AUTOMATIC SINGABLE TAMIL LYRIC GENERATION FOR A SITUATION AND TUNE BASED ON CAUSAL EFFECT

Rajeswari Sridhar
Department of Computer Science and Engineering
Anna University
Chennai, India
rajisridhar@gmail.com

Subasree Venkatsubhramaniyen
Department of Computer Science and Engineering
Anna University
Chennai, India
vsubasree33@gmail.com

Subha Rashmi S
Department of Computer Science and Engineering
Anna University
Chennai, India
rashmiceg@gmail.com

***Abstract*— *Poems reveal the linguistic capability and creativity of a person. Poets being driven by a creative impulse, frame imaginative poems that narrate a story, and express emotions in an artistic manner. Poets in film industries write poems considering a situation in the movie and the tune composed by the music director. In this paper, we discuss the design and analysis of an automatic lyric generation system for the Tamil language, which takes scene description (in prosaic form) and tune (in swara format) as inputs, and generates lyrics with the necessary poetic components, such as alliteration and simile, which adhere to the situation and the tune. The idea of the input scenario, which drives the lyric generation process is extracted by identifying causal-effect relationships. Also, the dialect of the speaker (in the input scene) will be reflected in the lyrics, if it is specified in the input scene.*

Keywords*— *Poem; scene, tune, Tamil; trigram, morphology, emotion, semantic, causal-effect, syllable, swara

I. INTRODUCTION

With the advent of computers and their ability to make voluminous mathematical calculations quickly and accurately than human beings, the idea of computational creativity came into consideration. The growth of internet and search engines made it possible for computers to generate and manipulate patterns in the field of music, language. Automatic language generation becomes challenging when a poetic version is needed rather than a prosaic one, as poetic features like rhyme scheme, alliteration of the language need to be considered in addition to syntactic correctness.

Here, we propose automatic lyric generation in the Tamil language, based on the context and tune, followed by the extraction of poetic features, viz rhyme, alliteration, and meter. Inputs to the system are verbal description of a scene and musical notes, which play a prominent role in choice of words to the output lyric.

The quality of the poem is directly reflected by the grammatical correctness and meaning conveyed. The poetic lines must be semantically related to the input. The generated lyrics must adhere to the input tune, such that singers will be able to sing the words with the correct pronunciation of the language. The paper is organized as follows: Section 2 throws light on the existing works on lyric generation and the techniques proposed in each of these works. Section 3 discusses the system architecture and implementation in detail, along with the complexity analysis. Section 4 presents the result analysis of the outputs achieved in the system, and Section 5 elaborates on the future work in the domain, and states the conclusions of the work done.

II. LITERATURE SURVEY

A survey on the existing works of Lyric generation was carried out and in this paper we highlight few works on automatic lyric generation. Context-based lyric generation in Tamil as [1], [2] takes an input scene description, followed by epitome extraction and lyric generation. The words in the scene are stemmed and mapped against an ontology to extract the context of the scene in the form of seed words, thereby ensuring semantic relatedness between input scene and final lyric. This paper focuses on statistical model of lyric generation, as N-gram model plays a vital role in framing the lines of the poem. This work restricts the domain and moreover, it could not deliver the right semantics in the lyric, when the input scenario starts with a complete mood of happiness and ends in total sadness. Also, the rhythm (singable characteristic of a poem), is not considered.

The technique proposed in [3] and [4] generates a mapping between melody and words. The input tune (midi file) is converted to the "ABC" format, which is then mapped to KNM (kuril nedil mei) patterns as described in [3]. Generating poem for the obtained KNM patterns limits the choice of words, and so the KNM patterns are converted to Asai (syllables) [4]. The words in the text corpus that correspond to the Asai pattern are generated. As their system is a tune based approach, it had

978-1-5386-5314-2/18/$31.00 ©2018 IEEE

restrictions on the choice of words. The work lacked poetic sentences and cohesion between multiple verses.

The lyric analysis in [5] proposes a data analysis model for words, rhymes and their usage in Tamil lyrics. The Rhyme analysis checks for 'Edugai' (Alliteration), 'Monai' (Rhyme) and 'Eyaibu' (End rhyme) [6] discusses a framework in the form of 4 main subsystems, Music to Template, Lyric Analysis, Lyric Generation, and Lyric Scoring. The input to the system is a MIDI file, that is converted to ABC notes and then mapped to Tamil place holders, called templates. The lyric generation process fills templates with words from a word net, according to the pattern (POS tags) mined from an existing corpus of lyrics, with due consideration to rhyme, meaning and flow.

