Pemuisi: a constraint satisfaction-based generator of topical Indonesian poetry

Fam Rashel[1] and **Ruli Manurung**[2]
Faculty of Computer Science
Universitas Indonesia
Depok 16424, Indonesia
[1]fam.rashel@ui.ac.id, [2]maruli@cs.ui.ac.id

Abstract

Pemuisi is a poetry generation system that generates topical poems in Indonesian using a constraint satisfaction approach. It scans popular news websites for articles and extracts relevant keywords that are combined with various language resources such as templates and other slot fillers into lines of poetry. It then composes poems from these lines by satisfying a set of given constraints. A Turing Test-style evaluation and a detailed evaluation of three different configurations of the system was conducted through an online questionnaire with 180 respondents. The results showed that under the best scenario, 57% of the respondents thought that the generated poems were authored by humans, and that poems generated using the full set of constraints consistently measured better on all aspects than those generated using the other two configurations. The system is now available online as a web application.

Introduction

Poetry is a form of literature with an emphasis on aesthetic aspects such as alliteration, repetition, rhyme and rhythm, which distinguishes it from other literary forms. In poetry, the specifically chosen wording is infused with much more meaning and expressiveness, hence the difficulty in translating poetry compared to translating prose.

Poetry generators are systems capable of automatically generating poetry given certain restrictions and contexts. Gervás (2002) presents an overall evaluation of various poetry generators. Other notable works include Manurung (2003), Colton et al. (2012), and Toivanen et al. (2013).

Colton et al. (2012) proposes an architecture for poetry generation that is able to generate poetry along with a commentary on the various decisions it chose in constructing the poem. Toivanen et al. (2013) show how constraint logic programming can be used to generate poems that satisfy various poetic and linguistic constraints.

Our system, *Pemuisi* (a rather archaic Indonesian word meaning *poet*), combines the architecture and approach proposed by Colton, particularly the fact that generated poems are based on current news articles, with the constraint satisfaction-based approach of Toivanen, and generates poems using a combination of handcrafted and automatically extracted Indonesian language resources.

The main contribution of this work, aside from the combination of these approaches, and the adaptation to the Indonesian language, is the user evaluation that was conducted, as both Colton et al. (2012) and Toivanen et al. (2013) present no user evaluation.

In the *Background* section below, relevant previous work will be presented, especially the generator described in Colton et al. (2012). The *Language Resources* section introduces the various language resources required by our system. *Pemuisi* utilizes two kinds of language resources, templates and slot fillers. Slot fillers are divided into poetic words and keywords. Each of these language resources play their own role in satisfying poetic properties. In the *Constraint Satisfaction Poetry Generation* section, we present our constraint satisfaction approach to poetry generation. Poetic features such as number of lines, syllable counts, and rhymes are defined as a set of constraints. Hence, the system will try to satisfy the constraints while composing the poem. The *Experiments and Evaluation* section details the various experiments we conducted. We took the output for evaluation through online questionnaire with 180 respondents. The results were analyzed based on several criteria, such as structure, topic, and message of the poem. Finally we briefly discuss our implementation of a live web application that continuously monitors popular news websites for articles and produces corresponding poems.

Background

Manurung (2003) claims that poetry must satisfy the three properties of meaningfulness, grammaticality, and poeticness. The property **meaningfulness** states that a text should aim to convey a message or concept that has meaning when readers try to interpret the text. This property could be a common element for any text, not just poetry. The property **grammaticality** states that a poem must comply with linguistic rules defined by a given grammar and lexicon. This property is also one of the most common needs that must be met by any natural language generation (NLG) system. The last one is **poeticness**. This property states that poetry must contain strong characteristics of poetry elements, e.g. pho-

netic features such as metre and rhyme. This is the key property to distinguish poetry from other texts. Such requirements imply that it is insufficient for poetry generation systems to simply produce random words.

Colton et al. (2012) states that the first poetry generator to be developed is most probably the *Stochastische Texte* system developed by Lutz that utilizes a small lexicon consisting of sixteen subjects and predicates from Kafka's *Das Schloβ*. The system randomly chooses words from Kafka's works and fits them into a grammatical template that previously has been defined.

Other poetry generators can be grouped into several categories. Referring to Gervás (2002) who provides a taxonomy of poetry generation systems based on the approach and techniques used, there are at least four different categories, namely (i) template-based systems, (ii) generate and test systems, (iii) evolutionary-based systems, and (iv) case based resoning systems.

