ANALISIS POLA DAN PROBABILITAS PADA PEMBUATAN KAMUS BAHASA INDONESIA DAN JAWA MENGGUNAKAN PEMODELAN MARKOV

Mike Yuliana[1], Tribudi Santoso[2], Nur Afifah[3]
Jurusan Teknik Telekomunkasi
Politeknik Elektronika Negeri Surabaya
Kampus PENS, Keputih, Sukolilo, Surabaya.
Telp : (031)5947280, mieke@eepis-its.edu, rosyid@eepis-its.edu, phiephah@gmail.com

Abstract

To overcome the problem of communication in two languages, we need a translator. The translator can be a dictionary. In the manufacture of electronic dictionaries, a method for parsing and searching is needed. In this study, the method that is used to parse a sentence is parsing tree and binary searching with the help of markov models to determine patterns of words that form the search.

The results showed that the words with the same prefix, would have the same probability value, and these form a pattern. The more words beginning with the same prefix, the smaller the probability value. The smallest probability value is 0.007462686 with the number of words are 134. While the probability of words beginning with the letter Y and O is 0.33333334, because it has the least amount of words. For the probability of words beginning with the letter F, O, V, and Z is 0, because there is no word in the dictionary database beginning with that letter.

Keyword: *binary,markov, parsing*

1. PENDAHULUAN

Manusia membutuhkan interaksi dengan manusia lain sebagai makhluk sosial. Proses komunikasi ini pada kondisi tertentu tidak bisa dilakukan karena kendala bahasa. Selain itu, bahasa merupakan cermin budaya dan identitas diri penuturnya, sehingga bahasa harus dilestarikan, misalnya Bahasa Jawa [1]. Untuk mengatasi permasalahan komunikasi dalam dua bahasa tersebut, maka dibutuhkan sebuah *trnaslator* atau penerjemah. *Translator* tersebut bisa berupa sebuah kamus.

Markov Model merupakan suatu sistem yang memodelkan simbol ke dalam suatu mesin *finite state* (keadaan yang terbatas), sehingga diketahui simbol apa yang dapat mewakili sebuah parameter vektor dari sebuah kata yang dimasukkan ke dalam mesin dan diestimasi berulang-ulang hingga dihasilkan parameter observasi dengan *mean* dan kovarian yang konvergen untuk setiap *state*-nya. *Markov Model* ini adalah model statistika, yang banyak digunakan untuk masalah-masalah *pattern recognition* (pengenalan pola), seperti *speech recognition* [2][3].

Pembuatan kamus dapat dipandang sebagai masalah pengenalan pola, dengan mengetahui urutan atau *state* karakter dan menghitung nilai probabilitas *Markov* yang membentuk pola tertentu. Karakteristik orang berbicara bisa diketahui dengan mengetahui pola yang terbentuk, baik kosakata yang sering muncul ataupun kosakata yang memiliki probabilitas paling kecil karena kosakata yang ada pada kamus merupakan kosakata yang umum digunakan masyarakat. Sehingga *markov model* ini juga bisa diaplikasikan pada pembuatan *translator*.

Pada penelitian ini dibuat *software* kamus, dengan metode *parsing tree* untuk mengurai kalimat, dan *binary searching* dengan bantuan *markov model* untuk menentukan pola kata yang terbentuk dari kata yang dicari. *Software* yang dikembangkan ini berbasis *android*, dan diharapkan tepat sasaran bagi pengguna perangkat *android* yang sedang berkembang pesat saat ini .

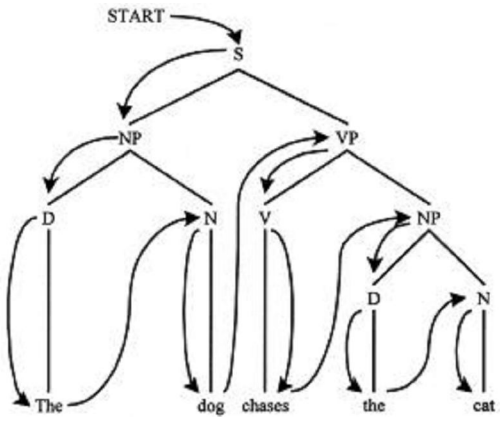
2. METODE

Pada proses *Parsing Tree* atau penguraian kalimat digunakan *Top-Down Parsing* karena cara menguraikan sebuah kalimat dimulai dari unsur yang terbesar yaitu kalimat tunggal sampai menjadi unsur yang terkecil yaitu kata. Penguraian dilakukan berdasarkan fungsi dan jenis kata, karena terdapat fungsi pembacaan subyek ketika kalimat dalam bahasa Indonesia akan diterjemahkan. Setelah itu dilakukan proses penerjemahan kata. Sebelum dilakukan proses pencarían kosakata atau *searching* akan digunakan metode *markov model* untuk pengenalan pola kata, sehingga diharapkan

markov model ini akan membantu mempercepat proses pencarían kata. Semua itu akan dihubungkan ke dalam satu *interface* yang dibuat menggunakan *compiler Eclipse*, dan akan dijalankan dengan *emulator AVD (Android Virtual Device)*.

