PENERAPAN ALGORITMA *ZHU-TAKAOKA*, *LEVENSHTEIN DAN CYK* (*COCKE-YOUNGER-KASAMI*) PADA KASUS PERBAIKAN KESALAHAN PENULISAN KATA DAN PERBAIKAN STRUKTUR KALIMAT DALAM BAHASA INDONESIA

Aep Saepudin[1], Adam Mukharil Bachtiar[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail: 1, adam@email.unikom.ac.id[2]

ABSTRAK

Bahasa ialah komunikasi yang paling lengkap dan efektif untuk menyampaikan ide, pesan, maksud, perasaan dan pendapat kepada orang lain. Suatu tata bahasa harus bisa dibuktikan secara ilmiah oleh setiap orang, di setiap tempat dan pada setiap waktu. Oleh karena itu, seorang penulis harus memperhatikan penulisan dan struktur kalimat dalam suatu bahasa karena penulisan dan struktur kalimat akan berpengaruh dalam proses penyampaian dan penerimaan pesan. Hal inilah yang mendasari perlunya diadakan penelitian tentang perbaikan kesalahan penulisan kata dan pemeriksaan struktur kalimat.

Algoritma yang digunakan dalam perbaikan kata dan perbaikan struktur kalimat, di antaranya algoritma Zhu-Takaoka, Levenshtein dan CYK (Cocke-Younger-Kasami). Algoritma Zhu-Takaoka merupakan salah satu algoritma pencocokan string. Dalam proses pencarian string algoritma *Zhu-Takaoka* terdapat beberapa tahap, yaitu preprocessing, right-to-left scan, bad character rule dan good-suffix rule. Algoritma Levenshtein merupakan algoritma yang digunakan untuk mencari jumlah operasi string yang paling sedikit untuk menstransformasikan suatu string menjadi string yang lain. Algoritma CYK merupakan salah satu algoritma yang dapat mengenali apakah sebuah string dapat dikatakan sebagai kalimat. Dari ketiga algoritma tersebut akan dilakukan penerapan, sehingga mengetahui algoritma Zhu-Takaoka, Levenshtain dan CYK dalam memberikan rekomendasi perbaikan kata dan perbaikan struktur kalimat.

Berdasarkan hasil pengujian yang dilakukan, didapat hasil dari penerapan algoritma Zhu-Takaoka, Levenshtein dan CYK dalam kasus perbaikan kesalahan penulisan kata dan perbaikan struktur dalam bahasa Indonesia menyatakan bahwa hasil pengujian perbaikan kata didapatkan nilai sebesar 76, 4%. Begitu juga dengan pengujian struktur didapat nilai sebesar 84, 21%. Sehingga dapat disimpulkan bahwa dari ketiga algoritma tersebut sudah cukup baik dalam memperbaiki kata dan memperbaiki struktur kalimat.

Kata kunci : Bahasa, Algoritma, Zhu-Takaoka, CYK, Levenshtein, Perbaikan Kata, Perbaikan Struktur

1. PENDAHULUAN

1.1 Latar Belakang

Bahasa ialah komunikasi yang paling lengkap dan efektif untuk menyampaikan ide, pesan, maksud, perasaan dan pendapat kepada orang lain [1]. Suatu tata bahasa harus bisa dibuktikan secara ilmiah oleh setiap orang, di setiap tempat dan pada setiap waktu [2].

Oleh karena itu, seorang penulis harus memperhatikan penulisan dan struktur kalimat dalam suatu bahasa karena penulisan dan struktur kalimat akan berpengaruh dalam proses penyampaian dan penerimaan pesan. Hal inilah yang mendasari perlunya diadakan penelitian tentang perbaikan kesalahan penulisan kata dan pemeriksaan struktur kalimat.

Berdasarkan penelitian Alinda dan Kania evita dewi pada "Perbandingan Algoritma Reverse Colusi dengan Algoritma Zhu-Takaoka dalam Pencocokan String". Algoritma Zhu-Takaoka dan Reverse Colusi memiliki tingkat akurasi sebanyak 100% dalam pencocokan string, kemudian Algoritma Reverse Colussi membutuhkan waktu rata-rata preprocessing selama 15.08 ms dan waktu rata-rata pencocokan pattern selama 0.13 ms sedangkan Algoritma Zhu-Takaoka membutuhkan waktu rata-rata preprocessing selama 5.43 ms dan waktu rata-rata pencocokan pattern selama 0.05 ms. Dapat disimpulkan bahwa dari kedua algoritma pengembangan Boyer Moore, algoritma Zhu Takaoka lebih cepat pada proses pencocokan string [3], algoritma Zhu-takaoka hanya bisa mencocokkan kata tidak bisa memperbaiki kata sehingga dibutuhkan algoritma yang bisa memperbaiki kata.

Berdasarkan penelitian Ni Made Muni Adiyani, I Wayan Santiyasa, Agus Muliantara pada "Implementasi Algoritma Levenshtein Distance dan Metode Empiris untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia", terdapat cara untuk membantu mengatasi permasalahan pada kesalahan

pengetikan kata [4]. Karena pada penelitian ini masih sebatas pengecekan kesalahan pengetikan bukan pengecekan struktur kalimat, maka dibutuhkan algoritma untuk pengecekan struktur kalimat.

