walaupun pengecekan karakter latin sudah sampai posisi ke-*n*, kadangkala harus mengecek kembali (penelusuran mundur) kondisi karakter-karakter sebelumnya lagi, sehingga dapat memenuhi syarat untuk diambil sebuah pola pemenggalan string latin. Kadangkala parsing tidak hanya dilakukan pada string sumber tetapi hasil pengolahan string sumber (misalkan penanganan angka romawi). Karakter dalam string masukan tidak hanya dibaca/diperiksa/dicek tetapi juga dimodifikasi/diubah/dikonversi dan disisipi/ditambahkan karakter lain supaya memenuhi syarat format penulisan, bahkan harus dicari nilai desimalnya dahulu jika karakter tersebut digunakan sebagai angka romawi. Jika elemen pertama adalah konsonan dan elemen berikutnya juga konsonan maka terjadi pengecekan berulang-ulang untuk menentukan pola pemenggalan string latinnya. Setiap menjumpai karakter konsonan maka banyaknya konsonan dan konsonan yang diperoleh (setelah dilakukan penyesuaian jika diperlukan) selalu dicatat. Banyaknya perulangan tidak dapat diketahui secara pasti tetapi yang diketahui adalah kapan perulangan dihentikan, yaitu jika setelah karakter konsonan dijumpai karakter selain konsonan. Untuk menentukan karakter awal pada penelusuran pemenggalan string latin berikutnya, ada dua kemungkinan tergantung kondisi karakter-karakter sebelum dan sesudahnya, yaitu:

1. Karakter yang tersisa (sebuah karakter atau beberapa karakter yang sudah diuji pada penelusuran pola pemenggalan string latin sebelumnya tetapi tidak digunakan untuk membangun pola pemenggalan string latin) dijadikan karakter awal untuk penelusuran pemenggalan string latin berikutnya.

2. Karakter terakhir yang digunakan untuk membangun pola pemenggalan string latin sebelumnya dijadikan karakter awal untuk penelusuran pemenggalan string latin berikutnya.

String matching adalah bagaimana menemukan semua kemungkinan kemunculan suatu string *x* dengan panjang *m* yang disebut dengan *pattern* dalam string lainnya *t* dengan panjang *n* yang disebut *teks* (Apostolico, 1997). Pencocokan string merupakan proses untuk menemukan pola susunan karakter string dalam string lain atau bagian dari isi dokumen teks (Nori, 1988); (Black, 2007). Pencocokan string secara garis besar dapat dibedakan menjadi dua yaitu pencocokan string secara eksak atau sama persis (*exact string matching*) dan pencocokan string berdasarkan kemiripan (*inexact string matching* atau *fuzzy string matching*) (Gusfield, 1997). Pencocokan string berdasarkan kemiripan masih dapat dibedakan menjadi dua yaitu berdasarkan kemiripan penulisan atau pola huruf (*approximate string matching*) (French, 1997) dan kemiripan ucapan (*phonetic string matching*) (Gusfield, 1997). Pada pencarian string melibatkan dua hal, yakni teks (string dengan panjang *n* karakter) dan *pattern* (string dengan panjang *m* karakter ($m<n$) yang akan dicari di dalam teks) (Muhammad, 2006).

Metode Komputasi Linguistik pada penelitian ini menggunakan metode *Pattern Matching* dengan beberapa pertimbangan berikut (Utami dkk, 2008a), (Utami dkk, 2008b):

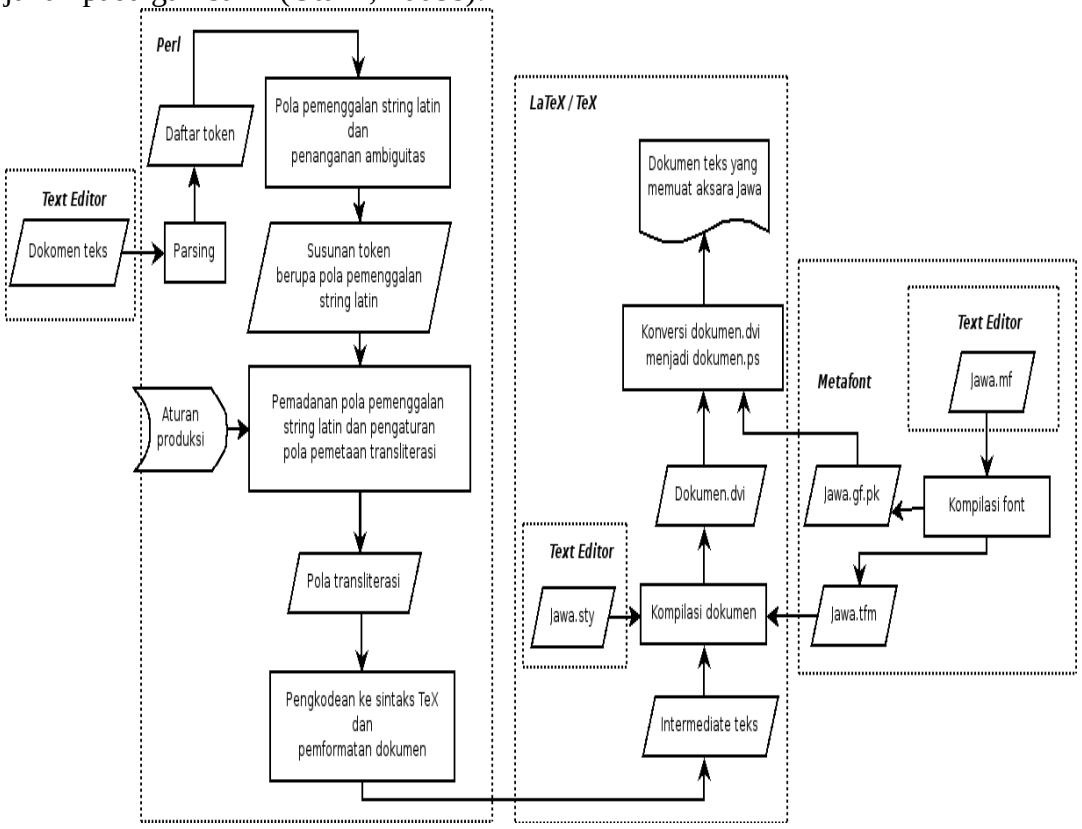
1. Karena sudah ditemukan atau terdefinisi 177 bentuk/model pola pemenggalan string latin sebagai token sehingga penelitian ini menggunakan *Pattern Matching*. *Pattern Matching* digunakan untuk mencari model pola pemenggalan string latin yang relevan dari setiap pola pemenggalan string latin yang telah didapatkan dari analisa sintaks. Setiap model mempunyai aturan produksi untuk alihaksara/pemetaan sendiri-sendiri. Aturan produksi untuk alihaksara yang sesuai dengan model tersebut digunakan untuk mengalihaksarakan pola pemenggalan string latin ke aksara-aksara Jawa dengan posisi yang tepat.
2. Pada Teori Bahasa Formal, setiap bahasa memiliki suatu aturan tata bahasa yang baku dan konsisten. Dalam kenyataannya tidak mungkin membatasi string yang akan dialihaksarakan menjadi aksara Jawa. Misalnya masyarakat mancanegara yang perbendaharaan katanya lebih kompleks. Sehingga setiap karakter input bisa bertemu atau diikuti dengan sembarang karakter yang bahkan tidak ada padanan abjadnya dalam aksara Jawa. Karena inputnya bisa berupa rangkaian string latin yang memuat kombinasi huruf bahkan kombinasi karakter yang bisa jadi tidak mungkin terjadi pada penulisan aksara Jawa (yang sebelumnya tidak bisa dialihaksarakan ke aksara Jawa). Diantaranya tidak ada pembatasan banyaknya konsonan bertingkat (berurutan), misalkan: *Arnold Schwarzenegger*, *mysql*, *postgresql*. Akibatnya tata bahasa yang digunakan seringkali tidak konsisten karena pedoman penulisan aksara Jawa yang ada tidak menjamin suatu string latin bisa dialihaksarakan.
3. Menurut Kamus Besar Bahasa Indonesia, suku kata adalah struktur yang terjadi dari satu atau urutan fonem yang merupakan bagian kata. Setiap suku kata ditandai dengan sebuah vokal (termasuk diftong). Saat ini definisi di atas tidak sepenuhnya benar bila diterapkan dalam sistem transliterasi. String masukan bisa terdiri huruf mati semua, misal string *mysql*, tidak salah karena tidak ada ketentuan urutan pemakaian fonem (huruf mati dan huruf hidup). Jadi ruang lingkupnya tidak terbatas dalam masyarakat Jawa tetapi masyarakat lebih luas yang menggunakan karakter latin dalam komunikasinya.
4. Tidak ada penghapusan atau penghilangan karakter dari string masukan. String latin *psychology* tidak akan diubah ejaannya menjadi *psikologi*, sehingga tidak dilakukan pengubahan lafal bunyi unsur bahasa yang bersangkutan. String latin dialihaksarakan apa adanya tanpa melalui transkripsi.
5. Penggunaan sebagian huruf latin dalam string tidak hanya sebagai konsonan saja tetapi juga berfungsi sebagai angka romawi. Dilakukan pengecekan kelegalan angka romawi yang akan ditransliterasi ke aksara wilangan. Format angka romawi yang tidak valid tidak akan dialihaksarakan. Sedangkan penanganan angka romawi yang valid dilakukan dengan menentukan nilai desimal dari rangkaian angka romawi terlebih dahulu, kemudian nilai desimal per karakter angka penyusunnya dialihaksarakan ke rangkaian aksara wilangan. Jadi token tidak hanya berupa