Another perspective from Colton et al. (2012) is that most existing poetry generation systems behave more as assistants, with varying degrees of automation, for the human user who has provided the majority of the resulting context of the poem. Departing from this view, they propose a fully autonomous computer system poet, which we refer to as Full-FACE. Full-FACE is a corpus-based poetry generator that utilizes various resources such as lexical databases, simile corpus, news articles, pronouncing dictionary, and sentiment dictionary. Given these resources, the system is able to generate poetry independently, to the extent of deciding its own form of poetry such as the number of lines, rhyme structure, message, and the theme of the poetry. Overall, this system consists of several stages. The first is **retrieval,** where the various resources needed to produce poetry are gathered, i.e. the Jigsaw Bard simile corpus, a set of constraints, and a collection of keyphrases from Guardian news articles that will be the topic of poetry. Then we go to **multiplication** stage, where the aforementioned resources are permutated to obtain variations in order for the resulting poetry to be more expressive. For example, the existing simile corpus yields similes in the form of a triple <*object, aspect, description*>, which contains information about the simile, e.g. the tuple <*child, life, happy*> represents the simile "*as happy as a child's life"*. Multiplication is done by applying three kinds of substitution methods: using the DISCO corpus, the simile corpus, or WordNet to find words that are similar. During the **combination** stage, Full-FACE produces lines of poetry through combining simile corpus, the simile multiplication result, and article keyphrases. This combination is done by following a certain template. For example, there is a keyphrase "*excess baggage*" that match the simile "*the emotional baggage of a divorce*" can be applied to the process of combination into line poem "*Oh divorce! So much emotional excess baggage*" in accordance with the specifications of the template. Finally, the results of the previous process are collated in accordance with the user-given constraints or existing template in the last stage called **instantiation**.

A fully autonomous computer system poet was established by handing over high-level control to the system itself. This was done by the system with context generation alongside with the commentary. Context generation is a process of how context, topics, templates to structure the poetry, such as lines and rhyme patterns, determined by the system to form poetry. In order to deliver the context, commentary generation is a process to produce a commentary on the poetry made. In general, the comments contain the condition of the heart/emotions at the time of making the poetry, a summary of the article reference, and how the process of writing poetry.

Language Resources

Our system requires at least two types of resources, templates and slot fillers. These resources are necessary pieces for the system to make poetry. To prepare these resources we need to go through several processes. Hereby is the explanation of each process.

Templates

A template is a ready-made sentence (canned text) that has one or more slots to be filled by certain words. Each slot is associated with a part-of-speech tag, such as noun, verb, adjective, or pronoun. Templates are used to fulfill the grammaticality property of a poem.

Firstly, we applied an Indonesian part-of-speech tagger on a corpus consisting of 213 poems written by famous Indonesian poets. Template extraction is then performed by removing words that have specific part-of-speech tags, i.e. nouns, verbs, adjectives, and pronouns. The positions of these removed words become slots to be filled later. A slot is also associated with a part-of-speech tag indicating what words may fill the slot. For example, consider the following sentence:

Aku mencintai kamu dengan sepenuh hati
I love you with full heart
I love you with all of my heart.

Each word is initially tagged with its part-of-speech. Subsequently, we remove all words tagged as <PR> (pronoun) and <NN> (noun) to obtain the following template:

<PR> mencintai <PR> dengan sepenuh <NN>
<PR> love <PR> with full <NN>
? love ? with all of (my/your/their) ?.

After extracting such templates, the feasibility and appropriateness of a template is evaluated by considering the semantic specificity embedded in the template. This consideration is important to prevent providing too much context to the system, and to avoid the risk of plagiarism against an existing line of poetry. Furthermore, with this evaluation we can determine the limits of human intervention concerning the poetic knowledge provided to the system. To illustrate, consider the following two templates (note, VBI indicates an intransitive verb):

ada yang <VBI>, ada yang <VBI>
some that <VBI>, some that <VBI>
some are ?, some are ?

ya, <PR> tahu mereka masih menggunakan <NN>
yes, <PR> know they still use <NN>
yes, ? know that they still use ?

From these two examples we can see that the latter template already carries with it a fairly specific semantic message. We believe such templates should be avoided. Furthermore, the former template is much more general and does not overconstrain the semantics. Such are the desired templates for our knowledge base. Using this consideration, we manually identified 22 templates to be used in our experiments. Theoretically it is possible to automate this process by computing the ratio of open class words remaining in the template, as opposed to function words, or closed class words.

The selected templates, along with illustrative English translations, are presented in Table 1. Note that due to grammatical differences, the translations may not be well-formed, but they are intended to illustrate the level of generality of the templates. In particular, note that almost all of the canned text contained within the templates consist of function words.

Additionally, other information that must be provided along with the template is the number of lexical slots available and the number of syllables that currently exist in the canned text of the template. This information is required for the selection process, such as to count the number of syllables and keywords. Figure *1* provides an example of how

```
TEMPLATE: SYLLABLE COUNT, SLOT COUNT
[<nn>,dan,<nn>,bisa,dibawa,<vbi>]: 6, 3
[<pr>,<vbi>,<pr>,<vbi>]: 0, 4
```

Figure 1. Two examples of templates

templates are represented in our system. It shows two templates (#11 and #5 from Table 1). The first template contains 6 syllables within its canned text ("**dan**", "**bi**", "**sa**", "**di**", "**ba**", "**wa**"), and has 3 lexical slots (2 nouns and an intransitive verb). The second template has 0 syllables within its canned text and has 4 lexical slots (2 pronouns and 2 intransitive verbs).

