Sentimental Poetry Generation

Kasper Aalberg Røstvold and **Björn Gambäck**
Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
kasperaarr@gmail.com, gamback@ntnu.no

Abstract

The paper investigates how well poetry can be generated to contain a specific sentiment, and whether readers of the poetry experience the intended sentiment. The poetry generator consists of a bi-directional Long Short-Term Memory (LSTM) model, combined with rhyme pair generation, rule-based word prediction methods, and tree search for extending generation possibilities. The LSTM network was trained on a set of English poetry written and published by users on a public website. Human judges evaluated poems generated by the system, both with a positive and negative sentiment. The results indicate that while there are some weaknesses in the system compared to other state-of-the-art solutions, it is fully capable of generating poetry with an inherent sentiment that is perceived by readers.

1 Introduction

Poetry generation is a type of linguistic creativity that requires certain qualities in both form and content, as well as the creation of understandable, meaningful and poetic language. A central part of poetry is the experience of the reader, including the emotions poetry can evoke. The overarching goal of this work is to explore and develop methods for generating poetry with a specific (pre-defined) inherent sentiment, which can be experienced by the readers. Earlier approaches to poetry generation followed a range of paths, such as methods based on templates (Gonçalo Oliveira, 2012) or corpora (Colton et al., 2012), evolutionary (Levy, 2001) or Case-Based Reasoning (Gervás, 2001) approaches, and generate-and-test (Gervás, 2000) or Blackboard (Misztal and Indurkhya, 2014) architectures. However, in recent years deep learners have proven powerful as poetry generators, including systems combining neural models with other techniques. As described below, Long Short-Term Memory (LSTM, Hochreiter and Schmidhuber, 1997) networks are the most used solutions in state-of-the-art systems, so an LSTM was implemented here, with an experimental focus on which specific network architecture and parameter settings would produce the best poetry word prediction model. The sentiment of the poems to be generated was decided in advance, with human judges rating their quality and how well the sentiment was perceived.

2 Related Work

Zhang and Lapata (2014) used a Recurrent Neural Network (RNN) to generate Chinese quatrains (stanzas with four lines), with the first line based on user-provided keywords giving the main concepts of the poem. Subsequent lines were generated based on previous lines, subject to admissible tonal pattern and structural constraints. Yi et al. (2016) also generated Chinese quatrains line-by-line based on user keywords, but using a sequence-to-sequence model with attention mechanism (Bahdanau et al., 2014), with a bi-directional RNN with gated recurrent units (GRU; Cho et al., 2014) as encoder-decoder to learn semantic relevance. Wang et al. (2016a) used a similar approach for character-by-character iambics generation, utilising a bi-LSTM as encoder and another LSTM as decoder to alleviate the quick-forgetting problem associated with conventional RNNs.

Ghazvininejad et al. (2017) combined hard format constraints with an RNN to generate 14-line classical sonnets in iambic pentameter, given a user-supplied topic and a set of related words, using *word2vec* (Mikolov et al., 2013). Rhyme words were found using CMU Pronouncing Dictionary (CMUdict),[1] with fixed pairs of often used words added to make the system find rhymes in rare topics. A Finite-state acceptor was built with paths for all

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Proceedings of the 17th International Conference on Natural Language Processing, pages 246–256
Patna, India, December 18 - 21, 2020.

conceivable vocabulary word sequences obeying formal rhythm constraints, and an RNN selected the best path. Similarly, Benhart et al. (2018) combined an RNN with sonnet format constraints, but added part-of-speech restrictions to alleviate erroneous word choices, and dynamically trainable word embeddings, so that the language model was able to learn some grammar before adjusting its word representations to suit the training corpus.

Wang et al. (2016b) implemented a 2-phase system: planning and generation, with an encoder-decoder generator using bi-GRUs. A special planning schema in advance selected sub-keywords (based on user input) guided by a language model and each line then generated with the planned word to improve coherence. Zhang et al. (2017) used the same attention-based RNN generation model, but with an additional memory component, effectively giving regularisation that constrains and modifies the behaviour of the model. Van de Cruys (2020) also utilised an encoder-decoder, but generated poems in both English and French. Human evaluators scored the output highly with regard to fluency, coherence, meaningfulness and poeticness, even though only non-poetic text was used as training data for the generator.