pemenggalan dari karakter atau rangkaian karakter dalam string sumber tetapi juga pemenggalan dari hasil pengolahan lebih lanjut dari karakter atau rangkaian karakter dalam string sumber

6. Satu karakter dalam string masukan, misalkan konsonan bisa mempunyai lebih satu kemungkinan padanan abjad dalam aksara Jawa tergantung karakter yang menyertainya, demikian juga tanda titik mempunyai kemungkinan sebagai penanda akhir, penanda desimal atau penanda singkatan, sehingga akan mempengaruhi bagaimana karakter-karakter tersebut membentuk kombinasi karakter sebagai pola pemenggalan string latin, melakukan modifikasi terhadap karakter-karakter yang lain dan juga merepresentasikan karakter-karakter apa yang menjadi penentu pemenggalan dari suatu rangkaian karakter latin.
7. Tidak ada string masukan yang ditolak atau tidak dikenali untuk dialihaksarakan.
8. String yang akan ditransliterasikan adalah rangkaian karakter ASCII yang berupa dokumen teks (bukan berupa dokumen grafis) sehingga tidak perlu diadakan pelatihan atau pembelajaran untuk mengenali bentuk suatu karakter.
9. Tidak ada string masukan yang keliru secara makna. Pembetulan atau pengkoreksian karakter dan penambahan atau penyisipan karakter hanya dilakukan pada tataran format penulisan sehingga hanya merubah bentuk bukan makna.

3. Konsep Transliterator JawaTeX

Transliterasi dokumen teks dalam karakter latin ke aksara Jawa dengan LaTeX ini dilakukan dengan mempergunakan tahapan proses transliterasi yang skemanya dapat ditunjukan pada gambar 1 (Utami, 2008b):



Gambar 1: Bagan proses transliterasi dokumen teks dalam karakter latin ke aksara Jawa

Dokumen teks ditulis dengan karakter latin menggunakan *text editor*. Sebelum penelusuran pola pemenggalan string latin dilakukan, harus dipastikan status setiap kata. Pengecekan kemungkinan kesalahan yang terjadi pada saat pengetikan dilakukan dengan mencocokan setiap kata dari teks sumber dengan daftar kata dalam kamus. Pengecekan ejaan kata hanya dilakukan pada kata-kata di dalam kamus yang mirip dengan kata yang sedang

dicek tersebut digunakan algoritma pencarian. Pengkoreksian kesalahan pengejaan kata menggunakan dua daftar kosakata bahasa Inggris (meliputi 29759 kata) dan bahasa Indonesia (meliputi 7962 kata) (Utami dkk, 2009a). Fasilitas *spell checker* ini dibuat untuk melakukan pencarian kemiripan kata untuk mengkoreksi kesalahan pengejaan kata.

Parsing dilakukan menggunakan algoritma *The Context-Free Recursive-Descent Parser* untuk mendapatkan token-token dari rangkaian karakter yang membentuk teks dibaca dari kiri ke kanan dalam dokumen teks (Utami dkk, 2008a). Untuk memenggal suatu string latin, penelusuran tidak hanya bergerak maju tetapi juga bergerak mundur. Pada penelitian ini untuk memenggal suatu string latin, walaupun pengecekan karakter latin sudah sampai posisi ke-*n* tetapi untuk mendapat pemenggalan yang benar, kadangkala harus mengecek kembali kondisi karakter-karakter sebelumnya lagi (penelusuran mundur) sehingga dapat memenuhi syarat untuk diambil sebuah pola pemenggalan string latin, disamping itu untuk menentukan karakter awal pada penelusuran pemenggalan string latin berikutnya, terdapat beberapa cara tergantung keberadaan karakter-karakter disekitarnya. Kadangkala parsing juga tidak hanya dilakukan pada string sumber tetapi hasil pengolahan string sumber.

Proses parsing menghasilkan daftar token berupa urutan karakter-karakter penyusun rangkaian string dalam dokumen teks. Daftar token dipilah-pilah menjadi penggalan-penggalan string latin pembentuk string (rangkaian karakter latin). Pemenggalan string latin tersebut mengakomodasi penanganan spasi supaya terhindar dari masalah ambiguitas, mengingat aksara Jawa tidak mengakomodasikan penempatan spasi untuk memenggal kata karena aksara Jawa tidak mengenal spasi (Utami dkk, 2008a). Proses pemenggalan string latin ini menggunakan konteks aturan produksi. Berikut ini perumusan aturan produksi untuk pemenggalan string Latin teks sumber secara garis besar:

