Model design for grammatical error identification in software requirements specification using statistics and rule-based techniques

F P Putra[1], D Enda[1]

[1] Department of Informatic Engineering, Politeknik Negeri Bengkalis, Jalan Bathin Alam, Riau, Indonesia

E-mail: fajri@polbeng.ac.id

Abstract. The statement of functional requirements statement that refers to software requirements specifications (SRS) is a reference for stakeholders in making software. The SRS document uses a different English grammar due to limited knowledge by the software development team, making it difficult for the development team to read and understand the specification documents for software requirements properly. To overcome this grammatical error, a method is needed to resolve grammatical errors. The method used to identify grammatical errors is based on a trigram language model. In the last research, trigram language models showed quite good performance in identifying grammatical errors based on probabilistic n-gram language models. To improve the performance of the n-gram language model this study also utilizes a corpus namely Penn Tree Bank Corpus. Provide recommendations, this study uses rules-based techniques. A number of rules are made to provide recommendations for inappropriate grammar. So that the software requirements specification compiler can check for grammatical errors and improve the quality of the software requirements specification document.

1. Introduction

Software requirements are specific attributes that are specifications of functional requirements and non-functional requirements of a software system. Software requirement specification (SRS) documents are generally compiled using natural language, which is the language we use in daily conversation [1]. Although natural language is more easily understood by stakeholders and is very flexible to accommodate various changing user needs. SRS describes all the requirements that must be owned by the system in order to conform to its function. These requirements are typically illustrating features of underdevelopment systems. These features not only describe its functional requirements FR but also its non-functional requirements NFR. Functional requirements and non-functional requirements have the same important role for system development. Both FR and NFR have an impact on the overall success of the system [2]. However, the use of natural language has the disadvantage that the authors of the SKPL document usually do not take great care of the grammatical errors used. The grammar in each language used has different characteristics, for example in Indonesian it has a specificity in the article clothing, prepositions, suffixes and conjunctions. English also has specificities to verbs, regular verbs, irregular verbs, nouns, adjectives and many others. The use of each word category has several rules that must be followed to produce proper and good grammar.

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

The preparation and writing of software requirements document generally follow the standard standards that have been established as layouts or official frameworks in the English version, for example, the IEEE 830-1998 standard. Most of the drafting teams of software requirements specification documents use English as the language for writing SRS documents. So that in this study using grammar in English as an object being discussed. Improper grammar can reduce the quality of software requirements specification documents, this makes it difficult for the development team to read and understand the software requirements specification documents properly. So that topics and methods regarding writing correct grammatical errors according to the rules and rules of grammar are very open to be developed and further researched.

This research proposes a method design to identify and provide recommendations for grammatical errors in the software requirements specification document. The method used to identify grammatical errors is based on trigram language models, as discussed in the Enda and Putra (2019) research [3], the trigram language model shows good performance in identifying grammatical errors based on probabilistic n-language models gram. To provide recommendations, this study uses rules-based techniques. A number of rules are made to provide recommendations for inappropriate grammar. So that the software requirements specification compiler can check for grammatical errors and improve the quality of the software requirements specification document. To improve the performance of the n-gram language model this study also utilizes a corpus namely Penn Tree Bank Corpus.

2. Theoretical basis

2.1. Software requirement

Previous research that has discussed grammar in preparing software requirements specification documents, like research by Enda and Putra (2019) [3] proposed a method for identifying grammatical errors in software requirements statements using the probabilistic n-gram language model. 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 precession, recall and f-measure values. Based on the tests that have been done, it can be seen that the acquisition of precession, recall and f-measure values in the trigram language model with a threshold = 0.1 has the highest values of 83%, 85%, and 86%, respectively. This shows that the trigram language model can identify grammatical errors well. when compared to the bigram testing scenario. Based on this research, to improve accuracy in identifying grammatical errors, this study uses a combination of n-gram language statistical models with rule based.

2.2. Spelling checker

Spelling checker is the initial process for identifying grammatical errors. Spelling checker is used to detect and handle word errors, word errors usually occur because during the process of typing text so that it can change the meaning of a word and even the meaning of a sentence. In addition, it is also due to mechanical failures such as hand or finger slip, and also arises due to the writer's ignorance in spelling words. Detecting and correcting errors of words has certain complexity. Predefined confusion sets, restricted numbers of corrections, and other assumptions are used as system constraints to simplify the problem [4].