Unlike the one-pass generation for previous neural networks models, Yan (2016) proposed a generative model with a polishing schema, refining the RNN-generated poem through several iterations. Also, while previous models were based on maximum likelihood estimation (MLE), which optimizes word-level loss and can lead to the systems remembering common patterns of the training corpus, Yi et al. (2018) added reinforcement learning to a basic generator pre-trained with MLE, simultaneously training two generators that learn both from the teacher (rewarder) and from each other. Automatic rewarders were designed corresponding to four criteria: fluency, coherence, meaningfulness, and overall quality.

Tikhonov and Yamshchikov (2018b) aimed to generate poetry in the style of a specific author, using an LSTM to predict the next word based on a previous word sequence, with the embeddings of the document currently being analysed used to support the model at every step. Two datasets were used to train the model, in English and Russian.

Several of the systems presented above implement a form of user input to influence the mood of the poetry, often related to a given sentiment. However, two such systems are particularly important in the way they include the use of sentiment: In the corpus-based approach of Full-FACE Poetry Generation (Colton et al., 2012), the system decides on a mood by checking the average sentiment of a set of newspaper articles posted during the previous 24 hours, and then selects one of the five articles with the highest resp. lowest sentiment value. Misztal and Indurkhya's (2014) system includes an emotional personality aspect implementing both sentiment analysis and emotional modelling. To extract sentiment, the Sentistrength (Thelwall et al., 2010) tool is used, rating positive and negative scores on a *valence* value scale. An average *arousal* value of the input is calculated using Affective Norms for English Words (ANEW, Bradley and Lang, 1999), and the poem's emotional state is set by combining valence and arousal. WordNet-Affect (Strapparava and Valitutti, 2004) is used to build a hierarchy of words describing emotional states in order to generate the affective content of poems.

3 Data set

The data set used in the experiments here is the English part of the data collected by Tikhonov and Yamshchikov (2018a,b). It consists of poems written and published by users on a public website, which leads to a variance in the quality of content, but both the large size and variance in content are positive factors for network training. The provided data set was already cleaned, with all types of punctuation removed and all letters converted to lowercase. However, there were some inconsistencies in how contractions were represented, with some appearing in a joined form (e.g., *wouldve*) and others as separate words (*would ve*). Hence all spaces between regular contractions were removed, as were line break markers (`<br>`), with every individual poem was instead represented by single individual lines, so that the structure of the generated poems would not be constrained.

The original data set contains $3,943,982$ poems and $155,066,504$ tokens, with a vocabulary of $708,727$ unique tokens. Most of the unique tokens come from misspellings, alternative spellings, irregular words and names. The training data was shortened to specifically reduce the vocabulary size, in order to remove words that very rarely appear in the data set, and to reduce the size of the matrix representation of data used in training the network. To reduce the data set, tests were run on vocabu-

	Lemmas	Poems	Tokens	Polar
Data set 1	10 000	205 230	15 847 356	1 849
Data set 2	15 000	306 942	25 883 608	2 424
Data set 3	20 000	395 057	35 178 076	2 900
Data set 4	30 000	530 121	50 505 342	3 519

Table 1: Data set sizes based on vocabulary

	Lemmas	Poems	Tokens
Training set 1	9 195	12 273	1 300 068
Training set 2	18 031	26 873	3 106 347
Test set 1	9 195	4 620	405 286
Test set 2	18 031	4 307	406 324

Table 2: Training and test data sets

lary sizes of $10,000 - 30,000$, finding how many of the poems only included words within a given vocabulary, as can be seen in Table 1. Since larger vocabulary results in that more possible words can be generated, but negatively effects training time and is reliant on how much the hardware used for training can handle, a simple test was run on a GeForce GTX 1070 graphics card. Creating a bidirectional LSTM using Keras[2] with a single hidden layer of $1,024$ neurons, and training on a random sample size of the $5,000,000$ tokens, with different vocabulary sizes, resulted in the GPU experiencing memory problems when exceeding a vocabulary size of $30,000$, so the vocabulary size was capped at that value, while the lower limit was set to $10,000$, as smaller vocabularies would result in too few words available for generation.

An important aspect of the vocabulary is the inclusion of words with sentiment values, since they would be generated to add sentiment to the poetry. Using Vader (Hutto and Gilbert, 2015), the data sets were investigated for how many unique words they contained with a polar (non-neutral) sentiment, i.e., either having a positive or a negative sentiment. Those are also reported in Table 1.

