OPEN ACCESS
Indo-JC Indonesia Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 4, Issue. 3, Dec 2019. pp. 51-64
doi:10.21108/indojc.2019.4.3.355

Deteksi Pola Ambiguitas Struktural pada Spesifikasi Kebutuhan Perangkat Lunak menggunakan Pemrosesan Bahasa Alami

Chlaudiah Julinar Soplero Lelywiary [#1], Sri Widowati [#2], Kemas Muslim L. [#3]

[#] *Fakultas Informatika, Telkom University*
Bandung, Indonesia

[1] chlaudiahjulinar@student.telkomuniversity.ac.id

[2] sriwidowati@telkomuniversity.ac.id

[3] kemasmuslim@telkomuniversity.ac.id

Abstract

Software Requirement Specifications (SRS) is a document that is produced from Requirement Engineering (RE) process and plays an important role in software development. About 87.7% of SRS documents are written using natural language. The biggest problem in writing with natural language is misinterpretations, by ambiguous words. If ambiguous happen and not detected as quickly as possible, then misinterpretations can lead to software results that are not in accordance with user needs. This makes the ambiguous problem in SRS very important to deal with. There have been various study on solutions to ambiguous handling in SRS, and most of them use SRS in English. This study aims to detect ambiguity that occurs due to the the structure of incorrect software requirement statements in Indonesian SRS. The proposed method is a natural language pattern based on Part-Of-Speech Tag Hidden Markov Model-Viterbi and the pattern is detected by Regular Expression Parsing. The proposed natural language pattern is evaluated by Kappa index Value. The results of natural language pattern analysis have the highest Kappa index value of 0.9139, which means that expert strongly agree on the results of structural ambiguous detection with natural language patterns.

Keywords: software requirement specifications, structural ambiguous pattern, part-of-speech tag, regular expression parsing, kappa index value

Abstrak

Spesifikasi Kebutuhan Perangkat Lunak (SKPL) merupakan dokumen yang dihasilkan dari proses rekayasa kebutuhan dan memegang peranan penting dalam pengembangan perangkat lunak. Sekitar 87.7% dokumen SKPL ditulis menggunakan bahasa alami. Masalah terbesar dalam penulisan dengan bahasa alami adalah kesalahan interpretasi yang disebabkan karena terdapat kata-kata yang ambigu. Jika terjadi ambigu dan tidak dideteksi secepat mungkin, maka kesalahan interpretasi dapat mengarah pada hasil perangkat lunak tidak sesuai dengan kebutuhan pengguna. Hal ini membuat masalah ambigu dalam SKPL sangat penting untuk ditangani. Sudah terdapat berbagai penelitian mengenai solusi penanganan ambigu dalam SKPL, dan hampir sebagian besar menggunakan SKPL dalam bahasa Inggris. Penelitian ini bertujuan untuk mendeteksi ambigu yang terjadi akibat struktur pernyataan kebutuhan perangkat lunak yang salah pada SKPL dalam bahasa Indonesia. Adapun metode yang diusulkan adalah pola bahasa alami berdasarkan *Part-Of-Speech Tag* Hidden Markov Model-Viterbi, dan pola tersebut dideteksi dengan *Regular Expression Parsing*. Pola bahasa alami yang diusulkan dievaluasi dengan nilai indeks Kappa. Hasil dari analisis pola bahasa alami memiliki nilai indeks Kappa tertinggi sebesar 0.9139, yang berarti ahli sangat sepakat terhadap hasil deteksi ambigu struktural dengan pola bahasa alami.

Kata Kunci: spesifikasi kebutuhan perangkat lunak, pola ambigu struktural, *part-of-speech tag*, *regular expression parsing*, nilai indeks kappa

Received on Nov 2019. Accepted on Dec 2019

I. Pendahuluan

DALAM siklus pengembangan perangkat lunak, memahami, mengumpulkan dan menentukan kebutuhan pengguna terhadap perangkat lunak merupakan tahap awal yang harus dilakukan dan dikenal sebagai proses rekayasa kebutuhan. Hasil dari tahap tersebut berupa dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL) yang berisi seluruh deskripsi kebutuhan fungsional dan non-fungsional dari sistem perangkat lunak yang dibangun [1]. Hal ini menyebabkan, kualitas dokumen SKPL yang dihasilkan akan memberikan pengaruh terhadap proses pengembangan perangkat lunak berikutnya. SKPL memiliki peran yang sangat penting dalam pengembangan perangkat lunak. SKPL digunakan dalam implementasi desain, pengamatan proyek, verifikasi dan validasi, serta dalam pelatihan perangkat lunak [1].

Sekitar 87.7% dari kebutuhan perangkat lunak ditulis menggunakan bahasa alami [2]. Kebutuhan perangkat lunak diperoleh dari penelusuran keinginan pengguna yang dilakukan oleh pengembang dan ditulis ke dalam dokumen SKPL menggunakan bahasa alami. Penggunaan bahasa alami dalam SKPL sangat rentan terjadi masalah berupa kesalahan interpretasi. Kesalahan interpretasi menandakan bahwa terdapat penggunaan kata-kata yang tidak tepat dalam menyusun kalimat, sehingga mengakibatkan suatu kalimat kebutuhan perangkat lunak dapat diartikan berbeda. Apabila beberapa orang memahami struktur dari pernyataan kebutuhan perangkat lunak secara berbeda-beda, menyebabkan pernyataan tersebut dianggap sebagai ambigu. Kebutuhan perangkat lunak sebagai masalah ambigu dalam fase pengembangan perangkat lunak mengambil bagian sebanyak 56%, yang artinya kesalahan interpretasi pada kebutuhan perangkat lunak paling sering terjadi [3].

Terjadinya ambigu di dalam dokumen SKPL dapat memberikan dampak yang negatif terhadap pengembangan perangkat lunak, terutama saat validasi perangkat lunak ditemukan ketidaksesuaian antara hasil perangkat lunak dengan kebutuhan pengguna. Selain itu, ambigu akan menyebabkan SKPL menjadi tidak sesuai dengan salah satu standar kriteria SKPL yang baik dan benar berdasarkan IEEE 84, yaitu sebuah SKPL tidak ambigu jika dan hanya jika kalimat kebutuhan perangkat lunak hanya memiliki satu interpretasi (*unambiguous*) [1]. Apabila ambigu pada SKPL tidak terdeteksi dan tidak dilakukan perbaikan secepatnya pada fase awal pengembangan, kesalahan interpretasi akan menimbulkan masalah. Masalah yang terjadi pada fase-fase selanjutnya seperti saat melakukan desain, menentukan fungsional, membuat algoritma, menentukan performa, maupun merancang antarmuka dari perangkat lunak [2]. Oleh karena itu, sangat penting untuk menangani masalah ambigu dalam SKPL agar dampak negatif seperti biaya perbaikan yang sangat tinggi, waktu mengeluarkan perangkat lunak tertunda, hingga kegagalan pengembangan perangkat lunak dapat dihindari.

Berbagai penelitian telah dilakukan untuk menangani masalah ambigu dalam SKPL beberapa tahun terakhir, dengan metode penanganan untuk berbagai jenis ambigu yang beragam. Pada penelitian tahun 2017 dilakukan deteksi jenis ambigu berupa ambigu struktural menggunakan pola bahasa alami dengan metode pemrosesan bahasa alami [4] [5]. Pada tahun 2018, kembali dilakukan penelitian untuk jenis ambiguitas struktural dengan metode pemrosesan bahasa alami yang berfokus pada ambiguitas struktural dalam bentuk kalimat pasif [2]. Tidak hanya melakukan deteksi terhadap ambigu, terdapat juga penelitian yang melakukan rekomendasi perbaikan untuk ambiguitas yang terjadi dalam SKPL, dimana metode yang digunakan yaitu pemrosesan bahasa alami [6].