Slot fillers

Slot fillers are simply words used to fill the slots contained in the template. They must also be associated with a part-of-speech tag and other information that is needed in the selection process. Slot fillers can be divided into two types, keywords and poetic words.

Keywords are slot fillers that will determine the theme of the constructed poem. These words are expected to fulfill a sense of meaningfulness in the poem so that readers of the poem will capture some message that is being conveyed.

At the beginning of the poetry generation process, we crawl popular Indonesian news websites such as kompas.com and detik.com. This is motivated by FullFACE, which crawls the Guardian news website to determine the theme of the poem. An article is selected based on a given criteria, such as most recent, most commented on, or most read. After selecting an article, keyword extraction is done to obtain the keywords. Keyword extraction is done using simple unigram statistics, with stopword removal.

Templates manually selected to be used	Illustrative translations of the templates
1. <PR>	1. <PR>
2. <PR> <VBI>	2. <PR> <VBI>
3. <PR> <VBI> <RB>	3. <PR> <VBI> <RB>
4. <PR> <VBT> <PR>	4. <PR> <VBT> <PR>
5. <PR> <VBI> <PR> <VBI>	5. <PR> <VBI> <PR> <VBI>
6. dari <NN> ke <NN>	6. from <NN> to <NN>
7. adalah <ADJ> <NN>	7. there is <ADJ> <NN>
8. tapi <PR> <VBI>	8. but <PR> <VBI>
9. <PR> dan <PR> <VBI>	9. <PR> and <PR> <VBI>
10. <PR> ini hanyalah <NN>	10. <PR> is just <NN>
11. <NN> dan <NN> bisa dibawa <VBI>	11. <NN> and <NN> can be brought <VBI>
12. <PR> <VBT> <NN> bersama <PR>	12. <PR> <VBT> <NN> with <PR>
13. <VBT> <PR> adalah <ADJ> untuk <PR>	13. <VBT> <PR> is <ADJ> for <PR>
14. dengan penuh <ADJ> dalam <NN>	14. with full <ADJ> in <NN>
15. tak ada lagi <ADJ> dan <NN>	15. no more <ADJ> and <NN>
16. adakah <NN> padaku atau <NN>	16. is there <NN> with me or <NN>
17. ada yang <VBI> ada yang <VBI>	17. some are <VBI> some are <VBI>
18. mengapa <NN> <VBI>	18. why <NN> <VBI>
19. oh <PR> begitu <ADJ>	19. oh <PR> is so <ADJ>
20. terlalu <ADJ> bagi <PR>	20. too <ADJ> for <PR>
21. <NN> menjadi <NN>	21. <NN> becomes <NN>
22. apa itu <NN	22. what is <NN>

Table 1. List of templates along with illustrative translations

```
WORD: POS, PRONOUNCE, SYLL.COUNT, FLAG
senja: nn, [s, eu, n, j, aa], 2, keyword
```

Figure 2. Example of keyword representation

```
WORD: POS, PRONOUNCE, SYLL.COUNT, FLAG
kalbu: nn, [k, aa, l, b, oo], 2, filler
```

Figure 3. Example of poetic word representation

Words that have the most frequency of occurrence will be the keywords candidate. An expanded collection of keywords is then constructed by identifying words that frequently occur together with the extracted words using the Wortschatz-Leipzig Corpora Collection (Quasthoff et al., 2006).

Other information that should be associated with each keyword is its pronunciation and syllable count. This information is used for the selection process, such as for the computation of rhyme and the number of syllables in a line. Figure 2 shows an example of how keywords are represented in our system. In this example, the keyword, senja, has a part-of-speech value of NN (noun), pronunciation (s, eu, n, j, aa), 2 syllables ("sen" and "ja"), and a "keyword" flag that indicates that senja is one of the keywords of the article.

For the experiments that we conducted, we selected 3 news articles and extracted a total of 247 keywords: 88 from the 1st article, 72 from the 2nd article, and 87 from the 3rd article.

Poetic words are obtained from the same corpus of poetry used for template extraction. They are designed to help the generated poem satisfy the property of poeticness. Unlike other constraints that are more focused on the structure, this property is more focused on the selection of words to add to the aesthetics of the poem.

The frequency of appearance of every word in the existing corpus is computed and stopwords are removed. The fifty words that most frequently appear in the corpus are selected. Finally, we apply an Indonesian POS Tagger to obtain their part-of-speech tags. Poetic words tend to convey a more general concept as opposed to the specific keywords based on news article. Furthermore, they tend to be more archaic in nature.. The technical representation of poetic words is similar to how keywords are represented, as they must also be associated with pronunciation, and number of syllables. Figure 3 shows an example of how poetic words are represented in our system. The poetic word kalbu has a part-of-speech value of NN (noun), pronunciation (k, aa, l, b, oo), 2 syllables ("kal" and "bu") and a "filler" flag that indicates that the word kalbu is a poetic word.