Due to the increased number of unique sentiment words with increased vocabulary sizes, but also due to memory restrictions and larger vocabularies resulting in greater training times, the vocabulary sizes for neural network training was chosen to be $10,000$ and $20,000$. As Table 1 shows, the number of individual tokens in the data set with a vocabulary size of $20,000$ is over 35 million, which is too large to train on, concerning the time it would take. The data sets were therefore further reduced to *training data sets*, containing only 10% of the original data. All poems were ordered after the user name of the person that published it, so every tenth poem was selected for creating training data, to get poems from as many different authors as possible. The training data set sizes are presented in Table 2.

To evaluate the trained networks, new *test* data was created from the original data sets, in the same way as the training data, but from poems not included in the training data and only containing tokens included in the training data vocabularies. The test data sets are also shown in Table 2.

The training of the neural network is done by creating input sequences to be fed through the network, but also creating the correct output which is then compared to the output of the network. The input is therefore created by choosing a sequence of the training data with a given length, and the token following that sequence as the correct output. A training data sequence length of 5 was chosen, both for memory storage reasons and since 5 was decided to be the shortest possible line length of the generated poetry. Resticting the sequence length will also make the network predict the next words based on just a short sequence, instead of all of the poem that is already generated, which could lead to more variation. Before creating the network input matrices, every poem was reversed, with the last word being the first, and so on, following the approach used by Benhart et al. (2018). This is done so it is possible to start with the ending rhyme word of a line of poetry, and generate the rest of the line backwards from that rhyme-word.

4 Architecture

This section introduces the architecture for the system implemented in this project. The first part describes the long short-term memory network that was implemented, while the second part describes the complete poetry generation process.

4.1 The LSTM network

A bi-directional Long Short-Term Memory neural network is the main component in the poetry generation process. After being trained on a large data set of human-written poetry, its task is to give a prediction on the next word that should follow after a given input sequence of words. The prediction consists of an array, with a predicted score of every unique word in the vocabulary.

Figure 1 shows the Bi-LSTM network. The input with a sequence length of 5 is transformed into

[2]https://keras.io/

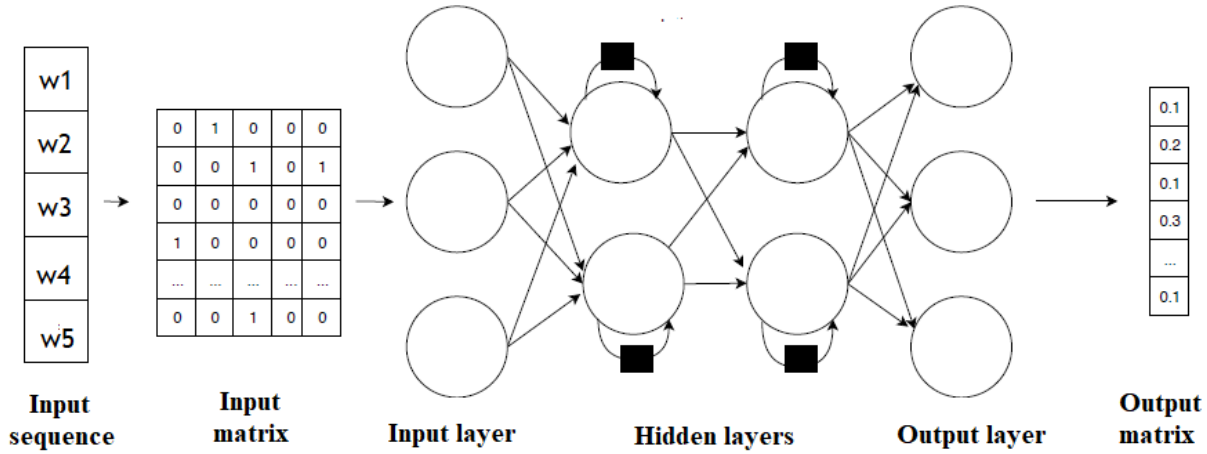

Figure 1: LSTM network prediction process

a matrix representation before being fed to the network. The network's input layer represents the matrix data given to the network, which accounts for both the data used for training and for the input during the poetry generation process. The output layer is a softmax activation function layer that outputs a matrix representing the network predictions for all possible words. During training, the result of these predictions are compared to the actual following word of a given sequence, which will update the hidden layer(s) based on a loss function, which the network tries to minimise. Between the input and output layers are hidden layers, consisting of (recurrent) LSTM cells that compute the possible values for the next predictions, based on the current input and the (stored memory of) the previous part of the input sequence.