Dari beberapa penelitian yang telah dipaparkan, dapat dilihat bahwa jenis ambigu yang paling populer untuk diidentifikasi adalah ambigu struktural. Lalu, pemrosesan bahasa alami masuk ke dalam tiga kategori teratas sebagai teknik yang dapat menyelesaikan masalah ambigu [2]. Akan tetapi, penelitian-penelitian tersebut masih melakukan identifikasi ambigu hanya pada SKPL dalam bahasa Inggris [2], [4], [5], [6]. Sehingga, penelitian yang mengidentifikasi ambigu pada SKPL dalam bahasa Indonesia masih sangat jarang untuk dilakukan dan menjadi topik yang menarik untuk dikaji. Adapun penelitian ini bertujuan untuk melakukan deteksi ambiguitas struktural pada SKPL dalam bahasa Indonesia menggunakan pola bahasa alami. Metode yang digunakan yaitu pemrosesan bahasa alami berupa POS-*Tagging* Hidden Markov Model (HMM)-Viterbi dan *Regular Expression Parsing*. Untuk evaluasi akhir pada penelitian ini, diukur ketepatan deteksi ambiguitas struktural berdasarkan pola bahasa alami.

II. PENELITIAN TERKAIT

Penelitian-penelitian terkait dibagi berdasarkan komponen penting yang terdapat dalam penelitian ini.

A. Spesifikasi Kebutuhan Perangkat Lunak (SKPL)

SKPL merupakan spesifikasi untuk produk perangkat lunak tertentu, program atau serangkaian program yang melakukan fungsi tertentu [1]. Di dalam SKPL, kebutuhan perangkat lunak dinyatakan sebagai antarmuka sistem dan bagian perangkat lunaknya, performansi eksternal (kebutuhan non-fungsional) dan kebutuhan fungsional atas bagian perangkat lunak [1].

Pada penelitian [2], [4], [5], dan [6] untuk melakukan deteksi ambigu pada SKPL, bagian yang digunakan adalah pernyataan-pernyataan kebutuhan perangkat lunak yang berada pada bagian kebutuhan spesifikasi (*specific requirements*). Kebutuhan spesifik berisi kebutuhan perangkat lunak fungsional dan non-fungsional, mulai dari kebutuhan antarmuka eksternal, kebutuhan performansi, kebutuhan basis data, fungsi perangkat lunak, batasan desain, atribut sistem perangkat lunak dan informasi pendukung [1]. Dalam empat penelitian tersebut, pernyataan-pernyataan kebutuhan perangkat lunak yang digunakan berasal dari dokumen SKPL dalam bahasa Inggris.

B. Ambiguitas Struktural dalam SKPL

SKPL dinyatakan tidak ambigu, jika dan hanya jika, setiap pernyataan kebutuhan perangkat lunak di dalamnya hanya memiliki satu interpretasi [1]. Dapat dikatakan bahwa, terjadinya ambigu dalam SKPL apabila pernyataan kebutuhan perangkat lunak memiliki lebih dari satu interpretasi. Berdasarkan penelitian [7] terdapat empat jenis ambiguitas dalam SKPL yang ditulis dengan bahasa alami, yaitu: (a) ambiguitas leksikal: memperhatikan arti dari suatu kata dalam kalimat; (b) ambiguitas struktural: memperhatikan urutan kata dalam kalimat; (c) ambiguitas semantik: memperhatikan makna dari suatu kalimat; (d) ambiguitas pragmatis: berfokus pada hubungan antara arti dari kalimat dan konteksnya.

Fokus utama dalam penelitian ini yaitu melakukan deteksi kesalahan urutan kata dalam pernyataan kebutuhan perangkat lunak yang menyebabkan ambigu. Ambiguitas struktural merupakan jenis ambiguitas yang terjadi apabila urutan kata dalam sebuah kalimat dapat diberikan lebih dari satu struktur tata bahasa dan masing-masing memiliki arti berbeda. Pada penelitian [4], ambiguitas struktural dalam SKPL dapat dibedakan menjadi tiga jenis ambiguitas struktural;

1) Ambiguitas analitis terjadi apabila bagian yang penting dalam suatu frasa atau kalimat bersifat ambigu. Diantara pola ambiguitas analitis, yang dapat terjadi adalah struktur dari kelompok kata benda memiliki satu lingkup pengubah. Contohnya seperti dua buah kata benda diterangkan oleh satu kata sifat [4]. Perhatikan pernyataan berikut: **"data mahasiswa baru"**, dapat dibaca sebagai **"data (mahasiswa baru)"** atau **"(data mahasiswa) baru"**.
2) Ambiguitas koordinasi terjadi ketika: (a) lebih dari satu kata penghubung yaitu **"dan"** atau **"atau"** digunakan pada satu kalimat; (b) terdapat satu kata penghubung yang digunakan bersamaan dengan satu kata pengubah, seperti kata kerja atau kata sifat. Pernyataan berikut sebagai contoh pada kejadian pertama: **"Sistem mengatur pengguna dan *admin* dan pengelola dapat masuk ke sistem"**, dapat dibaca sebagai **"(Sistem mengatur pengguna dan *admin*) dan (pengelola dapat masuk ke sistem)"** atau **"(Sistem mengatur pengguna) dan (*admin* dan pengelola dapat masuk ke sistem)"**. Untuk contoh pada kejadian kedua, perhatikan kalimat berikut: **"*username* atau *password* salah"**, dapat dibaca sebagai **"(*username* atau *password*) salah"** atau **"*username* atau (*password* salah)"**.
3) Ambiguitas lampiran terjadi pada saat bagian penting pada sintaksis tertentu dari suatu kalimat, seperti frasa preposisional atau klausa relatif, dapat dilampirkan ke dua bagian kalimat. Pola ambiguitas lampiran yang paling populer adalah frasa preposisional yang dapat mengubah kata kerja atau kata benda. Contoh pernyataan kebutuhan perangkat lunak sebagai berikut: **"pengguna masuk sistem dengan *id*"**, dapat diinterpretasikan sebagai **"sistem yang memiliki *id*"** atau **"pengguna menggunakan *id* untuk masuk sistem"**.