Parsing Tree Proses parsing tidak hanya dapat dilakukan dalam NLP, melainkan juga pada bidang lain seperti pada pembuatan sebuah *compiler*. Metode-metode parsing yang dibahas berikut khusus digunakan dalam NLP. Sebelumnya perlu diketahui arti dari istilah *constituent*, yaitu unsur-unsur pembentuk kalimat yang dapat berdiri sendiri, contohnya *frasa kata benda*, *frasa kata kerja* dan sebagainya, dan istilah *parser* yaitu program yang melakukan proses parsing. Dalam bidang tata bahasa dan linguistik, *parsing* adalah sebuah proses yang dilakukan seseorang untuk menjadikan sebuah kalimat menjadi lebih bermakna atau berarti dengan cara mengurai kalimat tersebut menjadi kata-kata atau frase-frase.

Metode *parsing* yang digunakan adalah *Top-down parser* yang bekerja dengan cara menguraikan sebuah kalimat mulai dari unsur yang terbesar sampai menjadi unsur yang terkecil [4][5]. Hal ini dilakukan terus-menerus sampai semua komponen yang dihasilkan adalah unsur terkecil dalam kalimat, yaitu kata. Sebagai contoh dapat dilakukan proses top-down parsing untuk kalimat "*The dog chased the cat"* yang ditunjukkan pada Gambar 1.



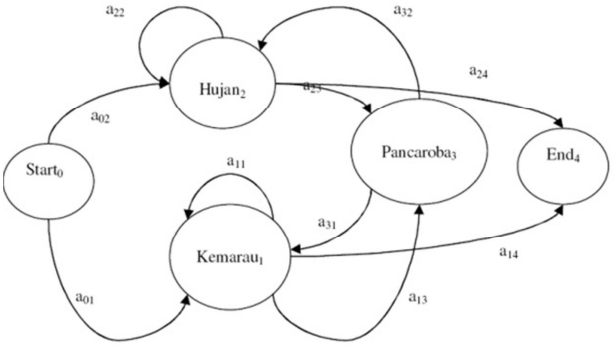
Gambar 1. Top-Down Parser [4]

Markov Model Rantai *Markov* atau *Markov Chain* adalah suatu teknik probabilitas yang menganalisis pergerakan probabilitas dari satu kondisi ke kondisi lainnya. Model ini dikenalkan oleh Andrey A. Markov, ahli matematika dari Rusia yang lahir tahun 1856 [6]. Rantai *Markov* merupakan suatu struktur yang terdiri dari entitas-entitas *stationer* yang disebut keadaan (*state*). Transisi antara atau di dalam suatu keadaan adalah suatu probabilitas yang merupakan perluasan dari *finite automaton*. *Finite automaton* sendiri adalah kumpulan *state* yang transisi antar *state*-nya dilakukan berdasarkan masukan observasi. Pada Rantai *Markov,* setiap busur antar *state* berisi probabilitas yang mengindikasikan kemungkinan jalur tersebut akan diambil dan dapat dihitung probabilitas urutan kejadian yang diamati [6].

Analisis rantai *Markov* tidak memberikan keputusan rekomendasi, melainkan hanya informasi probabilitas mengenai situasi keputusan yang dapat membantu pengambil keputusan untuk mengambil keputusan. Dengan demikian, analisa rantai *Markov* bukanlah teknik optimisasi, tetapi teknik deskriptif yang menghasilkan informasi probabilitas dimasa mendatang.

Untuk dapat menerapkan analisis rantai *Markov* kedalam suatu kasus, ada beberapa syarat yang harus dipenuhi [6]: 1) Jumlah probabilitas transisi untuk suatu keadaan awal dari sistem sama dengan 1, 2) Probabilitas-probabilitas tersebut berlaku untuk semua partisipan dalam sistem, 3) Probabilitas transisi konstan sepanjang waktu, 4) Kondisi merupakan kondisi yang *independent* sepanjang waktu.

Gambar 2 memperlihatkan contoh rantai *Markov* yang menggambarkan kondisi cuaca. Pada gambar ini, *aij* adalah probabilitas transisi dari *state i* ke *state j*.



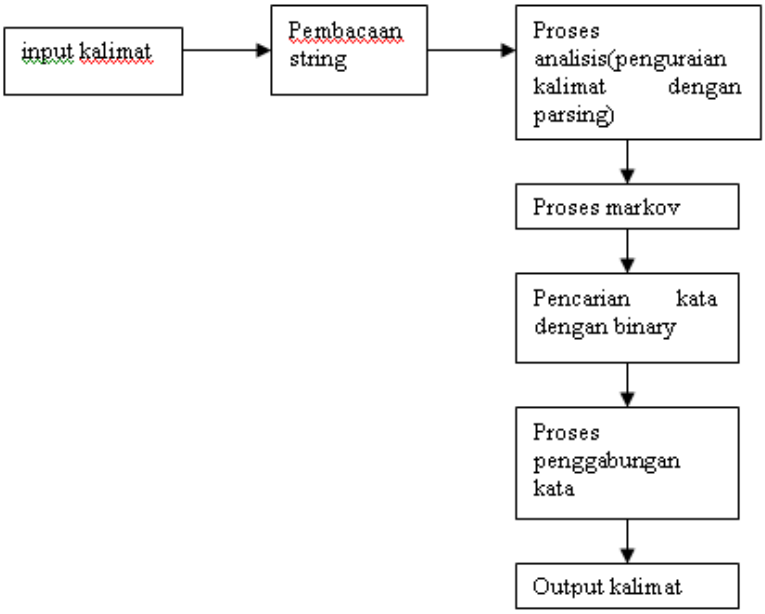
Gambar 2. Contoh Rantai *Markov* [6]