During training, the network tries to minimise the cross entropy loss:

$$L = -\frac{1}{N}\sum_{c=1}^{N} ln(p_c) \quad (1)$$

where N is the total training set, p the prediction, and c the category (the word) being looked at. The loss is the cross entropy between the distribution of the true labels and the network predictions. To minimise the loss function, backpropagation is used to update the network weights during training, by taking the error found by the loss function L and calculating the gradient of L with respect to the weights, w, in the network, $\frac{\partial L}{\partial w}$. The gradient is fed to the optimiser, which updates the weights in an attempt to minimise the loss function. The optimiser used is stochastic gradient descent, a first-order iterative optimisation algorithm. It is possible to get stuck in a local minimum when minimising the loss function, therefore the learning rate is initially set higher, and decreases during the training to try to find the global minimum. In addition, dropout (Srivastava et al., 2014) was used during network training to reduce overfitting.

4.2 Poetry generation system

The network output consist of an array containing the predicted value for each word in the vocabulary, to follow the input word sequence fed to the network. In addition to the LSTM, the poetry generator has three important components: rhyme pair generation, prediction score updating, and tree search. The generation of rhyme pairs is used as initial input for generating each poetry line and ensures that the poetry contains end rhymes. The score updating algorithm adjusts the prediction values gained from the LSTM, by adding rules and different weightings on certain types of possible words, to enhance the quality of the generated sequences. The search tree expands the number of possible sequences created during the generation, increasing the chance of finding the best possible sequence to create a poem from.

Rhyme word generation: The first part of the generation process consists of finding rhyme word pairs, that are used as input for producing a poetry line, as it is generated backwards from the rhyme words. For this, a unique word having a sentiment value matching the decided sentiment is randomly chosen from the vocabulary, using Vader (Hutto and Gilbert, 2015), with words with a sentiment value above 0.0 selected if the sentiment is to be positive, and less than 0.0 for negative.

The vocabulary is then searched for another word of the same sentiment rhyming with the first

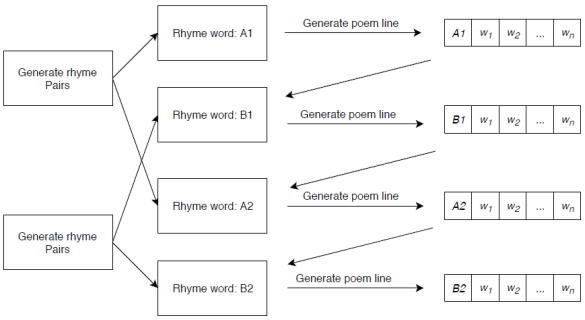

Figure 2: Poetry generation with rhyme word input

word, with CMUdict identifying the syllables. The conditions that need to be met to complete a rhyme do not form a *perfect* rhyme (i.e., with identical stressed vowel sounds in both words, as well as any subsequent sounds, but with different consonant preceding the stressed vowel), but rather a form of *imperfect* rhyme, where the last three sounds, including a consonant, are equal for two words. Each word in a rhyme pair is used for one line of poetry, and the rhyme form chosen is *ABAB*, so two pairs are needed to generate one 4-line stanza.

The first rhyme word found is used to predict the following words for the first line (which will be the final line of the poem, since it is generated backwards). The last four words of the generated sequence, plus the rhyme word for the next line, are used as input to generate the next line. This process is presented in Figure 2, where the rhyme word A1 is used to generate the first line, consisting of the rhyme word and a sequence of n words w. The next rhyme word B1 is then added to the sequence of the four words from w_{n-3} to w_n from the last sequence, and used as input to generate the next line. The last generated line with the rhyme word B2 is the first line of the complete poem.

Updating prediction scores: For every word generated in a sequence, predictions on all the possible words are given by the LSTM, based on network input consisting of the previously generated words. Four algorithms were implemented for updating and adjusting the prediction scores, to improve the generated poetry. These algorithms were similarly used in other state-of-the-art-solutions, including Benhart et al. (2018) implementing repetition and sentence structure restrictions, and Colton et al. (2012) measuring sentiment values of poetry lines against a set value.