Pada penelitian [2] dan [5] juga menambahkan satu jenis ambiguitas struktural, yaitu apabila dalam pernyataan kebutuhan perangkat lunak mengandung kata yang memiliki arti samar (*vague words*). Berdasarkan penelitian [8], kelompok kata yang mengandung arti samar dibagi menjadi lima;

1) Kata samar yang dapat merujuk pada lebih dari satu hal:
   - Kata ganti seperti **mereka**
   - Kata sifat penunjuk seperti **tersebut**, **ini** dan **itu**
2) Kata sifat yang memiliki arti samar:
   - karakteristik intrinsik seperti **lembut**, **keras**, **cepat**, **lambat**, **kuat**, **lemah**
   - karakteristik penilaian seperti **mudah**, **sulit**, **jelas**, **efisien**, **dapat diterima**, **memadai**, **bagus**, **jelek**, **masuk akal**, **cukup**, **berguna**, **penting**, **ramah pengguna**
   - karakteristik lokasi seperti **dekat** dan **jauh**
   - karakteristik pengurutan seperti **pertama**, **sebelumnya**, **selanjutnya**, **berikutnya**, **terakhir**
   - karakteristik sementara seperti **baru**, **tua**, **sekarang**, **nanti**, **lalu**, **segera**, dan **hari ini**
3) Kata yang menunjukkan posisi yang memiliki arti samar seperti **atas**, **bawah**, **depan**, **belakang**, **lebih**, **tinggi**, dan **rendah**
4) Kata kerja yang lebih bernilai kualitatif daripada kuantitatif seperti **meningkat**, **menurun**, **memaksimalkan**, **meminimalkan**
5) Frasa subyektif seperti **jika memungkinkan**, **hemat biaya**, dan **jika perlu**

C. Pemrosesan Bahasa Alami

Kebutuhan perangkat lunak paling sering ditulis menggunakan bahasa alami [1]. Pemrosesan Bahasa Alami merupakan kumpulan dari ilmu mengenai bagaimana manusia memahami dan menggunakan bahasa, sehingga dapat digunakan pada komputer agar dapat memahami dan memanipulasi bahasa alami sesuai dengan tujuan penggunaannya. Untuk menganalisis ambiguitas struktural, yang utama diperhatikan adalah tata bahasa dan struktur dari kalimat kebutuhan perangkat lunak, hal ini disebut sebagai analisis level sintaksis. Tiga teknik utama dalam pemrosesan bahasa alami yang digunakan untuk melakukan analisis pada level sintaksis khususnya pada ambiguitas struktural, berdasarkan penelitian [2], [4], dan [5];

1) Tokenisasi: Teknik untuk mempartisi pernyataan kebutuhan perangkat lunak yang akan di proses menjadi token yang terpisah, contohnya: kata-kata, angka, spasi, dan tanda baca [5]. Teknik ini dilakukan dengan tujuan untuk memungkinkan ekstraksi frasa yang efisien dan melakukan analisis teks berikutnya.

2) Part of Speech Tagging: Teknik *Part of Speech Tagging* biasa disebut dengan POS-*Tag*, dimana teknik ini akan memberikan *tag* pada setiap token yang dihasilkan dari proses tokenisasi, contoh *tag* seperti *noun* (NN), *verb* (VB), *adjective* (JJ) [5]. Pada berbagai penelitian yang telah disebutkan, SKPL yang dianalisis adalah SKPL dalam bahasa Inggris maka POS-*Tag* yang paling sering digunakan adalah POS-*Tagger* yang dibangun oleh *Stanford Natural Language Processing Group*. POS-*Tagger* tersebut hanya dapat digunakan pada empat bahasa yaitu bahasa Inggris, bahasa Cina, bahasa Jerman, dan bahasa Perancis. Oleh karena itu, salah satu cara untuk menganalisis SKPL dalam bahasa Indonesia dapat dengan membangun POS-*Tagger* yang memiliki kamus berisi kumpulan-kumpulan kalimat kebutuhan perangkat lunak dalam bahasa Indonesia yang sudah memiliki *tag*. Kamus tersebut dinamakan sebagai korpus anotasi. POS-*Tagger* yang dibangun, harus memperhatikan struktur kalimat dari korpus anotasi sebagai acuan dalam memberikan *tag* pada data pengujian berupa pernyataan-pernyataan kebutuhan perangkat lunak dari SKPL dalam bahasa Indonesia. Menurut penelitian [9], metode pemberian *tag* yang sesuai dengan konsep tersebut yaitu *Hidden Markov Model*-Viterbi (HMM-Viterbi). HMM-Viterbi mencapai nilai akurasi sekitar 90% untuk melakukan pemberian *tag* pada aplikasi masalah *Word Sense Disambiguation* dan nilai akurasi sekitar 80% pada aplikasi masalah *Grammatical Function Assignment* [9].

Korpus Anotasi yang digunakan dalam POS-*Tagger* HMM-Viterbi akan dibuat menggunakan *tagset* yang dibangun oleh Fam Rashel,dkk. pada penelitian [10]. *Tagset* ini berisi 250.000 kata dalam bahasa Indonesia beserta *tag* yang sesuai dan bersifat gratis dibawah lisensi *Creative Commons Licence*, dengan akurasi POS-*Tagger* sebesar 79% [10].

3) Shallow Parsing: Pada penelitian [4] dan [5], *shallow parsing* digunakan sebagai teknik yang akan mengidentifikasi pola bahasa alami dari tiap jenis ambiguitas struktural. Untuk dapat menggunakan *shallow parsing*, maka pola bahasa alami harus dibuat dalam bentuk ekspresi reguler berdasarkan hasil POS-*Tag*. Setelah itu, pola ekspresi reguler baru dapat di *parsing* untuk menganalisis pernyataan kebutuhan perangkat lunak yang sesuai dengan pola tersebut. Pola bahasa alami dalam bentuk ekspresi reguler yang sesuai dengan tiga jenis ambigu struktural yang sudah dibahas, telah didefinisikan pada penelitian [4];

- Pola ekspresi reguler ambiguitas analitis:
  Analytic: $\{< JJ >< NN.^* >< NN.^* >\}$
- Pola ekspresi reguler ambiguitas koordinasi:
  Coordination: $\{< JJ >< NN.^* >< CC >< NN.^* >\}$
- Pola ekspresi reguler ambiguitas lampiran:
  PP Attachment: $\{< VB.^* >< DT >? < JJ >^* < NN.^* >< IN >< DT >? < JJ >^* < NN.^* >\}$

Selain tiga jenis ambigu struktural diatas, telah disebutkan sebelumnya bahwa ambigu struktural dapat terjadi apabila pernyataan kebutuhan perangkat lunak mengandung kata dengan arti yang samar [2], [5]. Pola bahasa alami dari kejadian tersebut, telah didefinisikan pada penelitian [5];

- *Vague Terms*:
  $P_{VAG} = (Token.string \in Vague)$

Seluruh pola bahasa alami berdasarkan penelitian [4] dan [5], mengikuti tata bahasa pada bahasa Inggris. Sehingga, pada penelitian ini akan dilakukan modifikasi agar sesuai dengan tata bahasa bahasa Indonesia. Hal ini dikarenakan, walaupun secara umum tata bahasa pada bahasa Inggris dan bahasa Indonesia memiliki pola struktur yang sama seperti subjek-predikat-objek, tetap terdapat beberapa perbedaan [11]. Perbedaan paling utama yang berpengaruh terhadap penggunaan pola bahasa alami yaitu letak penggunaan kata pengubah seperti kata sifat dan kata kerja, dalam suatu kalimat. Pada bahasa Indonesia, pola kalimat adalah DM (Diterangkan–*word to modify* - Menerangkan–*modifier*), sedangkan dalam bahasa Inggris memiliki pola yaitu MD (Menerangkan–*modifier* - Diterangkan–*word to modify*) [11]. Contoh pola kalimat DM pada penelitian [11], seperti berikut:

$NP \rightarrow NN\ JJ$ (contoh: anak kecil)

$VP \rightarrow RB\ VB$ (contoh: sedang menulis)

