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
gamback@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 linguistics 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/, /æ/, /ɔ/, /a/), four semi-vowels, (/j/, /w/, /e̯/, /o̯/), 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*: /oi/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* (বর্গ). 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., /ʈ/vs /ʈ^h/). 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 (*alpo-prāṇ*: অল্প প্রাণ), the second to forth members are called high-stressed (*mahā-prāṇ*: মহাপ্রাণ), and the fifth and last is a nasal (*nāsik*: নাসিক্য).

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, VV, 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 /bɔ/kkho, bismaYa /biʃʃɔê/.
- **Single grapheme for multiple phonemes:** The vowel [e] is pronounced as either /e/or /æ/. 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/.

Borgo Name	Consonant Members				
ক (k)-*borgo*	ক (k)	খ (k^{h})	গ (g)	ঘ (g^{h})	ঙ (ŋ)
চ (tʃ)-*borgo*	চ (tʃ)	ছ (tʃ^{h})	জ (ʤ)	ঝ (ʤ^{h})	ঞ (n)
ট (ʈ)-*borgo*	ট (ʈ)	ঠ (ʈ^{h})	ড (ɖ)	ঠ (ɖ^{h})	ণ (n)
ত (t̪)-*borgo*	ত (t̪)	থ (t̪^{h})	দ (d̪)	ধ (d̪^{h})	ন (n)
প (p)-*borgo*	প (p)	ফ (p^{h})	ব (b)	ভ (b^{h})	ম (m)
অন্তঃস্থ(internal)-sound	য (ʤ)	য় (e̯)	র (ɾ)	ল (l)	
উষ্ম(warm)-sound	শ (ʃ)	ষ (ʃ)	স (s)	হ (h)	
তাড়নজাত(scolding)-sound	ড় (ɽ)	ঢ় (ɽ)			
পরাশ্রয়ী(parasitic)-sound				ঃ (h)	ঁ (ŋ)

Table 1: Bengali *borgo*-phonetic groups

[a] is pronounced as /o/word medially or word finally in specific contexts: nagara /nɔgor/, bakra /bɔkro/.

- **Vowel harmony or vowel height assimilation:** [a] and [e] are pronounced as /o/ resp. /e/ if followed by a high vowel (/u/ or /i/): patha /pɔth/, but pathika /pothik/; ekaTA /ækʈa/, but ekaTu /ekʈu/.
- **Schwa deletion:** [a] is deleted from word final or medial open syllables under specific conditions dependent on phonotactic constraints and etymology. For example: AmarA /amra/, darbAra /dɔrbar/.

Metres and Rhythms in Bengali

Bengali poetry has three basic and common metres: *akṣara-vṛtta*, *mātrā-vṛtta*, and *svara-vṛtta*. The first two were inherited from Sanskrit, while the third is more genuinely Bengali. However, before Tagore popularized it, the *svara-vṛtta* was used mainly for nursery rhymes and not really recognised as a serious poetic metre.

The *mātrā-vṛtta* and *svara-vṛtta* metres are based on the length of the vowels. The *akṣara-vṛtta* metre is in contrast in Sanskrit based on the number of letters in a line (*akṣara* is the Sanskrit letter); however, in Bengali poetry the number of syllables are counted rather than the number of letters. The letters অ (a), ই (i) and উ (u) are counted as being of one unit (*mātrā*) each, that is, a short vowel (*mora*), while এ (e), ঐ (ai), ও (o), and ঔ (au) are counted as being two units each, that is, a long vowel (*macron*). Furthermore, at the end of a line a short vowel may be counted as a long one.

The concepts of open and closed syllables are also central to Sanskrit prosody and poetry: closed syllables are those ending with a vowel sound, while those ending without vowels are called open. In Bengali, a syllable is considered as being one or two units long depending on its position in a line, rather than on whether it is open or closed. If a line begins with a closed syllable, 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 *mātrā-vṛtta* metre, the position of closed syllables does not matter; they are always counted as two units. In a similar fashion, in the *svara-vṛtta* each vowel (*svara*) is counted as one unit, regardless of whether the syllables are open or closed.