Constraint Satisfaction Poetry Generation

Our system adapts the approach proposed by Colton et al. (2012). The system creates poetry from the collection of templates combined with a particular set of words. The result of combining templates with keywords and poetic words will be the lines that will be collated to construct the poem. Overall, the system is implemented as three stages: retrieval, combination, and selection.

It differs from Full-FACE in the following ways. Firstly, *Pemuisi* is a much more knowledge-poor system, as there are far fewer lexical resources available for Indonesian as there are for English, in particular the Jigsaw Bard resource that appears to provide a major contribution to the poeticness and coherence to the poems generated by Full-FACE. Secondly, following Toivanen et al. (2013) (and to a lesser degree, Manurung (2003)), it explicitly treats the generation process as a constraint satisfaction problem, which affords a declarative formulation of the generation process, and the use of efficient off the shelf constraint solvers. Currently, *Pemuisi* is implemented as a logic program in Prolog. All lexical resources are encoded as factual assertions in the Prolog database, and the poetic constraints are implemented as clauses with subgoals that must be satisfied. Lastly, *Pemuisi* does not attempt the handing over of high level control that is implemented in Full-FACE, which is equipped with various definitions of aesthetics.

Retrieval

During this stage, a simple retrieval is performed by taking the relevant resources previously described from the knowledge base. Given an input news article, the system will populate the Prolog database with all relevant keywords, poetic words, and appropriate templates. The retrieval process can be set to randomly reorder the sequence of factual assertions, so that the systematic Prolog depth first search can yield novel results on repeated runs. Figure 4 shows an example of the output of this stage.

Combination

After collecting all the necessary resources, the system can start building the poem from the simplest unit, namely the poetry line. The combination process produces a poetry line through merging of a template with slot filler(s) by obeying certain rules. Each slot in the template must be filled with precisely one slot filler. A slot can only be filled with a slot filler with a corresponding part-of-speech tag. For example, a slot with a POS tag of NN (noun) can only be filled by a keyword or poetic word with a POS tag of NN. The system

```
TEMPLATE:
[<nn>,dan,<nn>,bisa,dibawa,<vbi>]: 6, 3
[<pr>,<vbi>,<pr>,<vbi>]: 0, 4

SLOT FILLER
aku:pr,[aa,k,oo],2,filler
kau:pr,[k,aa,oo],2,filler
senja:nn,[s,eu,n,j,aa],2,keyword
kalbu:nn,[k,aa,l,b,oo],2,filler
bayang:nn,[b,aa,y,aa,ng],2,keyword
pergi:vbi,[p,eu,r,g,ee],2,filler
kembali:vbi,[k,eu,m,b,aa,l,ee],3,filler
menunggu:vbi,[m,eu,n,oo,ng,g,oo],3,keyword
```

Figure 4. Example output of retrieval stage

will exhaustively consider all possible valid combinations of templates and slot fillers.

Consider the following example. Suppose that the resources obtained from the retrieval stage are as in Figure 4, which means the system must now combine 2 templates with 8 slot fillers consisting of: 2 <PR> slot fillers; 3 <NN> slot fillers, 2 of which are keywords; and 3 <VBI> slot fillers, 1 of which is a keyword.

Going by the previous explanation of how the process is done then all slot combinations are instantiated with the corresponding slot fillers to form the poem lines. Based on simple observation, it can be calculated that the number of combinations of lines of poetry can be generated from the collection of the above resources. 63 valid combinations of poetry lines can be obtained from the combination of templates and corresponding slot fillers.

Selection

After the combination stage, the system now has a large collection of poem lines that are ready to be built into larger units, i.e. the poem itself. This stage combines the lines that have been previously obtained as results of the combination. The resulting poem must satisfy the elements of poetry, such as the number of syllables, rhyme, rhythm, and number of lines. Such poetic elements are defined as constraints. Constraints that will be used include:

1) **Number of lines**: a constraint that states the number of poetry lines. As explained in the combination stage, the definition used for a single line is a result of a combination of a template with one or more slot filler.
2) **Rhyme**: a constraint that states the rules of rhyme in between lines of the poetry.
3) **Number of words**: a constraint that states the number of words contained in a single line of poetry. The number of words can be specified differently for each row.
4) **Number of syllables**: a constraint that states the number of syllables contained in a single line of poetry. Number of syllables can be specified differently for each row.
5) **The number of keywords relative to the number of slots**: a constraint that states the number of keywords relative to the number of slots contained in the whole poetry. In order to be more intuitive and easier, this constraint is expressed as a percentage. It can be used to control how the content of the poem focuses on a topic.

The above set of constraints must be met when choosing combinations of line results from the previous stage. This is an important point of the concept of constraint satisfaction approach as also seen in Toivanen et al. (2013).