Berdasarkan penelitian Sri Susanti pada "Analisis Perbandingan Algoritma LCP(Left-Corner-Parsing) dan Algoritma CYK(Cocke-Younger-Kasami) untuk Memeriksa Pola Kalimat Baku Bahasa Indonesia", terdapat cara untuk mengenali struktur kalimat dengan menggunakan algoritma LCP dan CYK. Dari hasil pengujian yang dilakukan Algoritma CYK memiliki akurasi sebesar 65% dengan laju error 0, 35 dan Algoritma LCP memiliki akurasi sebesar 60% dengan laju error 0, 40 [5]. Sehingga dapat disimpulkan bahwa Algoritma CYK lebih baik dalam mengenali struktur kalimat.

Berdasarkan permasalahan di atas, penelitian ini akan dilakukan penerapan algoritma Zhu-takaoka, Levenshtein dan CYK untuk memperbaiki kesalahan penulisan kata dan perbaikan struktur kalimat bahasa Indonesia pada abstrak skripsi. Algoritma Zhu-Takoaka digunakan untuk pencarian kata, algoritma Levenshtein digunakan untuk perbaikan kata dan memperbaiki struktur kalimat dan algoritma CYK digunakan untuk mengenali *tag* kata.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dari penelitian ini adalah bagaimana menerapkan *algoritma Zhu-Takaoka*, *Levenshtein* dan CYK untuk kesalahan penulisan kata dan perbaikan struktur kalimat dalam bahasa Indonesia.

1.3 Maksud dan Tujuan

Adapun maksud dari penelitian ini adalah menerapkan algoritma *Zhu-Takoaka, Levenshtein* dan CYK pada simulator untuk perbaikan kesalahan penulisan kata dan struktur kalimat.

Tujuan penelitian ini adalah untuk mengetahui algoritma *Zhu-Takaoka, Levenshtain* dan CYK dapat memberikan rekomendasi perbaikan kesalahan penulisan kata dan struktur kalimat dalam bahasa Indonesia pada abstrak skripsi.

1.4 Batasan Masalah

Batasan masalah pada algoritma *Zhu-Takaoka*, *Levenshtein* dan CYK dalam kasus perbaikan kesalahan penulisan kata adalah sebagai berikut.

1 Pengecekan terbatas hanya pada bahasa Indonesia.
2 Kamus dibuat berdasarkan POS *Tag* Indonesia.
3 Dokumen yang diperiksa berupa inputan dari *file* dengan ekstensi .doc atau *.docx.*
4 Pengecekan kalimat hanya kalimat sederhana dan kalimat luas.
5 Pengecekan kata tidak termasuk mengenali kata yang ambigu.
6 Data uji yang digunakan adalah dokumen abstrak skripsi dari elib Unikom.
7 Bahasa pemrograman yang digunakan adalah *C#*.

1.5 *Text Mining*

Text mining yang juga disebut sebagai Text Data Mining (TDM) atau Knowledge Discovery in Text (KDT), secara umum mengacu pada proses ekstraksi informasi dari dokumen-dokumen teks tak terstruktur (unstructured). Teks mining dapat didefinisikan sebagai penemuan informasi baru dan tidak diketahui sebelumnya [6].

Tujuan utama teks mining (text mining) adalah mendukung proses knowledge discovery pada koleksi dokumen yang besar. Pada prinsipnya, teks mining adalah bidang ilmu multidisipliner, melibatkan information retrieval (IR), text analysis, information extraction (IE), clustering, categorization, visualization, database technology, natural language processing (NLP), machine learning, dan data mining. Dapat pula dikatakan bahwa teks mining merupakan salah satu bentuk aplikasi kecerdasan buatan (artifisial intelligence / AI).

1.6 Sintaksis

Sintaksis membicarakan penataan dan pengaturan kata-kata ke dalam satuan yang lebih besar, yang disebut satuan-satuan sintaksis, yakni kata, frase, klausa, kalimat, dan wacana [7].

1.6.1 Frase

Frase adalah satuan sintaksis yang tersusun dari dua buah kata atau lebih, yang didalam klausa menduduki fungsi-fungsi sintaksis [7]. Frase dibagi menjadi beberapa kelompok:

1. Frase Nomina Koordinatif
   Frase ini dapat disusun dari dua buah kata berkategori nomina yang merupakan pasangan dan antonim relasional.
2. Frase Nomina Subordinatif
   Frase ini dapat disusun dari nomina + nomina (N + N), nomina + verba (N + V), nomina + ajektifa (N + A), adverbia + nomina (Adv + N), nomina + adverbia (N + Adv), nomina + numeralia (N + Num), numeralia + nomina (Num + N), dan nomina + demonstratifa (N + Dem).
3. Frase Verba Koordinatif
   Frase ini dapat disusun dari Dua buah kata berkategori verba yang merupakan anggota dari antonim relasional dan memiliki makna gramatikal 'menggabungkan' sehingga di antara keduanya dapat disisipkan kata 'dan'.
4. Frase Verba Subordinatif
   Frase ini dapat disusun dari adverbia + verba (Adv + V), verba + adverbia (V + Adv), verba + nomina (V + N), verba + ajektifa (V + A).
5. Frase Ajektifa Koordinatif
   Frase ini dapat disusun dari:
   a. Dua buah kata berkategori ajektifa yang merupakan anggota dari antonim relasional dan memiliki makna gramatikal 'pilihan'.

b. Dua buah kata berkategori ajektifa yang merupakan anggota dari pasangan bersinonim.
c. Dua buah kata berkategori ajektifa yang maknanya sejalan tidak bertentangan dan memiliki makna gramatikal ‘himpunan’.
d. Dua buah kata berkategori ajektifa yang maknanya tidak sejalan (bertentangan) dan memiliki makna ‘berkebalikan’.