The corpus-based generation of context and form in poetry [7], uses two different corpora, one to provide semantic content, and the other to generate the grammatical and poetic structure, to generate poetry in Finnish language. A background graph is used to obtain the association between the words in the corpus. The words related to the input keyword are extracted from the background graph. The GRIOT system [8] generates poetry line-by-line, in response to the user input narrative structure. An attempt was made to generate Haiku in English with Word Association Norms (WAN) [9]. Word Association Norms (WAN) are a collection of cue words and the set of free associations that were given as responses to the cues, accompanied by quantitative and statistical measures [9]. This Haiku generation algorithm includes five stages: theme selection, syntactic planning, content selection, filtered over generation and ranking.

Other works on lyric generation in other languages [10] [11] [12] [13] also exist. In our work, we have designed an algorithm for generating lyric, which is a poem that could be sung. Hence, the algorithm should consider the scenario as well as the tune. Based on these requirements, we considered a new approach by referring to some of the existing algorithms, made modifications to few, and proposed new algorithms. This work is discussed in the following section.

III. System Design

A. System Design

The overall system architecture is depicted in figure I. The system focuses on automating the lyric generation process, given an input scene description and tune. The system flow begins at the morphological analyzer, which stems the words and annotates with the Parts of Speech tags. The dialect of the speaker which highly influences the choice of words in the lyric is represented by his/her profession in the first sentence. Then, the sentence-wise emotion is recorded by making use of emotion-bearing keywords. Also, the causal-effect relations are extracted based on the observed patterns in sentences, aiming to understand the epitome of the input scene description. This results in seed words, which contribute to the words in the final lyric. The emotion and domain pertinent to the scene impact the poetic lines in the lyric.

Tamil poetic lines are generated with the aid of a rich knowledge base, which is developed by mining the existing Tamil poems. The lexicon is modeled as *tri*-gram, with words in root form along with suffixes. The suffixes of a word are based on the word following it in the n-gram model. For the words which are verbs in the corpus, the person and number suffix information is provided. To generate a complete poetic line, the morphological generator working at the word level provides the morphed form of the words. For each of the possible beginning words in the poetic lines, appropriate poetic enhancements are made with the help of a corpus. The tune inputted defines the desired sequence of words in the lyric. To avoid generating similar poems for the same input scene, randomness is incorporated. Finally, for the generated poem, features that distinguish a poem from prose, namely, rhyme, alliteration and meter are identified, for further enhancing the generated lyric.

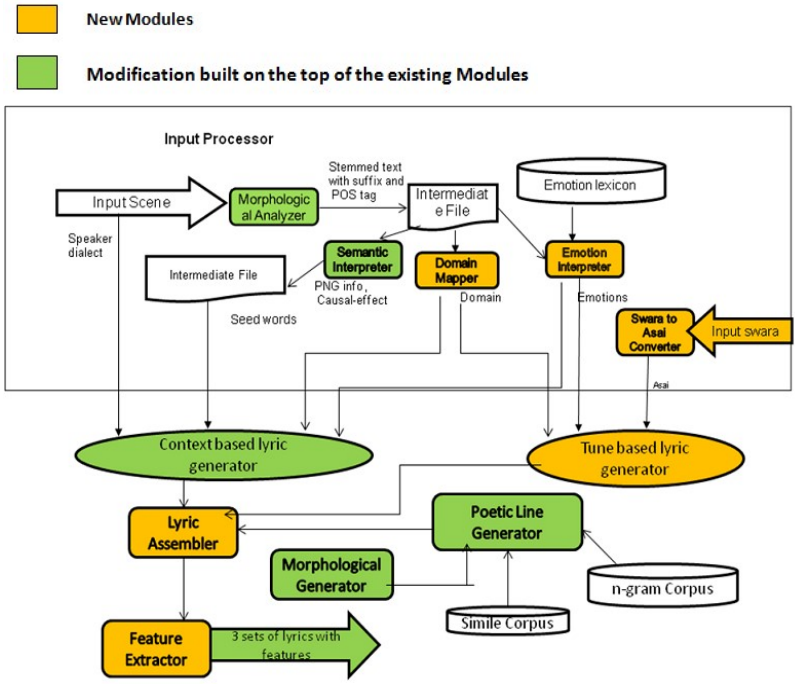

Fig. 1. Architecture of the lyric generation system

B. Module Design

1) Morphological Analyzer

The words in the input scene are analyzed with their root words and suffixes. The stemmed words along with the suffixes are stored to retain the original meaning. Also, the Part Of Speech (POS) tag is attached to every stemmed word. We use the Aksharam tool from the Tamil Computing lab (TACoLa) for this module. We further modified the tool output to incorporate the tag of the original word by observing the suffix. This is done to extract accurate emotions and causal effect relations which would be described in forthcoming sections.

