CESS (Journal of Computer Engineering System and Science) **p-ISSN :2502-7131**
Vol. 4 No. 2 Juli 2019 **e-ISSN :2502-714x**

IDENTIFIKASI KESALAHAN TATA BAHASA PADA PERNYATAAN KEBUTUHAN MENGGUNAKAN PROBABILISTIK MODEL BAHASA N-GRAM

Depandi Enda[1], Fajri Profesio Putra[2]
[1,2] *Politeknik Negeri Bengkalis*
[1,2]*Jalan Bathin Alam, Sei. Alam, Bengkalis, Riau, 28711*
[1]*depandienda@polbeng.ac.id,* [2]*fajri@polbeng.ac.id*

Abstrak—**Berbagai aspek seperti kemampuan untuk penulisan tata bahasa, latar belakang penggunaan bahasa Inggris dan keterbatasan pengetahuan yang dimiliki oleh tim penyusun kebutuhan perangkat lunak memungkinkan adanya kesalahan dalam pembuatan dokumen spesifikasi kebutuhan perangkat lunak. Hal ini dapat menyebabkan menurunnya kualitas dokumen spesifikasi kebutuhan perangkat lunak tersebut. Untuk mengatasi permasalahan tersebut maka diperlukan sebuah metode untuk mengidentifikasi kesalahan tata bahasa pernyataan kebutuhan perangkat lunak. Penelitian ini bertujuan mengusulkan pengembangan metode untuk mengidentifikasi kesalahan tata bahasa pada pernyataan kebutuhan perangkat lunak. Metode yang diusulkan ialah teknik berbasis statistik menggunakan probabilitas model bahasa n-gram, dimana model bahasa yang digunakan ialah model bahasa bigram dan trigram. Kinerja metode yang diusulkan dievaluasi menggunakan nilai precession, recall dan f-measure. Berdasarkan pengujian yang telah dilakukan dapat diketahui bahwa perolehan nilai precession, recall dan f-measure pada model bahasa trigram dengan threshold = 0.1 memiliki nilai yang tertinggi masing-masing sebesar 83%, 85%, dan 86% jika dibandingkan dengan skenario pengujian bigram. Hal ini menunjukkan bahwa model bahasa trigram dapat mengidentifikasi kesalahan tata bahasa dengan baik.**

Keywords— **Tata Bahasa Inggris, Pernyataan Kebutuhan Perangkat Lunak, N-Gram Language Model.**

Abstract **— Various aspects such as the ability to write grammar, the background of the use of English and the limited knowledge possessed by the compiler team of software needs allows for errors in the making of software requirements specification documents. This can cause a decrease in the quality of the document specifications of the software requirements. To overcome these problems, a method is needed to identify grammatical errors in the software requirements statement. This study aims to propose the development of a method for identifying grammatical errors in software requirements statements. The proposed method is a statistical-based technique using the probability of the n-gram language model, where the language model used is the bigram and trigram language models. The performance of the proposed method is evaluated using the value of precession, recall and f-measure. Based on the tests that have been carried out, it can be seen that the acquisition of precession, recall and f-measure values in the trigram language model with threshold = 0.1 has the highest value of 83%, 85% and 86% respectively, compared to the bigram test scenario. This shows that the trigram language model can identify grammatical errors well.**

Keywords—**English Grammar, Software Requirements Statement, N-Gram Language Model.**

I. PENDAHULUAN

Dokumen Spesifikasi Kebutuhan Perangkat Lunak atau yang lebih dikenal SKPL merupakan dokumentasi awal dalam pembuatan perangkat lunak. Didalam SKPL terdapat pernyataan kebutuhan fungsional yang merupakan acuan oleh *stakeholder* (pengembang perangkat lunak) dalam membangun perangkat lunak. Kebutuhan fungsional merupakan bentuk fungsi/layanan yang terdapat pada sistem. Kebutuhan ini menspesifikasikan fungsi-fungsi atau fitur-fitur yang harus ada pada sistem untuk dapat membantu pengguna mencapai tujuan ketika menggunakan sistem, yang pada akhirnya memenuhi tujuan bisnis.

Menelusuri dan mengidentifikasi sebuah kebutuhan adalah suatu hal yang rumit untuk dilakukan salah satunya jika penulisan dokumen menggunakan bahasa Inggris. Hal ini terkadang mengakibatkan kerancuan kebutuhan fungsional antara pelanggan dan pengembang. Sehingga dapat menurunkan kualitas dokumen spesifikasi kebutuhan perangkat lunak. Acuan dalam pembuatan kebutuhan fungsional diperoleh melalui komunikasi dua arah antara pelanggan dan pengembang baik secara lisan maupun tulisan yang dalam pelaksanaannya mungkin terdapat kesalahan penerjemahan kebutuhan [1]. Hal ini tentunya memberikan tantangan tersendiri kepada pengembang dalam menyusun dan menyajikan

dokumen spesifikasi kebutuhan yang berkualitas dan minim terhadap kesalahan.

Berbagai aspek seperti kemampuan untuk penulisan *grammar*, latar belakang penggunaan bahasa Inggris dan keterbatasan pengetahuan yang dimiliki oleh tim penyusun kebutuhan perangkat lunak memungkinkan adanya kesalahan didalam pembuatan dokumen spesifikasi kebutuhan perangkat lunak. Kesalahan dalam memahami kebutuhan fungsional pada dokumen Spesifikasi Kebutuhan Perangkat Lunak ini akan berdampak pada proses pengembangan perangkat lunak. Salah satunya terjadinya konflik antara pelanggan dan pengembang terutama dalam memahami dan memverifikasi capaian pengembang terhadap kebutuhan yang telah ditetapkan sejak awal [2]. Pada sisi pengembang, pernyataan kebutuhan fungsional yang ambigu akan membuat pengembang membuat banyak asumsi-asumsi terkait kebutuhan pengguna dan hal ini beresiko jika dilakukan.

