Poetic Machine: Computational Creativity for Automatic Poetry Generation in Bengali

Amitava Das
Department of Computer Science and Engineering
University of North Texas
Denton, Texas, USA
amitava.santu@gmail.com

Björn Gambäck
Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
gambback@idi.ntnu.no

Abstract
The paper reports an initial study on computational poetry generation for Bengali. Bengali is a morpho-syntactically rich language and partially phonemic. The poetry generation task has been defined as a follow-up rhythmic sequence generation based on user input. The design process involves rhythm understanding from the given input and follow-up rhyme generation by leveraging syllable/phonetic mapping and natural language generation techniques.

A syllabification engine based on grapheme-to-phoneme mapping has been developed in order to understand the given input rhyme. A Support Vector Machine-based classifier then predicts the follow-up syllable/phonetic pattern for the generation and candidate words are chosen automatically, based on the syllable pattern. The final rhythmic poetical follow-up sentence is generated through n-gram matching with weight-based aggregation. The quality of the automatically generated rhymes has been evaluated according to three criteria: poeticness, grammaticality, and meaningfulness.

Introduction
Cognitive abilities can be divided into three broad categories: intelligence, aesthetics, and creativity. Suppose someone has read a sonnet by Shakespeare and is asked the following questions:

- **Do you understand the meaning of this sonnet?**
  If the reader says yes, s/he has used her/his intelligence together with knowledge of the English language and world knowledge to understand it.

- **Do you like this sonnet?**
  Whatever is answer, the reader is using a subjective model of liking — and this is what is called aesthetic appreciation or sentiment.

- **Can you add two more lines to this sonnet?**
  So the reader has to write some poetry — and has to use her/his creative ability to do it.

Artificial Intelligence is a now six-to-seven decades matured research field. The majority of the research efforts until now have concentrated on the understanding of natural phenomena. During the latest two decades, we have witnessed a huge rise of research attention towards affect understanding, that is, the second level of cognition. However, there have so far been pretty few attempts towards making machines truly creative. The paradigm of computational creativity is actually still in infancy, and most of those efforts that have been carried out have concentrated on music or art. Still, computer systems have already made some novel and creative contributions in the fields of mathematical number theory (Colton 2005; Colton, Bundy, and Walsh 2000) and in chess opening theory (Kaufman 2012).

In this paper, in contrast, we look at computational linguistic creativity, and in particular poetry generation. Computational linguistic creativity has only in the last few years received more wide-spread interest by language technology researchers. A book on linguistic creativity was recently written by Veale (2012), and in particular the research group at Helsinki University is very active in this domain (Toivanen et al. 2012; Gross et al. 2012; Toivanen, Toivonen, and Valitutti 2013; Toivanen, Järvisalo, and Toivonen 2013). Some other interesting research attempts have also been made (Levy 2001; Colton, Goodwin, and Veale 2012, e.g.), but the approaches still vary widely.

The field of automatic poetry generation was pioneered by Bailey (1974), although Funkhouser (2009) quotes work going back to the 1950s. These systems were written by actual poets who were keen to explore the potential of using computers in writing poetry and were not fully autonomous. Thereafter, Gervás and his colleagues were the first to discuss sophisticated approaches to automatic poetry generation (Gervás 2000; 2001a; 2001b; 2002a; 2002b; Díaz-Agudo, Gervás, and González-Calero 2002; Gervás et al. 2007). Gervás’ work established the possibility of automatic poetry generation and has in the last decade been followed by a moderate number of attempts at linguistics creativity and in particular at automatic poetry generation.

The system developed by Manurung (2004) uses a grammar-driven formulation to generate metrically constrained poetry out of a given topic. In addition
to the scientific novelty, the work defined the fundamental evaluation criteria of automatic poetry generation: meaningfulness, grammaticality, and poeticness. A complete poetry generation system must generate texts that adhere to all these three properties. An alternative approach to evaluation would be to adopt the criteria specified by Ritchie (2007; 2001) for assessing the novelty and quality of creative systems in general based on their output.

All these previous efforts were inspiration points for the present work, but as we are unable to conclude what method performs best, we decided to propose a new architecture by following the rules and practices of Bengali poems and writings. There is no previous similar work in Bengali, nor on other Indian languages, except attempts at automatic analysis and generation of Sanskrit Vedas (Mishra 2010) and at automatic Tamil lyric generation (Ramakrishnan A, Kuppan, and Devi 2009; Ramakrishnan A and Devi 2010).