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 (*antyaprāsa*). 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 Aurobindo (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:

এরা যত বেশি পড়ে (1)

Ērā	*yata*	*bēśi*	*paṛē*
they	as much	more	read

'The more they read'

তত বেশি জানে (2)

Tata	*bēśi*	*jānē*
that	more	know

‘The more they learn’

তত কম মানে (3)

Tata	*kama*	*mānē*
that	less	obey

‘The less they obey’

For the present task, the follow-up model means that the system automatically generates a follow-up rhythmic line based on the user’s one-line poetry input.

For example, if the given sentence is:

এই দুনিয়ার সকল ভাল (4)

Ē’i	*duniẏāra*	*sakala*	*bhāla*
this	world	everything	good

‘All is well in the world’

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

আসল ভাল নকল ভাল (5)

Āsala	*bhāla*	*nakala*	*bhāla*
best	good	fake	good

‘Real is good, even fake is also good’

There are two essential modules for effective follow-up poetry generation in Bengali: rhyme structure understanding of the given user input and matching rhyme generation. The development of those modules is discussed in turn in the next two sections.

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)[1] were chosen. For the present task, we choose mainly poems written for children, as they mostly are written in *mātrā-vṛtta* metre and with *anto-mil* (tail) rhyme, which is relatively easy to start with for the task of automatic poetry generation. The poems chosen were mainly written by Sukumar Ray (1889–1923), as the rhyme structure of those poems is fairly easy to grasp. A few of Tagore’s poems, in particular those written for children, were also collected. Corpus size statistics are reported in Table 2.

This corpus was used later on to train a classifier to predict follow-up rhyme syllables. Therefore, from the collected poems only those pairs of lines were extracted that had both *mātrā-vṛtta* metre and *anto-mil* rhythm.

[1]http://www.bangla-kobita.com/

Type of units	**Number**
Sentences	3567
Words	9336
Unique tokens	7245

Table 2: Bengali poem corpus size statistics

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[2] 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 khēẏāla*’ (মেঘের খেয়াল).

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 ঠ(/ʈʰ/) and the final word of the second line ends with a member of the same *borgo*, namely ট (/ʈ/).

বুড়ো বুড়ো ধাড়ি মেঘ ঢিপি হয়ে উঠে (6)

Buṛō	*buṛō*	*dhāṛi*	*mēgha*	*ḍhipi*	*haẏē uṭhē*
old	old	inveterate	cloud	mound	becomes

‘The very old inveterate cloud looks like a hill’

শুয়ে বসে সভা করে সারাদিন জুটে। (7)

Śuẏē	*ba’sē*	*sabhā*	*karē sārādina*	*juṭē*
laid	sitting	meeting	all day	fellows

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

[2]http://ltrc.iiit.ac.in/showfile.php?filename=downloads/shallow_parser.php

Input	আকাশের	ময়দানে	বাতাসের	ভরে
	akasher	*maẏadane*	*bataser*	*vore*
	ākāśēra	*maẏadānē*	*bātāsēra*	*bharē*
English	In the sky with the air			
Syllables	*ākā-śē-ra*	*maẏa-dānē*	*bātā-sē-ra*	*bharē*
Syllable count	3	2	3	1
Open/Closed	o	c	o	c
Borgo	v	p	p	p

Table 3: Sample syllabification output

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.[3] 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 (Biśvās 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

[3]`http://www.anandabazar.com/`

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

আকাশের ময়দানে বাতাসের ভরে (8)

Ākāśēra	*maẏadānē*	*bātāsēra*	*bharē*
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 (ময়দান), 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 khēẏāla*' or 'Cloud Whims'):

ছোট বড় সাদা কালো কত মেঘ চরে। (9)