Untuk mengatasi kesalahan dalam tata bahasa pada dokumen SKPL diperlukan sebuah metode untuk melakukan pengecekan ejaan pada kalimat kemudian menghitung probabilitas penggunaan *grammar* yang tepat pada kalimat tersebut. Pengecekan ejaan dapat dilakukan dengan menggunakan pustaka Huspell Checker.

Teknik *n-gram* didasarkan pada pemisahan teks menjadi string dengan panjang n mulai dari posisi tertentu dalam suatu teks. *N-gram* untuk setiap string dihitung dan kemudian dibandingkan satu per satu. N-gram dapat berupa unigram (n=1), bigram (n=2), *trigram* (n=3), dan seterusnya (Lisangan, 2013). Dalam menentukan probabilitas *grammar* maka dilakukan perhitungan berdasarkan data probabilitas pengelompokan kata berdasarkan frasa (*chunking*) yang terdapat pada data Corpus. Model statistik *n-gram* yang digunakan pada penelitian ini terdiri dari dua model yaitu model bahasa *bigram* dan *trigram* yang menunjukkan kinerja cukup baik dalam memperbaiki kesalahan tata bahasa Athanaselis dkk. [3] dan [4]. Sedangkan untuk meningkatkan kinerja model bahasa *n-gram* penelitian ini juga memanfaatkan sebuah korpus untuk membuat model probabilitas pasangan *bigram* dan *trigram chunk* yaitu Penn Treebank Corpus.

Penerapan metode yang diusulkan diharapkan dapat mengurangi kesalahan dalam memahami *grammar* khususnya penggunaan *grammar* pada sebuah pernyataan kebutuhan fungsional perangkat lunak dengan menggantikan peran seorang tenaga ahli dalam menganalisis kebutuhan.

II. TINJAUAN PUSTAKA

A. *Kebutuhan Perangkat Lunak*

Kebutuhan perangkat lunak (*Software Requirements*) adalah atribut-atribut yang bersifat spesifik yang merupakan spesifikasi kebutuhan fungsional dan kebutuhan non fungsional dari sebuah sistem perangkat lunak. Dokumen SKPL umumnya disusun menggunakan bahasa alamiah, yaitu bahasa yang kita gunakan dalam percakapan sehari-hari. Walaupun bahasa alamiah lebih mudah dimengerti oleh pemangku kepentingan dan sangat fleksibel untuk menampung berbagai kebutuhan pengguna yang sering berubah-ubah. Namun, penggunaan bahasa alamiah memiliki kelemahan yaitu pada saat makna dari pernyataan kebutuhan perangkat lunak memiliki arti dan penafsiran yang berbeda-beda pada masing-masing orang atau bisa disebut sebagai kerancuan dalam pernyataan kebutuhan perangkat lunak [2]

B. *Grammar and Spell Checker*

Pada dasarnya ada tiga cara untuk mengimplementasikan pemeriksa tata bahasa, yaitu berbasis sintaksis, berbasis statistik, dan berbasis aturan. Penelitian ini merujuk pada ketentuan berikut:

1. Pemeriksaan berbasis sintaksis, dalam pendekatan ini, sebuah teks benar-benar diuraikan, yaitu kalimat dianalisis dan setiap kalimat diberi struktur pohon. Teks dianggap salah jika penguraian tidak berhasil.
2. Pengecekan berbasis statistik, seperti yang digunakan dalam penelitian ini. Dalam pendekatan ini, sebuah korpus POS (*Part-of-speech*) beranotasi digunakan untuk membangun daftar urutan penanda atau *tag* POS. Beberapa urutan penanda akan sangat umum (misalnya determiner, kata sifat, kata benda), yang lain mungkin tidak akan terjadi sama sekali (misalnya determiner, dan kata sifat). Urutan yang sering terjadi dalam korpus dapat dianggap benar dalam teks lain, juga, urutan yang tidak umum mungkin terdapat kesalahan.
3. Pemeriksaan berbasis aturan, dalam pendekatan ini, seperangkat aturan dicocokkan dengan teks yang setidaknya telah ditandai POS. Pendekatan ini mirip dengan pendekatan berbasis statistik, tetapi semua aturan dikembangkan secara manual.

Pemeriksaaan ejaan secara otomatis dapat dilakukan menggunakan pustaka Hunspell Checker. Hunspell adalah pustaka pemeriksa ejaan yang digunakan oleh LibreOffice, OpenOffice, dan banyak lainnya. Pustaka ini menyediakan sistem untuk t*okenizing*, *stemming* dan ejaan di hampir semua bahasa atau alfabet. Paket R memaparkan pemeriksa ejaan tingkat tinggi dan juga stemmer dan tokenizer tingkat rendah yang menganalisis atau mengekstrak kata-kata individual dari berbagai format (teks, html, xml, lateks). Pustaka Hunspell menggunakan format kamus khusus yang menentukan karakter, kata, dan konjugasi mana yang valid dalam bahasa tertentu. Proses pengecekan ejaan terdiri dari langkah-langkah berikut : [5].

1. Mengurai dokumen dengan mengekstraksi (*tokenizing*) kata-kata yang ingin kita periksa
2. Analisis setiap kata dengan memecahnya di root (*stemming*) dan imbuhan konjungsi.

3. Cari di kamus kombinasi antar kata dan imbuhan, jika valid untuk bahasa yang dipilih, maka tidak ditemukan kesalahan ejaan.
4. Untuk kata-kata yang salah, sarankan koreksi dengan menemukan kata-kata yang mirip (benar) dalam kamus.