The basic strategy adopted here is not to try to make the system create poetry on its own, but rather in collaboration with the user. And not a complete poem, but rather one poetry line at a time. The user enters a line of poetry and the system generates a matching, rhyming line. This task then in turn involves two subtasks: rhyme understanding and rhyme generation. Rhyme understanding entails parsing the input line to understand its poetic structure. Rhyme generation is based on the usage of a Bengali syllabification engine and a Support Vector Machine (SVM) based classifier for predicting the structure of the output sentence and candidate word generation, combined with bigram pruning and weighted aggregation for the selection of the actual words to be used in the generated rhyming line.

The rest of the paper is laid out as follows: To give an understanding of the background, we first discuss the Bengali language as such and the different rhythms and metres that are used in Bengali poems. Thereafter the discussion turns to the chosen methods for poetry line understanding and generation, starting by giving details of a corpus of poems collected for rhyme understanding, and then in turn describing the rhyme understanding and the rhyme generation tasks, and their respective subparts. Finally, an evaluation of the poetry generation model is given, in terms of the three dimensions poeticness, grammaticality, and meaningfulness.

**Bengali and Bengali Poetry**

Bengali (ethnonym: Bangla) is the seventh largest (in terms of speakers) language worldwide. It originates from Sanskrit and belongs to the modern Indo-Aryan language family. Bengali is the second largest language in India and the national language of Bangladesh. Bengali poetry has a vibrant history since the 10th century and the modern Bengali poetry inherited its basic ground from Sanskrit. As the first non-European Nobel Literature Laureate and known mainly for his poems, Rabindranath Tagore (1861–1941) was the pioneer who founded the firm basis of modern Bengali poetry.

**Bengali Orthography and Syllable Patterns**

Bengali, just as all Modern Indo-Aryan languages being derived from Sanskrit, is partially phonemic. That is, its pronunciation style depends not only on orthographic information, but also on Part-of-Speech (POS) information and semantics. Partially phonemic languages use writing systems that are in between strictly phonemic and non-phonemic. Bengali — and many other modern Indo-Aryan languages — still uses Sanskrit orthography, although the sounds and the pronunciation rules have changed to varying degrees.

The modern Bengali script contains the characters (known as akṣara) for seven vowels ([i]/[u]/[e]/[o]/[a]/[a]/[o]), four semi-vowels (⟨j⟩/⟨w⟩/⟨j⟩/⟨o⟩), and thirty consonants. Many diphthongs are possible, although they must always contain one semi-vowel, but only two of the diphthongs are represented directly in the script (i.e., have their own akṣara: aṣ/a and /ou/). All vowels can be nasalized (written as 〈ā/, etc.) and vowel deletion (e.g., schwa deletion) is common, particularly in word medial and final positions.

A phonetic group of Bengali consonants is called a borgo (বোর্গো). As we shall see below, these groups are particularly important in poetic rhymes. There are five basic borgos in Bengali and four separate pronunciation groups, as shown in Table 1, where each consonant is displayed together with its pronunciation in the International Phonetic Alphabet (IPA). Many consonant sounds can be either unaspirated or aspirated (e.g., /ʃ/ or /ʃ/). The first five borgos are named according to their first character. In each borgo, the first consonant takes the least stress when pronounced and the last takes the highest stress. The first member is thus called less-stressed (alpā-prāṇ: অল্প প্রাণ), the second to fourth members are called high-stressed (mahā-prāṇ: মহাপ্রাণ), and the fifth and last is a nasal (nāsik: নাসিক). Following the classification of Sarkar (1986), Bengali has 16 canonical syllable patterns, but CV (consonant-vowel) syllables constitute 54% of the whole language (Dan 1992). Patterns such as CVC, V, VC, CVV, CCV, and CCVC are also reasonably frequent. For more detailed recent overviews of Bengali phonetics, we refer the reader to, for example, Sircar and Nag (2014), Barman (2011) or Kar (2009), and just take the examples below of Bengali orthography — originally devised by Chatterji (1926) — to illustrate how it has deviated from the strictly phonemic orthography of Sanskrit.

- **Consonant clusters** are often pronounced as geminates irrespective of the second consonant. Thus: bAkya /bakko/, bakSha /by/kkho, bismaYa /biffey/.