1. Pecah teks sumber menjadi perbaris.
2. Pada awal masing-masing baris ditambahkan tanda penggal.
3. Pecah teks sumber dari perbaris menjadi perkarakter.
4. Jika diantara huruf vokal hanya terdapat 1 konsonan maka penggal sebelum konsonan.
5. Jika konsonan itu adalah *r, h, ng* maka penggal setelah konsonan.
6. Jika diantara huruf vokal hanya terdapat 2 konsonan dan bukan konsonan yang berderet maka penggal sebelum konsonan.
7. Jika konsonan pertama adalah *r, h, ng* maka penggal setelah konsonan pertama.
8. Jika konsonan lebih dari dua maka dihitung berapa panjang konsonan.
9. Jika konsonan berjumlah 3 atau lebih maka dicek kembali apakah mengandung konsonan berderet.
10. Jika terdapat berderet maka penggal seperti aturan 2 konsonan.
11. Jika tidak terdapat berderet maka dicek konsonan yang ke 3.
12. Jika konsonan ke-3 adalah *r, y* bukan *ny* maka penggal sebelum konsonan pertama.
13. Jika konsonan ke-3 bukan *r, y* maka cek konsonan ke 2 apakah *h, f, v, p, s, ks*:
14. Jika benar maka penggal sebelum konsonan pertama.
15. Jika salah maka penggal setelah konsonan pertama.
16. Gabung kembali karakter menjadi text sumber.
17. Text sumber dipecah kembali untuk proses validasi
18. Jika ditemukan tanda penggal ganda maka hilangkan satu.
19. Jika ditemukan simbol atau angka maka diapit penggal.
20. Jika ditemukan titik maka dicek
21. Jika sebelum karakter yang diikuti titik adalah spasi atau karakter pertama maka dipastikan format konsonan.
22. Jika singkatan adalah huruf spesial maka tetap di kapitalkan selainnya dikecilkan.
23. Jika ditemukan spasi maka tentukan kondisinya.
24. Jika spasi diapit oleh konsonan rangkap (contoh: *g_h, n_g*) maka spasi di biarkan.
25. Jika sebelum spasi adalah 2 konsonan maka spasi dipenggal, jika tidak maka spasi dihilangkan.
26. Jika ditemukan abjad spesial yang berdiri sendiri maka karakter dikecilkan.
27. Jika masih terdapat huruf *q*, *x* maka dirubah menjadi *kh*, *ks*.

28. Karakter disusun kembali menjadi teks sumber disimpan kedalam file ekstensi *.jw*.
29. Proses dilanjutkan.

Proses akan dilanjutkan apabila aturan produksi dipenuhi. Pemenggalan ini akan menentukan struktur dari pola pemenggalan string latin yang didapat. Jadi harus ditentukan karakter mana yang merupakan aksara pokok, karakter mana yang menyatakan *sandhangan* dan lain sebagainya.

Struktur pola pemenggalan string latin yang telah diperoleh selanjutnya menggunakan algoritma *Pattern Matching* dipadankan dengan aturan produksi untuk mendapatkan pola penulisan aksara Jawa yang relevan (Utami dkk, 2008a). Dalam hal ini pendekatan yang digunakan mengacu pada sejumlah aturan produksi yang digunakan untuk mendefinisikan secara tepat setiap pola pemenggalan string latin hasil penguraian rangkaian karakter latin yang akan ditransliterasikan. Jadi aturan produksi digunakan untuk pemadanan adanya suatu pola pemenggalan string latin yang diperoleh. Proses berikutnya adalah mencari pola pemetaan alihaksara yang relevan untuk mengganti bagian pola pemenggalan string latin tersebut menjadi aksara Jawa dengan terlebih dahulu menentukan letak atau posisi aksara Jawa.

Dengan membangun seperangkat aturan yang kompleks, sebuah rangkaian string latin yang ditulis menggunakan karakter latin dapat ditransliterasikan menjadi tulisan dalam aksara Jawa. Penyusunan kode-kode yang akan digunakan untuk mengalihaksarakan suku kata-suku kata yang telah didapatkan pola tata letaknya menggunakan format TeX /LaTeX yang disebut dengan intermediate teks (Utami dkk, 2008b). Intermediate teks merupakan dokumen teks yang berisi kode-kode aksara Jawa dalam ekstensi *.tex* yang telah mengikuti pola penulisan aksara Jawa yang benar. Metafont digunakan untuk mendesain dan menggambarkan bentuk rupa aksara Jawa yang benar. *Jawa.mf* merupakan kode sumber font aksara Jawa yang ditulis menggunakan text editor.

Proses kompilasi font mengubah kode sumber *.mf* menjadi kode font yang dikenal TeX yaitu *.gf* dan *.tfm* dengan program Metafont. File *Jawa.tfm* merupakan font hasil kompilasi dari Metafont. *Jawa.stf* merupakan aturan penulisan aksara Jawa dalam TeX berupa suatu class atau style (Utami dkk, 2008b). Semua aksara yang ada dalam aksara Jawa dikumpulkan dengan membuat perbendaharaan aksara, memberikan nama pada masing- masing aksara, menentukan urutan dari aksara, menjabarkan peraturan penggabungan aksara dan mendefinisikan aturan penulisan aksara Jawa (seperti: cara penempatan dan penyambungan) dan menentukan rupa aksara (*glyphs*) yang diperlukan dalam penggabungan. TeX Font Metrics (TFM) digunakan TeX untuk mengkompile dokumen. *dvips* merupakan suatu program yang digunakan untuk melakukan konversi dari dokumen *.dvi* menjadi suatu dokumen berbasis grafik, misalnya *postscript*. Hasil akhir dari transliterator adalah dokumen teks yang memuat aksara Jawa sebagai hasil transliterasi yang dihasilkan dari kode sumber teks yang telah diproses menggunakan Perl dan TeX.

4. Pengujian Program JawaTeX

Pengujian dilakukan dengan perangkat keras: processor AMD Atlon™ XP 2500+, RAM 256 MB dan Hardisk 40 GB, sedangkan perangkat lunak yang digunakan adalah sistem operasi GNU/Linux Debian 3.1 Sarge, Perl, LaTeX (e-TeX (Web2C 7.4.5) 3.14159-2.1) dan Metafont ((Web2C 7.4.5) 2.718)). Pengujian aplikasi dilakukan dengan memberikan input dokumen teks latin yang memuat, yakni: rangkaian string latin yang memuat kombinasi huruf yang mungkin terjadi pada suku kata dalam aksara Jawa, dan rangkaian string latin yang memuat kombinasi huruf yang bisa jadi tidak mungkin terjadi pada suku kata dalam aksara

Jawa yang sebelumnya tidak bisa ditentukan pola suku katanya. Terdapat 2 mekanisme transliterasi yang ditawarkan, yakni: pengguna menghendaki seluruh isi dokumen teks ditransliterasi, dan pengguna menghendaki sebagian isi dokumen teks ditransliterasi.