Word = Stemmed word + suffix1 + … + suffix n + Stemmed POS Tag

2) Emotion Interpreter

After the morphological analysis, the sentence wise emotion is recorded by identifying the Noun Adjective Verb Adverb (NAVA) words. Then, the emotion conveyed by these words and by dependencies like adverb-verb and adjective-noun, is identified with the help of an emotion corpus. Additionally, co-occurrence patterns that convey some emotion are taken into consideration to incorporate negation. The words corresponding to these dependencies are checked in the bi-gram corpus. If emotion is not found, the emotion for the adjective and adverb is identified from the unigram corpus. If no such dependency exists, the emotion for the NAVA word (noun, verb, adjective, and adverb) as such is identified from the unigram corpus. We have handled about 5 emotions, namely, fear, anger, surprise, happiness, and sadness.

Algorithm Emotion Interpreter

```
procedure LookupEmotion(S)                          ▷ S-Sentence
    initialize E to nil
    E1 ← SEARCHDEPENDENCY(S,Noun,Adjective)
    E2 ← SEARCHDEPENDENCY(S,Noun,Noun)
    E3 ← SEARCHDEPENDENCY(S,Verb,Adverb)
    add E1,E2,E3 to E
    return E
end procedure
procedure searchDependency(S,T1,T2)    ▷ S- Sentence,T1,T2- POS
    for each Word W2 in S with Tag1 do
        if preceded by Word W1 with Tag2 then
            if W1 W2 found in BigramEmotionCorpus then
                Check for negation
                Add emotion conveyed to E
            else
                if W1 found in UnigramEmotionCorpus then
                    Check for negation
                    Add emotion conveyed to E
                end if
            end if
        else
            if W2 found in UnigramEmotionCorpus then
                Check for negation
                Add emotion conveyed to E
            end if
        end if
    end for
    return E
end procedure
```

The complexity of the algorithm is O(n1)+O(n2).

where, n1=number of pairs in bi-gram corpus

n2=number of words in uni-gram corpus

3) Semantic Interpreter

PNG (person number gender) and tense information are extracted from the analyzed words. Intra-sentence and inter-sentence Causal-effect relationships are identified as described by Sobha Lalitha Devi et al, [14] in order to comprehend the context of the scenario. This is done by extracting patterns from sentences, like Noun-Verb, Noun-Adverb-Verb, Adverb-Verb, Verb, Noun, Adjective-Noun, and Noun-Verb-Verb, with their appropriate position of occurrence in sentences, namely end, middle or both. These patterns were obtained by mining commonly used Tamil sentences; and it is found that the words used for these patterns in the sentences contain causal-effect relations. Additionally, causal markers in Tamil like அதனை (athaNAl), இதனை (ithaNAl),), காரணம் (kAranam meaningreason), காரணமாக (kAranamaaga), காரணத்தால் (kAranatthAl), noun, and verb, noun tagged words followed by suffix ஆல் (Al) are extracted to aid in the causal-effect relation identification. The resulting words from the relations are seed words, which convey the context of the scene.

Algorithm Semantic Interpreter

```
procedure GetSemantics(T)                           ▷ T-Text
    initialize C,R to nil
    for each stemmed sentence S1 in T do
        ⟨C, R⟩ ← cue phrases within sentence
        if head word is a transitory keyword then
            add R of prev sentence in C of current sentence
        end if
    end for
    return C,R
end procedure
```

The complexity of the algorithm is O(m).

where, m=number of sentences in input scene

4) Domain Mapper

Domain of a poem helps a reader in identifying its context. This module attempts to identify the domain of the input scene in order to generate congruous lyrics. The NAVA words elicited from the stemmed input are scanned for domain. We generate lyrics for scenes whose domains are non-conflicting, in scenarios where multiple domain- related keywords are found. We mainly focus on four domains, namely love, friendship, nature and patriotism. This is one of the contributing factors in selecting the right corpus for the lyric generation process. The domain identifier maintains clusters of words for each of the above four domains to facilitate the above said process.

5) Asai Converter

In Tamil grammar, Kuril corresponds to the vowels and consonant-vowel compounds with short sounds, while Nedil corresponds to those with long sounds. A sequence of one or more of these units optionally followed by a consonant can form a ner asai (the Tamil word asai roughly corresponds to syllable) or a nirai asai depending on the duration of pronunciation. Ner and Nirai are the basic units of meter in Tamil prosody.

As discussed in [4], to select a word for a given tune without changing the duration (maathirai), the input tune in the swara pattern is parsed to resolve into the asai format. Asai (syllables) in Tamil is of two types, namely Nerasai and Niraiasai. An intermediate level of output is generated in the form of KNM (kuril nedil mei) components, from the swara sequence. Here, we represent space as a delimiter to separate the swara sequence for words within the line. A hyphen '-'is used as a delimiter for separating the notes that are sung together. Uppercase notation is used for higher octave notes.