- **Single grapheme for multiple phonemes:** The vowel [e] is pronounced as either /e/ or /a/. The ambiguity cannot be resolved by the phonological context alone as the etymology is often the underlying reason. For example: eka /æk/, but megha /megh/.
In the svara-vṛtta metre, they are always counted as two units. In a metre, the position of closed syllables does not matter; they are always counted as two units. In the syllable is counted as one unit, but if it occurs at the end of a line it is counted as two units. In the akṣara-vṛtta metre, the number of letters in a line is known as antyaprāsa, which is important rhyme for our purposes. There are three types of rhymes in Sanskrit poetry, depending on whether the rhyme is on the first syllable of each line (adiprāsa), or on the second syllable (dviteeyakshara prāsa), or if it is the final syllable of the line which is rhyming (antyaprasa). The most important rhyme for our purposes is antyaprāsa, which is known as tail-rhyme or end-alliteration in English, and as anto-mil in Bengali poetry.

There are many overviews and in-depth analyses of the metres and rhythms of Bengali poetry written in Bengali, but fairly few available in English. The reader is referred to Arif (2012), or the writings of Au-robindo (2004) that give a more poetic angle. Here, we will concentrate on poems written in mātrā-vṛtta metre with anto-mil rhyme, as these poems are relatively easy to understand and generate.

The Poetry Generation Model

The previous efforts on investigating computer poetic creativity vary widely in terms of the poetry generation approaches. Some have used document corpus-based models (Manurung 2004; Toivanen et al. 2012), while others have used constraint-programming based models (Toivanen, Järvisalo, and Toivonen 2013) or genetic programming based models (Manurung, Ritchie, and Thompson 2012).

In contrast, we choose a conversation follow-up model highly inspired by the Bengali movie ‘Hirak Rajar Deshe’ (‘Kingdom of Diamonds’, 1980) by Oscar winning director Satyajit Ray (the son of Sukumar Ray, the poet whose writings form the basis of our rhyme understanding corpus, as further discussed below).

In Satyajit Ray’s movie, the entire conversation was in rhythm. For example:

एরा যাত বেশি পড়ে (1)

Èrā yata bèśi pařé
they as much more read
‘The more they read’
For the present task, the follow-up model means that the system automatically generates a follow-up rhythm line based on the user’s one-line poetry input. For example, if the given sentence is:

এই দুনিয়ার সকল ভাল

Then the machine could generate a follow-up line such as:

আসল ভাল নকল ভাল

For example, if the given sentence is:

They were meeting all the day with the gathered friends. ‘They were meeting all the day with the gathered friends.

This helps in keeping the rhythm alive.

Rhyme Understanding

The initial step involves understanding the rhyme in an input line given by the user. The actual rhyme understanding module consists of syllable identification followed by borgo identification and open/closed syllable identification. Firstly, however, it is necessary to collect a corpus in order to understand the rhythm and metre structures of Bengali poems.

Corpus Acquisition

To collect the corpus, several dedicated Bengali poem sites (called Kobita in Bengali) were chosen. For the present task, we choose mainly poems written for children, as they mostly are written in mātrā-vṛtta present task, we choose mainly poems written for children.

Corpus size statistics are reported in Table 2.

Table 2: Bengali poem corpus size statistics

<table>
<thead>
<tr>
<th>Type of units</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentences</td>
<td>3567</td>
</tr>
<tr>
<td>Words</td>
<td>9336</td>
</tr>
<tr>
<td>Unique tokens</td>
<td>7245</td>
</tr>
</tbody>
</table>

Syllabification

Syllabification processes depend directly on the pronunciation patterns of any language. In Bengali poetry, open and closed syllables have been used deliberately to continue or stop rhythmic matras (units), as described in the section above on Bengali poetry. These are important features for syllabification.

In order to implement a syllabification engine, we developed a grapheme to phoneme (G2P) converter following the methods discussed by Basu et al. (2009). The consonants and vowels IPA patterns were inherited from that work, while the orthographic and contextual rules were rebuilt. An open-source Bengali shallow parser based POS tagger was used for the task.