4.1 Pengguna menghendaki seluruh isi dokumen teks ditransliterasi

Pengguna menuliskan teks sumber menggunakan teks editor dan disimpan dalam format *.txt*. Teks sumber diolah menggunakan Perl untuk menghasilkan daftar pola pemenggalan suku kata yang benar sesuai aturan produksi untuk pengecekan dan pemecahan suku kata berdasarkan *linguistics knowledge* penulisan aksara Jawa serta pemetaannya kedalam aturan penulisan LaTeX. Pengolahan disini termasuk juga *spell checker* (jika user menghendaki pengecekan kebenaran tulisan), penanganan ambiguitas, pengecekan kebenaran format bilangan romawi dan penghitungan nilai desimalnya, serta pengkodean sintaks. Proses ini menghasilkan 3 dokumen atau file, yaitu:

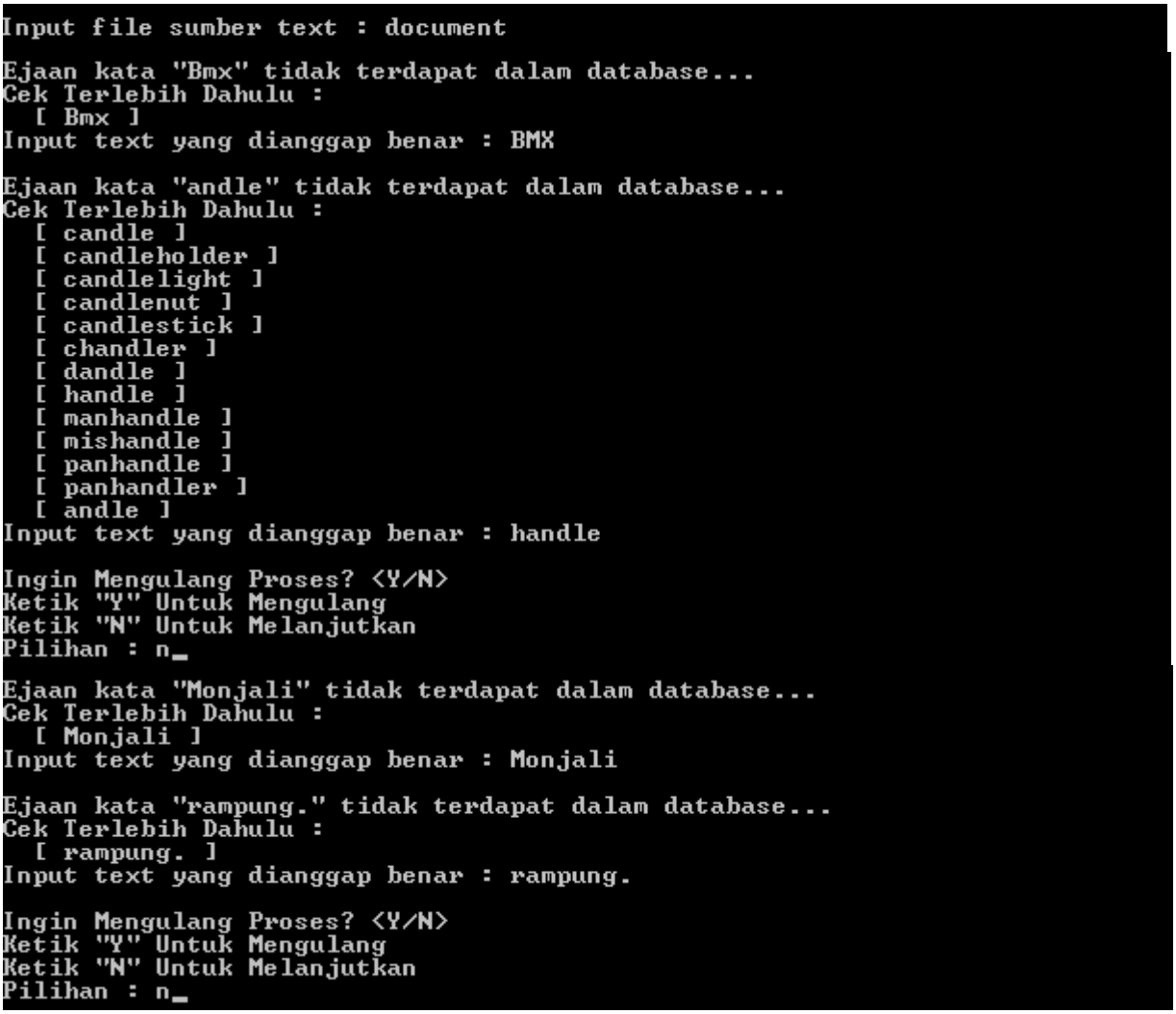
1. File dengan format *_rev.txt* yang berisi revisi isi dokumen setelah melalui *spell checker*.
2. File dengan format *.jw* yang berisi daftar pola pemenggalan string Latin.
3. File dengan format *.tex* yang berisi daftar kode sintaks untuk memetakan pola yang benar ke dalam aturan penulisan LaTeX menggunakan suatu class atau style.

Sistem pemenggalan dokumen teks diuji coba menggunakan file berbasis teks *.txt* (Utami dkk, 2009a) seperti yang tertampil pada gambar 2.

```
Bmx bicycle andle 1.9
Design photo Bank Monjali pas rampung.

Komple'ks E'xxon Xaveria dihari Jum'at.
Arnold Schwarzenegger mangan yogurt.
Postgresql,mysql and phsychology test.
Tiba-tiba hada Al-Toyib.

SMU Negeri VX
Perum Mulungan Wetan XV
S.M.P. N IX Yogya.

APPLYING OF THE CONTEXT FREE RECURSIVE DESCENT PARSER AND PATTERN MATCHING METHOD
Doctoral Program in Computer Science of Postgraduate School Gadjah Mada University
Graha Student Internet Center SIC 3rd floor
Faculty of Mathematic and Natural Sciences Gadjah Mada University
Sekip Utara Bulaksumur Yogyakarta.55281
Telp/fax:(0274)522443
Rakhshanda Qeyrra.
Nafeeza Hakim.
Nathanael Jericho Andhika.
```

Gambar 2: File *document.txt*

Cuplikan proses pengolahan teks sumber *document.txt* tersebut ditampilkan pada gambar 3.



Gambar 3: Cuplikan pemrosesan teks sumber

Teks dokumen yang telah diperbaiki, oleh sistem disimpan dengan nama *document_rev.txt* dan sistem dapat menghasilkan pola pemenggalan string latin dan oleh sistem disimpan dalam file *.jw* (Utami dkk, 2009a) seperti disajikan pada gambar 4.

```
\bmks\bi\cy\cle\ha\n\dle\1\.\9\
\de\si\gn\pho\to\Ba\nk\mo\nja\li\pa\s ra\mpung\.\

\Ko\mple'\kshe'\kskso\nksa\ve\ri\ya\di\ha\ri\ju\m\'\ha\t\.\
\Ar\no\ld\s\chwar\ze\ne\gger\ma\nga\n yo\gur\t\.\
\Po\st\gre\skhl\,\myskhl\ha\nd\phsy\cho\lo\gy\te\st\.\
\Ti\ba\-\ti\ba\ha\da\A\l\-\To\yi\b\.\

\Smu\Ne\ge\ri\vks\
\Pe\ru\ma\ha\nmu\lu\nga\nwe\ta\n\1\5\
\S.\m.\P.\n\9\yo\gya\.\

\A\p\plying\ho\fthe\co\nte\kst\fre\he\re\cur\si\ve\de\sce\nt\Par\ser\ha\nd\Pa\tter\
nma\t\ching\me\tho\d\
do\cto\ra\lPro\gra\mhi\nco\mpu\ter\Sci\he\nce\ho\fPo\s\tgra\du\ha\te\s\cho\ho\lGa\d
jah\ma\da\U\ni\ver\si\ty\
Gra\ha\Stu\de\nt\hi\nter\ne\tce\nter\Si\c\3\rd\flo\hor\
fa\cu\lty\ho\fma\the\ma\ti\cha\nd\Na\tu\ra\l\Sci\he\nce\sGa\djah\ma\da\U\ni\ver\si\
ty\
Se\ki\p\U\ta\ra\Bu\la\ksu\mur\yo\gya\kar\ta\.\5\5\2\8\1\
Te\lp\fa\ks\:\(\0\2\7\4\)\5\2\2\4\4\3\.\
ra\khsha\nda\khe\yrra\.\
Na\fe\he\za\ha\ki\m\.\
Na\tha\na\he\lje\ri\cho\A\ndhi\ka\.\
```