Keyboard notes of Tamil film songs are analyzed, to comprehend how lyrics are aligned with the tune. On analyzing, it is found that the notes can be mapped to the stressed and unstressed syllables in Tamil, in the form of short (kuril) and long (nedil) vowel sounds. The number of beats in

music varies according to the time-signature of a song. The first beat in a measure, called the downbeat is almost always the most stressed or heavy beat in the measure. So, we exploit this feature to map the notes to the KNM components. We follow the following legend:

Lower case notes: s-sa, r-ri, g-ga, m-ma, p-pa, d-da, n-ni

Upper-case notes- higher octave of the lower case notes

Comma ‘,’ to denote elongation of preceding note, space ‘ ‘ to separate note sequence of words, dash ‘-‘ for separating notes to be sung independently. We have handled a maximum of three notes like rgg that can be sung together.

The rules we formulated for mapping the swara notes of a word to KNM are as follows:

(i) If the beginning note consists of single note, then map to KM, else to KK if 2 notes (need to be sung together), else to KKM if 3 notes

(ii) Subsequent notes are mapped to K if a single note, KK if 2 notes, N if 2 notes in case they fall at the end of the sequence.

Exceptions are p-p-n (KM-**KM**-K), s-r-g(KM-**KM**-K), n-S-n(KM-**KM**-K), S-S-n(KM-**N**-K), S-S-R(KM-**N**-K)

Eg: M-P-P-P M-P M-G-R, mapped to KMKKK KMK KMKN

An automaton is designed based on Tamil Grammar for converting the KNM components into syllables, as shown in figure 2. Table 1 interprets the automaton to map the KNM sequence to the corresponding ‘asai’.

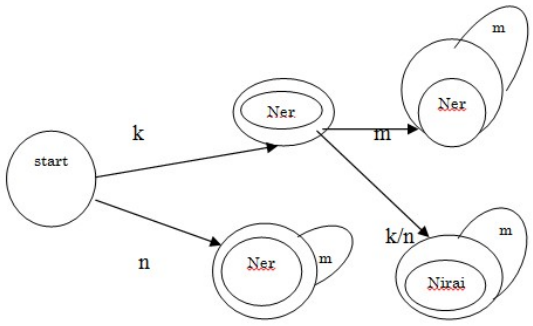

Fig. 2. Automaton for converting KNM components to asai

Vowel Consonant Sequence	Asai
K	Ner
N	Ner
KM	Ner
NM	Ner
KK	Nirai
KN	Nirai
KKM	Nirai
KNM	Nirai

Table 1. Grammar Rules to Convert KNM Components to Asai

6) Poetic Line Generator and Lyric Assembler

After determining the context from the input scene, the words that form part of the lyrics need to be identified. The first word is chosen based on the domain using the word selector module. The subsequent words are generated using a Trigram model designed by us. This trigram model, constructed based on the existing lyrics, is designed to maximize the probability of the occurrence of the word following the previous two words. The corresponding tri-gram model for lyric generation is selected based on the domain and emotion.

The lines generated are ensured to be grammatically correct with the help of a morphological generator. The choice of words acts as a major factor in making the lyrics compliant with the context of the input scene, tune, simile and dialect of the speaker. Emotion also plays a significant role in deciding the context-based words. These form the word selection rules. In the case of tune-based lyric generation, every word has to match its asai pattern. Words are chosen at random, and then are checked against the asai pattern. If there is a match, we proceed to select the next word for the subsequent asai pattern; else we select another word for the same asai pattern.

The lyric assembler module generates a lyric by utilizing the poetic line generator module. The poem is ensured to possess unique lines and different poems are generated each time the same scene is given as input. Moreover, this module provides the appropriate word selection rules, based on the user selection, i.e., tune-based, simple plain lyric or enhanced lyric. Also, this module generates poetic lines based on the dialect of the speaker identified. Further, to make the lyric concur with the input scene, lines are generated that begin with the seed words collected from the semantic interpreter.

7) Morphological Generator

This module is a word-level morphological generator. The POS tag of the word for which the suffix is to be added, helps in identifying the method of morphological generation. Suffix information is provided in the n-gram corpus. The tag is used to choose the appropriate technique of generation. The nature and ending of the root word are then examined, to decide on the paradigm to which the word belongs. The paradigm being identified, the suitable Tamil Grammar Rules are utilized to find out the form of the intermediate suffix, which has to be inserted in making up the morphed word [15] [16].

If the word is a noun, then, we observe only the word ending and suffix beginning, in case the suffix is a case suffix (ஐ (ai), ஆல் (Al), உக்கு (ukku), இன் (in), உடைய (udaiya), இல் (il), இலிருந்து (ilirunthu), இடம் (idam), இடமிருந்து (idamirundhu), ஓடு (Odu), உடன் (udan), கூட (kOda)).