6. Frase Ajektifa Subordinatif
   Frase ajektifa subordinatif disusun dengan struktur A + N, A + A, A + V, Adv + A, dan A + Adv.
7. Frase Preposisional
   Frase preposisional adalah frase yang berfungsi sebagai pengisi fungsi keterangan di dalam sebuah klausa.

1.6.2 Klausa

Klausa adalah satuan sintaksis yang bersifat prediktif. Artinya, di dalam satuan atau konstruksi itu terdapat sebuah predikat, bila di dalam satuan tidak terdapat predikat, maka satuan itu bukan disebut klausa. Kedudukan predikat ini sangat penting, sebab jenis dan kategori dari predikat itulah yang menentukan hadirnya fungsi subjek (S), fungsi objek (O), fungsi pelengkap, dan sebagainya [7]. Berdasarkan kategori yang mengisi fungsi P dapat dibedakan menjadi:

1. Klausa Verba (KV)
   Secara semantik ada tiga buah jenis verba, yaitu verba tindakan, verba kejadian, dan verba keadaan.
   a. Klausa Verba Tindakan
      Klausa verba tindakan bersasaran tak berpelengkap dapat disusun dari sebuah verba berkomponen makna (+ tindakan) dan (+ sasaran), sehingga klausanya memiliki fungsi sintaksis S, P, O.
      Klausa tindakan bersasaran berpelengkap dapat disusun dari sebuah verba berkomponen makna (+ tindakan), (+ sasaran), dan (+ pelengkap), sehingga klausanya memiliki fungsi S, P, O, dan Pel.
      Klausa verba tindakan tidak bersasaran dapat disusun dari sebuah verba yang memiliki komponen makna (+ tindakan) dan (- sasaran), sehingga klausanya hanya memiliki fungsi S dan fungsi P.
   b. Klausa Verba Kejadian
      Klausa verba kejadian dapat disusun dari predikat verba yang memiliki komponen makna (+ kejadian). Dalam hal fungsi sintaksis yang wajib hadir adalah fungsi S dan fungsi P. Fungsi S berupa nomina yang mengalami kejadian seperti disebutkan oleh predikat.
   c. Klausa Verba Keadaan
      Klausa Verba keadaan dapat disusun dari predikat verba yang memiliki komponen makna (+ keadaan). Dalam hal ini fungsi sintaksis yang muncul hanyalah fungsi S dan fungsi P. Fungsi S berupa nomina yang mengalami keadaan seperti yang disebutkan oleh predikat.
2. Klausa Nomina
   Klausa nomina hanya memiliki fungsi wajib S dan P. Klausa nomina ini dapat disusun dari fungsi S yang berupa kata atau frase berkategori nomina dan P yang berupa kata atau frase berkategori nomina. Klausa nomina, antara lain, dapat disusun jika:
   a. Nomina yang mengisi fungsi S merupakan jenis (spesifik) dari nomina pengisi fungsi P (generik).
   b. Nomina yang mengisi fungsi S mempunyai nama pada nomina pengisi fungsi P.
   c. Nomina pengisi fungsi P adalah profesi (jabatan, pekerjaan) bagi nomina pengisi fungsi S.
   d. Nomina pengisi fungsi P adalah relasi bagi nomina pengisi fungsi S.
   e. Nomina pengisi fungsi S mempunyai ciri atau sifat khas yang disebutkan oleh nomina pengisi fungsi S.
3. Klausa Ajektifa
   Klausa ajektifa memiliki fungsi wajib S dan P. Klausa ajektifa dapat disusun dari fungsi S yang berkategori N dan fungsi P yang berkategori A. Klausa ajektifa ini dapat disusun jika:
   a. Fungsi P yang berkategori ajektifa memiliki komponen makna (+ keadaan fisik).
   b. Fungsi P yang berkategori ajektifa memiliki komponen makna (+ sifat batin).
   c. Fungsi P yang berkategori ajektifa memiliki komponen makna (+ perasaan batin).
4. Klausa Preposisional
   Klausa preposisional adalah klausa yang fungsi P nya diisi oleh frase preposisional.
5. Klausa Numeral
   Klausa numeral adalah klausa yang fungsi P nya diisi oleh frase numeral.

1.6.3 Kalimat

Kalimat adalah satuan sintaksis yang disusun dari konstituen dasar, yang biasanya berupa klausa, dilengkapi dengan konjungsi bila diperlukan, serta disertai dengan intonasi *final*. Intonasi *final* dapat dilambangkan dengan tanda titik, tanda seru, atau tanda tanya [7].