***Binary* Search** *Binary Search* adalah algoritma pencarian yang lebih efisien daripada algoritma *Sequential Search* . Hal ini dikarenakan algoritma ini tidak perlu menjelajahi setiap elemen dari tabel. Kerugiannya adalah algoritma ini hanya bisa digunakan pada tabel yang elemennya sudah terurut baik menaik maupun menurun.

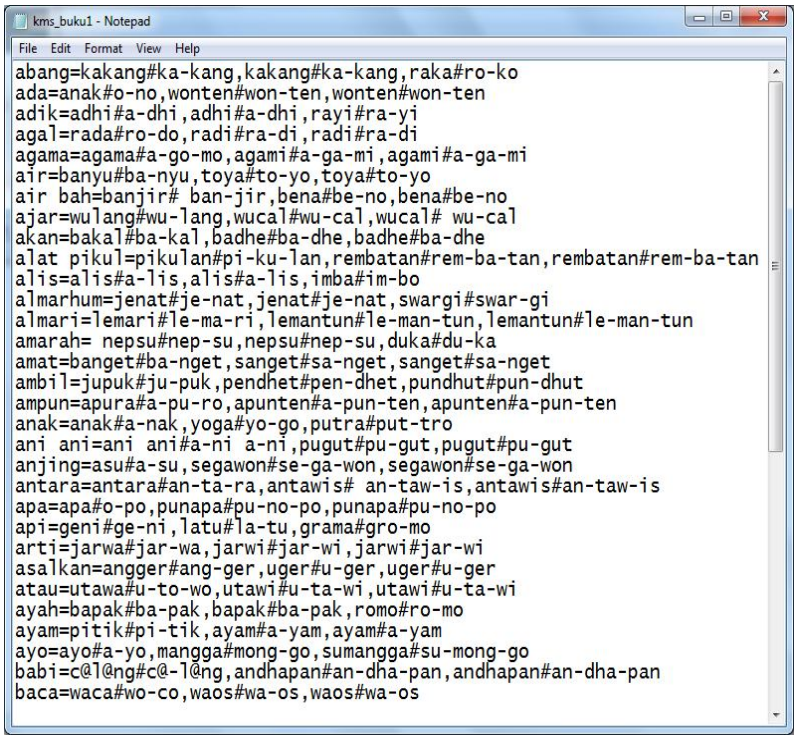
Pada intinya, algoritma ini menggunakan prinsip *divide and conquer*, yaitu sebuah masalah atau tujuan diselesaikan dengan cara membagi masalah menjadi bagian yang lebih kecil. Algoritma ini membagi sebuah tabel menjadi dua dan memproses satu bagian dari tabel itu saja .

3. DISKUSI

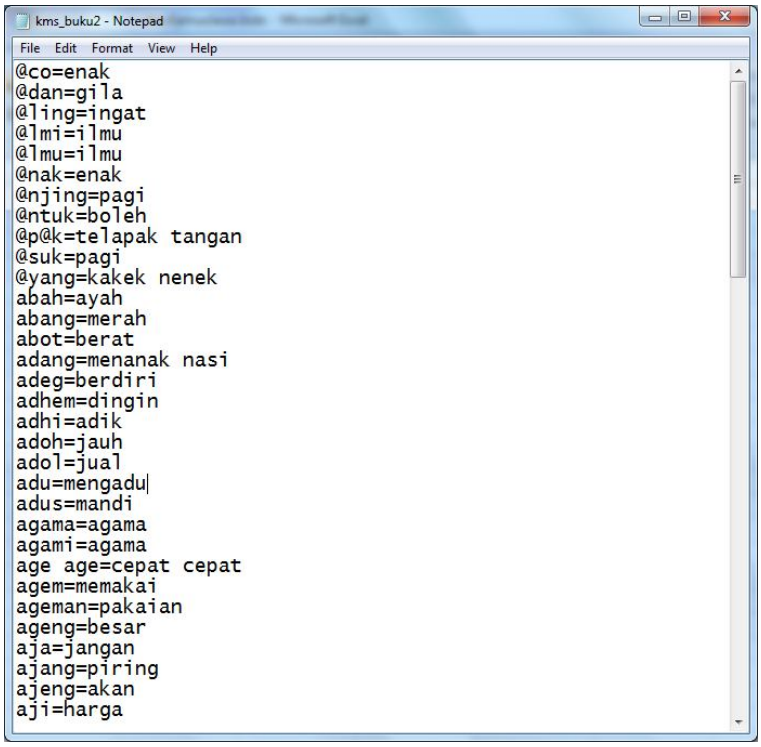
Perancangan sistem pada penelitian ini meliputi perancangan basisdata stándar, perancangan *interface*, dan metode pengolahan data yang terdiri dari proses *parsing kalimat*, proses pengenalan pola kata *markov*, serta proses *binary searching* dan penggabungan kata. Perancangan sistem pada basisdata didasarkan pada pembuatan basisdata standar yang berisi kumpulan kata-kata yang disimpan pada *file* teks(.txt) karena *database* sangat sederhana. Blok diagram sistem dari proses penerjemahan kalimat bisa dilihat pada Gambar 3.