From the previous example results obtained 63 lines of poetry that can be built into a combination of poetry. For instance, assume the following constraints:

1) The poem consists of 2 lines.
2) Line 1 and 2 share the same end-of-line rhyme.
3) Line 1 consists of 6 words with a total of 12 syllables.
4) Line 2 consists of 4 words with a total of 10 syllables.
5) 40% of all slots must be filled with content keywords.

If we only look at the first constraint, it can be calculated that there are 63^2 poems that could be generated. But the more we continue to meet the subsequent constraints, the less the combinations of lines of poetry that are able to meet all the constraints.

There are at least three cases that may occur after the selection process is done: (i) the system does not produce a single poem at all, (ii) it produces exactly a single poem, and (iii) it generates more than one poem.

If no poem is produced, it means there is no combination that successfully meets the constraints that have been defined. In this case, the constraints will be gradually relaxed and the selection stage repeated until eventually a poem can be produced. In loosening constraints, the constraint that has the lowest precedence is first chosen to be relaxed. This process is repeated until the system is capable of producing a poem that satisfies the remaining constraints.

If the system is able to produce one or more poems, it will randomly select one as its eventual output. Another alternative is to provide all the poetry as the output.

Pemuisi is currently equipped with six poem structures, i.e. sets of constraints, to be used during the experiments. The purpose of the provision of six alternative structures is for the poetry generated by the system to be more diverse.

An Illustrative Example

In this section we provide an example of the output of *Pemuisi.* It was run to construct a poem based on an article from an Indonesian news portal, kompas.com, about Sir Alex Ferguson's retirement in 2013 as Manchester United head coach. We situated *Pemuisi* to compose a poem with full constraint parameter and then randomly took 3 stanzas. Figure 5 shows the poem made by *Pemuisi*.

The corresponding constraints which became the reference for *Pemuisi* while generating this poem can be seen in Figure 6. While comparing Figure 5 and Figure 6, we can see that the set of constraints were all satisfied by the resulting poem.

Experiments and Evaluation

We conducted experiments using several constraint configurations through an online web-based questionnaire to see the respondents' opinions about the poetry generated by the system. Information about the experiment was distributed through various mailing lists and social media channels (e.g. Facebook, Twitter), targeting native Indonesian speakers including public groups, academic communities, and poetry appreciation communities in order to provide a more balanced and valid distribution of respondents, ranging from a layman's appreciation of poetry to communities that specifically discuss poetry appreciation. At the end of the data collection, we managed to obtain 180 respondents.

fergie pergi	*fergie is gone*
ferguson pensiun, ferguson berhenti	*ferguson retired, ferguson stopped*
adakah masa padaku atau juri	*is there time with me or jury*
fergie berhenti	*fergie stopped*
fergie pensiun sendirian	*fergie retired alone*
dengan penuh merah dalam perjuangan	*with full red in struggle*
tak ada lagi akrab dan perjalanan	*no more friendship and trips*
fergie pensiun sendirian	*fergie retired alone*
dengan penuh biru dalam kesedihan	*with full blue in sadness*
tak ada lagi akrab dan pertandingan	*no more friendship and matches*
ferguson, ini hanyalah kompetisi	*ferguson, this is just a competition*
usia dan keputusan bisa dibawa pensiun	*age and decisions can be brought in retirement*
fergie, ini hanyalah tradisi	*fergie, this is just a tradition*
pemain dan manajemen bisa dibawa pensiun	*players and management can be brought in retirement*

Figure 5. Illustrative output of *Pemuisi*

Constraint configurations

There are three constraint configurations that were applied. In the first configuration, the full set of poetic constraints are applied, and a ratio of 50% of the open slots must be filled by content keywords. The second configuration is similar to the first, but in this case all the open slots must be filled by content keywords. Finally, the loose constraint configuration is one where the system is more or less left unguided to generate poems, with the only constraints being the use of templates, part of speech tags, and the number of lines to be generated, i.e. poetic features such as syllable counts, rhymes, and content keyword ratios are ignored. Obviously, respondents were not made aware of the distinction of these three configurations, and were simpy asked to rate the perceived quality of the generated poems regardless of the configuration of the generator.

Stanza 1
Number of lines: 4
Line 1 – number of words: 2; number of syllables: 4
Line 2 – number of words: 4; number of syllables: 12
Line 3 – number of words: 5; number of syllables: 12
Line 4 – number of words: 2; number of syllables: 5
Line 1, 2, 3, and 4 rhyme with each other
Keywords composition: 100%

Stanza 2
Number of lines: 6
Line 1 – number of words: 3; number of syllables: 10
Line 2 – number of words: 5; number of syllables: 12
Line 3 – number of words: 6; number of syllables: 12
Line 4 – number of words: 3; number of syllables: 12
Line 5 – number of words: 5; number of syllables: 12
Line 6 – number of words: 6; number of syllables: 12
Line 1, 2, 3, 4, 5, and 6 rhyme with each other
Keywords composition: 100%