Detecting word errors can be done using the help of a computer application, the application is able to examine documents and look for writing errors in them and if necessary, warn the writer and offer some suggestions for correcting errors. Spelling checkers are divided into two types namely non-word error spell checkers and real-word spell checkers. Non-word error spell checkers handle misspelled words that are formed due to typos, while real-word error spell checkers prioritize handling words that replace error words in sentences [5].

In this study spell checking can automatically be done using the Hunspell Checker library. Hunspell is a spell checker and morphological analyser designed for languages with rich morphology and complex word compounding or character encoding. Hunspell has been used by LibreOffice, OpenOffice, and many others. This library provides a system for tokenizing, stemming and spelling in almost all languages or the alphabet. The R package presents a high-level spell checker and also a low-level stemmer and tokenizer that analyses or extracts individual words from various formats (text, html, xml, latex). The Hunspell library uses a special dictionary format that determines which characters, words and conjugations are valid in certain languages. The spell checker process consists of the following steps [6]:

- Tokenizing is process decompose documents by extracting the words we want to check.
- Stemming is process analyse each word by breaking it and conjunction affixes.
- Search dictionary combination of words and affixes, if valid for the selected language, no spelling errors are found.
- For wrong words, suggest corrections by finding similar (true) words in the dictionary.

2.3. N-Gram based statistical techniques

N-gram-based statistical techniques are techniques for mapping a sentence into n-letter sub-words or strings where n usually consists of one word (unigram), two words (bigram) or three words (trigrams) [7]. The n-gram model is a statistical technique that models word sequences into probability values. If a sentence is stated as {w1, w2, ..., wn-1, wn} and n is the number of words in a sentence, then to calculate the probability value of a sentence using a unigram statistical model it can be calculated using the resulting equation:

$$P(W_1^n) \approx \prod_i P(w_i) \tag{1}$$

Meanwhile, to calculate the probability value of each pair of two words in a sentence can be calculated using the maximum likelihood estimate equation following:

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

N (w_{i-1}, w_i) dan N(w_{i-1}) indicate the number of occurrences of a word or frequency N (w_{i-1}, w_i) and N(w_{i-1}) in the corpus. The probability value of a sentence using the bigram statistical model can be calculated by the Chain rule written with the following equation:

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

The probability value of each pair of three words in a sentence can be calculated using the following maximum likelihood equation:

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

Where N (w_{i-2}, w_{i-1}, w_i) dan N (w_{i-2}, w_{i-1}) indicates the number of occurrences of a word or frequency N (w_{i-2}, w_{i-1}, w_i) dan N(w_{i-2}, w_{i-1}) in the corpus. The probability value of a sentence using the trigram statistical model can be calculated using the following rules:

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

The rule-based approach was discussed by Naber in 2003 [9], which developed a method and tool that can check layout and grammatical errors in English sentences using rule-based techniques. To increase the accuracy of the proposed method, a large corpus, the British National Corpus, is used to ensure that the system built does not report too many errors in the correct sentence, so that the accuracy of the system can be increased. The rule-based technique proposed by Naber can be used to express rules that describe errors at the phrase level and not just at the word level. So that this method becomes one of the reliable methods in checking for grammatical errors.

A statistical based approach was discussed by Henrich and Reuter in 2009 [8] who proposed a method and tool to examine and correct grammatical errors that are independent of language types based on a statistical approach. The languages used as the language of research trials are English and German. Evaluation results show that approaches to different languages, although the accuracy of grammar checks varies, the reason is due to differences in the morphological richness of languages. The limitation of this method is that the quantity and quality of training data largely determine the method's performance. The proposed improvement method shows that there are still many problems in finding all grammatical errors [8].

3. Research methods

So that the research process can be carried out directed, the research procedure can be seen in Figure 1. Based on the picture 1. For data collection, this study uses data several statements of software requirements contained in the research Tjong in 2008 [10]. The study collected several datasets of requirements statements from several different problem domains to test the research in detecting confusion in the software requirements specification document.