The strategy we follow for nouns as per Tamil Grammar rules [15]. Then, appropriate suffix characters are appended based on the character at the beginning of the suffix and finally, characters of suffix except the first are appended to the word. This makes up the morphed word.

Eg: பெண் + இடம் = பெண்ணிடம் (pen + idam = penNidam)

Regarding the morphological generation for verbs, tense and PNG suffixes are added. We categorize the verbs in an offline process into paradigms like இன்-கிற்-வ், ந்த்-

க்கிற்-ப்ப், double-கிற்-வ், ட்-கிற்-ப், ற்-கிற்-ப், த்த்-க்கிற்-ப்ப், ந்த்-கிற்-வ், ட்-கிற்-வ், ற்-கிற்-வ், ன்-கிற்-வ், nil-கிற்-வ், based on the intermediate suffixes inserted for the three tenses, past, present, and future as given in [4], when these words undergo the morphological changes. Here, double indicates doubling of the final character and nil indicates no suffix. Then, the PNG suffix is added based on the requirement, in accordance with Tamil grammar rules.

Eg: செல் + 2+Singular+Male+Past = சென்றான் (sel + 2+Singular+Male+Past = senrAn)

Algorithm Morphological Generator

```
procedure MORPHGEN(W,S,T)W-Word,S-Suffix,T-POS
    initialize M to null
    if T=Noun then
        M ← obtain using Punarchi rules for noun
    else if T=Verb then
        M ← obtain using Punarchi rules for verb
    end if
    return M
end procedure
```

Algorithm 3. Morphological Generator

IV. Results and Discussions

A. Testing

The lexicon designed by referring to Tamil poems, and Tamil books like Kambaramayanam, EMILLE corpus and NalayiraDivyaPrabhandham comprises of roughly 10,000 unique words. The developed lexicon is categorized, according to the domains mentioned above. The number of lines generated depends based on the emotions identified from the input. The system has been tested for about 200 scenes and 20 tunes.

In addition to Tamil literature, the root words from the EMILLE corpus are scrubbed and cleaned and is also is used to construct the lyric corpus. The developed lexicon is categorized, according to the domains mentioned above, with 2000 words in each domain. The number of lines generated depends based on the emotions identified from the input. Each module in the system is tested separately, and the stage by stage testing of the system gives the following results.

Sample Input and Output:

அவன் ஒரு எழுத்தாளன். அவன் அவளை காதலிக்கிறான். அதனை அவளிடம் சொல்கிறான். அவளின் பதிலை எதிர்பார்த்து காத்திக்கிறான். அவள் அவன் வீட்ட்ற்கு ஒரு திருமணத்திற்கு செல்கிறாள். அங்கு அவள் தன் காதலினை ஆடியும் பாடியும் சம்மதிக்கிறாள். உடனே அவன் மகிழ்ச்சி அடைகிறான்.

Please find below the transliteration of the above Tamil description:

Avan oru ezhuththAlan. Avan avaLai kaathalikkiRAn. Athanai avaLidam solkiRAn. AvaLin bathilai ethirpAththu kAththirukkiRAn. AvaL avan vIttirku oru thirumaNaththirku sekiRAL. Angu avaL than kAthalinai Adiyum pAdiyum sammathikkiRAL. Udanae avan makizhchchi adaikiRAn.

Overall Emotion identified - [happy]

Here, the domain of the input scene is identified as love based on the keyword 'Kadhali'. This drives the system to generate poem based on the identified emotion and domain i.e. love and optimism.

The following is the generated lyric.

thamizh kavi vA
nee oru kAviyam
kanavu ninaivu pudhumai
kAnbavai yaavum kaaviyam
sillendra kAttru kaRaiyAERu
uRavu uyir vaLar

Tune-Based Lyric is generated for the tune input as follows:

Tune input:

S-G GS G-M-MP
S-G GS G-M-MP
M-P-P-P M-P M-G-R,
S-G GS G-M-MP
S-G GS G-M-MP
M-P-P-P M-P M-G-R,

Thamizh kaviyae vA
kAviyam nee aagu
kanavu ninaivu pudhumai
kAnbavai yaavum kAviyam
sillennRa kAttru karaiyaeRukiRadhu
uRavukku uyir vaLaththAEn

Enhanced lyric:

ThAEn thamizh kaviyae vA
kamban kAviyam nee aanAi
vanna kanavum ninaivum pudhumaiyAnadhu
kAnbavai yaavum kAviyam
sillennRa kAttru karaiyaeRukiRadhu
ponnAna uRavukku uyir vaLaththAEn

B. Performance Studies

The lyric efficiency depend entirely on how well the corpus is designed, the correctness of the input tune, and proper description of the input scene. We have compared the context-based lyric generation system as against our tune-based system with context incorporated. The parameters as discussed in one of our previous works [17] and their significance are discussed in the following sections:

1) Assortment of Words (AW)

Any lyric is characterized by the choice of words used in it. The lyric will sound ineffective, unimaginative and hackneyed, if repeated words are found in it. Therefore, unique words used in the poem can serve as a measure, to score a poem. The number of lines generated in the lyric depends upon the input scene, and therefore, is not taken into

account. So, the metric is normalized based on the number of lines. Hence, the ratio of the number of distinct words to the total number of words in the lyric is computed as an assortment of words.

$$\text{Assortment of words} = DW/TW*100 \quad (1)$$

Where, DW= Number of distinct words in lyric and TW= Total number of words.