C. Model Bahasa N-Gram

Salah satu teknik yang digunakan untuk memperbaiki kesalahan tata bahasa ialah teknik statistik berbasis *n-gram*. Teknik ini memetakan sebuah kalimat kedalam n-huruf sub urutan kata atau string dimana n biasanya terdiri dari satu kata (*unigram*), dua kata (*bigram*) atau tiga kata (*trigram*) [4]. Model *n-gram* adalah salah satu teknik statistik yang memodelkan urutan kata kedalam nilai probabilitasnya. Jika sebuah kalimat dinyatakan dengan {w1, w2, ..., wn-1, wn} dan n adalah jumlah kata dalam kalimat, maka untuk menghitung nilai probabilitas kalimat menggunakan model statistik unigram dapat dihitung menggunakan persamaan berikut :

$$P(W_1^n) \approx \prod_i P(w_i) \quad (1)$$

Sedangkan untuk menghitung nilai probabilitas tiap pasangan dua kata didalam sebuah kalimat dapat dihitung menggunakan persamaan perkiraan kemungkinan maksimum (*maximum likehood estimation*) berikut :

$$P(w_i|w_{i-1}) = \frac{N(w_{i-1}, w_i)}{N(w_{i-1})} \quad (2)$$

Dimana N(wi-1, wi) dan N(wi-1) menunjukkan jumlah kemunculan kata atau frekuensi N(wi-1, wi) dan N(wi-1) pada korpus. Nilai probabilitas sebuah kalimat menggunakan model statistik *bigram* dapat dihitung dengan aturan Chain yang ditulis dengan persamaan berikut :

$$P(W_1^n) \approx \prod_{i=1}^{n} P(w_i|w_{i-1}) \quad (3)$$

Nilai probabilitas tiap pasangan tiga kata didalam kalimat dapat dihitung dengan menggunakan persamaan kemungkinan maksimum berikut :

$$P(w_i|w_{i-2},w_{i-1}) = \frac{N(w_{i-2}, w_{i-1}, w_i)}{N(w_{i-2},w_{i-1})} \quad (4)$$

Dimana N(wi-2, wi-1, wi) dan N(wi-2, wi-1) menunjukkan jumlah kemunculan kata atau frekuensi N(wi-2, wi-1, wi) dan N(wi-2, wi-1) pada korpus. Nilai probabilitas sebuah kalimat menggunakan model statistik *trigram* dapat dihitung menggunakan aturan berikut :

$$P(W_1^n) \approx \prod_{i=3}^{n} P(w_i|w_{i-2},w_{i-1}) \quad (5)$$

D. Chunk-Based Sentence Pattern

POS Taging adalah proses utama membuat potongan dalam kalimat yang sesuai dengan bagian tertentu dari sebuah kalimat atau ucapan. Penandaan POS adalah proses penugasan penandaan tiap bagian kata seperti kata benda, kata kerja, kata ganti, kata depan, kata keterangan, kata sifat atau penanda lain untuk setiap kata dalam kalimat. Kata benda dapat dibagi lagi menjadi kata benda tunggal dan jamak, kata kerja dapat dibagi menjadi kata kerja *past tense* (lampau) dan kata kerja *present tense* (sedang berlangsung/saat ini) dan sebagainya.

Membuat potongan (*chunk*) adalah proses untuk mengurai kalimat menjadi bentuk yang merupakan struktur kalimat berbasis potongan. Chunk adalah unit teks dari tag POS yang berdekatan yang menampilkan hubungan antara kata-kata internal mereka. Input kalimat bahasa Inggris dibuat dalam struktur chunk dengan menggunakan aturan tulisan tangan. Hal ini mewakili bagaimana potongan-potongan ini cocok untuk membentuk konstituen kalimat.

Aplikasi NLP seperti *Grammar Checker* membutuhkan parser dengan model parsing opsional. Parsing adalah proses menganalisis teks secara otomatis dengan menetapkan struktur sintaksis sesuai dengan tata bahasa. Parser digunakan untuk memahami sintaks dan semantik dari kalimat bahasa alami yang terbatas pada tata bahasa.

Ada dua metode untuk penguraian seperti penguraian *top-down* dan pengurutan *bottom-up*. Penguraian *top-down* dimulai dengan simbol awal dan upaya untuk mendapatkan kalimat input dengan mengganti sisi kanan produksi untuk non terminal. Penguraian *bottom-up* (*shift-reduce*) dimulai dengan kalimat input dan menggabungkan kata-kata ke dalam potongan level yang lebih tinggi sampai unit akhirnya menjadi sebuah kalimat. Parser *bottom-up* menangani kelas tata bahasa yang besar. Dalam sistem ini, penguraian *bottom-up* digunakan untuk menguraikan kalimat.

Memilah potongan dengan menggunakan CFG: Memotong atau memecah segmen kalimat ke dalam urutan konstituen sintaksis atau potongan, yaitu urutan kata-kata yang berdekatan dikelompokkan berdasarkan sifat linguistik [6]. Struktur potongan kalimat yang sintaksis diperlukan untuk menentukan kebenaran tata bahasanya. Dalam sistem yang diusulkan, sepuluh tipe *chunk* umum digunakan untuk membuat struktur *chunk* seperti yang ditunjukkan pada Tabel 1 [7].