Stanza 3
Number of lines: 4
Line 1 – number of words: 4; number of syllables: 12
Line 2 – number of words: 6; number of syllables: 16
Line 3 – number of words: 4; number of syllables: 10
Line 4 – number of words: 6; number of syllables: 16
Line 1 and 3 rhyme with each other
Line 2 and 4 rhyme with each other
Keywords composition: 100%

Figure 6. Constraint configuration used for poem in Figure 5

Turing Test

Before conducting the main experiment to see how respondents' evaluated the computer generated poems in terms of various aspects, we first conducted a simple Turing Test-like experiment to to determine how the system is able to imitate human behavior, in this case writing poetry. For this experiment, we selected snippets from four poems created by famous Indonesian poets (such as Chairil Anwar, Sutardji Calzoum Bachri, and WS Rendra), four poems generated by the system with the full constraint configuration, and four poems generated by system with the loose constraint configuration.

For this Turing Test, the system only used poetic words as slot fillers so that the poetry does not specifically discuss a particular topic. These poems were randomized in the questionnaire and respondents were asked to annotate each poem by guessing whether the poem was written by a human or system. Figure 7 shows some poem examples for the Turing Test section.

The questionnaire results for the Turing Test are shown in Table 2. 74% of the respondents correctly identified human-authored poems, but 26% of the human-authored poem judgments were erroneous (i.e. deemed to be machine-authored). As for the poems generated with the full set of constraints, 57% of the judgments were erroneous, i.e. they were deemed to be human-authored, and for the poems generated with the loose constraints, in only 35% of the cases did respondents falsely identify them as human-authored.

Human authored (*Hilang* (Lost), by Sutardji Calzoum Bachri)	
batu kehilangan diam	*A stone loses silence*
jam kehilangan waktu	*A clock loses time*
pisau kehilangan tikam	*A knife loses stab*
mulut kehilangan lagu	*A mouth loses song*
Full constraint	
tak ada lagi pilu dan rindu	*no more pain and yearning*
dari rindu ke mentari	*from yearning to the sun*
ada yang terdiam	*some lay silent*
ada yang menunggu	*some lay in waiting*
Loose constraint	
cinta kau adalah sakit untuk kau	*your love is pain for you*
aku melayang, aku melayang	*I fly, I fly*
cinta kau adalah sakit untuk kau	*your love is pain for you*

Figure 7. Poem examples for Turing Test

Main experiment

For the main experiment, the three constraint configurations were each applied to three different news articles, resulting in 9 different poems being assessed. The poems were randomly obtained from the system output.

In this section of the experiment, we aim to analyze how the poems generated by the system under different configurations were appraised by respondents. The questionnaire randomly presents one of the three chosen news articles along with the three poems produced from that article under the previously discussed constraint configurations. Each poem is the result of concatenating three stanzas that were generated and selected randomly. Respondents were asked to give an assessment of the poems based on the following criteria:

1) **Structure**: a criterion to evaluate the overall structure of the poem, i.e. whether or not it fulfilled the respondent's subjective expectations of what constitutes a poem.

	Human authored	**Full constraints**	**Loose constraints**
Human	74%	57%	35%
Machine	26%	43%	65%

Table 2. Results for Turing Test experiment

2) **Diction**: a criterion to evaluate the choice of words used in the poetry generated.
3) **Grammar**: a criterion to evaluate how well the grammar was in the poem.
4) **Unity**: a criterion to evaluate the unity between the form and content of poetry produced.
5) **Message/theme**: a criterion to evaluate the suitability of the poetry content with the reference article.
6) **Expressiveness**: a criterion to evaluate the level of expression of the resulting poem.

An overview of the data analysis results of the questionnaire can be seen in Figure 8. The blue bar represents 50% keywords-full constraint, the red bar represents 100% keywords-full constraint, and the green bar represents loose constraint.

Every respondent's assessment is transformed to number scale with range of 0-3 then accumulated for the six criteria that have been mentioned previously. From the overview we can see that in general 50% keywords-full constraint and 100% keywords-full constraint parameter give better performance than loose constraint parameter in every criterion.

As can be seen from Figure 8, poems made with 50% keywords-full constraint and 100% keywords-full constraint have a better structure than loose constraint. The structure is evaluated from the number of lines, number of syllables, and rhyme in the poem. We can predict this result as the full constraint configuration is meant to give a strict rule for the system when composing poems that the loose constraint