1. Kalimat Sederhana
   Kalimat sederhana adalah kalimat yang dibentuk dari sebuah klausa dasar atau klausa sederhana, yaitu klausa yang fungsi-fungsi sintaksisnya hanya diisi oleh sebuah kata atau sebuah frase sederhana.
2. Kalimat Luas

Sebuah kalimat biasanya terangkum informasi lebih banyak di dalamnya dan disebut dengan kalimat luas.

1.7 *Chomsky Normal Form* (CNF)

CNF merupakan salah satu bentuk normal yang sangat berguna untuk CFG. Bentuk CNF dapat dibuat dari sebuah CFG yang telah mengalami penyederhanaan dengan menghilangkan produksi *useless*, unit, dan *empty* (himpunan kosong yang dilambangkan dengan ε) [8]. Contoh CFG yang akan disederhanakan dapat dilihat pada aturan (1).

$S \rightarrow bC \mid aB \mid AB$
$A \rightarrow bAC \mid a$
$B \rightarrow aCB \mid b$
$C \rightarrow cBA \mid c$ (1)

Hasil aturan produksi dapat dilihat pada aturan (2).

$S \rightarrow V_1C$
$S \rightarrow V_2B$
$S \rightarrow AB$
$A \rightarrow V_1V_3$
$A \rightarrow V_2$
$B \rightarrow V_2V_4$
$B \rightarrow V_3$
$C \rightarrow V_5V_6$
$C \rightarrow V_3$
$V_1 \rightarrow b$
$V_2 \rightarrow a$
$V_3 \rightarrow AC$
$V_4 \rightarrow CB$
$V_5 \rightarrow c$
$V_6 \rightarrow BA$ (2)

1.8 POS Tag

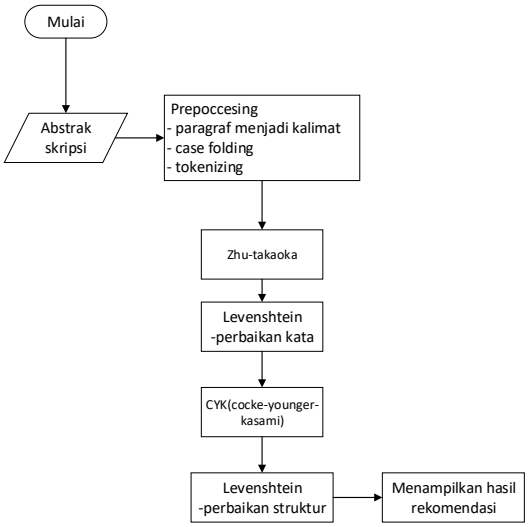
POS (*Part-of-Speech*) Tag merupakan suatu cara pengkategorian kelas kata, seperti kata benda, kata kerja, kata sifat, dan lain-lain. POS *Tagging* adalah suatu aktivitas menganotasi setiap kata/token dengan nilai *part-of-speech* tag yang sesuai. Informasi nilai POS Tag ini merupakan hal yang mendasar bagi keperluan pengolahan bahasa manusia [9]. POS Tag yang digunakan sebagai bantuan dalam mengenali token/tag setiap kata diambil dari POS Tag Indonesia. POS Tag akan mengenali kata mana yang termasuk kata benda, kata kerja, kata sifat, kata keterangan, kata depan, kata sambung, kata ganti benda, dan kata bilangan. POS Tag ini menggunakan pendekatan *Rule-Based* berdasarkan aturan tata bahasa Indonesia. Pertama-tama, POS Tag akan melakukan tokenisasi terhadap teks menggunakan kamus bahasa Indonesia. Selanjutnya kata-kata yang termasuk ke dalam jenis *closed-class word* diproses. Lalu setiap kata yang ambigu diproses menggunakan aturan-aturan yang sudah didefinisikan untuk menemukan kelas kata yang tepat.

Tabel 1 Aturan produksi

Nama Kelompok Kata	Diubah Menjadi
Noun / Kata Benda (N)	NN, NND, NNP, PRP, atau WH
Adjective / Kata Sifat (A)	JJ
Verb / Kata Kerja (V)	VB
Adverb / Kata Keterangan (Adv)	RB
Numeralia (Num)	OD dan CD
Preposisi (P)	IN
Demonstrative (Dem)	PR
Konjungsi koordinatif	CC
Artikula	DT
Verba bantuan	MD
Kata ingkar	NEG
Partikel	RP
Konjungsi subordinatif	SC
Simbol	SYM
Tanda baca	Z

2. ISI PENELITIAN

2.1 Arsitektur Sistem



Gambar 1 Arsitektur sistem

2.1 Analisis Data Masukan

Data masukan untuk simulator ini berupa abstrak skripsi yang berekstensi .doc atau .docx.

2.2 Analisis Preprocessing

Cara yang digunakan dalam mempelajari suatu data teks, adalah dengan terlebih dahulu menentukan fitur-fitur yang mewakili setiap kata untuk setiap fitur yang ada pada dokumen. Sebelum menentukan fitur-fitur yang mewakili, diperlukan tahap *preprocessing* yang dilakukan secara umum dalam teks *mining* pada dokumen [6], yaitu pemecahan paragraf menjadi kalimat, *case folding*, dan *tokenizing*.