The distribution of scores for the assortment of words, for both context-based lyric generation and tune-based (with context incorporated) is given in figure 3. The graph shows that the assortment of words is high for both the context and tune-based (context incorporated) lyrics due to the efficient design of the corpus.

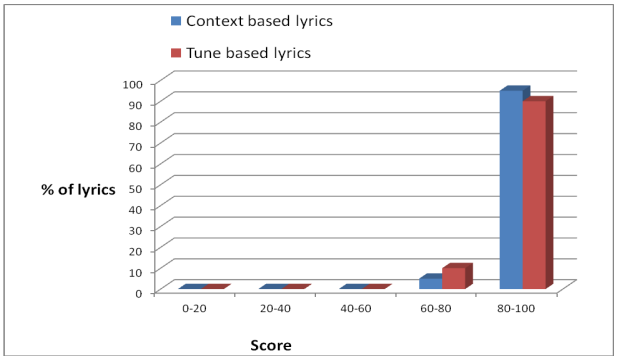

Fig 3. Distribution of score for assortment of words

2) Semantic Relatedness

This measures the extent to which the engendered lyric concurs with the input scene. Some poems may be compliant with the scene in such a manner, that the scene can be predicted by looking at the lyric. This is measured by obtaining the feedback from the end users, who rate the lyric on a 5 point scale (maximum rating indicates good lyrics). This rating is converted to 100. The emotion conveyed in the input scene is ensured to reflect in the lyric, which adds to its semantic relatedness. The distribution of scores for the semantic relatedness is shown in figure 4.

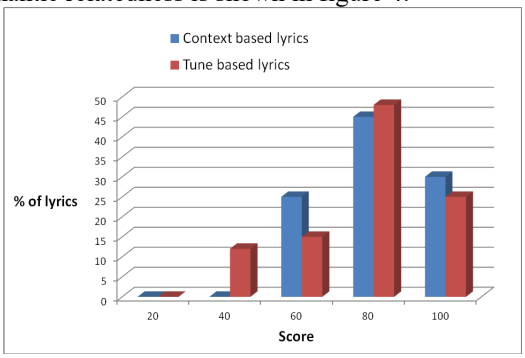

Fig 4. Distribution of score for semantic relatedness

As can be seen from the graph, the semantics have been conveyed in the lyrics due to the causal-effect relations extracted from the processed scene and thus both the context-based and tune-based values are almost same. The tune-based lyrics have incorporated the context, but due to the choice of words beings chosen to adhere to the tune, the semantic relatedness is a little less when compared to the context-based lyrics.

3) Musical Score

Rhythm is an important feature of a poem. Therefore, if the poem is singable to some tune, then it is said to be good. Poetic components like edhugai, monai, and iyaibu, make the poem sound rhythmic. All these components are taken into account to assign a musical score to a poem automatically. For the tune-based lyric, the musical score is high, as it is ensured to be singable in the input tune.

For the tune-based lyric

$$\text{Musical score} = 50 + (RC/TW)*100 \quad (2)$$

where, RC=Number of rhythmic components (edhugai, monai, iyaibu) and TW= total number of rhythmic components possible in the lyric

For other lyrics,

$$\textit{Musical Score} = (RC/TW)*100 \quad (3)$$

The musical score distribution is shown in figure 5. The musical score for tune-based lyrics is high as they are singable in the input tune, but context-based lyrics need not be singable.

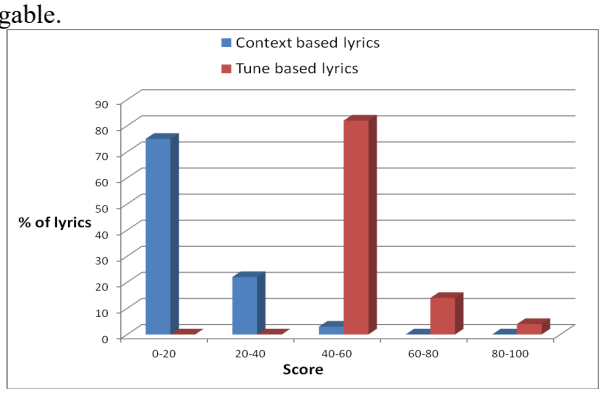

Fig. 5. Distribution of the musical score

4) Logical Coherence

Coherence is a measure of the flow of words in a piece of writing. Coherence between words in a line is ensured based on the n-gram corpus, framed by mining various poems. Inter-sentence coherence is vital to make the user understand the context correctly. This also characterizes the writer's (poet) ability to connect subsequent sentences (verses). This is measured through survey from users. The survey was conducted from the students, and staff of the Department of Computer Science and Engineering, Anna University. The distribution of scores for logical coherence is shown in figure 6.