(i) Since some popular words often appear in the data set, they will have a high prediction value. To avoid a repetitive use of these words and to achieve better variety, **related words** are found using WordNet (Miller, 1995) for the 20 unique words with the highest prediction value, and the predictions for those words are increased, thus increasing the probability of choosing a less used word.

(ii) To reduce the likelihood of a line of poetry containing **repeated words**, a word's predicted score is reduced during sequence generation, if it has previously appeared in the sequence.

(iii) It was observed that the generated poetry consistently had obvious part-of-speech (POS) errors. Sequences were thus POS-tagged using the Natural Language ToolKit (Bird et al., 2009), and used to implement **sentence structure** restrictions, such as disallowing a pronoun directly preceding another pronoun and a verb directly preceding another verb.

(iv) Based on the intended **sentiment** for the poems, the scores for the possible words with a corresponding sentiment were increased, using Vader to find all the possible next words in a sequence having a sentiment value corresponding to the intended sentiment and increasing their predicted score, resulting in an increased chance of choosing words with the correct sentiment value when generating sequences. The degree of increasing predicted scores is based of the sentiment value of each word. Words that Vader evaluates as having a higher correct sentiment value get a larger predicted score increase; words with the opposite sentiment get their predicted scores decreased.

Search tree algorithm: To increase the chances of finding the best possible sequences to create a poem from, a tree search algorithm was implemented to expand the number of sequences created. Instead of generating a single next word based on the highest score, the search takes a number of the possible words with the highest score values, and generates different possible sequences. This is repeated for every new word in the sequence.

The first word w_1 can be one of the rhyme words used to generate the rest of the sequence, but it can also be a sequence of previously generated words plus the new rhyme word. Using this as the input to the LSTM network, we get the predictions for the next words. These prediction scores are then updated by the four algorithms described above, before 20 new sequences are created, that all consist of the first word w_1 plus one of the next words

x with one of the highest scores, where every sequence has a unique next word x. This generation step is then performed on all the possible sequences created in the previous step, for adding the third word in the sequence, and so on.

The search continues with a number of steps i equal to the total number of words to be generated in a given sequence. As the search tree grows exponentially, the possible sequences that are created with the lowest *scores* are continuously pruned throughout the process.

When all the possible sequences have been created, the sentiment values for each complete sequence are evaluated using Vader. These sentiment values are again used to adjust the score for all the possible sequences. After this final score update, the sequence with the highest score is chosen, and used to generate a new line of the current poem.

The process described above is used to generate 8-line poems consisting of two 4-line stanzas with a rhyming scheme of *ABAB CDCD*. The system adds commas after each of the first three lines of a stanza, a period after the last line, and an empty line between the two stanzas. The first letter of the first word of every line is capitalised, in addition to other letters where capitalising is grammatically correct. Finally, Vader is used to calculate the sentiment value of the entire complete poem. This final sentiment value is used in the experiments where it is compared to human evaluations.

5 Network Experiments

Experiment were conducted to decide the final architecture of the LSTM network, and what parameters to use in the implementation of the poetry generation system. The network training was performed on a Tesla P100-PCIE-16GB GPU. The learning rate was initially set to 0.9 for the training of all networks. A monitor was implemented on the validation loss calculated on validation samples each epoch, reducing the learning rate after a given period when the validation loss has not decreased. The period before reducing the learning rate was set to 5 epochs, and the reduction of the learning rate to a factor of 0.3. The minimum limit for the learning rate was set to 0.001. The dropout probability rate was 0.6 for all non-recurrent units. This value was chosen based on Zaremba et al. (2014), where the dropout rate used for medium LSTM (650 units per layer) was 0.5, and 0.65 for large LSTM (1500 units per layer).

	M1	M2	M3	M4	M5
CA	2.25	2.38	2.46	2.41	2.33
L	8.997	8.097	7.951	8.131	8.813
WP	8,077	3,286	2,839	3,400	6,722

Table 3: Results using Training Data set 1

Two different data sets were used in the experiments and the parameters tested were the numbers of hidden layers, units in each hidden layer, and training epochs. The test data input was fed through the networks, and predictions were measured against true values, using three measures:

(i) *Categorical accuracy* (CA) is calculated for the entire test data set, by taking the mean accuracy rate across all the predictions, to check if the predicted word is equal to the true word.