TABEL I
JENIS CHUNK

Jenis Chunk	Deskripsi	Contoh
NC	Noun Chunk	A young, the girls
VC	Verb Chunk	Is playing, goes, went
AC	Adjective Chunk	More beautiful, younger, old
RC	Adverb Chunk	Usually, quickly
PTC	Particle Chunk	Up, down
PPC	Prepositonal Chunk	At, on, in, under
COC	Conjunction Chunk	And, or , but
QC	Question Chunk	Where, who, when
INFC	Infinitive Chunk	To
TC	Time Chunk	Tomorrow, yesterday

III. METODE PENELITIAN

Pengembangan metode identifikasi kesalahan tata bahasa pada pernyataan kebutuhan perangkat lunak yang diusulkan terdiri dari tiga tahapan yaitu pengumpulan dataset penelitian, pembuatan kamus frekuensi pola pasangan *bigram* dan *trigram chunk*, dan pengembangan metode identifikasi kesalahan tata Bahasa.

A. Data Penelitian

Penelitian ini menggunakan beberapa pernyataan kebutuhan pada penelitian Tjong (2008) [8]. Penelitian tersebut mengumpulkan beberapa dataset pernyataan kebutuhan yang berasal dari beberapa domain permasalahan yang berbeda untuk melakukan pengujian terhadap penelitiannya dalam mendeteksi kerancuan pada dokumen spesifikasi kebutuhan perangkat lunak. Berikut adalah dataset yang digunakan oleh Tjong (2008).

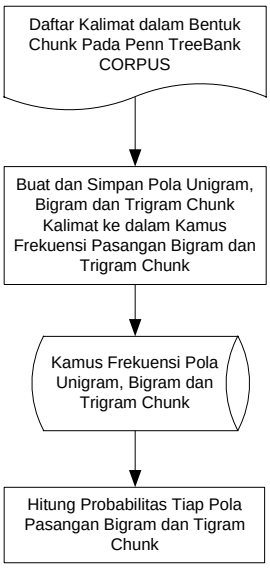
Total dataset yang digunakan pada penelitian ini berjumlah 12 dataset yang dapat dilihat pada Tabel 2. berikut.

TABEL II
DATASET PENELITIAN

No	Dataset	Sumber
1	ACM's OOPSLA DesignFest®	(Hussain, 2007)
2	Lift Controller System	(Tjong, 2008)
3	Yacht Race Results (YRR)	(Tjong, 2008)
4	Batch Poster System (BPS)	(Tjong, 2008)
5	Cask Loader Software (CLS)	(Tjong, 2008)
6	Data Cycle System (DCS)	(Tjong, 2008)
7	EVLA Array Operations (EVLA)	(Tjong, 2008)
8	Large Area Telescope (LAT) Science Analysis Software (SAS) Level III Specification	(Tjong, 2008)
9	PESA High-Level Trigger Selection	(Tjong, 2008)
10	Sort Algorithm Demonstration Program (SAD)	(Tjong, 2008)
11	New Adelaide Airport System	(Tjong, 2008)
12	MCSS System	(Tjong, 2008)

B. Pembuatan Kamus Frekuensi Bigram dan Tigram Chunk

Kamus frekuensi pola pasangan *bigram* dan *trigram chunk* digunakan untuk menentukan probabilitas tiap pola pasangan *bigram* dan *trigram chunk*. Adapun tahapan pembuatan kamus frekuensi *bigram* dan *trigram chunk* dapat dilihat pada gambar 1.

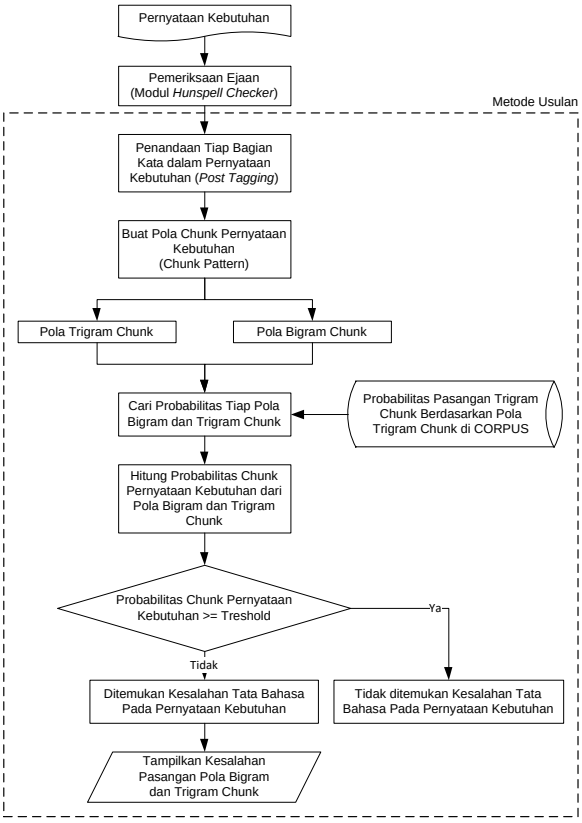


Gbr 1. Tahapan Pembuatan Kamus Frekuensi *Bigram* dan *Trigram Chunk*

Kamus frekuensi *bigram* dan *trigram chunk* dibuat berdasarkan jumlah kemunculan pola pasangan *bigram* dan *trigram chunk* pada korpus. Korpus yang digunakan pada penelitian ini adalah Penn TreeBank Corpus, dimana di dalam korpus ini terdapat sekitar 3914 ragam kalimat yang telah diberi *pos_tag* dan *chunk_tag*.

C. Pengembangan Metode Identifikasi Kesalahan Tata Bahasa

Metode identifikasi kesalahan tata bahasa yang diusulkan pada penelitian ini menggunakan probabilistic model bahasa *N-Gram*, dimana identifikasi kesalahan tata bahasa akan dianalisis pada tingkat frasa dari kelompok kata (*chunking*) pada sebuah pernyataan kebutuhan perangkat lunak. Adapun usulan metode identifikasi kesalahan tata bahasa pada penelitian ini dapat dilihat pada Gambar 2.