Chōṭa	*baṛa*	*sādā*	*kālō*	*kata*	*mēgha*	*carē*
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 ≤ 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:

$$\sum_{i=0}^{n_1} w_i^1 \cdot \sum_{i=0}^{n_2} w_i^2 \tag{10}$$

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

$$w^1 - w^2, w^2 - w^3, w^3 - w^4, w^4 - w^5, \ldots, w^{m-1} - w^m$$

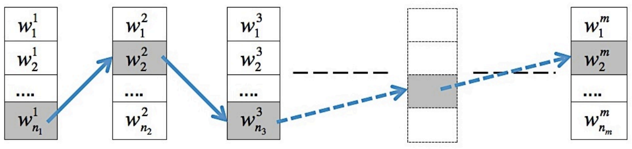

Figure 1: Word sequence selection by n-gram pruning

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.

Evaluators	**Dedicated experts**			**Randomly chosen**	
	#1	#2	#3	≤ 5 words	unrestricted
Poeticness	2.4	1.2	2.1	2.3	1.9
Grammaticality	1.7	1.0	1.4	1.8	0.6
Meaningfulness	1.5	0.9	1.1	1.6	0.8

Table 4: Evaluation of the Bengali poetry generator

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' (ছন্দের যাদুকর) 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.

পালকী চলে ! (11)
'Palanquin moves!'
দুলকি চালে
'Trot pace'

স্তব্ধ গাঁয়ে (12)
'Stunned village'
রুদ্ধ দ্বারে
'Cloggy doors'

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 with a prior assessment of having a possible poetic sequence. He chose lines from the Bartanam newspaper.[4] The best system output was the one in Example 13, where first line again is the input line and the second line has been generated by the system.

কে হবেন প্রধানমন্ত্রী ? (13)
'Who will be the prime minister?'
গদি নেওয়ার ষড়যন্ত্রী
'Conspirator for the throne'

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.

গভীরে যাও (14)
'Dive into the depth of your heart'
শুধরে নাও
'Rectify yourself'

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.

[4] http://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 straight-forward 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

Arif, H. 2012. Prosody. In *Banglapedia: the National Encyclopedia of Bangladesh*. Asiatic Society of Bangladesh, 2 edition.

Aurobindo, S. 2004. *Letters on Poetry and Art*, volume 27 of *The complete works of Sri Aurobindo*. Pondicherry, India: Sri Aurobindo Ashram Publication.

Bailey, R. W. 1974. Computer-assisted poetry: The writing machine is for everybody. In Mitchell, J. L., ed., *Computers in the Humanities*. Edinburgh University Press. 283–295.

Barman, B. 2011. A contrastive analysis of English and Bangla phonemics. *Dhaka University Journal of Linguistics* 2(4):19–42.

Basu, J.; Basu, T.; Mitra, M.; and Mandal, S. 2009. Grapheme to phoneme (G2P) conversion for Bangla. In *Proceedings of the Oriental International Conference on Speech Database and Assessments COCOSDA*, 66–71. IEEE.

Biśvās, Ś. 2000. *Samsad Bengali-English dictionary*. Calcutta, India: Sahitya Samsad, 3 edition.

Chatterji, S. K. 1926. *The Origin and Development of the Bengali Language*. Calcutta University Press.

Colton, S.; Bundy, A.; and Walsh, T. 2000. Automatic invention of integer sequences. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, 558–563. AAAI.

Colton, S.; Goodwin, J.; and Veale, T. 2012. Full-FACE poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, 95–102.

Colton, S. 2005. Automated conjecture making in number theory using HR, Otter and Maple. *Journal of Symbolic Computation* 39(5):593–615.

Dan, M. 1992. *Some issues in metrical phonology of Bangla: The indigenous research tradition*. Phd Thesis, Deccan College, University of Poona, Pune (Poona), India.

Díaz-Agudo, B.; Gervás, P.; and González-Calero, P. A. 2002. Poetry generation in COLIBRI. In *Advances in Case-Based Reasoning*. Springer. 73–87.

Funkhouser, C. 2009. *Prehistoric Digital Poetry: An Archaeology of Forms, 1959–1995*. Modern and Contemporary Poetics. University of Alabama Press.