(ii) *Categorical cross entropy loss* gives the loss function (L; Eq. 1) used during network training.

(iii) *Word perplexity* (WP) measures how well a probability distribution can predict a sample: the lower the perplexity, the less confused a network is about predicting the next word. It is calculated by using the loss function as exponent to the power of the constant e:

$$\text{WP} = e^{L} = exp(-\frac{1}{N}\sum_{c=1}^{N} ln(p_c)) \qquad (2)$$

5.1 LSTM network training

Five LSTM networks were trained over 25, 50 or 75 training epochs on each of the training data sets: two network models with two hidden layers, with 256 (denoted as model M1 below) resp. 1024 (M2) hidden units; and three models with three hidden layers, with the number of hidden units being 256 (M3), 512 (M4) or 1024 (M5).

The LSTM models trained on Training set 1 (with a vocabulary size of $9,195$) were only evaluated on the Test set 1, since it has the same size. Table 3 displays the categorical accuracy, cross entropy loss, and perplexity for those networks.

The LSTM network models trained on the Training set 2 on the other hand (with a vocabulary size of $18,031$), were evaluated on both Test set 1 and Test set 2, in order to be able to compare LSTM models trained on different vocabulary sizes, with results presented in Table 4.

	M1	M2	M3	M4	M5
Evaluation Data set 1 results					
CA	2.53	2.38	2.51	2.41	2.25
L	7.986	8.617	8.000	8.315	9.729
WP	2,939	5,522	2,981	4,084	16,789
Evaluation Data set 2 results					
CA	2.34	2.24	2.38	2.24	2.07
L	8.266	8.893	8.617	8.540	9.983
WP	3,888	7,279	5,522	5,117	21,653

Table 4: Results using Training Data set 2

5.2 LSTM network evaluation

The perplexity varied greatly for the different LSTMs, ranging from $2,939$ to $21,653$, and increasing with increased network complexity. However, networks trained on a smaller vocabulary did not have better perplexity scores than those trained on the larger vocabulary, so the ones trained on the smaller vocabulary were disregarded, since a smaller vocabulary means fewer possible unique words in the generated poems.

The network trained on the large vocabulary with best perplexity ($3,888$ for Test set 2) had 2 hidden layers and 256 hidden units per layer. However, generating text solely based on predictions given by this network, it showed signs of being highly overfitted, repeating a few select words. Since this network did not achieve perplexity scores significantly better than the alternatives, the larger network with the next-best perplexity was chosen instead. It has 3 hidden layers, 512 hidden units per layer, and was trained for 25 epochs, achieving a perplexity of $5,117$ for Test set 2, with a cross entropy loss of 8.540 and a categorical accuracy of 0.0224. A network with the same architecture, but trained for an additional 25 epochs, resulted in a much higher word perplexity score ($21,653$), likely due to underfitting the training data, since it had a much harder time predicting correct words.

While the perplexity differed greatly for the LSTMs, it was consistently very high, representing poor network training results. Zaremba et al. (2014) achieved a word perplexity of 78.4 with a regularised LSTM where dropout was used, training on the Penn Tree Bank (PTB) 10k vocabulary (Marcus et al., 1993). While the vocabulary size of the training data does not differ, there are several notable differences that will impact the results: Word perplexity will be greater for data with a larger vocabulary size (plainly due to the word possibilities increasing, clearly shown in Table 4. Furthermore, the PTB consists of grammatically correct literature, while the data used here consists of publicly written poetry, which is more irregular and with greater variation (especially in sentence structure and grammaticality), which could impact the networks' ability to learn patterns from the texts. The networks here were also trained on short sequences with a length of 5, making it harder for them to learn connections and general rules in the data.

	Positive sentiment	Negative sentiment
Rated positive	58.8%	2.7%
Rated neutral	36.2%	27.6%
Rated negative	5.0%	69.7%
Human average	0.360	-0.372
Vader average	0.977	-0.925

Table 5: Sentiment evaluation results

6 Evaluating the generated poetry

Human judges evaluated the generated poetry both in itself and with regard to the intended sentiment. The poetry was rated along the dimensions suggested by Manurung (2004): Grammaticality, Meaningfulness, and Poeticness, on a 1–3 scale (1 being "not", 2 "partially, and 3 "fully"). For the sentiment rating, the human judges evaluated the poetry using three categories: Negative sentiment, No sentiment (neutral), and Positive sentiment. If a poem was evaluated as having negative or positive sentiment, it was graded with a score of 1–3 ("Slightly", "Quite", and "Very" negative/positive).