The logical coherence is high for context and tune-based lyrics, due to the contribution of the extracted seed words in

the lyrics. Figures 7 and 8 depict the comparison between the machine and human scores for the lyrics.

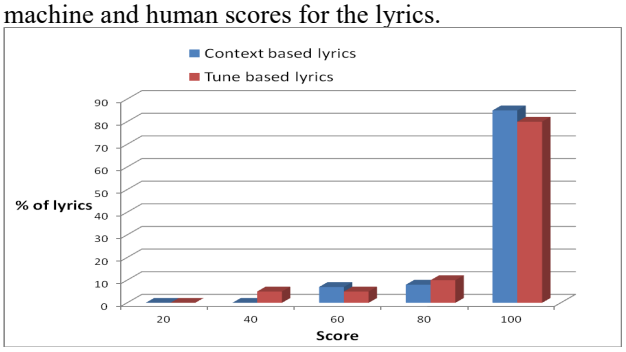

Fig. 6. Distribution of scores for logical coherence

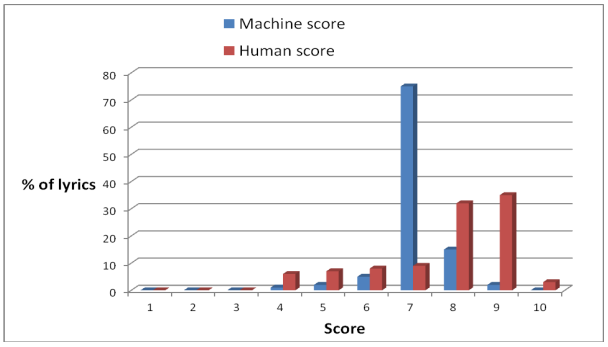

Fig. 7. Human and machine scores for context based lyrics

The human scores for context-based lyrics are relatively higher, as people consider only the domain and emotion when evaluating them. But, machine scores are obtained by considering the various metrics mentioned above.

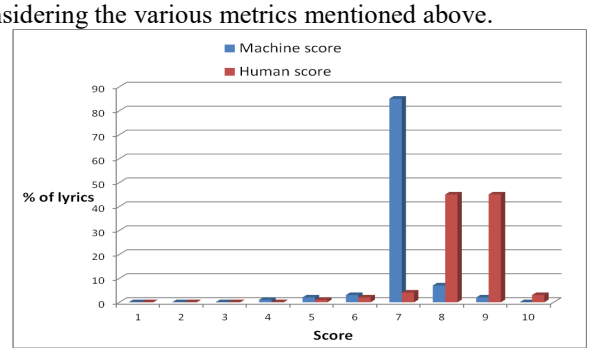

Fig. 8. Human And Machine Score For Tune Based Lyrics

The graph shows high human scores for tune-based lyrics, as people mainly observe the musical characteristics of the lyric, while machine scores are assigned by considering the other metrics also.

V. Conclusion and Future Work

The statistical analysis obtained via feedback from the end users shows, that the developed system can produce lyrics for all scenes in the handled domain that are grammatically correct and contain emotion-bearing words. Thus, the system is successful in composing lyrics for a tune and situation. A lexicon, formulated in the form of n-gram lyric model is built carefully according to the emotion. Enhancements in the lyrics are made with the help of a corpus that contains possible similes, alliterations to the starting words in the generated lyric. The system is scalable to multiple domains, to which the n-gram corpus has to be added.

The present system generates the poem for a scene and a tune, but not in any 'Pa'. Also, the system does not add any rhythmic components to the lyric. Also, emotion, which is identified from the input, is based on keywords. Therefore, a poem will not be generated if there is no emotion-bearing keyword in the input scene. Also, only a tri-gram model is constructed, thus making the system accept tunes of film songs containing only 3 words per line.