2.2.1 Proses Pemecahan Paragrag Menjadi Kalimat

Tabel 2 Proses pemecahan paragraf menjadi kalimat

Dokumen	Hasil pemecahan paragraf menjadi kalimat	
Teknologi permainan dalam kemajuan dari sisi desain nenarik. Teknologi pada objek krena desain baik satu kemajuan.	No	Kalimat
	1	Teknologi permainan dalam kemajuan dari sisi desain nenarik.
	2	Teknologi pada objek krena desain baik satu kemajuan.

2.2.2 Case Folding

Tabel 3 *Case folding*

Sebelum	Sesudah
Teknologi permainan dalam kemajuan dari sisi desain nenarik.	teknologi permainan dalam kemajuan dari sisi desain nenarik.
Teknologi pada objek krena desain baik satu kemajuan.	teknologi pada objek krena desain baik satu kemajuan.

2.2.3 Tokenizing

Tabel 4 *Tokenizing*

Kalimat	Karakter
teknologi permainan dalam kemajuan dari sisi desain nenarik.	– teknologi – permainan – dalam – kemajuan – dari – sisi – desain – nenarik – .
teknologi pada objek krena desain baik satu kemajuan.	– teknologi – pada – objek – krena – desain – baik – satu – kemajuan – .

2.3 Analisis Pencarian Kata dengan Algoritma Zhu-Takaoka

Algoritma *Zhu-Takaoka* merupakan salah satu algoritma pencocokan *string* (*String Matching*) yang merupakan pengembangan dari algoritma *Boyer-Moore*. Algoritma ini dipublikasikan oleh Zhu Rui Feng dan Tadao Takaoka pada tahun 1986 [3]. Dalam penelitiannya, Zhu dan Takaoka menyebut algoritma pencocokan *string* ini sebagai BM" *Algorithm* (*Boyer-Moore Algorithm*) karena merupakan modifikasi dari algoritma pencocokan *string Boyer-Moore*, yaitu algoritma yang dibuat oleh Boyer R.S. dan Moore J.S. Algoritma Zhu-Takaoka dan algoritma *Boyer-Moore* memiliki ciri yang sama dalam proses pencarian *string*, yaitu terdapat tahap *Preprocessing*, *Right-to-left scan*, *Bad character rule* dan *Good-suffix rule* [3].

2.3.1 Membuat Tabel ZtBC

Untuk memudahkan proses mencari tabel ztBC langkah awal adalah dengan menemukan Nilai OH, yaitu dengan mengidentifikasi setiap perpindahan, tabel ztBC sebelum diisi.

Tabel 5 OH awal

Index	0	1	2	1
Karakter	d	a	r	i
Nilai OH	?	?	?	?

Untuk mengisi Nilai OH, langkah pertama adalah dengan melakukan pencacahan di setiap karakter pada *pattern*. Pencacahan dimulai dari karakter ke-2 paling kanan (*Index = Length* (*pattern*) - 2) dari *pattern*. Untuk langkah pertama perpindahan diisi dengan angka 1.

Tabel 6 Pencacahan pertama

Mulai : *Index = length(pattern) – 2* = 4 – 2 = 2				
Index	0	1	2	3
Karakter	d	a	r	i
Pindah = 1				

Bandingkan setiap Bandingkan setiap karakter yang dicacah terhadap karakter yang terdapat pada tabel ztBC sebelumnya, jika karakter yang dicacah tidak ditemukan dalam tabel tersebut, maka tambahkan karakter tersebut ke dalam ztBC dan Nilai OH (*Occurrence Heuristic*) diisi dengan jumlah pindah karakter yang sedang dibandingkan. Untuk perbandingan pertama, bandingkan karakter pertama dengan *Null.*

Tabel 7 Perbandingan pertama

Bandingkan d dengan *null*		Hasil	
ztBC(awal)		ztBC(akhir)	
Karakter	null	Karakter	r
Nilai OH	null	Nilai OH	1

Tabel 8 Hasil akhir nilai OH

Index	0	1	2	3
Karakter	d	a	r	i
Nilai OH	3	2	1	4

2.3.2 Membuat tabel ZtGS

Setelah mendapatkan hasil akhir pada tabel ztBC, maka dilanjutkan dengan membuat tabel akhiran/*suffix* dari *pattern*.

Tabel 9 Awal ZtGS

ztGS				
Index	0	1	2	3
Karakter	d	a	r	i
Nilai MH	?	?	?	?

Tabel 10 Proses pertama ZtGS

Process #1-01 suffix tabel		
Index	3	Move
Prefix	i	
Suffix	null	
Suffix comparator	dar	1
	da	2
	d	3
	null	4
Nilai MH		

ztGS				
Index	0	1	2	3
Karakter	d	a	r	i
Nilai MH	?	?	?	1

Tabel 11 Akhir ZtGS

Process #4-01 suffix tabel		
Index	2	move
Prefix	d	
Suffix	ari	
Suffix comparator	dar	1
	da	2
	d	3
	null	4
Nilai MH	?	