Twenty poems were evaluated (i.e., 40 stanzas and 160 lines), generated to contain 10 each with positive and negative sentiment. Thirty human judges participated, evaluating a selection of 6 or 7 poems each, with a near-equal amount of positive and negative sentiments.

The 20 poems were scored with an average mean of Grammaticality: 1.488 ± 0.0476, Meaningfulness: 1.582 ± 0.0338, and Poeticness: 2.012 ± 0.0342. Table 5 shows the percent of the judges who rated the poems to contain positive, neutral or negative sentiment. The evaluators also rated the degree of perceived sentiment for each poem they had evaluated to contain a positive or negative sentiment, with ratings for every poem normalised to values between 0–1. The poems included in the experiments were also rated using Vader. Table 5 also shows the average sentiment degree scores.

The results show that on average the majority of judges perceived the poems to contain the sentiment value intended by the system, but the degrees of the sentiment value, both for negative and positive poems, are rated considerably lower by the human judges than the ratings given by Vader. One reason for this is the very high ratings that Vader assigns, both for negative and positive sentiment scores, with the generator discouraging the use of words with an opposite sentiment value (compared to the intended value), while encouraging the use of words with a "correct" sentiment value, given by Vader. The degree ratings from the human judges on the other hand reflect that a lack of the words with an opposite sentiment value does not result in a high degree of sentiment being perceived. It is also interesting to note the consistency of the degree ratings given by the human judges, where positive and negative sentiment poems are on average rated with almost the same degree of sentiment.

Three of the generated poems are presented below: #8 with one of the lowest scores in the experiment results, and #3 and #14 with some of the highest. Along with each poem are its scores, with the first array presenting the grammaticality, meaningfulness and poeticness scores. The second array shows the percentage of human judges who found the poem to contain negative, neutral, or positive sentiment. Poem 8 achieved a score of 1.22 for both grammaticality and meaningfulness, while Poem 14 achieved a score of 1.73 for those. The poeticness scores for both Poem 8 and 14 are below average, while Poem 3 achieved the highest poeticness score of all the poems with 2.5, and also high scores for grammaticality and meaningfulness.

A common trait among all the generated poems is incorrect use of articles and conjunctions, in addition to erroneous use of other word classes and poor sentence structure. Examples of this are the sequences *So most from an till* and *Amid to our so most from currently* in Poem 8. Another noticeable aspect is the rhyme pairs in the poetry not always rhyming, e.g., *exceptions* and *solutions* at the end of lines 6 and 8 in Poem 14. This is due to the system pairing rhyme words based on the last three syllables using CMUdict. A similar aspect is the consistent lack of repetitiveness found in the poetry, since repeated use of words is highly discouraged by the system, to avoid constant use of the same high predictions words, resulting in the poems lacking an often used poetic technique, to consciously

Poem 3

Of ravishing sin was naturally deprivation, 1
Of war suffering it tired than two then a situation, 2
And the go on no lies at finding they frustration, 3
Of voice are their seven then limitation. 4

The forest in mystic hands understood, 5
A trapped on must words most from your unarmed, 6
With wind raging to our misunderstood, 7
As a entire my that alarmed. 8

Metric evaluation scores: [1.7, 1.5, 2.5]
Sentiment evaluation scores: [0%, 20%, 80%]

Poem 8

For worst the all sleep as hearted of unpredictable, 1
So most from an till sensations, 2
Poor tall being eye in goes horrible, 3
Without the when todays so most from contradictions. 4

Kept on humanitys your soft in night a frightening, 5
Amid to our so most from currently, 6
In unwanted of feminine with sickening, 7
By no lies at impatiently. 8

Metric evaluation scores: [1.22, 1.22, 1.89]
Sentiment evaluation scores: [0%, 56%, 44%]

Poem 14

Without the oozing in night quickly divine thin lovable, 1
Like non focused best gods so wearing on contentment, 2
Bliss most from victory like favorable, 3
Without the respected of your improvement. 4

A dusty amid to governments, 5
Like mother of god be will certain the exceptions, 6
Amid to our so most from the do innocents, 7
In love a feeling most which are their solutions. 8

Metric evaluation scores: [1.73, 1.73, 1.91]
Sentiment evaluation scores: [82%, 18%, 0%]

repeat words or phrases. Other noticeable factors include misspellings (e.g., *humanitys* in Poem 8), and use of rare and special words. The generator vocabulary consists of the $20,000$ most frequently used words in the chosen data set, so any spelling error would mean that a high frequency of that misspelling occurs in the data.