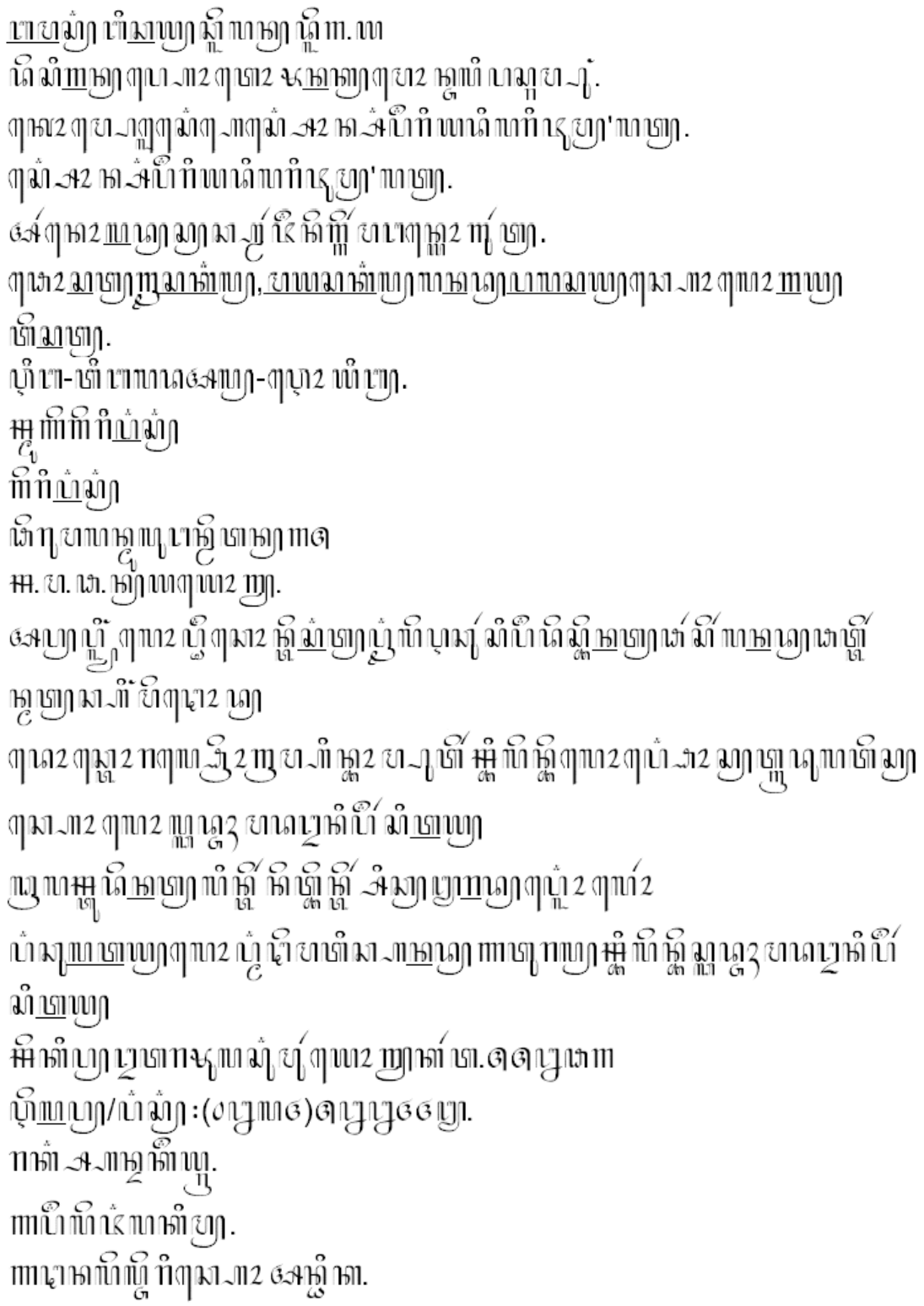
Gambar 4: File *document.jw*

File ketiga yang dihasilkan oleh sistem adalah pola pemetaan yang disimpan dalam file *.tex* seperti yang ditunjukkan pada gambar 5.

```
\gb{\ba\ma}\ks\bi\gb{\ca}\y\ca\Le\ha\n\da\Le\1\ttk\9
\de\si\gb{\ga}\n\pE\Ho\to\mba\gb{\na}\k\mo\na\Ja\li\pa\sa\Ra\ma\Pung\ttk

\mko\mE\Pa\LE\ksE\HE\ksE\KSo\na\KSa\ve\ri\ya\di\ha\ri\ju\m\prm\ha\t\ttk
\AR\no\gb{\la}\d\s\ca\Ha\War\ze\ne\ga\Ger\ma\nga\nE\Yo\gur\t\ttk
\mpo\gb{\sa}\t\gre\gb{\sa\kha}\l\kma\gb{\ma\ya\sa\kha}\l\ha\gb{\na}\d\gb{\pa\ha\sa}
\y\cE\Ho\lo\gb{\ga}\y\te\gb{\sa}\t\ttk
\mti\ba\min\ti\ba\ha\da\A\l\min\mto\yi\b\ttk

\msa\Mu\Ne\ge\ri\gb{\va}\ks
\mpe\ru\ma\ha\na\Mu\lu\nga\na\We\ta\n\1\5
\msa\ttk\ma\ttk\mpa\ttk\n\9\yo\gya\ttk

\A\p\pa\Lying\ho\fa\THe\co\na\Te\gb{\ksa}\t\fre\he\re\cur\si\ve\de\sa\Ce\gb{\na}\t\
mpar\ser\ha\gb{\na}\d\mpa\ta\Ter\na\Ma\t\ca\Hing\me\tho\d
\dO\cE\To\ra\lE\mPro\gra\ma\Hi\nE\Co\ma\Pu\ter\msa\Ci\he\na\Ce\ho\fE\mPo\s\ta\Gra\d
u\ha\te\s\cE\Ho\ho\la\mGa\da\Jah\ma\da\U\ni\ver\si\gb{\ta}\y
mgra\ha\msa\Tu\de\gb{\na}\t\hi\na\Ter\ne\ta\Ce\na\Ter\Si\c\3\gb{\ra}\d\fE\Lo\hor
fa\cu\gb{\la\ta}\y\ho\fa\Ma\The\ma\ti\ca\Ha\gb{\na}\d\mna\tu\ra\l\msa\Ci\he\na\Ce\s
a\mGa\da\Jah\ma\da\U\ni\ver\si\gb{\ta}\y
mse\ki\p\U\ta\ra\mbu\la\ksu\mur\yo\gya\kar\ta\ttk\5\5\2\8\1
mte\gb{\la}\p\grg\fa\ks\tda\lkr\0\2\7\4\rkr\5\2\2\4\4\3\ttk
ra\kha\Sa\Ha\na\Da\khe\ya\Rra\ttk
mna\ffe\he\za\ha\ki\m\ttk
mna\tha\na\he\la\Je\ri\cE\Ho\A\na\DHi\ka\ttk
```

Gambar 5: File *document.tex*

Dokumen *.tex* tersebut kemudian diproses dengan menggunakan LaTeX dengan memanggil kode sintaks yang telah ditulis dalam file style bernama *JawaTeX.sty*. Hasil dari pemrosesan file dokumen tersebut adalah file *.dvi* yang kemudian dapat diproses menjadi file *.ps* dan *.pdf* seperti yang tertampil pada gambar 6.