Selain itu, dalam bahasa Indonesia juga tidak memperhatikan kata penentu (*determiner*) seperti dalam bahasa Inggris terdapat kata penentu *"the"* atau *"a"* [11]. Oleh karena itu, seluruh pola bahasa alami maupun pola ekspresi reguler pada jenis ambiguitas struktural yang memiliki pola MD, akan diubah ke dalam pola DM dan menghilangkan *tag* DT jika terdapat dalam pola. Hasil modifikasi pola-pola tersebut, seperti di bawah ini:

- Pola ekspresi reguler ambiguitas analitis:
  Analitis: $\{< NN.^* >< NN.^* >< JJ >\}$
- Pola ekspresi reguler ambiguitas koordinasi:
  Koordinasi: $\{< NN.^* >< CC >< NN.^* >< JJ >\}$
- Pola ekspresi reguler ambiguitas lampiran:
  Lampiran Preposisi: $\{< VB >< NN.^* >^* < JJ >< IN >< NN.^* >^* < JJ >\}$

D. Confusion Matrix

Confusion Matrix merupakan sebuah metrik yang digunakan pada penelitian [5] dan [6] sebagai bahan evaluasi. Evaluasi yang dilakukan yaitu untuk mengukur efektivitas dari pola bahasa alami dalam melakukan deteksi ambiguitas struktural pada pernyataan kebutuhan perangkat lunak. Komponen perhitungan dalam *confusion matrix* dapat dilihat pada Tabel I.

Evaluasi dengan *confusion matrix* akan berfokus pada masing-masing jenis ambiguitas struktural yang terdeteksi pada setiap pernyataan kebutuhan perangkat lunak [5]. Berdasarkan fokus tersebut, maka definisi perhitungan untuk tiap komponen dalam *confusion matrix* menurut penelitian [5] sebagai berikut:

Tabel I
KOMPONEN *Confusion Matrix*

	Actual Positive	*Actual Negative*
Predicted Positive	TP	FP
Predicted Negative	FN	TN

1) TP sebagai *True Positive*:
   Jumlah pernyataan kebutuhan perangkat lunak yang benar terdeteksi sebagai jenis ambiguitas struktural berdasarkan pola bahasa alami
2) FP sebagai *False Positive*:
   Jumlah pernyataan kebutuhan perangkat lunak yang salah terdeteksi sebagai jenis ambiguitas struktural berdasarkan pola bahasa alami
3) FN sebagai *False Negative*:
   Jumlah pernyataan kebutuhan perangkat lunak yang sebenarnya ambiguitas struktural tetapi tidak terdeteksi berdasarkan pola bahasa alami
4) TN sebagai *True Negative*:
   Jumlah pernyataan kebutuhan perangkat lunak yang benar bukan ambiguitas struktural dan tidak terdeteksi berdasarkan pola bahasa alami

Berdasarkan komponen-komponen di atas, pada penelitian [5] dihitung nilai dari hasil evaluasi berupa nilai ketepatan (*Precise*) dan nilai perolehan (*Recall*). Persamaan 1 merupakan bentuk persamaan untuk nilai ketepatan (*Precise*) dan Persamaan 2 merupakan bentuk persamaan untuk nilai perolehan (*Recall*).

$$Precise = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Nilai perolehan (*Precise*) dipengaruhi secara negatif oleh jumlah pernyataan kebutuhan perangkat lunak yang salah terdeteksi (FP), sedangkan nilai perolehan (*Recall*) dipengaruhi secara negatif oleh jumlah pernyataan kebutuhan perangkat lunak yang sebenarnya merupakan ambiguitas struktural tetapi tidak terdeteksi (FN) [5].

Pada penelitian [6], komponen-komponen *confusion matrix* digunakan pada cohen's Kappa untuk menilai hasil dari sistem yang dibuat. Nilai dari cohen's Kappa digunakan sebagai tolak ukur dari kinerja metode yang diusulkan dengan cara membandingkan pengetahuan yang dimiliki oleh ahli dengan keluaran yang dihasilkan oleh sistem. Persamaan 3 merupakan bentuk persamaan cohen's Kappa.

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \tag{3}$$

Adapun langkah-langkah perhitungan tiap komponen pada cohen's Kappa dijelaskan pada penelitian [12], sebagai berikut:

1) $P(A)$ yaitu nilai probabilitas dari hasil analisis yang sama. Nilai tersebut diperoleh dari jumlah *True Positive* dengan *True Negative* dan dibagi dengan total seluruh komponen *confusion matrix*
2) $P(E)$ yaitu nilai probabilitas dari perbedaan hasil analisis. Untuk memperoleh nilai tersebut, terlebih dahulu dihitung nilai probabilitas lainnya seperti berikut:
   - Probabilitas hasil ahli yang mengatakan bahwa pernyataan merupakan ambigu diperoleh dari jumlah *True Positive* dengan *False Negative* dan dibagi dengan total seluruh komponen *confusion matrix*
   - Probabilitas hasil ahli yang mengatakan bahwa pernyataan tidak ambigu diperoleh dari jumlah *False Positive* dengan *True Negative* dan dibagi dengan total seluruh komponen *confusion matrix*
   - Probabilitas hasil deteksi sistem yang menyatakan bahwa pernyataan ambigu diperoleh dari jumlah *True Positive* dengan *False Positive* dibagi dengan total seluruh komponen *confusion matrix*

- Probabilitas hasil deteksi sistem yang menyatakan bahwa pernyataan tidak ambigu diperoleh dari jumlah *True Negative* dengan *False Negative* dibagi dengan total seluruh komponen *confusion matrix*
- Probabilitas hasil ahli dan hasil deteksi sistem yang menyatakan bahwa pernyataan merupakan ambigu diperoleh dari perkalian antara probabilitas hasil ahli ambigu dan probabilitas hasil deteksi sistem ambigu
- Probabilitas hasil ahli dan hasil deteksi sistem yang menyatakan bahwa pernyataan tidak ambigu diperoleh dari perkalian antara probabilitas hasil ahli tidak ambigu dan probabilitas hasil deteksi sistem tidak ambigu
- Nilai $P(E)$ diperoleh dari jumlah probabilitas hasil ahli dan hasil deteksi sistem yang menyatakan bahwa pernyataan merupakan ambigu dengan probabilitas hasil ahli dan hasil deteksi sistem yang menyatakan bahwa pernyataan tidak ambigu

Berdasarkan penelitian [13], nilai yang dihasilkan dari cohen's Kappa memiliki lima jenis kelas kesepakatan, yang dapat dilihat pada Tabel II.

Tabel II
KELAS KESEPAKATAN PADA NILAI INDEKS KAPPA

kappa	**kelas kesepakatan**
$<= 0.20$	Tidak Sepakat (*insufficient*)
$0.21 - 0.40$	Lumayan Sepakat (*satisfactory*)
$0.41 - 0.60$	Cukup Sepakat (*sufficient*)
$0.61 - 0.80$	Sepakat (*good*)
$0.81 - 1.00$	Sangat Sepakat (*excellent*)