With the help of this list, the syllabification engine marks every input word according to its borgo. If a word stars with a vowel, the system marks it as a ‘v’ group. Only the rules mentioned in the paper by Basu et al. have been included, whereas a few things that are not clearly described in the paper remain unattended, for example, some orthographic and exception rules. An example of syllabification output is given in Table 3, where the input is the first line of Sukumar Ray’s poem ‘Cloud Whims’, ‘Mēghēra khējāla’ (মেঘেরা খেজালা).

Borgo Identification

For open syllabic words, identification of the borgo class for the final character is quite important. In case no rhythmic follow-up word is available for the last word in the given sentence, an alternative approach is to choose a word that ends with a consonant belonging to the same borgo. This helps in keeping the rhythm alive.

For example, in the following sequence (also from Sukumar Ray’s poem ‘Cloud Whims’) the first line ends with /t/ (t/) and the final word of the second line ends with a member of the same borgo, namely /t/ (t/).

বুঝা বুঝা ধারি মেঘা ঢিপি হাজু যন্ত্র

The very old inveterate cloud looks like a hill

বুঝে বুঝে সভা কের সারাদিন জুটে

They were meeting all the day with the gathered friends.

http://www.bangla-kobita.com/
Rhyme Generation
The automatic rhyme generation engine consists of several parts. First, an SVM-based classifier predicts syllable sequence patterns. Then, a set of candidate output words are selected from preprocessed syllable-marked word lists. In order to preserve the rhythm in the generated sentence, a few other parameters are checked, such as borgo classes, anto-mil, and whether the syllables are open or closed. Finally, bigrams are used to prune the list of candidate words and weighted sentence aggregation used to generate the actual system output. These steps are described in detail in turn below.

Syllabic Sequence Prediction
A machine-learning classifier was trained for the syllabic rhyme sequence prediction. The Weka-based Support Vector Machine (SVM) implementation (Hall et al. 2009) was chosen as basis for the classifier. The collected poetry corpus described above was used here for training and testing. The training corpus was split into rhythmic pairs of sentences, where the first line would represent the user-provided input whereas the second line would be the one that has to be generated by the system. The input features for the syllabic sequence prediction are: the syllable count sequence of the given line, open/closed syllable pattern sequence of the given line, and the borgo group marking sequence of the first given line. The output labels for the training and testing phases are the syllable counts of each word.

For simplicity only those pairs of sentences were chosen where the number of words are same in both the lines. The overall task has been designed as a sequence syllable count prediction, but there are tricky trade-offs for initial position and the last position. The common rhythmic pattern in Bengali poems is anto-mil (tail-rhyme), so it is necessary to take care of the last word’s syllables separately. Therefore three different ML engines have been trained: One for the initial position, one for the final position, and one for other intermediate positions. Feature engineering has been kept the same for each design, whereas different settings have been adopted for the intermediate positions.

Word Selection
A relatively large word collection was used for the word selection task. The collection consists of the created poem corpus and an additional news corpus. For rhythmic coherence, all words are kept in their inflected forms. In practice, stemming changes the syllable count of any word and may therefore affect the rhythm of the rhythmic sequence.

All word forms are pre-processed and labelled with their syllable counts using the G2P syllabification module. For the word selection, the following strategies have been incorporated serially in the same sequential order as they are described here, in order to narrow down the search space.

Syllable-wise: All words with similar syllabic patterns are extracted from the word list.

Closed Syllable / Open Syllable: Depending on the word in the previous line at the corresponding position, either open or closed syllabic words are chosen. The rest of the words are discarded.

Semantic Relevance: Semantic relevance is very essential to keep the generated rhyme meaningful. There is neither any WordNet publicly available for Bengali nor any relational semantic network like ConceptNet. Therefore the English ConceptNet (Havasi, Speer, and Alonso 2007) and an English-Bengali dictionary (Biswās 2000) were used to measure the semantic relevance of the automatically chosen words.