Gambar 6: File *document.pdf*

Sistem mengalihaksarakan string latin berdasarkan pola pemenggalan string latin dan tidak berdasarkan: pembentukan kata dan bunyi pengucapan suatu kata atau lafal bunyi unsur bahasa yang bersangkutan.

4.2 Pengguna menghendaki sebagian isi dokumen teks ditransliterasi

Pengguna menuliskan kode sintaks di LaTeX. Pengguna dituntut mempunyai pengetahuan tentang cara penulisan aksara Jawa yang benar. Pengguna secara manual memahami penulisan aksara Jawa disesuaikan dengan format style JawaTeX yang merupakan kode sintaks dari pemenggalan suku kata dari bagian dokumen yang akan ditransliterasi. Di sini pengguna mengerti bagaimana memenggal suku kata berdasarkan cara penulisan aksara Jawa dan dituntut menghapal daftar kode sintaks penulisan LaTeX yang telah dibuat. Pada gambar 7 berikut ini dicontohkan pengguna menghendaki pengalihaksaraan beberapa bagian dari dokumen teks. Dokumen teks ditulis menggunakan teks editor dan disimpan dengan format *.tex*.

```
\documentclass{article}
\usepackage{JawaTeX}
\begin{document}
In many countries research has been done to develop character computeration for
their local culture. Right now Javanesse society write and read using Latin
characters and not familiar using Javanesse characters. This is happened because of
they did not have enough education to be able write and read Javanesse characters.
Writing Javanesse characters also not simple as Latin so make other problem. There
is only less research on digital Javanesse character. One of reason is that there
is only few researcher that expert in Javanesse grammar and has capability in
digital technology. Latin text document can be transliterated to Javanesse
character. That will be transliterated is parsed or broke to get token list
(sylabic). \begin{jw}
\mte\gb{\ksa}\t\dO\cu\me\gb{\na}\t\msa\Pa\Li\ta\mPa\ta\Ter\nE\mBro\wa\Sing\mba\se\d
E\Ho\na\Li\ngu\hi\sa\Ti\c\mkE\No\wa\Le\da\Ge\ho\fa\Wri\ting\ja\va\ne\sa\Se\msa\Cri\
gb{\pa}\t\hu\sing\mna\tu\ra\la\La\ngu\ha\ge\mpro\ce\sa\Sing\ttk\The\ha\lE\Go\ri\gb{
\tha}\m\ho\fa\La\ti\na\Te\gb{\ksa}\t\dO\cu\me\gb{\na}\t\sa\Pa\Li\ta\Ba\se\dE\Ho\na\
Li\ngu\hi\sa\Ti\c\kE\No\wa\Le\da\Ge\ho\fa\Wri\ting\ja\va\ne\sa\Se\sa\Cri\gb{\pa}\t\
hu\sing\The\co\na\Te\gb{\ksa}\t\min\fre\he\re\cur\si\ve\min\de\sa\Ce\gb{\na}\t\mpar
\ser\ha\lE\Go\ri\gb{\tha}\m\co\na\Si\gb{\sa\ta}\s\ho\f\tda\ffi\le\wri\ting\ha\gb{\n
a}\d\re\ha\ding\pro\ce\gb{\sa}\s\kma\for\ma\ta\Ting\pro\ce\gb{\sa}\s\tda\ro\ma\na\N
u\ma\Ber\ca\He\ca\King\pro\ce\gb{\sa}\s\kma\sa\Pe\gb{\la}\l\ca\He\ca\Ker\for\ffi\la
\Te\ring\wrong\wor\gb{\da}\s\pro\ce\gb{\sa}\s\kma\ha\gb{\na}\d\la\ti\n\sa\Tring\sa\
Pa\Li\ta\Pa\ta\Ter\nE\Bro\wa\Sing\pro\ce\gb{\sa}\s\ttk\E\ver\yE\Pro\ce\gb{\sa}\s\ha
\sE\Ho\gb{\wa}\n\pro\du\ca\Ti\ho\na\Ru\le\ttk\gb{\ba}\y\bu\hi\la\Ding\ha\se\tE\Co\m
a\Pa\Le\ksE\Ho\fa\Ru\le\kma\la\ti\n\sa\Tring\se\khu\he\na\Ce\wa\Hi\gb{\ca}\h\hi\sa\
Wri\ta\Te\na\Hi\na\La\ti\n\ca\Ha\ra\ca\Ter\ca\na\Be\pro\ce\sa\Se\dE\So\tha\gb{\ta\s
a}\y\la\La\ba\Le\sa\Pa\Li\ta\Pa\ta\Ter\gb{\na}\s\ca\na\Be\pro\du\ce\dE\Cor\re\gb{\c
a\ta\la}\y\ttk\A\ma\Bi\gu\hi\gb{\ta}\y\pro\ba\Le\ma\Ca\na\Be\ha\vo\hi\de\gb{\da\ba}
\y\ha\n\da\Ling\ho\fa\Sa\Pa\ce\ca\Ha\ra\ca\Ter\ha\gb{\na}\d\be\he\ksa\Pe\ca\Te\dE\T
o\so\la\Ve\co\ma\Pa\Le\ksE\Pro\ba\Le\ma\Wi\tho\hu\ta\Ha\nyE\Pro\ba\Le\gb{\ma}\s\ttk
\thi\sa\La\ti\na\Te\gb{\ksa}\t\dO\cu\me\gb{\na}\t\sa\Pa\Li\gb{\ta}\s\ha\re\re\ha\gb
{\da}\y\to\be\pro\ce\sa\Se\da\Hi\nE\To\The\ne\gb{\ksa}\t\sa\Te\pa\THa\ta\Hi\sa\THe\
pro\ce\gb{\sa}\s\ho\fE\Co\na\Ver\ting\gb{\sa}\y\la\La\ba\Le\sa\Pa\Li\ta\Pa\ta\Ter\g
b{\na}\s\to\be\ja\va\ne\sa\Se\ca\Ha\ra\ca\Ter\ha\gb{\na}\d\The\ma\pa\Ping\ho\fa\THe
\hir\wri\ting\s\ca\He\me\ttk\end{jw} This research belongs to Linguistics Computing
area, conducting the natural language processing using symbol by computer
technology. The concept of this text document split model can improve the existing
machine of Latin string split which was made before. From the testing it is seen
that by using The Context-Free Recursive-Descent Parser algorithm, text document in
Latin writing can be processed so that syllable split patterns can be produced
correctly. Honestly this research has not come to the final result, it has only
succeeded to produce syllable split patterns. The produced syllable split patterns
are ready to be processed into the next step that is the process of converting
syllable split patterns to be Javanesse character and the mapping of their writing
scheme. The concept of the next document split built in this paper can be used as
the base to be developed in other case.
\end{document}
```