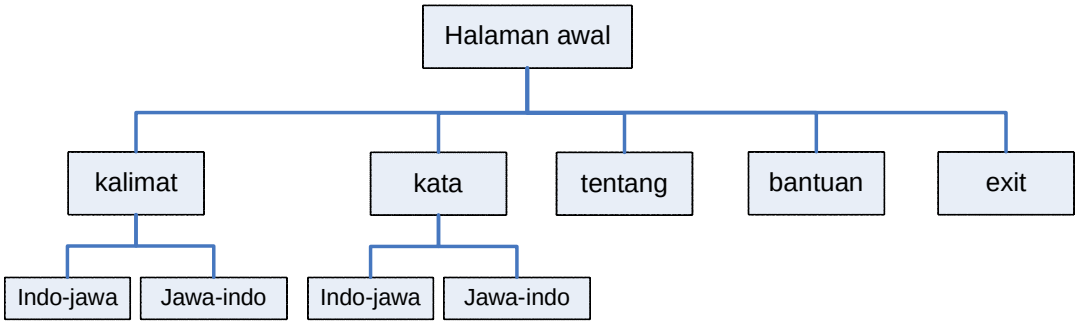
Gambar 3. Blok Diagram Proses Penerjemah



Gambar 4. Kamus Indonesia-Jawa



Gambar 5. Kamus Jawa-Indonesia



Gambar 6. Perancangan *Interface*

Perancangan Basis Data Standar Basis data berfungsi sebagai tempat penyimpanan kata. Karena dalam hal ini dibuat *software* penerjemah, maka akan dibutuhkan banyak sekali kata baik dalam bahasa Indonesia dan kata dalam bahasa Jawa. Karena basis data sederhana, tidak dibuat pemrograman khusus untuk *database*, namun hanya memanfaatkan *file* teks yang disimpan pada *notepad*. Sehingga nantinya akan ada perintah yang mengintegrasikan *file* teks ke dalam *software*. Pada kasus ini, basis data terdiri dari daftar kata yang terurut, karena digunakan metode *Binary* untuk proses *searching*

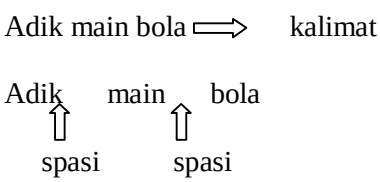
Gambar 4 dan 5 menunjukkan contoh kamus Indonesia Jawa dan Jawa Indonesia, dimana pada kamus Indonesia Jawa hasil terjemahan bahasa Indonesia terdapat 3 kata, yaitu kata dalam bahasa *ngoko, ngoko halus,* dan *krama* yang masing-masing kata dipisahkan dengan tanda koma ",".

Perancangan *Interface* Alur desain sistem *interface* kamus elektronik ini dapat dilihat pada Gambar 6.

Metode Pengolahan Data Setelah proses perancangan selesai, langkah selanjutnya adalah proses pengolahan data. Proses pengolahan data yang dilakukan meliputi *parsing* kalimat, pengenalan pola *Markov* dan *searching* kata.

***Parsing* Kalimat** Langkah pertama ketika kalimat sudah dimasukkan, akan dilakukan proses

penguraian kalimat menjadi unsur atau kata. Algoritma dari proses penguraian kalimat menjadi kata-kata adalah sebagai berikut : 1) *Input* kalimat, 2) Inisialisasi *variable* i=0, 3) Cek jumlah kata atau panjang kalimat. Dengan menggunakan prosedur for, bandingkan i dengan jumlah kata (keyStrArr.length) : 1) Jika i lebih kecil atau sama dengan jumlah kata, cek apakah input String []keyStrArr bertemu pemenggal kata spasi (’’). Jika Iya, gabungkan kata simpan ke array : ArrayList<String> results = new ArrayList<String>(); Increment variable i, dan kembali ke proses 3. Jika Tidak, masukkan variabel ke results. Return results, selesai, 2) Jika i lebih besar dari jumlah kata, return results, dan selesai.



Syarat dari sebuah kalimat adalah terdiri dari 2 kata atau lebih. Kalimat tersebut merupakan contoh kalimat tunggal dalam bahasa Indonesia, dan diurai berdasarkan spasi (‘ ‘) yang memisahkan tiap kata atau unsur.