Gbr 2. Metode Usulan N-Gram

Data yang digunakan sebagai masukkan dari sistem ialah pernyataan kebutuhan perangkat lunak. Kalimat atau pernyataan kebutuhan terlebih dahulu akan dilakukan pemrosesan awal yaitu pemeriksaan kesalahan ejaan dimana modul yang digunakan pada penelitian ini ialah *Hunspell Checker*. Langkah berikutnya ialah penandaan tiap bagian kata (Pos-Tagging). Penandaan tiap bagian kata dilakukan dengan menggunakan pustaka *TaggerIBase* berbasis Penn TreeBank Tagset. Hasil POS-Tagging pernyataan kebutuhan selanjutnya akan dijadikan input untuk membuat pola pengelompokan kata berdasarkan frasa (*chunking process*). Proses pengelompokan kata (*chunking*) pada penelitian ini menggunakan pustaka *ClassifierChunkParser*. Hasil dari proses *chunking* pernyataan kebutuhan akan membentuk pola *bigram* dan *trigram chunk.*

Langkah berikutnya ialah mencari nilai probabilitas tiap pola *bigram* dan *trigram chunk* berdasarkan frekuensi pola yang ada di korpus pembelajaran. Setelah mendapatkan nilai probabilitas dari tiap pasangan *bigram* dan *trigram chunk*, langkah berikutnya ialah mengidentifikasi kesalahan tata bahasa berdasarkan nilai probabilitas *bigram* dan *trigram chunk*. Penemuan kesalahan tata bahasa dilakukan dengan membandingkan nilai probabilitas tiap-tiap *bigram* atau *trigram chunk* pada pernyataan kebutuhan dengan nilai ambang batas yang telah ditentukan. Penentuan nilai ambang batas (*threshold*) yang tepat sangat menentukan performa metode identifikasi. Sehingga pada penelitian ini digunakan nilai ambang batas untuk pola *bigram* maupun *trigram chunk* sebesar 0.1, 0.15, dan 0.2.

IV. HASIL DAN PEMBAHASAN

A. Pengolahan Dataset

Dataset pada penelitian ini berjumlah 12 dataset. Pada setiap dataset akan diambil pernyataan kebutuhan secara acak dengan total jumlah data pernyataan kebutuhan sebanyak 200 pernyataan kebutuhan. Pernyataan kebutuhan ini untuk selanjutnya akan dijadikan sebagai input dari metode yang diusulkan. Langkah awal yang dilakukan ialah memeriksa kesalahan ejaan dengan menggunakan pustaka *HunspellChecker*. Contoh hasil implementasi pengolahan awal dataset pada pernyataan kebutuhan "*The operator a admin log will record all warning mesages prompteds by the sistem.*" menggunakan pustaka *HunspellChecker* adalah sebagai berikut.

```
>>> from nltk import word_tokenize
>>> from hunspell import Hunspell
>>> requirement_sent = 'The operator a admin log will record all warning mesages prompteds by the sistem.'
>>> token_sent = word_tokenize(requirement_sent)
>>> checker = Hunspell('en_US')
>>> spell_check_result = {}
>>> for token in token_sent:
...  if not checker.spell(token):
...   spell_check_result[token] = checker.suggest(token)
...
>>> for token in spell_check_result:
...  print(token, ':', spell_check_result[token])
...
admin : ('admen', 'adman', 'admit', 'ad min', 'ad-min', 'admiring')
mesages : ('messages', 'message', 'menages', 'me sages', 'me-sages', 'mes ages', 'mes-ages', 'massages', 's
ausages', 'sages', 'meagres')
prompteds : ('prompted', 'prompters', 'prompt eds', 'prompt-eds', 'prompted s', 'prompter', 'prompts')
sistem : ('sister', 'system', 'stem')
>>>
```

Gbr 3. Hasil Implementasi Pengolahan Awal Dataset

Pada pernyataan kebutuhan yang diuji kata yang salah eja yaitu kata *admin*, *mesages*, *prompteds*, dan *sistem*. Sistem tidak akan melanjutkan proses berikutnya yaitu menandai tiap bagian kata (*POS-Tagging*) pada pernyataan kebutuhan sebelum kata yang salah ejaan diperbaiki atau kata tersebut dimasukkan kedalam daftar pengecualian dari kata yang salah ejaan. Perbaikan kata yang salah ejaan dapat dilakukan dengan memilih salah satu kata yang ada pada daftar kata hasil rekomendasi. Kata *admin* memiliki rekomendasi kata *'admen'*, *'adman'*, *'admit'*, *'ad min'*, *'ad-min'*, *'admiring'*, kata mesages memiliki rekomendasi kata *'messages'*, *'message'*, *'menages'*, *'me sages'*, *'me-sages'*, *'mes ages'*, *'mes-ages'*, *'massages'*, *'sausages'*, *'sages'*, *'meagres'*, kata prompteds memiliki rekomendasi kata yaitu *'prompted', 'prompters', 'prompt eds'*, *'prompt-eds'*, *'prompted s'*, *'prompter'*, *'prompts'*,dan kata sistem memiliki rekomendasi kata *'sister'*, *'system'*, dan *'stem'*.