Process #4-02 suffix tabel		
Index	2	move
Prefix	d	
Suffix	ari	
Suffix comparator	dar	1
	da	2
	d	3
	null	4
Nilai MH	?	

Process #4-03 suffix tabel		
Index	2	move
Prefix	d	
Suffix	ari	
Suffix comparator	dar	1
	da	2
	d	3
	null	4
Nilai MH	?	

Process #4-04 suffix tabel		
Index	2	move
Prefix	d	
Suffix	ari	
Suffix comparator	dar	1
	da	2
	d	3
	null	4
Nilai MH	4	

ztGS				
Index	0	1	2	3
Karakter	d	a	r	i
Nilai MH	4	4	4	1

2.3.3 Pencarian Kata

Sumber kata : baikdalamdaridesainkarenakerenkemajuanmenariobjekpadapermainansatusisiteknologi

Pattern : dari

Pencocokan 1

Index	1	2	3	4	5	6	7	8	9
Sumber kata	b	a	i	k	d	a	l	a	m
Pattern	d	a	r	i					

Dari hasil pencocokan ke-1, dapat dilihat bahwa karakter akhir sumber *string* yang dicari yaitu karakter i sejajar dengan karakter k. Artinya, pada pencocokan ke-1 terjadi ketidakcocokan, maka dilakukan pergeseran sejauh empat karakter. Nilai pergeseran 4 ini diperoleh dari tabel ztBC di mana dicocokannya dua karakter akhri yang tidak sama. Maka cek pada tabel ztBC baris i dan kolom k maka bernilai 4.

Pencocokan 5

Index	17	18	19	20	21	22	23
Sumber kata	d	a	r	i	d	e	s
Pattern	d	a	r	i			

Dari hasil pencocokan ke-5, dapat dilihat bahwa karakter akhir yang dicari yaitu karakter i sejajar dengn karakter i pada kamus. Artinya, pada pencocokan ke-5 terjadi kecocokan, maka pencocokan dilakukan terhadap karakter sebelum karakter a, yaitu r, a, d pada sumber kata, bandingkan dengan karakter yang sejajar terjadi kecocokan. Kemudian algoritma menandai lokasi penemuan dan untuk melanjutkan pencocokan gunakan pergeseran. maka nilai pergeseran selanjutnya di ambil dari tabel ztGS dengan karakter d maka pergeseran sejauh 4 karakter.

2.4 Perbaikan Kata Dengan Algoritma Levenshtein

Algoritma Levenshtein merupakan algoritma pencarian jumlah perbedaan *string* yang ditemukan oleh Vladimir Levenshtein, seorang ilmuwan Rusia, pada tahun 1965. Algoritma ini digunakan secara luas dalam berbagai bidang, misalnya mesin pencari, pengecek ejaan (*spell checking*), pengenalan pembicaraan (*speech recognition*), pengucapan dialek, analisis DNA, dan lain-lain. Dua buah *string* dibandingkan untuk mengetahui nilai *edit distance*-nya. *Edit distance* adalah suatu pengukuran (metrik) yang dihasilkan melalui perhitungan jumlah perbedaan yang terdapat pada dua *string* [4].

Setiap kata yang dianggap keliru akan dibandingkan dengan setiap kata yang ada pada *library* yang sudah dibuat kemudian dicari nilai *edit distance* yang terkecil dengan algoritma Levenshtein.

Contoh kata nenarik dibandingkan dengan menarik.

Tabel 12 Perbandingan "nenarik dan menarik"

		n	e	n	a	r	i	k
	0	1	2	3	4	5	6	7
m	1	1	2	3	4	5	6	7
e	2	2	1	2	3	4	5	6
n	3	2	2	1	2	3	4	5
a	4	3	3	2	1	2	3	4
r	5	4	4	3	2	1	2	3
i	6	5	5	4	3	2	1	2
k	7	6	6	5	4	3	2	1

Contoh kata nenarik dibandingkan dengan menari.

Tabel 13 Perbandingan “nenarik dan menari”

		n	e	n	a	r	i	k
	0	1	2	3	4	5	6	7
m	1	1	2	3	4	5	6	7
e	2	2	1	2	3	4	5	6
n	3	2	2	1	2	3	4	5
a	4	3	3	2	1	2	3	4
r	5	4	4	3	2	1	2	3
i	6	5	5	4	3	2	1	2

Tabel 14 Hasil perbandingan kata

Kata	Saran perbaikan	Nilai edit distance
nenarik	menari	2
	menarik	1

2.5 Algoritma Cocke-Younger-Kasami (CYK)

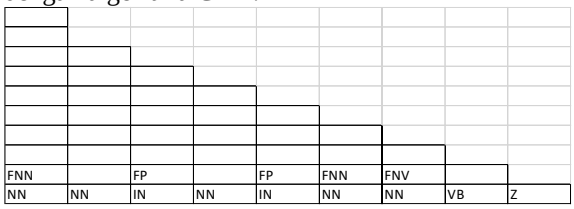
Algoritma CYK merupakan algoritma *parsing* dan keanggotaan (*membership*) untuk tata bahasa bebas konteks. Syarat untuk menggunakan algoritma ini adalah tata bahasa harus berada dalam bentuk CNF [5]. Tujuan dari algoritma ini adalah untuk menunjukkan apakah suatu *string* dapat diterima dari suatu tata bahasa.