Proses Markov Pada penelitian ini akan digunakan metode pengenalan pola kata dengan *markov model*. Metode ini digunakan untuk kebutuhan pendekatan pengenalan pola melalui perbandingan pola dari kata. Proses ini akan menghasilkan nilai probabilitas dari kata yang dimasukkan terhadap daftar kata yang ada pada basisdata. Hasil probabilitas transisi merupakan urutan kejadian atau urutan huruf atau karakter yang menyusun kata yang dimasukan tersebut.

Secara umum, proses pengenalan kata dibagi menjadi tiga tahap, yaitu : 1) Pembacaan string, 2) Pemodelan, 3) Pengenalan. Pembacaan string dilakukan di awal proses, dimana perlu dilakukan pembacaan setiap karakter dan juga dibutuhkan untuk proses pengurutan kata. Selanjutnya dilakukan pemodelan, dengan pemodelan statistik yaitu menggunakan rantai *Markov*. Dari pemodelan ini akan didapatkan parameter yang selanjutnya akan digunakan dalam proses pencarían kata untuk membantu mempercepat waktu eksekusi.

Algoritma dari *markov model* adalah sebagai berikut: 1)Input kata, 2) *Readfile*(fileteks_name), akses data database kamus.txt, 3) Dengan for, Cek karakter. Jika kata !=1 && i!=ch.length-1 maka : ambil karakter-1 dan compare dengan obyek kamus, kamus = compare.getResult(), 4) Perhitungan probabilitas transisi, 5) Ambil nilai Probablitas transisi, Hitung perkalian dari proabilitas transisi prob = prob*compare.getProb().

Cara kerja *markov model* untuk proses pengenalan pola kata dan perhitungan nilai probabilitas markov yaitu, dimisalkan terdapat basisdata seperti Tabel 1.

Tabel 1. Lexicon atau basisdata kamus

Bahasa Indonesia	**Bahasa Jawa**
Minum	Ngombe
Main	Dolan
Makan	Mangan
Nasi	Sego
Saya	Aku
Suka	Seneng
Susah	Angel

Dari Tabel 1, terdapat 7 kosakata yang sudah terurut sesuai alphabet. *Markov model* dalam penelitian ini digunakan untuk proses pengenalan model dan perhitungan nilai total probabilitas kata yang dimasukkan terhadap kosakata yang ada pada basisdata. Kosakata yang terdapat pada Tabel 1, merupakan contoh daftar kosakata yang sering diucapkan oleh masyarakat yang mendekati karakteristik orang berbicara. Contoh :
Masukkan kata yang akan diterjemahkan (Indo-Jawa) = MAIN. Dibentuk dari susunan huruf : M-A-I-N.
Pada kata ‘MAIN’ akan dilakukan pembacaan string perkarakternya, dan kata tersebut langsung di-*compare* (perbandingan) terhadap kosakata yang diawali oleh huruf M yang ada pada *lexicon*. Pada *event* pertama proses *compare* menuju ke huruf M, karena *input* (masukan) berupa kata ‘MAIN’ yang berawalan ‘M’, sehingga didapat data sementara seperti Tabel 2 .

Tabel 2. Data sementara

Bahasa Indonesia	**Bahasa Jawa**
Minum	Ngombe
Main	Dolan
Makan	Mangan

Probabilitas Transisi I :
S1 ‘Main’ (State 1) = P(I|M) = 1/3
S1 ‘Makan’ (State 1) = P(A|M) = 2/3

Pada Probabilitas Transisi I atau *event* kedua adalah *compare* huruf ‘A‘ yaitu huruf selanjutnya yang tersusun setelah huruf ‘M’. Huruf tersebut

di*compare* terhadap kosakata yang diawali oleh huruf ‘A’ yang ada pada *lexicon*. Sehingga didapat data seperti Tabel 3

Tabel 3. Data probabilitas transisi I

Bahasa Indonesia	Bahasa Jawa
Main	Dolan
Makan	Mangan

Probabilitas Transisi II :
S2 ‘main’ = P(I|A) = ½
S2 ‘makan’ = P(K|A) = ½

Pada Probabilitas Transisi II atau *event* ketiga adalah *compare* huruf ‘I‘ yaitu huruf selanjutnya yang tersusun setelah huruf ‘M’ dan ‘A’. Huruf ‘I’ tersebut di-*compare* terhadap kosakata yang diawali oleh huruf ‘I’ yang ada pada *lexicon*. Sehingga didapat data seperti Tabel 4