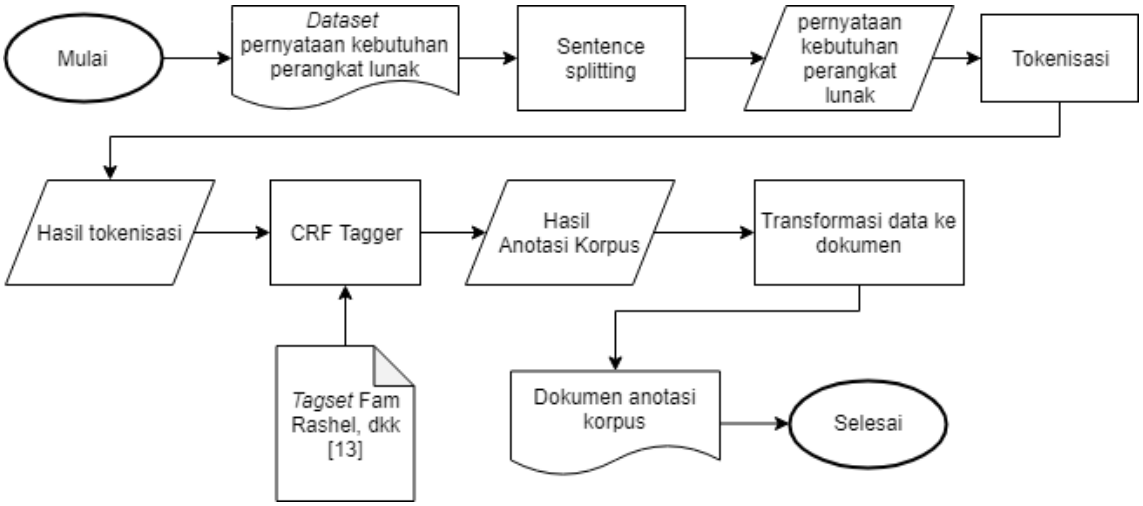
III. PERANCANGAN SISTEM

Sistem deteksi pola ambiguitas struktural dibangun menggunakan bahasa pemrograman *Python 3*, pada kernel *Jupyter Notebok*. Sistem terdiri atas tiga proses utama dimulai dari pembuatan korpus anotasi, lalu dokumen korpus anotasi digunakan dalam proses selanjutnya yaitu pra-proses dan POS-*Tagging* HMM-Viterbi untuk dataset pengujian dan proses terakhir adalah melakukan analisis struktur ambiguitas pada dokumen hasil *tagging* dari proses sebelumnya.

A. Pembuatan Korpus Anotasi

Untuk dapat melakukan POS-*Tagging* pada data pengujian, diperlukan korpus anotasi sebagai kamus dalam POS-*Tagger* yang dibangun. Dalam penelitian ini akan dibuat satu buah korpus anotasi, dimana dataset pernyataan kebutuhan perangkat lunak yang dijadikan sebagai korpus diambil dari *OpenScience tera-PROMISE repository* [14]. Pernyataan-pernyataan kebutuhan perangkat lunak dari dataset tersebut diambil dari berbagai SKPL dalam bahasa Inggris, sehingga di dalamnya berisi pernyataan kebutuhan perangkat lunak dalam bentuk bahasa Inggris. Dengan bantuan ahli dalam bidang Rekayasa Perangkat Lunak, pernyataan-pernyataan tersebut diubah ke dalam bahasa Indonesia dan digunakan sebagai dataset dalam penelitian ini. Gambar 1 mengilustrasikan proses pembuatan korpus anotasi.

Dokumen korpus anotasi dalam bahasa Indonesia disimpan dalam format *file* teks UTF-8. Secara keseluruhan, dataset terdiri dari 619 pernyataan kebutuhan perangkat lunak. Setelah dilakukan POS-*Tagging* menggunakan bantuan CRF-*Tagger* dan tagset dari Fam Rashel, dkk. [10], diperoleh dokumen korpus anotasi dengan jumlah kata dan jumlah *tag* sebanyak 11.391 buah untuk 619 kalimat.

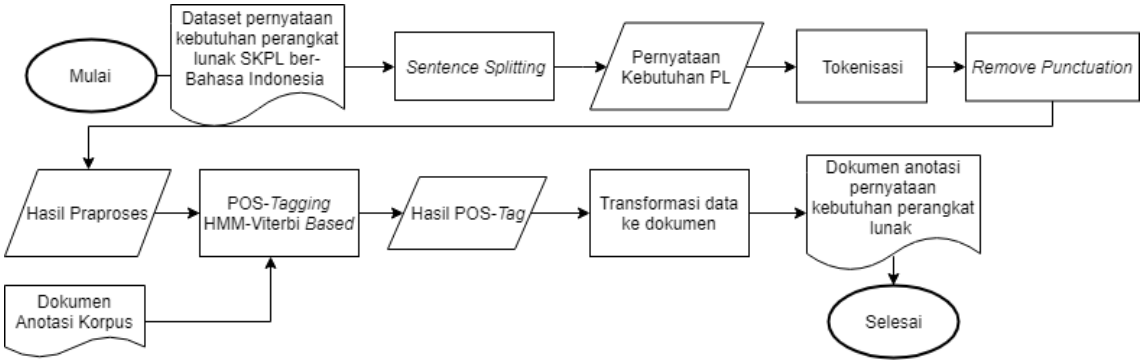


Gambar 1. Proses pembuatan korpus anotasi

B. Pra-proses dan POS-Tagging HMM-Viterbi

Pada proses ini, dilakukan pra-proses dan pemberian *tag* pada data pengujian. Terdapat lima dokumen SKPL dalam bahasa Indonesia yang pernyataan kebutuhan perangkat lunak dari tiap dokumen tersebut telah dipilih secara manual. Hasil dari pemilihan secara manual tersebut berupa lima dataset pernyataan kebutuhan perangkat lunak sebagai data pengujian yang digunakan dalam proses ini. Setiap data pengujian akan melewati pra-proses dan POS-*Tagging*. Gambar 2 merupakan alur pada pra-proses dan proses POS-*Tagging*.

Proses *sentence splitting*, *remove punctuation*, dan tokenisasi merupakan bagian dari pra-proses. Pada proses *remove punctuation*, tidak semua tanda baca dihilangkan dari pernyataan kebutuhan, melainkan hanya "." sebagai akhir dari kalimat, "!" "?" sebagai penunjuk kalimat perintah, kalimat tanya yang sangat jarang digunakan dalam pernyataan kebutuhan. Untuk tanda baca yang berada di tengah kalimat seperti "," tidak akan dihilangkan karena akan mempengaruhi struktur kalimat awal. Setelah hasil pra-proses telah selesai, masuk pada proses pemberian *tag* menggunakan metode HMM-Viterbi. Di bagian inilah, dokumen korpus anotasi yang telah dibuat pada proses pertama digunakan. POS-*Tagger* yang dibangun akan membaca pola struktur kalimat dan urutan *tag* pernyataan kebutuhan perangkat lunak dari korpus anotasi. Setelah urutan *tag* telah dibaca maka POS-*Tagger* mencari urutan pola *tag* dengan probabilitas paling bagus untuk dijadikan sebagai urutan pola tag pada pernyataan kebutuhan perangkat lunak dalam data pengujian.



Gambar 2. Alur pra-proses dan POS*Tagging*

Kelima dataset diproses secara satu persatu pada pra-proses dan POS*Tagging*, dan hasil dari pemberian tag pada tiap dataset disimpan ke dalam *file* dengan format teks UTF-8. Tabel III menunjukkan jumlah pernyataan kebutuhan perangkat lunak dari setiap dataset pengujian yang digunakan.