Before the semantic relevance judgement, each Bengali word from the given input is stemmed using the morphological analyser, packaged with the Bengali shallow parser. After stemming, those words are translated to English by dictionary look-up. The translated English words are then checked in the ConceptNet and all the semantically related words are extracted. Now, if a selected word co-occurs with the given word in the ConceptNet extracted list, then it is considered as relevant. Otherwise it is discarded. For the ConceptNet

---

Table 3: Sample syllabification output

<table>
<thead>
<tr>
<th>Input</th>
<th>আকাশের</th>
<th>ময়দানে</th>
<th>বাতাসের</th>
<th>ভের</th>
</tr>
</thead>
<tbody>
<tr>
<td>akasher</td>
<td>majadane</td>
<td>bataser</td>
<td>vore</td>
<td></td>
</tr>
<tr>
<td>অকাশের</td>
<td>ময়দানে</td>
<td>বাতাসের</td>
<td>ভের</td>
<td></td>
</tr>
<tr>
<td>akāśēra</td>
<td>maja-dānē</td>
<td>bātāsēra</td>
<td>bhārē</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>English</th>
<th>In the sky with the air</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syllables</td>
<td>aka-śē-ra</td>
</tr>
<tr>
<td>Syllable count</td>
<td>3</td>
</tr>
<tr>
<td>Open/Closed</td>
<td>o</td>
</tr>
<tr>
<td>Borgo</td>
<td>v</td>
</tr>
</tbody>
</table>

3http://www.anandabazar.com/
search, only nouns and verbs are considered. For example (same as in Table 3) if the given line is:

\[\text{আকাশের মাদানে বাতাস ভরে}
\]

sky field air filled

‘The sky is filled with the air from the fields’

The words that will be searched in ConceptNet are sky (আকাশ), field (বাতাস), and air (বাতাস). The extracted word list will then definitely contain words such as cloud (মেঘ), which was used by Sukumar Ray in the original poem (again ‘Mēghēra khēāla’ or ‘Cloud Whims’):

\[\text{ছোট বড় সাদা কালো কত মেঘ চের।}
\]

small large white black many clouds grazing

‘Many large and small, black and white clouds are grazing.’

**Borgo-wise**: Borgo-wise similarity is checked and only words ending in the same borgo classes are kept for the last position word. The other words are checked for first letter borgo-similarity, and the non-matching are discarded.

**Anto-mil**: For anto-mil or tail-rhyme matching, an edit distance (Levenshtein 1966) based measure has been adopted. If the Minimum Edit Distance is \( \leq 2 \), then any word is considered as homophonic and kept. This strategy only works for the final word position. The remaining members are excluded.

**Pruning and Grammaticality**

The methods described so far are able to produce word-lists for each word member from the input. Appropriate pruning and natural language techniques are required to generate grammatically correct rhythm sequences from these word options.

N-gram (bigram) matching followed by aggregation is used for the final sentence generation. The n-grams have been generated using the same word collection as described above, that is, the poem corpus plus the news corpus. The system computes weights (frequency/total number of unique n-grams in the corpus) for each pair of n-grams. For example, suppose that the total number of generated word candidates for the first position word is \( n_1 \) and for the second position word it is \( n_2 \). Then \( n_1 \cdot n_2 \) valid comparisons have to be carried out. The possible candidates will be:

\[
\sum_{i=0}^{n_1} w_1^i \cdot \sum_{i=0}^{n_2} w_2^i
\]

Where the sums intend to represent the relevance of using one term after another to create a meaningful word sequence. Suppose the targeted sentence has \( m \) number of words. The process will then be continued for each successive bigram pair, for example, for \( w^1 - w^2, w^2 - w^3, w^3 - w^4, w^4 - w^5, \ldots, w^{m-1} - w^m \)

Finally, the best possible combination is chosen by maximizing the total weighted path as a multiplication function (that is, by maximizing over the dot product of all the possible n-gram sequences). The process is illustrated in Figure 1.

**Experiments and Performance**

The generated system has been evaluated in two ways: through a set of in-depth studies by three dedicated expert evaluators and in more free-form studies by ten randomly selected evaluators.

As discussed in the introduction, three major criteria for the quality assessment of automatic poetry generation have been used previously: poeticness, grammaticality, and meaningfulness (Manurung 2004). The same evaluation measures have been applied to the present task. The evaluation process is manual and each of the three dimensions is assessed on a 3-point scale:

- **Poeticness**:
  - (3) Rhythmic
  - (2) Partially Rhythmic
  - (1) Not Rhythmic

- **Grammaticality**:
  - (3) Grammatically Correct
  - (2) Partially Grammatically Correct
  - (1) Not Correct

- **Meaningfulness**:
  - (3) Meaningful
  - (2) Partially Meaningful
  - (1) Not Meaningful