Setelah dilakukan perbaikan kesalahan ejaan pada pernyataan kebutuhan yang akan diuji, langkah berikutnya ialah menganalisis tata bahasa pernyataan kebutuhan tersebut dengan bantuan seorang ahli di bidang tata bahasa spesialisasi bahasa Inggris dan memiliki pengalaman studi atau pernah bekerja di luar negeri dengan menggunakan bahasa Inggris sebagai bahasa komunikasi. Tugas ahli ialah menganalisis pernyataan kebutuhan apakah terdapat kesalahan tata bahasa dan menandai pola *bigram* dan *trigram chunk* yang salah pada pernyataan kebutuhan yang diuji. Berikut adalah contoh hasil analisa ahli dalam menganalisis salah satu pernyataan kebutuhan.

TABEL III
HASIL ANALISIS AHLI PADA PERNYATAAN KEBUTUHAN

The	**operator**	**a**	**admin**	**log**	**will**	**record**	**all**	**warning**	**messages**	**prompted**	**by**	**the**	**system**	**.**
DT	NN	DT	NN	NN	MD	VB	DT	NN	NNS	VBN	IN	DT	NN	.
NP		NP			VP		NP			VP	IN	NP		.
Catatan : silahkan tambahkan kata penghubung "*dan*" pada pasangan *bigram chunk* tersebut. Contoh : The operator and a admin log will record all warning messages prompted by the system.														

Pada Tabel 3 merupakan salah satu contoh hasil analisis kesalahan tata bahasa pernyataan kebutuhan perangkat lunak "*The operator a admin log will record all warning mesages prompteds by the sistem.*" dimana terdapat kesalahan pola *bigram chunk* "NP_NP" pada pernyataan kebutuhan tersebut, tepatnya pada bagian pernyataan "*The operator*" dan "*a admin log*".

B. Kamus Frekuensi Bigram dan Trigram Chunk

Kamus frekuensi pola *bigram* dan *trigram chunk* dibangun menggunakan Penn TreeBank Corpus berdasarkan *chunk* kalimat yang terdapat didalam korpus. Jumlah atau frekuensi pola *bigram* dan *trigram chunk* yang terjadi di korpus akan menentukan nilai probabilitas kamus frekuensi *bigram* dan *trigram chunk* yang dibangun. Artinya semakin sering muncul pola pasangan *bigram* dan *trigram chunk* tersebut di dalam korpus maka semakin tinggi nilai probabilitasnya di kamus frekuensi. Hasil implementasi kamus frekuensi *bigram chunk* berbasis Penn TreeBank Corpus dapat dilihat pada Gambar 4 berikut.

```
D:\Project_Python\nlp\grammar_check>
chunk1,chunk2 = probability
('PP', 'NP') = 0.937452092595
('STA', 'NP') = 0.648063573255
('VP', 'NP') = 0.544688995215
('OUT', 'NP') = 0.442577030812
('NP', 'VP') = 0.285128068382
('NP', 'OUT') = 0.272299008933
('OUT', 'OUT') = 0.248855978035
('NP', 'PP') = 0.237367228123
('VP', 'OUT') = 0.224421052632
('STA', 'OUT') = 0.222963098283
('VP', 'PP') = 0.179559808612
('OUT', 'VP') = 0.15591979366
('NP', 'END') = 0.119876153652
('STA', 'PP') = 0.109152356595
('OUT', 'PP') = 0.102615303547
('NP', 'NP') = 0.0853295409093
('OUT', 'END') = 0.0500318939457
('VP', 'END') = 0.0395406698565
('PP', 'VP') = 0.0280162501916
('STA', 'VP') = 0.019820971867
('PP', 'PP') = 0.0185114211252
('PP', 'OUT') = 0.0152920435382
('VP', 'VP') = 0.0117894736842
('PP', 'END') = 0.00072819254944
```

Gbr 4. Kamus Frekuensi *Bigram Chunk*

Pada Gambar 4 diketahui bahwa nilai probabilitas pola *bigram chunk* dengan urutan 3 tertinggi berturut-turut yaitu "PP_NP" sebesar 0.93, "STA_NP" sebesar 0.64 dan "VP_NP" sebesar 0.54. Hal ini menunjukkan bahwa pola *bigram chunk* "PP_NP", "STA_NP" dan "VP_NP" adalah pola *bigram chunk* yang memiliki struktur *chunk* paling dapat diterima menjadi suatu frasa kalimat yang dinilai baik dan memiliki frekuensi yang paling banyak muncul didalam korpus.

Hasil implementasi kamus frekuensi *trigram chunk* berbasis Penn TreeBank Corpus dapat dilihat pada Gambar 5 berikut.

```
D:\Project_Python\nlp\grammar_check>python2
chunk1,chunk2,chunk3 = probability
('NP', 'PP', 'NP') = 0.951319977532
('VP', 'PP', 'NP') = 0.939032189299
('PP', 'PP', 'NP') = 0.919254658385
('STA', 'PP', 'NP') = 0.899581589958
('OUT', 'PP', 'NP') = 0.89
('PP', 'VP', 'NP') = 0.730506155951
('STA', 'STA', 'NP') = 0.648063573255
('VP', 'VP', 'NP') = 0.63961038961
('STA', 'OUT', 'NP') = 0.620647275707
('OUT', 'VP', 'NP') = 0.616506581288
('STA', 'NP', 'VP') = 0.5681465821
('STA', 'VP', 'NP') = 0.534562211982
('NP', 'VP', 'NP') = 0.515249129735
('OUT', 'NP', 'VP') = 0.498370723148
('NP', 'OUT', 'NP') = 0.449431478157
('PP', 'OUT', 'NP') = 0.448621553885
('OUT', 'OUT', 'NP') = 0.448010698763
('VP', 'NP', 'PP') = 0.401194659171
('NP', 'NP', 'VP') = 0.379340277778
('PP', 'OUT', 'OUT') = 0.340852130326
('PP', 'NP', 'OUT') = 0.338348323794
('VP', 'OUT', 'NP') = 0.338222752857
```

Gbr 5. Kamus Frekuensi *Trigram Chunk*

Pada Gambar 5 diketahui bahwa nilai probabilitas pola *trigram chunk* dengan urutan 3 tertinggi berturut-turut yaitu "NP_PP_NP" sebesar 0.95, "VP_PP_NP" sebesar 0.94 dan "PP_PP_NP" sebesar 0.91.