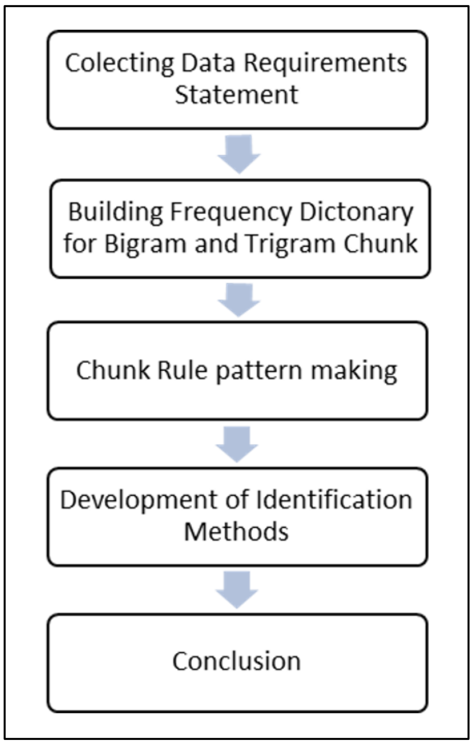
Colecting Data Requirements Statement
Building Frequency Dictonary for Bigram and Trigram Chunk
Chunk Rule pattern making
Development of Identification Methods
Conclusion

Figure 1. Research flow.

After the data collection of software requirements statement, then making a dictionary of bigram frequency and trigram chunk is done to get the probability value of each bigram and trigram pair based on the frequency found in the learning corpus. This frequency dictionary will later be used as a reference dictionary for the proposed method. The next step is creating a chunk rule pattern. The resulting rule pattern will be used as a rule to correct chunk pattern errors that have been detected in the sentence, at this stage the rule-based technique works. The next stage is the development of the proposed identification method, then continues to draw conclusions about the development of the method.

4. Results

4.1. Research dataset

The following dataset will be used for each data set that has several software requirements statements. The total dataset used in this study is 12 datasets which can be seen in the following Table 1.

Table 1. Research dataset.

No	Dataset	Source
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)

4.2. Building frequency dictionary for bigram and trigram chunk
The dictionary of the frequency of the bigram and trigram chunk pair patterns is used to determine the probability of each bigram and trigram chunk pair pattern. The stages of making a dictionary of bigram frequency and trigram chunk can be seen in Figure 2.

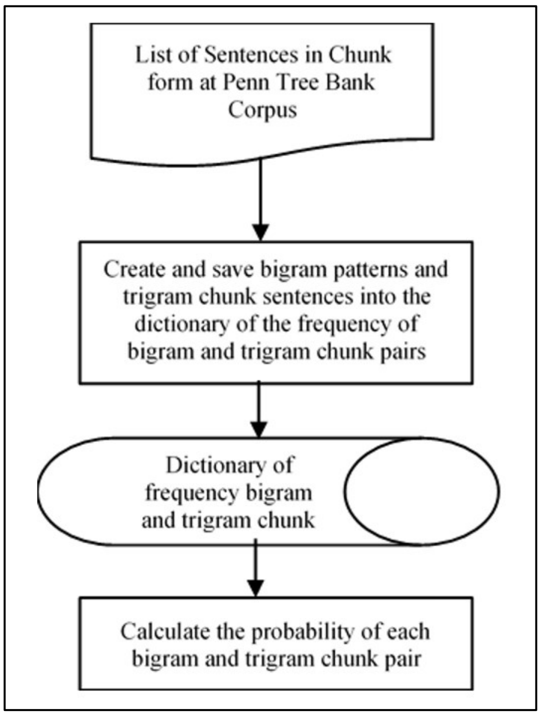
List of Sentences in Chunk form at Penn Tree Bank Corpus
Create and save bigram patterns and trigram chunk sentences into the dictionary of the frequency of bigram and trigram chunk pairs
Dictionary of frequency bigram and trigram chunk
Calculate the probability of each bigram and trigram chunk pair

Figure 2. Stages building frequency dictionary for bigram and trigram chunk.

The dictionary of bigram and trigram chunk frequencies is based on the number of occurrences of the bigram and trigram chunk pair patterns in the corpus. The corpus used in this study is Penn TreeBank Corpus, where in this corpus there are around 3914 various sentences that have been given pos_tag and chunk_tag.