Tabel 4. Data probabilitas transisi II

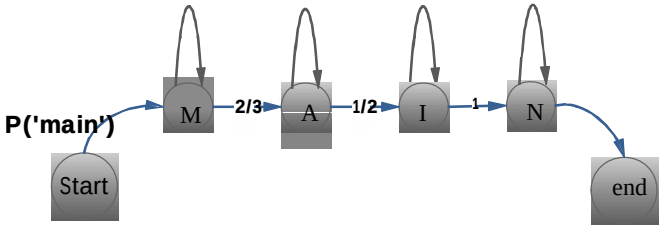
Bahasa Indonesia	Bahasa Jawa
Main	Dolan

Probabilitas Transisi III :
S3 ‘suka’ = P(N|I) = 1

Pada Probabilitas Transisi III atau *event* keempat adalah *compare* huruf ‘N’, yaitu huruf selanjutnya yang tersusun setelah huruf ‘M’, ‘A’ dan ‘I’. Huruf ‘N’ tersebut *dicompare* terhadap huruf ‘N’ yang ada pada *lexicon*. Proses *compare* selesai, karena huruf atau karakter yang membentuk kata ‘MAIN’ telah selesai di-*compare*. Dan didapat kesamaan atau kemiripan huruf-huruf yang tersusun pada kata masukan ‘MAIN’ dengan rangkaian *state* atau *event* yang terjadi, sehingga bisa dilakukan perhitungan nilai total probabilitas transisi atau probabilitas urutan *event* yang membentuk kata tersebut.Perhitungan Total Probabilitas untuk urutan karakter MAIN :

= P(A|M). P(I|A) . P(N|I)
= 2/3 . ½. 1
= 2/6=0.333333333

pada proses ini terdapat empat model *state* berupa rangkaian *state* ***X*** = 0,1,2,3 untuk membangun urutan o_0 sampai $o_{3,}$ dimana o_0 sampai o_3 disini adalah probababilitas transisi I sampai dengan IV.



Gambar 7. Rangkaian *state* kata ‘MAIN’

Proses Searching Pada penelitian ini akan digunakan metode pencarian dengan *Binary Search*. Metode ini digunakan untuk kebutuhan pencarian dengan waktu yang cepat. Prinsip pencarian dilakukan dengan membagi data atas dua bagian. Algoritma dari *Binary Search* adalah: 1) Posisi awal = 1**,** 2) Posisi Akhir = N, 3) Data Tengah = (Posisi Awal+Posisi Akhir)/2, 4) Bandingkan data yang dicari dengan data tengah, 5) Jika lebih kecil, proses dilanjutkan dengan posisi akhir = posisi tengah-1, Jika lebih besar, proses dilanjutkan dengan posisi awal+1, 6) Diproses berulang hingga data tengah = data yang dicari.

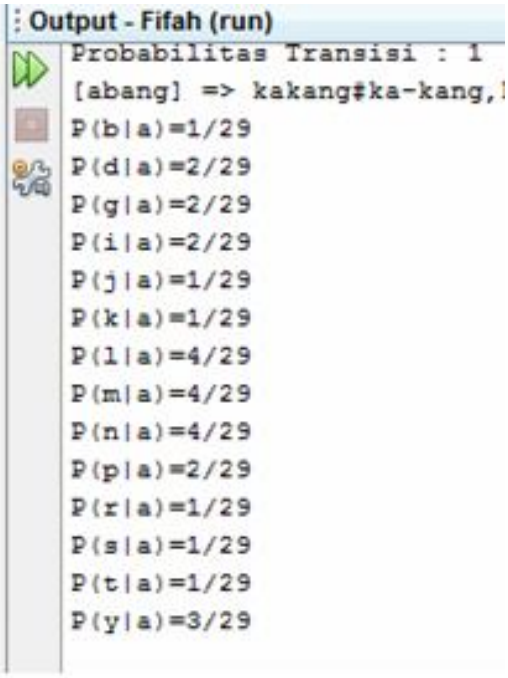
Proses Penggabungan Kata Keluaran dari program akan menghasilkan kalimat bahasa jawa yang merupakan hasil terjemahan untuk *translator* Indonesia ke Jawa, yaitu berdasarkan pembacaan subyek, apakah termasuk dalam tingkatan I (ngoko), II (madya) atau III (krama). Adapun algoritma dari penggabungan kata adalah sebagai berikut: 1) Banyaknya arti kata (*binary*SearchResult) yang sudah di-*searching*, 2) Bandingkan apakah binarySearchResult==null. Jika ya, retResult+ = data.get(idxMid).getJawa(). Jika tidak, binarySearchResult+ = keyStrArr[x], 3) Pembacaan subyek (subyek_krama.txt). Jika ya, *output* krama. Jika tidak, *output* ngoko.

4. HASIL

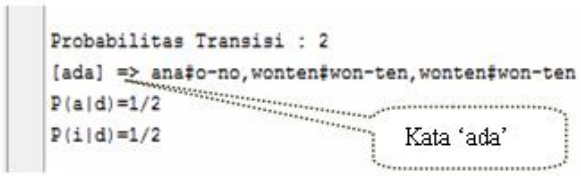
Pada penelitian ini akan dilakukan analisis pengenalan pola dan probabilitas dari kata yang dimasukkan. Proses pengenalan pola kata yang menggunakan metode *markov model* ini bertujuan untuk mengetahui nilai probabilitas dari kata yang dimasukkan terhadap kata yang terdapat pada *database* kamus. Selain probabilitas juga akan di dapatkan pola pembentukan kata atau urutan *state* karakter atau huruf yang membentuk kata tersebut. Sebelum proses *searching* kata, terlebih dahulu dilakukan proses markov ini untuk mempercepat proses *searching*. Sehingga program akan mencari ke pola kata tersebut, bukan ke semua kata yang ada pada *database* kamus. Data sementara tersebut disimpan pada Array List.