C. Pengujian Metode Identifikasi Kesalahan Tata Bahasa

Proses uji coba metode rekomendasi dilakukan berdasarkan 2 skenario uji coba yaitu uji coba metode rekomendasi menggunakan model bahasa *bigram*, dan uji coba metode rekomendasi menggunakan model bahasa *trigram*. Pada setiap skenario uji coba, baik menggunakan model bahasa *bigram* dan *trigram chunk* akan dilakukan pengujian dengan nilai *threshold* yang berbeda. Ada tiga nilai *threshold* yang dipilih yaitu sebesar 0.1, 0.15, dan 0.2. Hal ini dilakukan untuk mengetahui nilai akurasi dan presisi pada tiap-tiap nilai *threshold* dan model bahasa yang digunakan.

Hasil uji coba metode yang diusulkan menggunakan model bahasa *bigram* dan *threshold* sebesar 0.1 dapat dilihat pada Gambar 6.

```
(S
  (NP The/DT operator/NN)
  (NP a/DT admin/NN log/NN)
  (VP will/MD record/VB)
  (NP all/DT warning/NN messages/NNS)
  (VP prompted/VBN)
  (PP by/IN)
  (NP the/DT system/NN)
  ./.)

Hasil analisis pola bigram chunk (Threshold= 0.1 ) :
STA ~ NP = 0.648063573255
NP ~ NP = 0.0853295409093
ditemukan kesalahan pada pola bigram chunk :  ('NP', 'NP')
NP ~ VP = 0.285128068382
VP ~ NP = 0.544688995215
NP ~ VP = 0.285128068382
VP ~ PP = 0.179559808612
PP ~ NP = 0.937452092595
NP ~ END = 0.119876153652
```

Gbr 6. Hasil Uji Coba Model Bahasa *Bigram* (Threshold=0.1)

Hasil uji coba pada Gambar 6 menunjukkan terdapat 1 kesalahan pola *bigram chunk* yaitu "NP_NP". Hasil uji coba metode yang diusulkan menggunakan model bahasa *bigram* dan *threshold* sebesar 0.15 adalah sebagai berikut.

```
(S
  (NP The/DT operator/NN)
  (NP a/DT admin/NN log/NN)
  (VP will/MD record/VB)
  (NP all/DT warning/NN messages/NNS)
  (VP prompted/VBN)
  (PP by/IN)
  (NP the/DT system/NN)
  ./.)

Hasil analisis pola bigram chunk (Threshold= 0.15 ) :
STA ~ NP = 0.648063573255
NP ~ NP = 0.0853295409093
ditemukan kesalahan pada pola bigram chunk :  ('NP', 'NP')
NP ~ VP = 0.285128068382
VP ~ NP = 0.544688995215
NP ~ VP = 0.285128068382
VP ~ PP = 0.179559808612
PP ~ NP = 0.937452092595
NP ~ END = 0.119876153652
ditemukan kesalahan pada pola bigram chunk :  ('NP', 'END')
```

Gbr 7. Hasil Uji Coba Model Bahasa *Bigram* (Threshold=0.15)

Hasil uji coba pada Gambar 7 menunjukkan terdapat 2 kesalahan pola *bigram chunk* yaitu "NP_NP" dan "NP_END". Hasil uji coba metode yang diusulkan menggunakan model bahasa *bigram* dengan *threshold* sebesar 0.2 adalah sebagai berikut.

```
(S
  (NP The/DT operator/NN)
  (NP a/DT admin/NN log/NN)
  (VP will/MD record/VB)
  (NP all/DT warning/NN messages/NNS)
  (VP prompted/VBN)
  (PP by/IN)
  (NP the/DT system/NN)
  ./.)

Hasil analisis pola bigram chunk (Threshold= 0.2 ) :
STA ~ NP = 0.648063573255
NP ~ NP = 0.0853295409093
ditemukan kesalahan pada pola bigram chunk :  ('NP', 'NP')
NP ~ VP = 0.285128068382
VP ~ NP = 0.544688995215
NP ~ VP = 0.285128068382
VP ~ PP = 0.179559808612
ditemukan kesalahan pada pola bigram chunk :  ('VP', 'PP')
PP ~ NP = 0.937452092595
NP ~ END = 0.119876153652
ditemukan kesalahan pada pola bigram chunk :  ('NP', 'END')
```

Gbr 8. Hasil Uji Coba Model Bahasa *Bigram* (Threshold=0.2)

Hasil uji coba pada Gambar 8 menunjukkan terdapat 3 kesalahan pola *bigram chunk* yaitu "NP_NP", "VP_PP" dan "NP_END".

D. Evaluasi Metode

Dataset pernyataan kebutuhan yang diuji coba berjumlah 200 pernyataan kebutuhan dimana data ini akan dibagi menjadi 3 bagian (*3-fold cross validatation*) untuk dilakukan pengujian.

Hasil evaluasi terhadap metode yang diusulkan terdiri dari 2 skenario uji coba yaitu uji coba metode menggunakan model bahasa *bigram* dan model bahasa *trigram* yang dapat dilihat pada Tabel 4 hingga Tabel 7. Hasil perhitungan untuk *bigram* dan *trigram* dievaluasi menggunakan *precision*, *recall* dan *f-measure* pada masing-masing *fold* menggunakan *threshold* (T) = 0.1, 0.15, 0.2. Tabel 4 hingga Tabel 6 adalah hasil evaluasi menggunakan model bahasa *bigram* untuk tiap-tiap *fold* pada dataset uji coba.