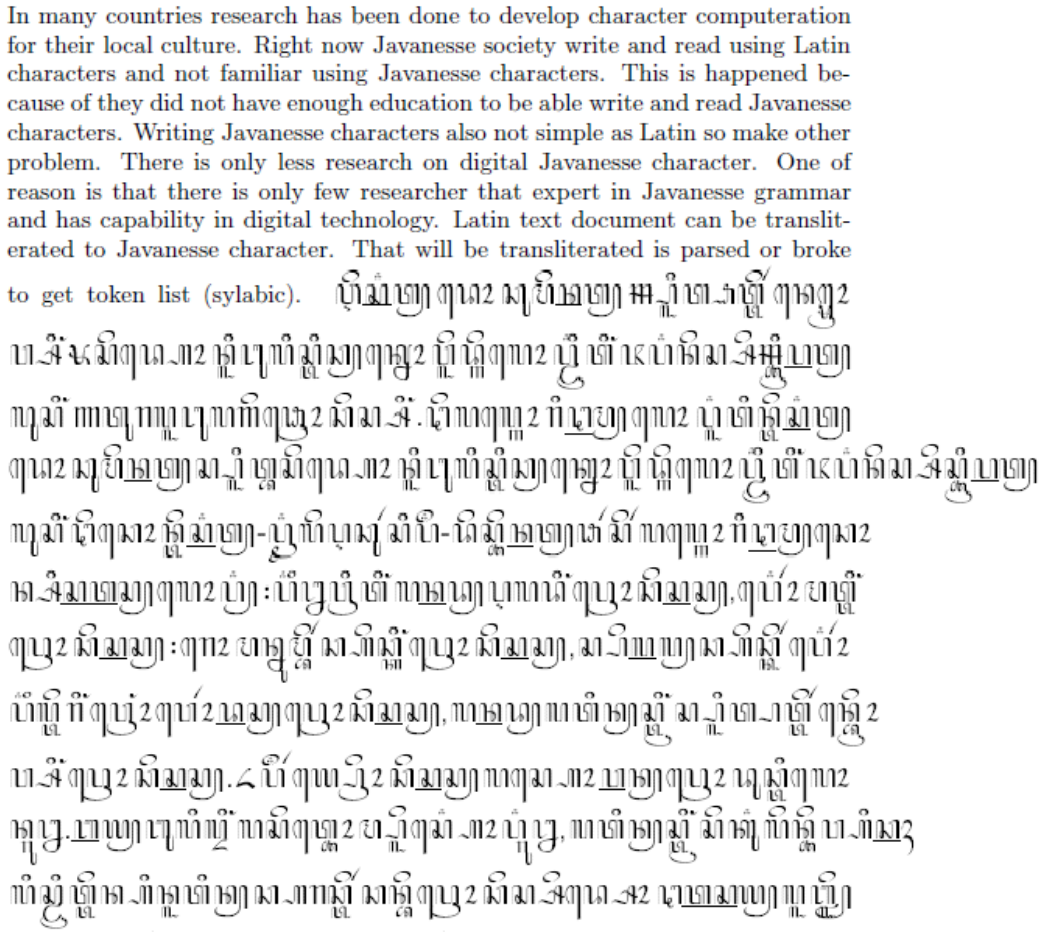
Gambar 7: Contoh penulisan dokumen teks yang hanya menghendaki sebagian isi dokumen teks ditransliterasi

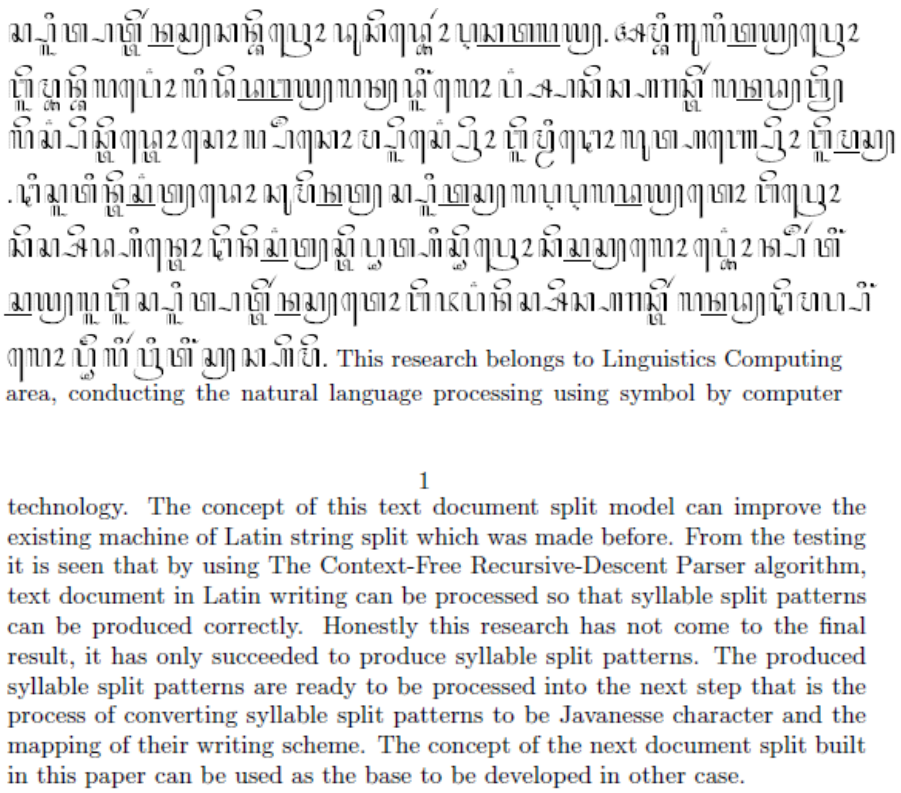
File berformat *.tex* yang telah ditunjukkan pada gambar 7 selanjutnya dicompile dengan perintah:

```
ema@debian:/home/data/s3/JawaTeX/$ latex double.tex
ema@debian:/home/data/s3/JawaTeX/$ ls double.*
double.aux  double.dvi  double.log  double.tex
ema@debian:/home/data/s3/JawaTeX/$ dvips double.dvi
ema@debian:/home/data/s3/JawaTeX/$ ls double.*
double.aux  double.dvi  double.log  double.ps  double.tex
ema@debian:/home/data/s3/JawaTeX/$ ps2pdf double.ps
ema@debian:/home/data/s3/JawaTeX/$ ls double.*
```

```
double.aux  double.dvi  double.log  double.pdf  double.ps
double.tex
```

dan menghasilkan file dokumen teks berformat *.pdf* yang memuat hasil alihaksara seperti yang disajikan pada gambar 8


In many countries research has been done to develop character computeration for their local culture. Right now Javanesse society write and read using Latin characters and not familiar using Javanesse characters. This is happened because of they did not have enough education to be able write and read Javanesse characters. Writing Javanesse characters also not simple as Latin so make other problem. There is only less research on digital Javanesse character. One of reason is that there is only few researcher that expert in Javanesse grammar and has capability in digital technology. Latin text document can be transliterated to Javanesse character. That will be transliterated is parsed or broke to get token list (sylabic).