Gambar 8 menunjukkan tampilan *software* kamus yang di buat, dengan pencarian kata ‘ada’. Sedangkan Gambar 9 sampai dengan 11 menunjukkan urutan *state* serta nilai probabilitas Markov dari kata ‘ada’.

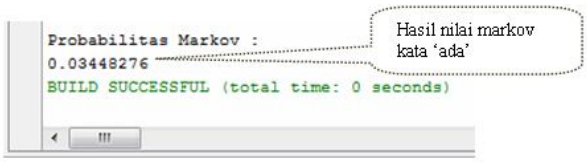
Gambar 8. Pencarian kata ‘ada’



Gambar 9. Probabilitas transisi ke I



Gambar 10. Probabilitas transisi ke II



Gambar 11. Nilai Probabilitas Markov dari kata ‘ada’

Tabel 5. Rekap kosakata berdasarkan awalan huruf

Huruf (α)	Jumlah	Nilai Probabilitas Kata berawalan Huruf (α)
A	29	0.03448276
B	103	0.009708738
C	18	0.05555556
D	31	0.032258064
E	5	0.2
F	0	0
G	17	0.05882353
H	29	0.03448276
I	17	0.058823533
J	23	0.04347826
K	76	0.013157896
L	35	0.02857143
M	134	0.007462686
N	9	0.11111111
O	3	0.33333334
P	62	0.016129034
Q	0	0
R	19	0.05263158
S	69	0.014492754
T	73	0.013698632
U	15	0.06666667
V	0	0
W	5	0.2
X	0	0
Y	3	0.33333334
Z	0	0
Total	**775**	-

Pada *database* atau *file* teks kamus untuk kalimat Indonesia ke Jawa terdapat 775 kosakata, sedangkan kamus untuk kalimat Jawa ke Indonesia terdapat terdapat 1530 kata. Tabel 5 adalah rekap jumlah kata yang memiliki awalan huruf atau karakter yang sama . Dari tabel di atas terlihat, bahwa nilai probabilitas kata berawalan huruf “M”, memiliki nilai probabilitas paling kecil yaitu 0.007462686 dengan jumlah kata yang berawalan huruf “M” sebanyak 134 kata. Untuk nilai probabilitas kata berawalan huruf “F”,”Q”,”V”, dan “Z” adalah 0, karena tidak ada kata pada *database* kamus yang berawalan huruf tersebut. Sedangkan nilai probabilitas kata berawalan huruf sama yang paling besar ada pada huruf “Y”dan “O” , yaitu 0.33333334, karena memiliki jumlah kata paling sedikit. Nilai total probabilitas dari setiap awalan huruf (α) selalu berjumlah 1, sehingga nilai 1 tersebut akan terbagi nilai

probabilitasnya terhadap banyaknya jumlah kata yang ada pada *database* kamus. Banyaknya jumlah kata yang ada pada *database* kamus, menunjukkan bahwa himpunan kata tersebut yang sering digunakan di masyarakat pada umunya, yaitu himpunan kata yang berawalan huruf "M", dengan jumlah kata 134. Tabel 6 menunjukkan rekap dari proses markov, yang dicompile melalui program yang telah dibuat.

Data pada Tabel 6 terdiri dari nilai probabilitas markov dan rangkaian *state* karakter yang membentuk suatu kata, yang berawalan huruf 'A. terdapat 29 kosakata yang berawalan huruf 'A', dimana jumlah kata tersebut sesuai dengan jumlah kata yang ada pada *database* kamus.Terlihat bahwa untuk kata yang memiliki awalan huruf yang sama, akan memiliki nilai probabilitas yang sama dan ini membentuk suatu pola. Setelah di dapatkan pola, selanjutnya akan dilakukan proses *searching* dengan metode *binary*. Proses *searching* tersebut dilakukan tanpa harus mencari ke semua kata yang ada pada *database*, tetapi cukup pada himpunan kata yang memiliki awalan huruf sama yang sudah tersimpan pada *ArrayList()*. Diharapkan dengan adanya proses markov pada *searching*, akan mempersingkat waktu dalam pencarían kata.