TABEL IV
PERHITUNGAN *PRECISION*, *RECALL* DAN *F-MEASURE*
UNTUK *BIGRAM* DENGAN NILAI T=0.1

Fold	***Precision*(%)**	***Recall*(%)**	***F-measure*(%)**
I	73	81	80
II	77	79	83
III	80	83	85

TABEL V
PERHITUNGAN *PRECISION*, *RECALL* DAN *F-MEASURE*
UNTUK *BIGRAM* DENGAN NILAI T=0.15

Fold	***Precision*(%)**	***Recall*(%)**	***F-measure*(%)**
I	70	69	69
II	74	71	74
III	76	76	76

TABEL VI
PERHITUNGAN *PRECISION*, *RECALL* DAN *F-MEASURE*
UNTUK *BIGRAM* DENGAN NILAI T=0.20

Fold	***Precision*(%)**	***Recall*(%)**	***F-measure*(%)**
I	73	74	80
II	77	79	83
III	80	80	81

Selanjutnya pada Tabel 7 sampai Tabel 9 adalah evaluasi menggunakan model bahasa *Trigram.*

TABEL VII
PERHITUNGAN *PRECISION*, *RECALL* DAN *F-MEASURE*
UNTUK *TRIGRAM* DENGAN NILAI T=0.1

Fold	***Precision*(%)**	***Recall*(%)**	***F-measure*(%)**
I	79	81	83
II	80	83	84
III	83	85	86

TABEL VIII
PERHITUNGAN *PRECISION*, *RECALL* DAN *F-MEASURE*
UNTUK *TRIGRAM* DENGAN NILAI T=0.15

Fold	***Precision*(%)**	***Recall*(%)**	***F-measure*(%)**
I	72	71	80
II	74	71	83
III	76	76	84

TABEL IX
PERHITUNGAN *PRECISION*, *RECALL* DAN *F-MEASURE* UNTUK *TRIGRAM* DENGAN NILAI T=0.2

Fold	***Precision*(%)**	***Recall*(%)**	***F-measure*(%)**
I	70	73	72
II	74	74	74
III	76	76	76

Berdasarkan hasil yang diperlihatkan pada Tabel 7 diketahui bahwa perolehan *precission*, *recall* dan *f-measure* pada model bahasa *trigram* dengan *treshold* = 0.1 memiliki nilai yang lebih tinggi jika dibandingkan dengan *fold* yang lain. Hal ini menunjukkan bahwa model bahasa *trigram* dapat mengidentifikasi kesalahan tata bahasa dengan baik.

V. KESIMPULAN

Berdasarkan hasil evaluasi metode identifikasi metode yang telah dilakukan pada 200 sampel data kebutuhan fungsional perangkat lunak maka dapat disimpulkan bahwa metode bahasa trigram dapat digunakan untuk mengidentifikasi kesalahan tata bahasa dengan baik. Hal ini dapat dilihat dari hasil evaluasi untuk *precision*, *recall* dan *f-measure* untuk 2 scenario bahasa n-gram (*bigram* dan *trigram*). Hasil evaluasi diperoleh model bahasa *Trigram* dengan *Threshold* = 0.1 memiliki nilai yang lebih tinggi jika dibandingkan dengan *fold* yang lain.

Berdasarkan pengujian yang dilakukan perlu dilakukan kombinasi dengan beberapa metode lainnya seperti metode *Rule Based* untuk meningkatkan akurasi metode dalam mengidentifikasi kesalahan tata bahasa dengan baik.

REFERENSI

[1]. Siahaan, D. O., 2012. *Analisa Kebutuhan Dalam Rekayasa Perangkat Lunak, Edisi 1.* Yogyakarta : Andi.

[2]. Enda, D. dan Siahaan, D. 2018. Rekomendasi Perbaikan Pernyataan Kebutuhan Yang Rancu Dalam Spesifikasi Kebutuhan Perangkat Lunak Menggunakan Teknik Berbasis Aturan, *Jurnal Teknologi Informasi dan Ilmu Komputer(JTIIK).* Vol 5 No.2 hal 207-216.

[3]. Athanaselis, T. et al., 2011, A Corpus ased Technique for Repairing Ill- Formed Sentences With Word Order Errors Using Co-Occurrences of N- Grams, *International Journal on Artificial Intelligence Tools*, 20 (3), hal. 401–424. DOI: 10.1142/S0218213011000218.

[4]. Wu, J., Chang, J. dan Chang, S. J., 2013 Correcting Serial Grammatical Errors based on N-grams and Syntax, *International Journal of Computational Linguistics & Chinese Language Processing*, Vol. 18, No. 4, December 2013-Special Issue on Selected Papers from ROCLING XXV, 18(4), hal. 31–44.

[5]. Anonim, 2018, *The hunspell package: High-Performance Stemmer, Tokenizer, and Spell Checker for R.* https://cran.r-project.org/web/packages/hunspell/vignettes/intro.html. Diakses 20 Desember 2018

[6]. Abney, S.P. 1996, Partial parsing via finite-state cascades. Natural Language Engineering 2(4) 337–344.

[7]. Lin, Y.N., Soe, K.M., dan Thien, N.L Developing a Chunk-based Grammar Checker for Translated English Sentences. *Prociding. 25th Pacific Asia Conference on Language, Information and Computation,* hal 245–254.

[8]. Tjong, S. F. 2008, *Avoiding ambiguitiy in requirements specifications*, Faculty of Engineering & Computer Science.