Gervás, P.; Pérez y Pérez, R.; Sosa, R.; and Lemaitre, C. 2007. On the fly collaborative story-telling: Revising contributions to match a shared partial story line. In *Proceedings of the 4th International Joint Workshop in Computational Creativity*, 13–20. Goldsmiths, University of London.

Gervás, P. 2000. WASP: Evaluation of different strategies for the automatic generation of Spanish verse. In *Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects of AI*, 93–100. AISB.

Gervás, P. 2001a. An expert system for the composition of formal Spanish poetry. *Knowledge-Based Systems* 14(3):181–188.

Gervás, P. 2001b. Generating poetry from a prose text: Creativity versus faithfulness. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*, 93–99. AISB.

Gervás, P. 2002a. 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, the 15th European Conference on Artificial Intelligence.*

Gervás, P. 2002b. Linguistic creativity at different levels of decision in sentence production. In *Proceedings of the AISB 02 Symposium on AI and Creativity in Arts and Science*, 79–88. AISB.

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.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter* 11(1):10–18.

Havasi, C.; Speer, R.; and Alonso, J. B. 2007. ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In *Proceedings of the 6th International Conference on Recent Advances in Natural Language Processing.*

Kar, S. 2009. *The syllable structure of Bangla in Optimality Theory and its application to the analysis of verbal inflectional paradigms in Distributed Morphology.* Phd Thesis, Neuphilologischen Fakultät, Universität Tübingen, Tübingen, Germany.

Kaufman, L. 2012. *The Kaufman Repertoire for Black and White: A Complete, Sound and User-friendly Chess Opening Repertoire.* Alkmaar, The Netherlands: New In Chess.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* 10(8):707–710.

Levy, R. P. 2001. A computational model of poetic creativity with neural network as measure of adaptive fitness. In *Proceedings of the Workshop on Creative Systems, International Conference on Case-Based Reasoning.*

Manurung, R.; Ritchie, G.; and Thompson, H. 2012. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence* 24(1):43–64.

Manurung, H. M. 2004. *An Evolutionary Algorithm Approach to Poety Generation.* Phd Thesis, School of Informatics, University of Edinburgh, Edinburgh, UK.

Mishra, A. 2010. Modelling Aṣṭādhyāyī: An approach based on the methodology of ancillary disciplines (*Vedāṅga*). In *Sanskrit Computational Linguistics.* Springer. 239–258.

Ramakrishnan A, A., and Devi, S. L. 2010. An alternate approach towards meaningful lyric generation in Tamil. In *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity*, 31–39. ACL.

Ramakrishnan A, A.; Kuppan, S.; and Devi, S. L. 2009. Automatic generation of Tamil lyrics for melodies. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, 40–46. ACL.

Ritchie, G. D. 2001. Assessing creativity. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*, 3–11. AISB.

Ritchie, G. D. 2007. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines* 17(1):67–99.

Sarkar, P. 1986. Aspects of Bengali syllables. In *National Seminar on the Syllable in Phonetics and Phonology.* Hyderabad, India: Osmania University.

Sircar, S., and Nag, S. 2014. Akshara–syllable mappings in Bengali: a language-specific skill for reading. In Winskel, H., and Padakannaya, P., eds., *South and Southeast Asian psycholinguistics.* Cambridge University Press. 202–211.

Toivanen, J. M.; Toivonen, H.; Valitutti, A.; and Gross, O. 2012. Corpus-based generation of content and form in poetry. In *Proceedings of the Third International Conference on Computational Creativity*, 175–179.

Toivanen, J. M.; Järvisalo, M.; and Toivonen, H. 2013. Harnessing constraint programming for poetry composition. In *Proceedings of the Fourth International Conference on Computational Creativity*, 160–167.

Toivanen, J. M.; Toivonen, H.; and Valitutti, A. 2013. Automatical composition of lyrical songs. In *Proceedings of the Fourth International Conference on Computational Creativity*, 87–91.

Veale, T. 2012. *Exploding the Creativity Myth: The computational foundations of linguistic creativity.* Bloomsbury Academic.