Tabel III
SPESIFIKASI DATASET PENGUJIAN

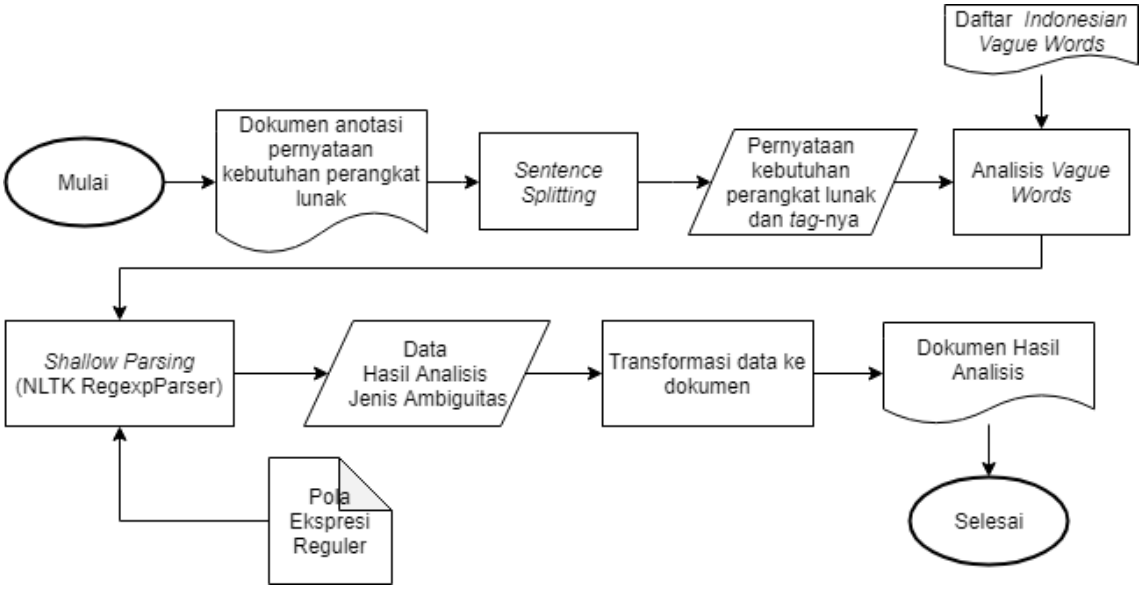
SKPL No.	**Dataset SKPL**	**Jumlah pernyataan**
1	*S*istem Perencanaan Biaya Perjalanan Ibadah Haji [15]	19 kalimat
2	Sistem Informasi Pelayanan Publik [16]	37 kalimat
3	SKPL untuk PDPT Dikti [17]	43 kalimat
4	*Web* Semantik Berbasis Ontologi dan *Thesaurus* [18]	18 kalimat
5	*S*INAPRA Berbasis Sistem Informasi Terpadu [19]	30 kalimat

C. Deteksi Pola Ambiguitas Struktural

Deteksi pola ambiguitas struktural dibangun berdasarkan dua model pola; (a) pola *vague words* berdasarkan penelitian [5]; (b) pola ekspresi reguler (analitis, koordinasi, preposisi) berdasarkan penelitian [4] yang telah dimodifikasi pada bagian II-C3. Pada proses ini, setiap pernyataan kebutuhan perangkat lunak dari masing-masing dataset pengujian dideteksi apakah mengandung satu atau lebih jenis ambiguitas struktural atau tidak, lalu diberi tanda. Gambar 3 merupakan ilustrasi dari langkah-langkah pendeteksian.

Pada proses analisis *vague words*, dokumen yang berisikan kata-kata samar (*vague*) telah dibuat dan berisi 33 kata samar dalam bahasa Indonesia dan telah disesuaikan dengan aturan pada penelitian [8]. Selanjutnya, proses analisis terakhir adalah *shallow parsing*, dimana pada proses ini digunakan bantuan NLTK RegexpParser untuk melakukan *parsing* dan mengecek pola ekspresi reguler terhadap setiap pernyataan kebutuhan perangkat lunak. Dalam satu pernyataan kebutuhan perangkat lunak, sangat mungkin terdapat lebih dari satu jenis ambiguitas struktural.

Seluruh dataset pengujian setelah selesai dilakukan analisis dan pemberian tanda jenis ambiguitas struktural, maka hasil akan ditampilkan. Tampilan berupa sebuah tabel berisi pernyataan kebutuhan perangkat lunak, urutan *tag*, serta jenis ambiguitas struktural yang terkandung di dalam pernyataan kebutuhan perangkat lunak tersebut. Kemudian, program akan menyimpan hasil tersebut ke dalam sebuah *file* CSV sebagai akhir dari proses deteksi.



Gambar 3. Langkah-langkah pendeteksian ambiguitas struktural

IV. EVALUASI DAN ANALISIS

A. Hasil Deteksi

Tahap deteksi dilakukan dengan mendeteksi jenis ambiguitas struktural dalam lima dokumen SKPL dalam bahasa Indonesia pada Tabel III. Di setiap deteksi pada masing-masing dokumen SKPL, dihitung jumlah tiap jenis ambiguitas struktural yang terdeteksi dalam setiap pernyataan kebutuhan perangkat lunak. Tabel IV merupakan jumlah setiap jenis ambiguitas struktural yang terdeteksi dalam setiap pernyataan kebutuhan perangkat lunak pada masing-masing dokumen SKPL dalam bahasa Indonesia.

Tabel IV
JUMLAH SETIAP JENIS AMBIGUITAS STRUKTURAL YANG TERDETEKSI

Jenis Ambiguitas Struktural	**SKPL No.1**	**SKPL No.2**	**SKPL No.3**	**SKPL No.4**	**SKPL No.5**
Vague Words	0	2	1	2	1
Ambiguitas Analitis	2	5	6	2	2
Ambiguitas Koordinasi	2	1	0	1	4
Ambiguitas Lampiran Preposisi	0	0	1	0	0
Total per SKPL	4	8	8	5	7

B. Analisis Hasil Deteksi

Sesuai dengan tujuan evaluasi pada pendahuluan, maka objektivitas analisis diukur pada kesesuaian antara hasil analisis secara manual dengan hasil deteksi dengan sistem. Skenario yang dilakukan untuk analisis yaitu hasil deteksi sistem yang terdiri atas lima *file* hasil analisis, digabungkan menjadi satu dokumen evaluasi dengan *file* CSV. total pernyataan kebutuhan perangkat lunak dalam *file* evaluasi tersebut sebanyak 146 pernyataan. *File* evaluasi yang berisi hasil deteksi sistem, dianalisis lagi hasilnya oleh ahli dan dihitung jumlah pernyataan kebutuhan perangkat lunak yang sesuai dan tidak sesuai antara hasil deteksi sistem dengan hasil analisis ahli. Perhitungan tersebut menggunakan *confusion matrix* seperti pada Tabel I, dengan hasil dari perhitungan dapat dilihat ada Tabel V, VI, VII, dan VIII.

Pada Tabel V, dapat dilihat bahwa dalam mendeteksi *Vague Words* di dalam pernyataan kebutuhan perangkat lunak, sistem tidak melakukan kesalahan dengan mendeteksi kata yang tidak berada di dalam daftar kata samar yang dibuat. Akan tetapi, terdapat dua kata yang dianggap oleh ahli sebagai kata dengan arti yang samar tetapi dua kata tersebut tidak terdeteksi oleh sistem karena tidak berada dalam daftar kata samar, dua kata tersebut ialah **"kemudahan"** dan **"kenyamanan"**.