The standard used to mark each part of a word in a sentence or Part of Speech Tag (POS-Tag) is the Penn TreeBank Tagset. Penn TreeBank Tagset consists of 46 different tagset where the number of each occurrence or frequency of each tagset in Penn TreeBank Corpus can be seen in Table 2:

Table 2. Frequency of Penn TreeBank tagset in corpus.

No	POS-Tag	Frequency	No	POS-Tag	Frequency	No	POS-Tag	Frequency
1	PRP$	766	17	NN	13166	33	RBS	35
2	VBG	1460	18	FW	4	34	RBR	136
3	VBD	3043	19	,	4886	35	CD	3546
4	``	712	20	.	3874	36	-NONE-	6592
5	VBN	2134	21	TO	2179	37	EX	88
6	POS	824	22	PRP	1716	38	IN	9857
7	"	694	23	RB	2822	39	WP$	14
8	VBP	1321	24	-LRB-	120	40	MD	927
9	WDT	445	25	:	563	41	NNPS	244
10	JJ	5834	26	NNS	6047	42	-RRB-	126
11	WP	241	27	NNP	9410	43	JJS	182
12	VBZ	2125	28	VB	2554	44	JJR	381
13	DT	8165	29	WRB	178	45	SYM	1
14	#	16	30	CC	2265	46	UH	3
15	RP	216	31	LS	13			
16	$	724	32	PDT	27			

Each frequency in the Penn TreeBank tagset will determine the probability value of a pair of bigram and trigram tags. The higher the tag frequency on the corpus, the greater the probability value.

4.3. Building frequency dictionary for bigram and trigram chunk

The next stage is the creation of chunk rule patterns, this rule pattern is used as a rule pattern dictionary in determining rule patterns that are suitable for correcting identified chunk pattern errors. The following are the steps to create a chunk dictionary pattern which can be seen in Figure 3.

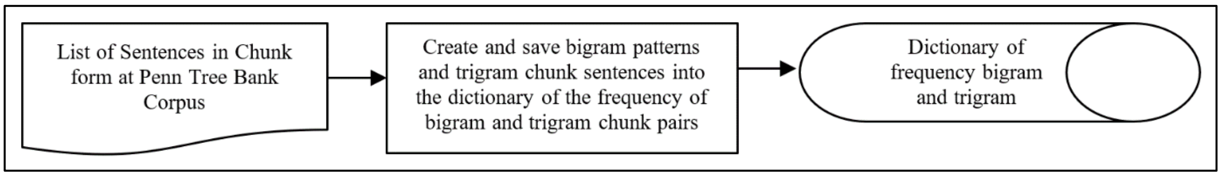
List of Sentences in Chunk form at Penn Tree Bank Corpus
Create and save bigram patterns and trigram chunk sentences into the dictionary of the frequency of bigram and trigram chunk pairs
Dictionary of frequency bigram and trigram

Figure 3. Chunk rules pattern dictionary.

The list of sentences in the form of chunk in the Penn Tree Bank corpus will be used as input data to create a chunk rule pattern, where the resulting rule pattern is a sequence of tag patterns in the sentences that make up the chunk. The resulting rule pattern will be saved into a chunk rule pattern dictionary which will later be used as a pattern matching reference dictionary to provide improvements to the chunk patterns found wrong in the sentence.

4.4. Building frequency dictionary for bigram and trigram chunk

The grammatical error identification method proposed in this study uses a probabilistic N-Gram language model, where the identification of grammatical errors will be analysed at the phrase level (chunking) in a statement of software requirements. The proposed method for identifying grammatical errors in this study can be seen in Figure 4. Based on Figure 4 explained as follows:

- Data used as input from the system is a statement of software requirements.
- The sentence or statement of need will first be done preliminary processing that is checking spelling errors where the module used in this study is the Hunspell Checker. This module works using n-gram similarity techniques, rule-based and pronunciation dictionaries. The result of this module is a list of words containing spelling errors and recommendations for improvement.
- The next step is to tokenize sentences and change each word resulting from the process of tokenize sentences into basic words. Phrase filtering and language style checking are performed to increase system performance in detecting grammatical errors. Marking each part of the word is done using the Tagger Base library based on the Penn Tree Bank Tagset.
- The results of the POS-Tagging statement of needs will then be used as input to create a pattern of grouping words based on phrases (chunking process). The process of grouping words (chunking) in this study uses the Classifier Chunk Parser library.
- Based on the chunk pattern this statement of need will be made a bigram and trigram chunk pattern. Probability values are obtained from queries of bigram and trigram chunk pairs in the bigram frequency dictionary and trigram chunk that have been created.
- After getting the value of each bigram chunk pair, the next step is to identify the grammatical error based on the probability value of the bigram chunk.
- The discovery of grammatical errors is done by comparing the probability values of each bigram and trigram chunk on the statement of needs with a predetermined threshold value.
- Determination of the appropriate threshold value will determine the performance of the identification method. So that in this study the threshold value for the bigram chunk pattern was 0.1, 0.15, and 0.2. This is done to find the best threshold value from several value variations. If the chunk probability value is equal to or exceeds the threshold value specified then the grammatical error statement needs not found and displays the results of the analysis, on the contrary if it does not meet the threshold value then the grammatical error needs statement found.
- After getting the bigram pattern pair and the trigram chunk identified as the wrong pattern pair the next step is to do the chunk rule pattern matching in the chunk rules dictionary that has been created.
- The final stage will display recommendations for matching chunk pairs as a method output.

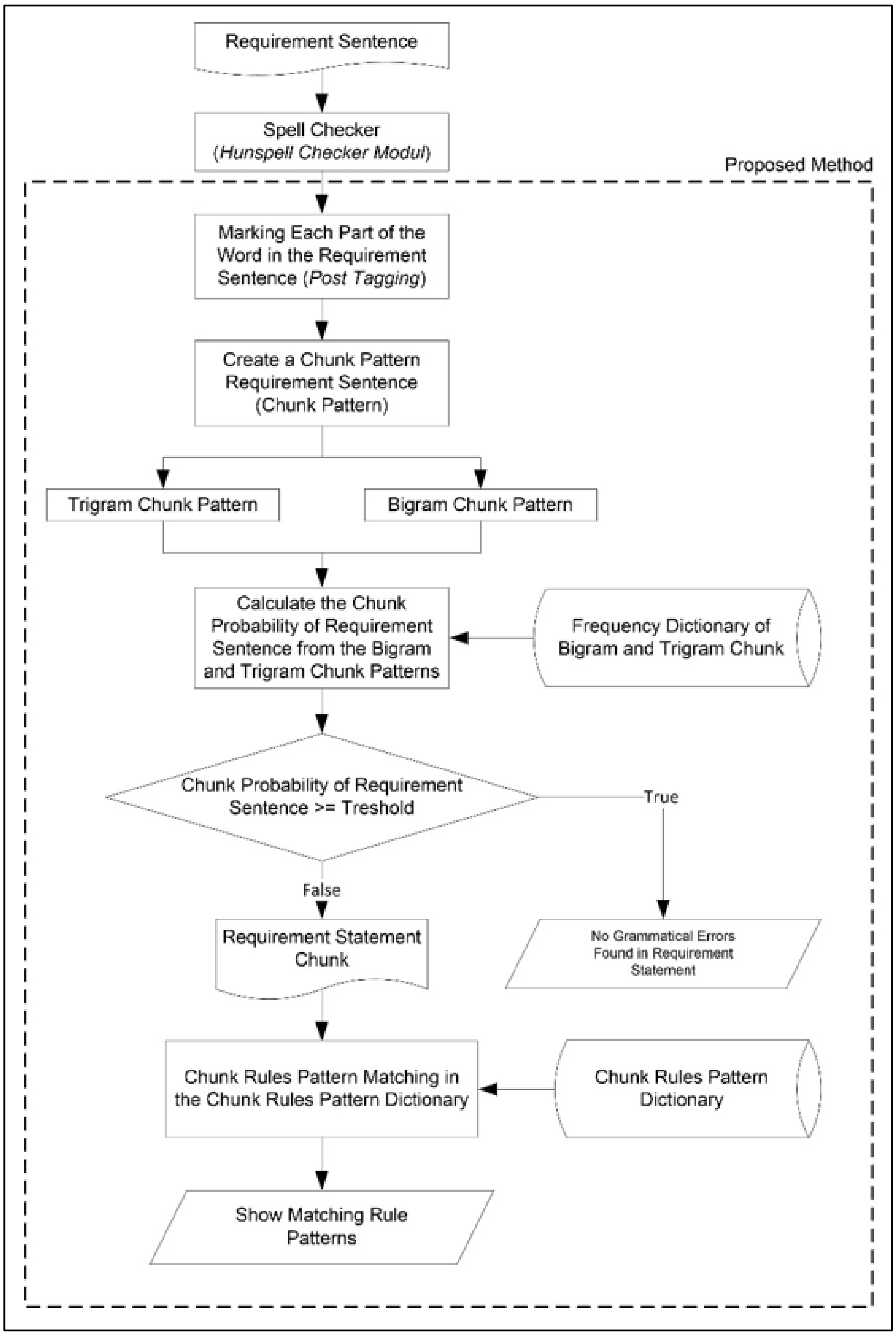
Requirement Sentence
Spell Checker (*Hunspell Checker Modul*)
Proposed Method
Marking Each Part of the Word in the Requirement Sentence (*Post Tagging*)
Create a Chunk Pattern Requirement Sentence (Chunk Pattern)
Trigram Chunk Pattern
Bigram Chunk Pattern
Calculate the Chunk Probability of Requirement Sentence from the Bigram and Trigram Chunk Patterns
Frequency Dictionary of Bigram and Trigram Chunk
Chunk Probability of Requirement Sentence >= Treshold
True
False
Requirement Statement Chunk
No Grammatical Errors Found in Requirement Statement
Chunk Rules Pattern Matching in the Chunk Rules Pattern Dictionary
Chunk Rules Pattern Dictionary
Show Matching Rule Patterns