Tabel 6. Rekap Data Probabilitas dari Proses Markov

No	Kata Indonesia	Nilai Probabilitas Markov	Urutan *State* Huruf
1	abang	0.03448276	P(b|a)=1/29 * P(a|b)=1/1 * P(n|a)=1/1 * P(g|n)=1/1
2	ada	0.03448276	P(d|a)=2/29 * P(a|d)=1/2
3	adik	0.03448276	P(d|a)=2/29 * P(i|d)=1/2 * P(k|i)=1/1
4	agak	0.03448276	P(g|a)=2/29 * P(a|g)=2/2 * P(k|a)=1/2
5	agama	0.03448276	P(g|a)=2/29 * P(a|g)=2/2 * P(m|a)=1/2 * P(a|m)=1/1
6	air	0.03448276	P(i|a)=2/29 * P(r|i)=2/2
7	air bah	0.03448276	P(i|a)=2/29 * P(r|i)=2/2 * P(|r)=1/1 * Pb(b|)=1/1 * P(a|b)=1/1 * P(h|a)=1/1
8	ajar	0.03448276	P(j|a)=1/29 * P(a|j)=1/1 * P(r|a)=1/1
9	akan	0.03448276	P(k|a)=1/29 * P(a|k)=1/1 * P(n|a)=1/1
10	alat pikul	0.03448276	P(l|a)=4/29 * P(a|l)=1/4 * P(t|a)=1/1 * P(|t)=1/1 * P(p|)=1/1 * P(i|p)=1/1 * P(k|i)=1/1 * P(u|k)=1/1 * P(l|u)=1/1
11	alis	0.03448276	P(l|a)=4/29 * P(i|l)=1/4 * P(s|i)=1/1

Tabel 6. Rekap Data Probabilitas dari Proses Markov (lanjutan)

No	Kata Indonesia	Nilai Probabilitas Markov	Urutan *State* Huruf
12	almarhum	0.03448276	P(l|a)=4/29 * P(m|l)=2/4 * P(a|m)=2/2 * P(r|a)=2/2 * P(h|r)=1/2 * P(u|h)=1/1 * P(m|u)=1/1
13	almari	0.03448276	P(l|a)=4/29 * P(m|l)=2/4 * P(a|m)=2/2 * P(r|a)=2/2 * P(i|r)=1/2
14	amarah	0.03448276	P(m|a)=4/29 * P(a|m)=2/4 * P(r|a)=1/2 * P(a|r)=1/1 * P(h|a)=1/1
15	amat	0.03448276	P(m|a)=4/29 * P(a|m)=2/4 * P(t|a)=1/2
16	ambil	0.03448276	P(m|a)=4/29 * P(b|m)=1/4 * P(i|b)=1/1 * P(l|i)=1/1
17	ampun	0.03448276	P(m|a)=4/29 * P(p|m)=1/4 * P(u|p)=1/1 * P(n|u)=1/1
18	anak	0.03448276	P(n|a)=4/29 * P(a|n)=1/4 * P(k|a)=1/1
19	ani-ani	0.03448276	P(n|a)=4/29 *P(i|n)=1/4 * P(-|i)=1/1 * P(a|-)=1/1 * P(n|a)=1/1 * P(i|n)=1/1
20	anjing	0.03448276	P(n|a)=4/29 * P(j|n)=1/4 * P(i|j)=1/1 * P(n|i)=1/1 *P(g|n)=1/1
21	antara	0.03448276	P(n|a)=4/29 * P(t|n)=1/4 * P(a|t)=1/1 * P(r|a)=1/1 * P(a|r)=1/1
22	apa	0.03448276	P(p|a)=2/29 * P(a|p)=1/2
23	api	0.03448276	P(p|a)=2/29 * P(i|p)=1/2
24	arti	0.03448276	P(r|a)=1/29 * P(t|r)=1/1 * P(i|t)=1/1
25	asalkan	0.03448276	P(s|a)=1/29 * P(a|s)=1/1 * P(l|a)=1/1 * P(k|l)=1/1 * P(a|k)=1/1 * P(n|a)=1/1
26	atau	0.03448276	P(t|a)=1/29 * P(a|t)=1/1 * P(u|a)=1/1
27	ayah	0.03448276	P(y|a)=3/29 * P(a|y)=2/3 * P(h|a)=1/2
28	ayam	0.03448276	P(y|a)=3/29 * P(a|y)=2/3 * P(m|a)=1/2
29	ayo	0.03448276	P(y|a)=3/29 * P(o|y)=1/3
TOTAL PROBABILITAS		1	-

Kesimpulan Dari penelitian yang telah dilakukan dapat diambil kesimpulan yaitu:

1. Suatu pola akan terbentuk jika kata yang berawalan huruf sama memiliki nilai probabilitas yang sama
2. Semakin banyak jumlah kata yang berawalan sama maka nilai probabilitas juga akan semakin kecil. Hal ini ditunjukkan dengan kata berawalan huruf "M", yang memiliki nilai probabilitas paling kecil yaitu 0.007462686 karena jumlah kata yang berawalan huruf "M" paling banyak yaitu sebesar 134 kata.

5. DAFTAR PUSTAKA

[1] S. Sry Satriya Tjatur Wisnu, *Unggah Ungguh Bahasa Jawa*, Yayasan Paramalingua, Jakarta, 2004.

[2] S. Theodoridis, K. Koutroumbas, *Pattern Recognition,* Academic Press, 2006

[3] Dan Jurafsky, James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition,* Prentice Hall, 2009.

[4] L. Kallmeyer,*Parsing Beyond Context-Free Grammars,* Springer, 2010.

[5] Reno Leermakers, "The Functional Treatment of Parsing", *The Kluwer international series in engineering and computer science*, ISBN 0-7923-9376-7, Volume 21, Number 1,1993

[6] M. H. A. Davis, *Markov Models and Optimization,* Chapman & Hall, 1993.

.