The evaluation results are reported in Table 4, where the scores assigned by three in-depth evaluators are reported separately, while the randomly selected evaluators have been grouped according to whether they should give short (not more than five words) input lines or whether they could give unrestricted length input. The whole assessment process is elaborated on below, including explanations for the scores given by the different evaluators.
### Table 4: Evaluation of the Bengali poetry generator

<table>
<thead>
<tr>
<th>Evaluators</th>
<th>Dedicated experts</th>
<th>Randomly chosen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#1</td>
<td>#2</td>
</tr>
<tr>
<td>Poeticness</td>
<td>2.4</td>
<td>1.2</td>
</tr>
<tr>
<td>Grammaticality</td>
<td>1.7</td>
<td>1.0</td>
</tr>
<tr>
<td>Meaningfulness</td>
<td>1.5</td>
<td>0.9</td>
</tr>
</tbody>
</table>

### In-Depth Evaluation

Three dedicated expert evaluators were chosen for an in-depth evaluation. One of them is a Bengali literature student, the second a Bengali journalist, and the third a technical undergraduate student. Each of them were asked to test the system performance on 100 input sentences, chosen by themselves.

**Evaluator 1: Literature Student**

The Bengali literature student was instructed to collect 100 simple poem lines from various poets, whose poems were not included in our training set. Through discussion with the evaluator, we decided to choose lines from Satyendranath Dutta’s (1882–1922) poems since he is known for his rhyme sense and renowned as the ‘wizard of rhymes’ (ছেহর যায়কর) in Bengali literature. Also, his creatures are very easy to understand.

We started with the famous ‘The Song of the Palanquin’, ‘Palkir Gan’ (পালকির গান). Following are some examples of the output the system produced. The second lines in the examples were generated by the system, while the first lines were given to the system as input.

<table>
<thead>
<tr>
<th>Palanquin moves!</th>
<th>Trot pace</th>
</tr>
</thead>
<tbody>
<tr>
<td>পালকী চেল !</td>
<td>ডুলকি চালে</td>
</tr>
</tbody>
</table>

The output in Example 11 is surprisingly good. Actually, the same line has been used as follow-up to this input line in one of the paragraphs of the original poem. The output in Example 12 is also good in terms of poeticness, but is less meaningful, while the first output is fabulous for all the evaluation criteria poeticness, meaningfulness and grammaticality. However, we obviously also got many bad output sequences.

**Evaluator 2: Journalist**

The journalist evaluator was requested to judge the system’s performance on news line input and was instructed to chose short sentences. In the evaluation, she assigned a high score to poeticness, but lower scores to grammaticality and meaningfulness. Thus the system performed better than in the news domain, but inferior to the poetry domain. The best output produced by the system is shown in Example 14.

<table>
<thead>
<tr>
<th>‘Who will be the prime minister?’</th>
<th>‘Conspirator for the throne’</th>
</tr>
</thead>
<tbody>
<tr>
<td>কে হবেন প্রধানমন্ত্রী ?</td>
<td>প্রধানমন্ত্রীর গতিময়তার</td>
</tr>
</tbody>
</table>

However, most of the system output in the news domain was unsatisfactory. From discussions with the evaluator, it was eminent that it also is very difficult for humans to generate poetic sequences for any given line, so it is naturally quite difficult for a machine to do this, in particular if the lines are coming from a non-rhythmic news domain.

**Evaluator 3: Technology Student**

The technical undergraduate student was asked to chose lines from modern Bengali songs, and was instructed to chose smaller and simpler sentences. In the evaluation, she assigned a high score to poeticness, but lower scores to grammaticality and meaningfulness. Thus the system performed better than in the news domain, but inferior to the poetry domain. The best output produced by the system is shown in Example 14.

<table>
<thead>
<tr>
<th>‘Dive into the depth of your heart’</th>
<th>‘Rectify yourself’</th>
</tr>
</thead>
<tbody>
<tr>
<td>গভীরে যাও</td>
<td>রুখ ধারে</td>
</tr>
</tbody>
</table>

**Evaluation by Random Evaluators**

Ten randomly selected evaluators (not connected to the research in any way) were asked to evaluate the system’s performance on sentences given by themselves, with the only restriction given that they should provide simple examples with possible tail-rhymes.

The first five of them were instructed to limit their input to five words only. This is in order to understand system performance on longer vs shorter sentences. As a result, we found that system performance is good on all the three aspects on shorter sentences, but that it degrades drastically when longer sentences are given as input. As can be seen in Table 4, this is in particular the case for the dimension of grammaticality, and also true for meaningfulness, while the scores on poeticness are not that bad overall.