Langkah awal pemeriksaan pola kalimat dengan algoritma CYK adalah melakukan pengisian baris dari tabel dua dimensi algoritma CYK dengan melihat variabel pada dua blok, kemudian dilihat variabel mana yang menghasilkan kedua variabel tersebut. Blok selanjutnya diisi dengan langkah yang sama sesuai dengan blok yang akan diisi. Pengisian pada blok kedua dilakukan dengan mengambil variabel dari kolom kedua dan ketiga.

Tabel 15 Pengisian blok

Kalimat	Blok	Tag 1	Tag 2	Aturan Produksi
Ke 1	[1,1] & [1,2]	NN	NN	FNN
	[1,2] & [1,3]	NN	IN	-
	[1,3] & [1,4]	IN	NN	FP
	[1,4] & [1,5]	NN	IN	-
	[1,5] & [1,6]	IN	NN	FP
	[1,6] & [1,7]	NN	NN	FNN
	[1,7] & [1,8]	NN	VB	FNV
	[1,8] & [1,9]	VB	Z	-

Seperti penjelasan yang sudah dijelaskan tentang langkah-langkah algoritma CYK maka hasil akhir dari pemeriksaan pola kalimat bahasa Indonesia dengan algoritma CYK.



Gambar 2 Hasil akhir pemeriksaan

2.5 Algoritma Levenshtein

Setiap struktur yang dianggap keliru akan dibandingkan dengan setiap struktur yang ada pada *library* yang sudah dibuat kemudian dicari nilai *edit distance* yang terkecil dengan algoritma Levenshtein. Contoh struktur "NN IN NN SC NN JJ CD NN Z" dibandingkan dengan "NN IN NN SC NN CD NN Z", "NN NN SC NN CD NN Z" dan "NN IN NN SC NN IN NN JJ Z"

Tabel 16 Hasil perbandingan struktur

Perbandingan	Struktur	Edit distance
1	NN IN NN SC NN CD NN Z	6
2	NN NN SC NN CD NN Z	9
3	NN IN NN SC NN IN NN JJ Z	9

3. PENGUJIAN

3.1 Pengujian Akkurasi

Pada pengujian ini digunakan sebanyak 5 abstrak dengan total 89 kata keliru dan 19 struktur kalimat yang dianggap keliru. Dokumen-dokumen yang diuji merupakan dokumen yang sudah diperiksa oleh ahli tata bahasa. Struktur kalimat yang benar disimpan dalam sebuah dokumen untuk dibandingkan hasilnya dengan sistem. Rekomendasi perbaikan diambil dari struktur yang sudah diperbaiki. Pada tahap awal dilakukan pengecekan kata menggunakan algoritma *zhu-takaoka*. Algoritma ini akan mengenali kata yang ada pada dokumen.

3.1.1 Pengujian Kata

Pengujian ini bertujuan untuk melihat berapa banyak kata yang dapat diperbaiki. Tahap pertama pencarain kata pada *library* kamus menggunakan algoritma *Zhu-Takaoka*. Pengujian selanjutnya dilakukan dengan pembandingan kata keliru dengan *library* kamus. Tahap ini juga merupakan pengujian sejauh mana algoritma *Levenshtein* ini dapat menemukan kata yang sesuai. Nilai *edit distance* yang digunakan adalah nilai yang terkecil dari perbandingan kata.

Tabel 17 Hasil pengujian kata

No	Nama Dokumen	Jumlah Kata Keliru	Jumlah Kata yang Sesuai
1	abstrakskripsi01	13	9
2	abstrakskripsi02	12	10
3	abstrakskripsi03	10	6
4	abstrakskripsi04	23	20
5	abstrakskripsi05	31	23
Jumlah Total		89	68

Dari hasil pengujian terdapat 89 kata yang mendapat perbaikan. Terdapat 21 kata tidak dapat diperbaiki. Jumlah kata yang mendapat perbaikan yang sesuai ada 68. Nilai keberhasilan dari pengujian perbaikan kata dapat diukur dengan cara:

$$\frac{jumlah\ total\ perbaikan\ kata\ yang\ sesuai}{total\ jumlah\ kata\ yang\ keliru}\ x\ 100$$

$$\frac{68}{89}\ x\ 100 = 76,4\%$$

Hasil pengujian didapatkan nilai sebesar 76, 4% untuk pengujian algoritma *Levenshtein* dalam memperbaiki kata.

3.1.2 Pengujian Struktur Kalimat

Pengujian ini bertujuan untuk melihat berapa banyak struktur kalimat yang dapat diberikan rekomendasi. Tahap pertama mengenali *tag* dari masing-masing kata dan mengenali aturan produksi dari masing-masing kalimat dengan menggunakan algoritma CYK. Pengujian selanjutnya dilakukan dengan pembandingan struktur kalimat yang keliru dengan *library* struktur. Tahap ini juga merupakan pengujian sejauh mana algoritma *Levenshtein* ini dapat menemukan struktur kalimat yang sesuai. Nilai *edit distance* yang digunakan adalah nilai yang terkecil dari perbandingan struktur kalimat. Pengujian struktur kalimat yang diperbaiki dapat dilihat pada lampiran D.