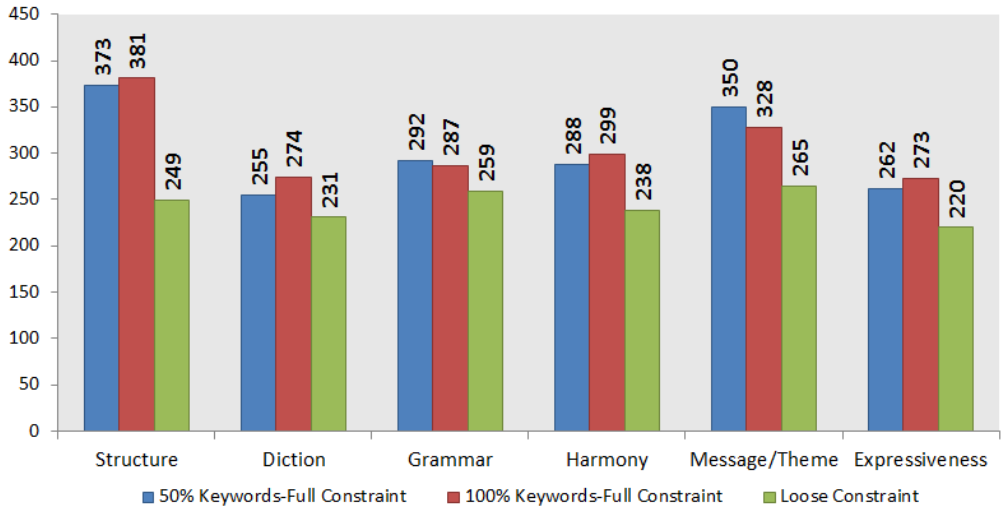

Figure 8. Overview of main experiment results

configuration does not have to obey. This phenomenon was also seen in unity and message aspect. Poems made with 50% keywords-full constraint and 100% keywords-full constraint seem to have a message and stay in specific theme/topic rather than loose constraint. The system is expected to achieve a good performance for discussing a specific theme given the way that the keywords are selected. The keyword ratio constrains the poems to remain on topic while the loose constraint configuration does not. However, it is important to remember that *Pemuisi* is not deliberately conveying a particular semantic message as it is simply constructing lines of poetry by randomly filling slots (given constraints). Thus, we claim that *Pemuisi* composes poems that can be said to be related to the article rather than faithful to the article. Tables 3 and 4 show the detail between topic and message aspect retrieved from the questionnaire response. 50%-FC stands for 50% keywords-full constraint, 100%-FC stands for 100% keywords-full constraint, and LC stands for loose constraint.

	50%-FC		**100%-FC**		**LC**	
	Topic	**Msg**	**Topic**	**Msg**	**Topic**	**Mes**
TA	29%	10%	11%	6%	5%	2%
A	59%	61%	76%	61%	54%	49%
D	10%	25%	11%	31%	34%	42%
TD	1%	4%	2%	3%	7%	8%

TA: Totally Agree; A: Agree; D: Disagree; TD: Totally Disagree

Table 3. The existence of topic and message

	50%-FC		**100%-FC**		**LC**	
	Topic	**Msg**	**Topic**	**Msg**	**Topic**	**Msg**
TA	24%	4%	11%	7%	5%	2%
A	64%	66%	68%	61%	46%	41%
D	12%	28%	20%	31%	42%	49%
TD	1%	2%	1%	1%	7%	8%

TA: Totally Agree; A: Agree; D: Disagree; TD: Totally Disagree

Table 4. The relation of topic and message with the article

The unity between the form and content is better in 50% keywords-full constraint and 100% keywords-full constraint than loose constraint. This aspect shows us about the unity of poem structure and content.

50% keywords-full constraint and 100% keywords-full constraint have a slight lead in expressiveness aspect. This could be due to the composition between poetic words and keywords that is regulated by the keywords ratio. While keeping the poem to stay on topic, we allow the system to also be expressive by using poetic words. Finally, an almost tie result is shown in diction and grammar aspect with 50% keywords-full constraint and 100% keywords-full constraint, with both yielding a slightly better result than loose constraint. We can infer that for every parameter we use the same templates set that already holds for grammaticality property.

Pemuisi: Up-to-date Poem Feed

We have developed a web application as a showcase to publish *Pemuisi* poems at http://budaya.cs.ui.ac.id/pemuisi. The core generation system runs as a background process of the site and is scheduled at noon everyday to crawl various news portals. In order to make *Pemuisi* up-to-date with the world situation, *Pemuisi* will find a recent article published by looking into the news portal RSS feed. The entire preprocessing work is automated.

Pemuisi composes a poem consisting of 3-4 stanzas about the chosen news article. As *Pemuisi* would produce all poem combinations which satisfy the set of given constraint, we demand a fast processing and relevant poem. We provide seven sets of constraints which represent various kinds of Indonesian traditional poem form structure. We also provide 22 templates and 50 poetic words as static language resources. These constraints and language resources can be added anytime later. The *Pemuisi* web application also randomly shuffles the order of all the language resources and set of constraints before generation commences in order to raise the diversity level of the output.

The poem produced by *Pemuisi* is then published to the site page. The first line of the poem is also tweeted by the *Pemuisi* Twitter account (@pemuisi) along with the website page link. In the site page connected with Twitter and Facebook, viewers can comment and share their thoughts about the poem to social media.