---

4http://bartamanpatrika.com/
Conclusion

This paper has reported some initial experiments on automatic generation of Bengali poems. Bengali is a morph-syntactically rich language which has inherited the characteristics and fundamentals of its poems from Sanskrit. Automatic rhyme generation for Bengali is therefore a relatively complex problem. The approach taken here is novel and based on interaction with the user who enters a line of poetry, which the system then aims to understand in order to generate a corresponding text line, adhering to the rules and metres of Bengali poetry and rhyming with the input.

This basic system has many drawbacks and limitations, especially in the understanding of wide varieties of rhythms and in terms of grammaticality. The rhyme generation utilises a Bengali syllabification engine and an SVM-based classifier for predicting the structure of the output sentence and for the candidate word generation, which is based on a notion of semantic relevance in terms of proximity mappings derived from ConceptNet translations. The final selection of the actual poetic words is presently done through bigram pruning and aggregation.

Using the notion of semantic relevance is a computationally cheap way to automatically create meaningful rhymes, although poetry written by humans obviously do not always contain semantically related words. However, this is initial work and using ConceptNet is a straightforward approach; and even though conceptual similarity hardly is the ultimate way to measure word relevance for poems, it is probably one of the easiest ways. In the future, we would aim to involve further natural language generation techniques to create more meaningful poetry.

Acknowledgments

Many thanks go to the evaluators for all their efforts, assistance and comments. We would furthermore like to thank the anonymous reviewers for several comments that helped to substantially improve the paper.

We are grateful to the late Satyajit Ray (1921–1992) for directing the movie ‘Kingdom of Diamonds’ (‘Hirak Rajar Deshe’) which originally inspired our approach.

A very special token of appreciation goes to the three Bengali poets who in the early years of the previous century wrote the verses that were used in the building, training and evaluation of our system: Sukumar Ray, Rabindranath Tagore and Satyendranath Dutta.

References


Gross, O.; Toivonen, H.; Toivanen, J. M.; and Valitutti, A. 2012. Lexical creativity from word associations. In *Seventh International Conference on Knowledge, Information and Creativity Support Systems*, 35–42. IEEE.


Abstract: Poetry Generation involves teaching systems to automatically generate text that resembles poetic work. A deep learning system can learn to generate poetry on its own by training on a corpus of poems and modeling the particular style of language. In this paper, we propose taking an approach that fine-tunes GPT-2, a pre-trained language model, to our downstream task of poetry generation. We extend prior work on poetry generation by introducing creative elements. Specifically, we generate poems that express emotion and elicit the same in readers, and poems that use the language of dream Poetic machine: computational creativity for automatic poetry generation in Bengali. In Proceedings of 5th International Conference on Computational Creativity, ICCC 2014, Ljubljana, Slovenia. Google Scholar. Fellbaum, C. (ed.), 1998. WordNet: An Electronic Lexical Database (Language, Speech, and Communication). Adapting a generic platform for poetry generation to produce Spanish poems. In Proceedings of 5th International Conference on Computational Creativity, ICCC 2014, Ljubljana, Slovenia. Google Scholar. Gonçalo Oliveira, H., and Oliveira Alves, A. 2016. Poetry from concept maps â€” yet another adaptation of PoeTryMeâ€™s flexible architecture. In Proceedings of 7th International Conference on Computational Creativity, ICCC 2016, Paris, France. Google Scholar. The paper reports an initial study on computational poetry generation for Bengali. Bengali is a morpho-syntactically rich language and partially phonemic. The poetry generation task has been defined as a follow-up rhythmic sequence generation based on user input. The design process involves rhythm understanding from the given input and follow-up rhyme generation by leveraging syllable/phonetic mapping and natural language generation techniques. A Support Vector Machine-based classifier then predicts the follow-up syllable/phonetic pattern for the generation and candidate words are chosen automatically, based on the syllable pattern. The final rhythmic poetical follow-up sentence is generated through n-gram matching with weight-based aggregation. Poetic Machine: Computational Creativity for Automatic Poetry Generation in Bengali. A Das, B Gambäck. ICCC, 230-238, 2014. 41. 2014. The system can't perform the operation now. Try again later.