Figure 4. Proposed method.

5. Conclusion
From the research that has been done, researcher has succeeded in designing a design model to solve the grammatical problems in SRS related documents asking for software requirements. The approval method uses a combination of two methods, namely the statistical-based method (N-gram) and the rule-based method. In the process of verifying spelling errors of the language at the start using the Hunspell Checker module, further spelling is done by spelling to be a basic word using the Penn TreeBank Set

dictionary. The results of the tag set are used as input for the chunking process (Classifier Chunk Parser) and obtain the probability value of the bigram and trigram chunk. After getting the bigram pattern pair and the trigram chunk identified as the wrong pattern pair, the next step is to do the chunk rule pattern matching in the chunk rules dictionary that has been created and set the suitable chunk pair pattern as a method.

The results of this design are ready to be further developed because the model is quite easy to understand. For prototype design can be evaluated by calculating the value of precision, recall and f-measure. The range of measured values is between 0 and 1 and sometimes measured in percentage values. The results of this design are ready to be implemented in an RPL documentation management software tool.

6. References

[1] Enda D and Siahaan D 2018 *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)* **05** 207-216
[2] Asif M, Ali I, Malik M S A, Chaudary M H, Tayyaba S and Mahmood M T J 2019 *IEEE Access* **7** 26164-26176
[3] Enda D and Putra F P 2019 *Journal of Computer Engineering, System and Science* **4** 130-137
[4] Lu J Chris, Aronson A R, Shooshan S E and Demner-Fushman D 2019 *Journal of The American Medical Informatics Association* **26** 211-218
[5] Soleh M Y and Purwarianti A 2011 *International Conference on Electrical Engineering and Informatics* 1-6
[6] Hornik K and Murdoch D 2011 *The R Journal* **3** 22-28
[7] Wu J, Chang J and Chang S J 2013 *International Journal of Computational Linguistics & Chinese Language Processing* **18** 31–44
[8] Henrich V and Reuter T 2009 *LIS Grammar Checker: Language Independent Statistical Grammar Checking* (Iceland: Hochschule Darmstadt & Reykjavík University)
[9] Naber D 2003 *A Rule-Based Style and Grammar Checker* (Germany: Bielefeld University)
[10] Tjong S F 2008 *Avoiding Ambiguitiy In Requirements Specifications* (England: University Nottingham)

Acknowledgments
We would like to thank Bengkalis State Polytechnic which has provided research support in the form of PNBP grants so that this research can be carried out well. Acknowledgments were also given to UPT P3M Bengkalis State Polytechnic for supporting the research to run smoothly.