Tabel V
Confusion Matrix UNTUK *Vague Words*

		Ahli	
		Ambigu	**Tidak Ambigu**
Sistem	**Ambigu**	6	0
	Tidak Ambigu	2	108

Tabel VI menunjukkan bagaimana sistem melakukan deteksi terhadap ambiguitas analitis dengan pola bahasa alami yang diusulkan. Dapat dilihat bahwa terdapat dua pernyataan kebutuhan perangkat lunak yang terdeteksi sebagai ambiguitas analitis, tetapi dianggap oleh ahli bukan sebagai ambiguitas analitis. Salah satunya pada pernyataan **"jenis perguruan tinggi"**, dimana pernyataan tersebut tidak mengandung ambiguitas analitis karena kata **"perguruan tinggi"** sudah sangat lazim dianggap oleh seluruh orang sebagai **"universitas"**.

Tabel VI
Confusion Matrix UNTUK AMBIGUITAS ANALITIS

		Ahli	
		Ambigu	**Tidak Ambigu**
Sistem	**Ambigu**	12	2
	Tidak Ambigu	0	108

Selanjutnya, sistem melakukan deteksi terhadap ambiguitas koordinasi yang hasilnya ditunjukkan pada Tabel VII. Hasil perbandingan analisis ahli dan deteksi sistem dalam ambiguitas koordinasi merupakan hasil yang paling beragam. Terdapat satu pernyataan yang dianalisis oleh sistem sebagai ambiguitas koordinasi, tetapi oleh ahli dinyatakan bukan ambiguitas yaitu **"keluar dan deskripsi singkat"**. Jika dilihat dari pola bahasa alami untuk ambiguitas struktural, pernyataan tersebut memang termasuk ambigu, akan tetapi jika memperhatikan seluruh kata dalam pernyataan dapat dilihat bahwa kata sifat **"singkat"** tersebut sangat jelas mengarah pada kata **"deskripsi"**, dan juga sebelum kata **"keluar"** terdapat kata **"menu"** yang menyebabkan pernyataan menjadi lebih spesifik. Adapun dua pernyataan kebutuhan perangkat lunak yang dianggap sebagai ahli adalah ambiguitas koordinasi merupakan ambigu yang terjadi karena penggunaan kata penghubung lebih dari satu di dalam pernyataan, dimana untuk kejadian tersebut tidak memiliki pola bahasa alami yang dibuat dalam sistem.

Tabel VII
Confusion Matrix UNTUK AMBIGUITAS KOORDINASI

		Ahli	
		Ambigu	**Tidak Ambigu**
Sistem	**Ambigu**	7	1
	Tidak Ambigu	2	108

Ambiguitas struktural terakhir yang dianalisis adalah ambiguitas lampiran preposisi. Tabel VIII memperlihatkan bahwa tidak ada kesamaan yang diperoleh dari hasil analisis ahli dan hasil analisis sistem terhadap pola bahasa alami pada ambiguitas lampiran preposisi. Pernyataan yang terdeteksi oleh sistem sebagai ambiguitas lampiran preposisi dianggap oleh ahli bukan sebagai ambigu karena terjadinya kesalahan pemberian *tag*, dimana kata **"akademik"** dalam pernyataan **"menampilkan jumlah dosen per jabatan akademik"** diberi *tag* JJ (kata sifat). Untuk pernyataan kebutuhan perangkat lunak yang dianalisis sebagai ambiguitas lampiran analisis oleh ahli tetapi tidak terdeteksi oleh sistem, terjadi karena perbedaan pola. Dalam hasil analisis ahli, pernyataan-pernyataan yang dianggap ambigu lampiran preposisi lebih banyak mengandung kata dengan *tag* NN (kata benda) tanpa *tag* JJ sebagai kata sifat, sehingga menyebabkan pernyataan tidak terdeteksi oleh sistem.

Tabel VIII
Confusion Matrix UNTUK AMBIGUITAS LAMPIRAN PREPOSISI

		Ahli	
		Ambigu	**Tidak Ambigu**
Sistem	**Ambigu**	0	1
	Tidak Ambigu	4	108

Sebagai hasil akhir untuk memperoleh objektivitas analisis, pada penelitian ini digunakan cohen's Kappa untuk menilai kinerja dari pola bahasa alami yang diajukan dalam mendeteksi ambiguitas struktural pada SKPL dalam bahasa Indonesia. Dalam menghitung tingkat kesepakatan dengan cohen's Kappa, ahli bertindak sebagai penguji pertama dan sistem yang dibangun bertindak sebagai penguji kedua. Setiap pola bahasa alami dari jenis ambiguitas struktural dihitung nilai Kappa untuk mengetahui kinerja dari pola tersebut.

Dengan diperolehnya seluruh komponen nilai dari *confusion matrix* pada setiap jenis ambiguitas struktural, maka dengan menggunakan Persamaan 3 dapat diperoleh nilai indeks Kappa untuk setiap pola bahasa alami dalam mendeteksi ambiguitas struktural dengan kelas kesepakatan berdasarkan Tabel II. Nilai Kappa untuk tiap pola bahasa alami dapat dilihat pada Tabel IX.

Tabel IX
NILAI INDEKS KAPPA UNTUK DETEKSI AMBIGUITAS STRUKTURAL DENGAN POLA BAHASA ALAMI

	Pola Bahasa Alami			
	Vague Words	**Analitis**	**Koordinasi**	**Lampiran Preposisi**
Nilai Kappa	0.8481	0.9139	0.8098	−0.0143
Kelas Kesepakatan	Sangat Sepakat	Sangat Sepakat	Sepakat	Tidak Sepakat

Dapat dilihat pada tabel diatas, nilai indeks Kappa untuk ambiguitas lampiran preposisi bernilai negatif. Hal ini disebabkan karena nilai probabilitas dari perbedaan hasil analisis ahli dan deteksi sistem, lebih besar daripada nilai probabilitas kesamaan hasil analisis ahli dan deteksi sistem. Probabilitas perbedaan hasil analisis memiliki nilai yang tinggi daripada kesamaan analisis terjadi karena ahli hanya setuju pada hasil deteksi sistem yang tidak ambigu (*True Negative*), sedangkan terdapat satu hasil deteksi sistem yang menyatakan ambigu tetapi tidak disetujui oleh ahli. Lalu, ahli juga menemukan pernyataan yang seharusnya ambigu tetapi dideteksi oleh sistem bukan sebagai ambigu. Kedua hal tersebut membuat nilai probabilitas pernyataan yang benar sebagai ambigu hampir mencapai nilai 0.0.

Berdasarkan hasil pengukuran nilai indeks Kappa terhadap empat pola bahasa alami, dapat diketahui bahwa pola untuk ambiguitas lampiran preposisi memiliki nilai indeks Kappa yang sangat rendah dibandingkan dengan ketiga pola bahasa alami lainnya. Pola bahasa alami untuk *vague words*, ambiguitas analitis, dan ambiguitas koordinasi memiliki nilai indeks Kappa yang tinggi yaitu 0.8481, 0.9139 dan 0.8098, sedangkan untuk ambiguitas lampiran preposisi hanya memperoleh nilai indeks Kappa sebesar -0.0143. Sehingga, pola ambiguitas lampiran preposisi berada pada kelas kesepakatan paling rendah yaitu ahli tidak sepakat dengan hasil deteksi sistem. Hal ini disebabkan oleh kata preposisi (IN) yang dilampirkan tidak hanya mempengaruhi kata sifat (JJ), tetapi juga dapat mempengaruhi kata lainnya seperti kata benda (NN) untuk menjadi ambigu. Selain itu, kesalahan pemberian *tag* juga mempengaruhi nilai indeks Kappa dari ambiguitas lampiran preposisi.