Tabel 18 Jumlah kalimat yang diberikan rekomendasi

No	Nama Dokumen	Jumlah struktur yang dianggap keliru	Jumlah struktur yang sesuai
1	abstrakskripsi01	3	3
2	abstrakskripsi02	4	1
3	abstrakskripsi03	2	2
4	abstrakskripsi04	7	7
5	abstrakskripsi05	3	3
Jumlah Total		19	16

Pada tabel 18 dapat dilihat bahwa jumlah kalimat yang sesuai perbaikannya dengan pola kalimat yang diperiksa oleh ahli tata bahasa berjumlah 16 kalimat dari 19 kalimat yang keliru. Nilai keberhasilannya dapat diukur dengan cara:

$$\frac{jumlah\ total\ perbaikan\ struktur\ yang\ sesuai}{total\ jumlah\ struktur\ yang\ keliru}\ x\ 100$$

$$\frac{16}{19}\ x\ 100 = 84,21\%$$

Hasil pengujian didapatkan nilai sebesar 84, 21% untuk pengujian algoritma *Levenshtein* dalam mencocokkan pola kalimat yang digunakan untuk rekomendasi perbaikan.

4. PENUTUP

4.1 Kesimpulan

Berdasarkan penelitian mengenai penerapan algoritma *Zhu-Takaoka, levenshtein* dan CYK. Pengujian berhasil mengetahui bahwa algoritma *Zhu-Takaoka* dapat mencari kata dengan baik, algoritma CYK dapat mengenali struktur kalimat dengan baik sesuai aturan produksi dan algoritma *Levenshtein* juga sangat baik dalam memperbaiki kata dan struktur kalimat. Hasil dari penelitian ini menunjukkan bahwa perbaikan kata didapatkan nilai sebesar 76,4% dan pengujian struktur kalimat didapat nilai sebesar 84, 21%. Nilai ini sudah cukup baik dalam memperbaiki kata dan struktur kalimat yang keliru namun masih terdapat kekurangan dari penelitian tentang penerapan ini karena algoritma *Levenshtein* menentukan perbaikan dari nilai edit *distance* yang terkecil, tidak dapat mengenali struktur kalimat yang seharusnya di satukan atau dipisah, tidak dapat mengenali kata yang seharusnya disatukan atau dipisah dan kamus POS *Tag* yang digunakan masih banyak kata yang tidak dikenali.

4.2 Saran

Berdasarkan hasil pengujian yang telah dilakukan, maka saran yang diharapkan setelah dilakukannya penerapan algoritma *Zhu-Takaoka, Levenshtein* dan CYK pada kasus perbaikan kesalahan penulisan kata dan perbaikan struktur kalimat dalam bahasa Indonesia :

1. Dapat menggunakan algoritma *approximate string matching* yang lebih baik pada perbaikan kata dan perbaikan struktur kalimat.
2. Dapat Menggunakan algoritma yang lebih baik pada pemeriksaan struktur kalimat.
3. Menggunakan POS *Tag* lain yang lebih banyak daftar kata karena masih banyak kata yang tidak ada, untuk meningkatkan rekomendasi perbaikan kata.

DAFTAR PUSTAKA

[1] Walija, "Bahasa Indonesia dalam Perbincangan," Jakarta, IKIP Muhammadiyah Jakarta Press, 1996.

[2] P. D. Iswara, "Penelitian Variasi Pola Kalimat," 24 Februari 2009. [Online]. Available: http://iswara.staf.upi.edu/2009/02/24/penelitian-variasi-pola-kalimat.. [Diakses 19 September 2016].

[3] Alinda dan Kania Evita Dewi, "perbandingan Algoritma Reverse Colussi dengan Algoritma Zhu-Takaoka dalam Pencocokan String," Jurnal Ilmiah Komputer dan Informatika (KOMPUTA), 2016.

[4] N. Made muni Adriyani, I. Wayan santiyasa dan A. Muliantara, *Implementasi Algoritma Levenshtein Distance dan Metode Empiris untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia*, pp. 1-7.

[5] Susanti, Sri, "Analisis Perbandingan Algoritma LCP(Left-Corner-Parsing) dan Algoritma CYK (Cocke-Younger-Kasami) untuk Memeriksa Pola Kalimat Baku Bahasa Indonesia," Ilmu Komputer dan Informatika (KOMPUTA), 2015.

[6] F. Muhammad, I. Veri, N. Muhammad, "Implementasi Algoritma Dynamic Programming untuk Koreksi Kesalahan Ejaan Kata Pada Search Engine," 2015.

[7] A. Chaer, Sintaksis Bahasa Indonesia (Pendekatan Proses), Jakarta: Rineka Cipta, 2009.

[8] F. Utdirartatmo, Teori Bahasa dan Automata, Yogyakarta: Graha Ilmu, 2005.

[9] A. Dinakaramani, F. Rashel, A. Luthfi dan R. Manurung, "POS Tag Indonesia," 20 October 2014. [Online]. Available: http://bahasa.cs.ui.ac.id/postag/corpus. [Diakses 20 March 2015].