The system has listed the meter and related information in the generated poem, but as an extension, a poem can be modified to be written in one of the 'Pa's. This might require simple steps like the rearrangement of lines or addition of vocative words, etc. Also, images and geographic information can act as factors for the poem. Also, some pattern-based poem can be generated like <if you are X then I am Y>, <X beautifies Y>, <X is my wish>, <X is a wonder>, <Suppose I do X, you do Y>. The present system identifies direct emotion keywords. It can be extended to identify indirect emotions from the scene by making the system comprehend the situation. Further, the system produces incorrect lyrics, if the scene contains words from multiple domains. As an add-on, based on the lyric generated, the tune can be suggested, based on the 'osai' predicted.

Acknowledgment

We kindly thank Tamil Computing Lab, DCSE, Anna University Chennai for providing the Aksharam Morphological analyzer tool to carry out our work.

References

[1] Rajeswari Sridhar, Jalin Gladis D, Ganga K., Dhivya Prabha G. 2013. N-Gram Based Approach to Automatic Tamil Lyric Generation by Identifying Emotion. In *Proceedings of International Conference on Advances in Computing, Advances in Intelligent Systems and Computing*. Vol. 174, ISBN: 978-81-322-0739-9, pp: 919 – 926.

[2] Rajeswari Sridhar, Jalin Gladis D, Ganga K., Dhivya Prabha G. 2014. Automatic Tamil Lyric Generation based on ontological interpretation for semantics. In Journal Sadhana – Academy Proceedings in Engineering Science. Vol. 39, Issue 1, pp 97-121.

[3] Ananth Ramakrishnan A, Sankar Kuppan, Sobha Lalitha Devi. June 2009. Automatic Generation of Tamil Lyrics for Melodies. In Proceedings of the NAACL HLT Workshop on Computational Approaches to Linguistic Creativity, pages 40–46, Boulder, Colorado,. C 2009 Association for Computational Linguistics

[4] Ananth Ramakrishnan A, Sobha Lalitha Devi . 2010. An alternate approach towards meaningful lyric generation in Tamil. *In Proceedings of the Workshop on Computational Approaches to Linguistic Creativity* (CALC 2010), Association for Computational Lingusitics (ACL), LA, USA, pp.31-39.

[5] Karthika R, TV Geetha, Ranjani P and Madhan Karky. 2011. Lyric Mining Word, Rhyme & Concept Co-occurrence Analysis. www.infitt.org/ti2011/papers/presentations/02LyricMining.pdf. In Proceedings of Penn, pp 276-281

[6] Sowmiya Dharmalingam and Madhan Karky. 2011. LaaLaLaa – Tamil Lyric Analysis and Generation Framework. In *Proceedings of Penn*, pp 287-292.

[7] Toivanen J M, Toivonen H, Valitutti A, Gross O. 2012. Corpus-Based Generation of Content and Form in Poetry. In *International Conference on Computational Creativity*, pp 175-179.

[8] Harrell D F. 2005. Shades of Computational Evocation and Meaning: The GRIOT System and Improvisational Poetry Generation. *Digital arts and culture*.

[9] Yael Netzer, David Gabay, Yoav Goldberg†, Michael Elhadad. June 2009. Gaiku : Generating Haiku with Word Associations Norms. In *Proceedings of the NAACL HLT Workshop on Computational Approaches to Linguistic Creativity*, pages 32–39, Boulder, Colorado.

[10] H. R. Goncalo Oliveira, F. A. Cardoso, F. C. Pereira. 2007. Tra-la-lyrics: an approach to generate text based on rhythm. In Proceedings of the 4th International Joint Workshop on Computational Creativity, London, UK.

[11] Manurung H M. 2003. An evolutionary algorithm approach to poetry generation. In Institute for Communicating and Collaborative Systems, School of Informatics, University of Edin-burgh.

[12] Jing He*, Ming Zhou, Long Jiang . 2012. Generating Chinese Classical Poems with Statistical Machine Translation Models. In *Association for the Advancement of Artificial Intelligence.*

[13] Simon Colton, Jacob Goodwin, Tony Veale. 2012. Full face poetry generation. In proceedings of International Conference on computational creativity, pages 95-102.

[14] Sobha Lalitha Devi, Menaka S. 2011. Semantic Representation of Causality. Language in India. Special Volume: Problems of Parsing in Indian Languages .

[15] M. Selvam and A.M. Natarajan. 2009. Improvement of Rule Based Morphological Analysis and POS Tagging in Tamil Language via Projection and Induction Techniques. In International Journal of Computers, Issue 4, Volume 3, 2009

[16] Anand Kumar M, Dhanalakshmi V V and Rajendran S. 2010. A Novel Data Driven Algorithm for Tamil Morphological Generator. In *International Journal of Computer Applications* (0975 – 8887)Volume 6– No.12, September 2010.

[17] Sridhar, R., Dharmaraj, N., Damodaran, J., & Sampath, S. (2015, June). Automatic tamil lyric generation based on image sequence and derived tune. In *Advance Computing Conference (IACC), 2015 IEEE International* (pp. 861-866). IEEE.