Untuk ketiga pola bahasa alami dengan perolehan nilai indeks Kappa yang tinggi membuat pola *vague words*, pola ambiguitas analitis, dan pola ambiguitas koordinasi berada pada kelas kesepakatan paling tinggi yaitu ahli sangat sepakat terhadap hasil deteksi sistem. Kelebihan pada pola *vague words* yaitu pada daftar kata samar dalam bahasa Indonesia sangat beragam. Adapun kelebihan pada pola ambiguitas analitis dan ambiguitas koordinasi adalah struktur dasar dari bahasa Indonesia yaitu kata benda (NN) diikuti oleh kata sifat (JJ) sebagai *modifier* sangat sering ditemui pada pernyataan kebutuhan perangkat lunak dalam bahasa Indonesia.

Akan tetapi ketiga pola bahasa alami tersebut belum dapat mencapai nilai indeks Kappa 1.0 karena masih terdapat berbagai kelemahan. Untuk kelemahan pola *vague words* dipengaruhi oleh terdapatnya kata-kata samar dalam bahasa Indonesia yang memiliki awalan dan akhiran sehingga tidak terdeteksi oleh sistem, karena dalam daftar kata samar yang dibuat hanya terdapat kata dasarnya saja. Pada pola bahasa ambiguitas analitis dan ambiguitas koordinasi, kelemahan terjadi apabila pola bertemu kata majemuk dengan struktur kata benda (NN) diikuti oleh kata sifat (JJ) dan dideteksi sebagai ambigu, sedangkan kata majemuk tersebut sebenarnya hanya memiliki satu arti sehingga tidak dapat disebut ambigu.

V. Kesimpulan

Berdasarkan analisis hasil deteksi ambiguitas struktural menggunakan pola bahasa alami untuk *vague words*, ambiguitas analitis, dan ambiguitas koordinasi berada pada kelas kesepakatan yang paling tinggi yaitu ahli sangat sepakat terhadap hasil deteksi sistem. Hal ini menyatakan bahwa ketiga pola bahasa alami dengan nilai indeks Kappa 0.8481, 0.9139 dan 0.8098 memiliki tingkat ketepatan yang tinggi. Sehingga pola *vague words*, pola ambiguitas analitis dan pola ambiguitas koordinasi dapat digunakan untuk mendeteksi ambiguitas struktural yang terjadi pada SKPL dalam bahasa Indonesia. Akan tetapi, ahli tidak sepakat dengan hasil deteksi sistem dengan pola bahasa alami untuk ambiguitas lampiran preposisi. Dengan nilai indeks Kappa sebesar -0.0143, membuat pola ambiguitas lampiran preposisi memiliki tingkat ketepatan yang buruk. Sehingga, pola bahasa alami untuk ambiguitas lampiran preposisi perlu dilakukan perbaikan lebih lanjut di penelitian selanjutnya agar dapat digunakan dalam mendeteksi ambiguitas struktural pada SKPL dalam bahasa Indonesia.

Tingkat ketepatan yang tinggi untuk pola *vague words*, pola ambiguitas analitis dan pola ambiguitas koordinasi membuat hasil deteksi ambiguitas struktural pada SKPL dalam bahasa Indonesia untuk ketiga pola tersebut dapat dipercaya. Sehingga, dengan menerapkan pola *vague words*, pola ambiguitas analitis dan pola ambiguitas koordinasi pada sistem yang dibangun dapat membantu menangani masalah ambiguitas struktural pada SKPL dalam bahasa Indonesia.

PUSTAKA

[1] IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board. Ieee recommended practice for software requirements specifications. Institute of Electrical and Electronics Engineers, 1998.

[2] ALI OLOW JIM'ALE SABRIYE and WAN MOHD NAZMEE WAN ZAINON. An approach for detecting syntax and syntactic ambiguity in software requirement specification. *Journal of Theoretical & Applied Information Technology*, 96(8), 2018.

[3] T Jaison Vimalraj and B Seema. Identification of ambiguity in requirement specification using multilingual word sense. *2016 International Journal of Adavanced Research in Computer and Communication Engineering*, 5(6):386–388, 2016.

[4] Reza Khezri. Automated detection of syntactic ambiguity using shallow parsing and web data. 2017.

[5] Benedetta Rosadini, Alessio Ferrari, Gloria Gori, Alessandro Fantechi, Stefania Gnesi, Iacopo Trotta, and Stefano Bacherini. Using nlp to detect requirements defects: an industrial experience in the railway domain. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 344–360. Springer, 2017.

[6] Depandi Enda and Daniel Siahaan. Rekomendasi perbaikan pernyataan kebutuhan yang rancu dalam spesifikasi kebutuhan perangkat lunak menggunakan teknik berbasis aturan. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 5(2):207–216, 2018.

[7] Daniel M. Berry, Ph. D. Computer Science, Michael M. Krieger, and Ph. D. Mathematics. From contract drafting to software specification: Linguistic sources of ambiguity - a handbook version 1.0, 2000.

[8] Donald Firesmith. Specifying good requirements. *Journal of Object Technology*, 2(4):77–87, 2003.

[9] Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. In *Third Conference on Applied Natural Language Processing*, 1992.

[10] Fam Rashel, Andry Luthfi, Arawinda Dinakaramani, and Ruli Manurung. Building an indonesian rule-based part-of-speech tagger. In *2014 International Conference on Asian Language Processing (IALP)*, pages 70–73. IEEE, 2014.

[11] Rosa A Sukamto and Dwi H Widyantoro. Probabilistic parsing for indonesian language. 06 2019.

[12] Yunata Dede Pratiwi, Daniel Oranova Siahaan, and Sarwosri Sarwosri. Kakas bantu analisis kerancuan kebutuhan perangkat lunak berbasis aturan. *Jurnal Teknik ITS*, 1(1):A228–A230, 2012.

[13] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[14] Openscience tera-promise repository. https://terapromise.csc.ncsu.edu/!/#repo/view/head/requirements/nfr. *Online*; Diakses 5 Juni 2019.

[15] Yudhi Kurniawan, Yuswanto , Program Studi, Sistem Informasi, Fakultas Sains, Dan Teknologi, Studi Program, Fakultas Akuntansi, Dan Ekonomi, Chung Ma, Chung , Puncak Villa, N Tidar, Malang , and Jawa Timur. Software requirement specification sistem perencanaan biaya perjalanan ibadah haji sesuai dengan standard ieee 830-1998. 02 2014.

[16] Deny Wiria Nugraha. Software requirement dalam membangun sistem informasi pelayanan publik. *Mektrik*, 13(3), 2011.

[17] PT. PRADIPTA INTIMEDIA SELARAS. Spesifikasi kebutuhan perangkat lunak untuk pangkalan data pendidikan tinggi. www.ai3.itb.ac.id/~basuki/inherent/PDPT/SRS\%20Dikti.doc. *Online*; Diakses 5 Juni 2019.

[18] Fajar Triadmojo. Rancang bangun web semantik berbasis ontologi dan thesaurus berpedoman pada web semantic design method (wsdm) guna memperoleh hasil pencarian resep masakan provinsi sulawesi selatan yang relevan. http://repository.bakrie.ac.id/262/, 2016. *Online*; Diakses 5 Juni 2019.

[19] Nur Hadi Waryanto. Software requirements specification sinapra berbasis sistem informasi. *Pythagoras: Jurnal Pendidikan Matematika*, 7(2), 2012.