Gambar 8: Dokumen teks yang sebagian teksnya ditransliterasi

Paket program JawaTeX terdiri dari program pengecekan dan pemenggalan string latin untuk penelusuran pola pemenggalan string Latin dan *style* LaTeX yang digunakan untuk mengkodekan sintaks LaTeX. Sepanjang teks sumbernya telah ditata ejaan maka JawaTeX mempunyai beberapa kemampuan sebagai berikut:

1. Mampu mengalihaksarakan lebih dari satu kalimat, bahkan alinea.
2. Mampu mengalihaksarakan string Latin berdasarkan pola pemenggalan string Latin yang penelusurannya tanpa transkripsi.
3. Mampu melakukan pencarian kemiripan kata untuk mengkoreksi kesalahan pengejaan kata menggunakan dua daftar kosakata bahasa Inggris dan bahasa Indonesia.
4. Mampu melakukan pemformatan tulisan.
5. Mampu menangani penggunaan karakter-karakter latin yang tidak mempunyai padanan abjadnya dalam aksara Jawa.
6. Mampu menangani penggunaan diftong.
7. Mampu menangani penggunaan bilangan Romawi.
8. Mampu menangani ambiguitas penggunaan tanda titik sebagai penanda akhir kata, penanda singkatan dan penanda desimal.
9. Mampu menangani ambiguitas penggunaan tanda spasi, tanda petik (`), dan tanda penghubung (-).
10. Mampu menangani penggunaan karakter konsonan bertumpuk lebih dari tiga.
11. Mampu melakukan pengkodean pola pemetaan dan menghasilkan pola transliterasi sesuai dengan daftar model pola pemetaan yang telah disusun.
12. Mampu melakukan alihaksara dengan penempatan aksara Jawa sesuai dengan skema penulisan yang telah dirancang.
13. Mampu melakukan berbagai perubahan bentuk aksara Jawa tergantung posisinya dalam suatu kalimat, serta aksara lain yang mengikuti dan diikutinya.
14. Mampu dijalankan di beberapa sistem operasi, telah diuji di Windows dan Linux.
15. Dapat dijalankan melalui web, diuji di http://www.jawatex.org

5. KESIMPULAN

Dari hasil pengujian terlihat bahwa dengan menggunakan algoritma *The Context-Free Recursive-Descent Parser*, dokumen teks dalam tulisan latin dapat diproses sehingga dihasilkan pola pemenggalan string Latin dengan benar. Pola pemenggalan suku kata yang dihasilkan siap mengantarkan ke tahapan selanjutnya yaitu proses pengubahan pola pemenggalan string latin ke aksara Jawa dan pemetaan skema penulisannya menggunakan algoritma *Pattern Matching*.

Algoritma transliterasi dokumen teks karakter Latin berdasarkan pengetahuan linguistik penulisan aksara Jawa ini meliputi:

1. Aturan produksi untuk penelusuran pola pemenggalan string Latin, meliputi:
   a. Aturan produksi penataan ejaan.
   b. Aturan produksi pengecekan kesalahan ketikan setiap kata dalam teks sumber untuk melakukan pencarian kemiripan kata untuk mengkoreksi kesalahan pengejaan kata.
   c. Aturan produksi pemformatan file text untuk membaca, memeriksa, memodifikasi dan menyisipi karakter lain ke dalam karakter penyusun string masukan supaya dokumen teks memenuhi syarat format penulisan. Di dalamnya termasuk aturan perhitungan bilangan Romawi menjadi nilai desimal.
   d. Aturan produksi untuk pemenggalan string Latin yang digunakan untuk menentukan sejauh mana pengecekan karakter dilakukan pada suatu string Latin dari rangkaian karakter yang telah dicek sehingga dapat memenuhi syarat untuk diambil sebuah pola pemenggalan string Latin. Sebuah pola pemenggalan string Latin dapat dibedakan dari suatu rangkaian karakter dengan melihat susunan kombinasi karakter.
2. Daftar model pola pemenggalan string latin, dalam algoritma transliterasi ini terdapat 177 model pola pemenggalan string Latin.
3. Daftar model pola pemetaan, dalam algoritma transliterasi ini terdapat 138 model pola pemetaan.
4. Aturan produksi untuk pengubahan pola pemetaan menjadi aksara Jawa tertentu.
   Proses pengubahan pola pemenggalan string Latin ke aksara Jawa adalah memetakan pola ke dalam aksara Jawa sehingga setiap karakter terwakili dalam hasil transliterasi.

JawaTeX juga dapat digunakan tanpa harus menggunakan program pengecekan dan pemenggalan string Latin namun pengguna dituntut mampu melakukan pemenggalan string Latin serta pengkodekan dalam format LaTeX. Penelusuran pola pemenggalan string Latin dan pola pemetaan dilakukan sendiri oleh pengguna. Hasil pengujian menunjukkan bahwa jika pengkodean dilakukan secara manual maka pengguna diberi kebebasan untuk memilih bagian dokumen yang akan ditransliterasikan, sehingga pengguna memiliki pilihan apakah akan mentransliterasikan seluruh isi dokumen teks Latin atau sebagian isi dokumen teks Latin. Konsep pemenggalan dokumen teks dan pengalihaksaraan yang dibangun dalam tulisan ini dapat dijadikan dasar untuk dikembangkan dalam kasus yang lain.

DAFTAR PUSTAKA

Apostolico, A; Galil, Z. 1997. *Pattern Matching Algorithms.* Oxford University Press. Oxford. UK

Black, P. 2007. *Dictionary of Algorithms and Data Structures*. Nasional Institute of Standards and Technology. Online pada **www.nist.gov/dads/**. 26 Juni 2008

French, J.; Powell, A.; Schulman, E. 1997. *Applications of Approximate Word Matching*. ACM ISSN: 0-89791-970-x. Online at www.cs.virginia.edu/papers/p9-french.pdf. 26 June 2008.

Gusfield, D. 1997. Algorithms on Strings, Trees, and Sequences: Computer Science. Cambridge University Press.

Mohammad, A; Saleh, O; Abdeen, R. 2006. *Occurrences Algorithm for String Searching Based on Brute-force algorithm*. Journal of Computer Science 2(1): 82-85, 2006. ISSN 1549-3636. Science Publications. Online pada www.scipub.org/fulltext/jcs/jcs2182-85.pdf. 25 Juni 2008.

Nori, K.; Kumar, S. 1988. Foundations of Software Technology and Theoretical Computer Science. Springer.

Schildt, Herbert. 1987. *Artificial Intelligence Using C*. Osborne-McGraw Hill, California.

Utami, E.; Istiyanto, J.; Hartati, S.; Marsono; Ashari, A. 2008a. Pemanfaatan Teknologi Informasi dan Komunikasi Untuk Membangun Model Transliterasi Dokumen Teks Karakter Latin ke Aksara Jawa Melalui Komputasi Linguistik sebagai Alternatif Menarik dalam Melestarikan Kebudayaan Jawa. Prosiding Seminar Nasional Teknologi Informasi, Komunikasi dan Multimedia (SNASTIA) 2008, ISSN: 1979-3960, 31 May 2008, page 154-159.

Utami, E.; Istiyanto, J.; Hartati, S.; Marsono; Ashari, A. 2008b. *JawaTeX: Javanese Typesetting and Transliteration Text Document*. Prosiding International Conference on Advanced Computational Intelligence and Its Application (ICACIA) 2008, ISBN: 978-979-98352-5-3, 1 September 2008, page 149-153.

Utami, E.; Istiyanto, J.; Hartati, S.; Marsono; Ashari, A. 2009a: *Text Document Split Pattern Browsing Based on Linguistic Knowledge of Writing Javanese Script using Natural Language Processing*, Proceeding International Conference on Rural Information and Communication Technology 2009, ISBN: 978-979-15509-4-9, 17-18 Juni 2009, 372-377