Conclusions and Future Work

We have developed an automatic poetry generation system that is capable of automatically generating poems in Indonesian based on specific context restrictions defined by existing constraints and reference news articles.

The system combines the general architecture of the Full-FACE system introduced in Colton et al. (2012), particularly the aspect that generated poems are based on current news articles, with the explicit treatment of the generation process as a constraint satisfaction problem as in Toivanen et al. (2013) (and to a lesser degree, Manurung (2003)), which affords a declarative formulation of the generation process, and the use of efficient off the shelf constraint solvers (although in our current system we use Prolog, we plan to use purpose-built constraint solvers such as ECLiPSe[1]).

The main contribution of this work, aside from this combined approach, and the adaptation to Indonesian, is the user evaluation that was conducted, as both Colton et al. (2012) and Toivanen et al. (2013) present no user evaluation. Lastly, *Pemuisi* is in effect a much more knowledge-poor system than Full-FACE, as there are far fewer lexical resources available for Indonesian as there are for English, in particular the Jigsaw Bard resource that appears to provide

[1] http://eclipseclp.org

a major contribution to the poeticness and coherence to the generated poems.

From the experimental results, it was found that when all the implemented constraints are applied the system is able to produce poetry that is deemed more similar to human-authored poetry rather than the poetry generated under the loosely-constrained configuration. They were also deemed to have better structure, more focus on a topic and conveyed the message from the reference article better.

Many aspects from the system are still rudimentary, and there are still many opportunities to improve the system, such as expanding the types of constraints that can be handled, developing a better interface for the user, and improving the language resources. A careful qualitative evaluation from poets and other poetry experts would be valuable in order to gain feedback about the output of the system. With the developed web application, viewers can leave comments about the generated poem, thus this provides a channel for collecting information for a deep analysis on human perception about the generated poems.

References

Colton, S., J. Goodwin, and T. Veale. 2012. Full-FACE Poetry Generation. In *Proceedings of the 3rd International Conference on Computational Creativity, 2012*. Dublin, Ireland.

Gervás, P. 2002. Exploring quantitative evaluations of the creativity of automatic poets. In *Proceedings of the 2nd. Workshop on Creative Systems, Approaches to Creativity in Artificial Intelligence and Cognitive Science, 15th European Conference on Artificial Intelligence (ECAI).* Lyon, France.

Manurung, H. 2003. *An evolutionary algorithm approach to poetry generation*. PhD. Dissertation, University of Edinburgh, Edinburgh, United Kingdom.

Quasthoff, U., M. Richter, and C. Biemann. 2006. Corpus Portal for Search in Monolingual Corpora. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC 2006)*, Genoa, pp. 1799-1802

Toivanen, J. M., M. Järvisalo, and H. Toivonen. 2013. Harnessing Constraint Programming for Poetry Generation. In *Proceedings of the 4th International Conference on Computational Creativity 2013*. Sydney, Australia.

Pemuisi: a constraint satisfaction-based generator of topical Indonesian poetry Fam Rashel1 and Ruli Manurung2 Faculty of Computer Science Universitas Indonesia Depok 16424, Indonesia 1 , Abstract Pemuisi is a poetry generation system that generates topical poems in Indonesian using a constraint satisfaction approach.Â The results showed that under the best scenario, 57% of the respondents thought that the generated poems were authored by humans, and that poems generated using the full set of constraints consistently measured better on all aspects than those generated using the other two configurations. The system is now available online as a web application. Pemuisi is a poetry generation system that generates topical poems in Indonesian using a constraint satisfaction approach. It scans popular news websites for articles and extracts relevant keywords that are combined with various language resources such as templates and other slot fillers into lines of poetry. It then composes poems from these lines by satisfying a set of given constraints. A Turing Test-style evaluation and a detailed evaluation of three different configurations of the system was conducted through an online questionnaire with 180 respondents. "Pemuisi: a constraint satisfaction-based generator of topical Indonesian poetry" (2014) by Fam Rashel;Ruli Manurung; "Quantifying Creativity in Art Networks" (2015) by Ahmed Elgammal;Babak Saleh; Explore this analogy. "Using Computational Models to Harmonise Melodies" (2010) by Raymond Whorley;Geraint Wiggins;Christophe Rhodes;Marcus Pearce; Explore this analogy. "Blending in the Hub Towards a computational concept invention platform" (2014) by Oliver Kutz;Fabian Neuhaus;Till Mossakowski;Mihai Codescu; Explore this analogy. "Non-Conformant Harmoniz The in-text citation can take two forms: parenthetical and narrative. Both types are generated automatically when citing a source with Scribbrâ€™s APA Citation Generator. Parenthetical citation: According to new research â€¦ (Smith, 2020). Narrative citation: Smith (2020) notes that â€¦ Multiple authors and corporate authors. The in-text citation changes slightly when a source has multiple authors or an organization as an author. Pay attention to punctuation and the use of the ampersand (&) symbol. Author type.