The average scores for both grammaticality and meaningfulness are rather low. This correlates directly with the results from the LSTM training: the poor word perplexity scores for the LSTM networks have an effect on the words chosen by the poetry generator, creating poems having poor grammaticality, and therefore being more difficult to perceive meaningfulness from. The variables that update the word predictions during the generation could be improved to positively impact the evaluation scores, in particular the feature that updates prediction values based on sentence structure.

Poeticness had a significantly higher score, which could be due to factors not influenced by

the LSTM model's performance, specifically including the generation and use of rhyming pairs and the form of the poetry. The form of the poetry is also not influenced by the LSTM predictions, as the length of each line is randomly decided between two outer bounds, and proper punctuation is added after each line, including a line break between stanzas. While consistent rhyme form, punctuation, and varying line lengths likely account for the good poeticness results, possible weaknesses that might have affected the results are the lack of perfect rhymes and the lack of known poetry forms with consistent syllable lengths, such as sonnets. The lack of repetitiveness as encouraged by the system might also negatively affect the score.

7 Conclusion and Future Work

A system capable of generating poetry with inherent sentiment has been designed and implemented, with the main system component being a bi-directional Long Short-Term Memory network used for generating word predictions based on a given input sequence. The network was trained on a data set consisting of poetry written by humans. Other components of the final generation system are algorithms and rule-based methods for influencing word predictions and word choices during the generation process, and a search algorithm for expanding the possibilities of generated sequences.

The implemented system was used to generate 20 poems in total, all consisting of two stanzas with four lines each. 10 of the poems were generated to contain an inherent positive sentiment value, while the other 10 were generated to contain a negative sentiment value. Several experiments were conducted, both regarding the LSTM network and on the generated poetry. The first experiment was on training different LSTM networks with varying architecture details, with the goal of training the best performing network model to use in the implementation of the final poetry generation system.

Two other experiments were conducted on the final generated poetry, both involving human judges evaluating the generated poetry. The first of these experiment consisted of the judges evaluating the poetry based on three standard evaluation criteria. This enabled evaluation of the performance of the poetry generation system, and comparison to other works that have been conducted in this field. In the second experiment the human judges evaluated the sentiment value they perceived generated poetry to contain, in order to investigate whether the system was capable of generating poetry with an inherent sentiment value that would be perceived as intended by human readers.

The results of the experiments varied, with LSTM experiments giving word perplexity scores worse than state-of-the-art solutions. Applying some standard evaluation metrics showed one of the metrics achieving similar values to state-of-the-art solutions, while the other metric gave poorer results, one reason being the influence from the subpar prediction performance of the LSTM network. The experiment for evaluating the sentiment of the generated poetry produced good results. While there is a lack of similar experiments by others to compare to, the results show a clear trend of the human judges perceiving the poetry to contain the intended sentiment value.

Possible future work could include implementing additional features or other architectures, such as word embedding models or language models like BERT (Devlin et al., 2018) that show prominent results for text analysis, or use mutual reinforcement, which has given state-of-the-art results in poetry generation (Yi et al., 2018). The data set used to train the neural network model has a considerable effect on the system's performance and the generated poetry. Hence using different data sets, especially data containing poetry of a generally accepted higher quality, would probably improve the system. Adding only perfect rhymes for rhyme pair generation, or a strict poetic form based on syllables, such as the sonnet form, could improve the poetic qualities of the output.

The main feature of the generation system is to generate poetry with an inherent sentiment, and this can also be further developed. First, the system needs to generate poetry with a wider range of sentiment value words, as it current only uses words with neutral sentiment or sentiment values corresponding to the intended sentiment. Adding more words with opposite sentiment could increase the poetry's emotional dynamic. The sentiment feature could also be extended to generate poetry with a wider range of emotions, e.g., by using emotional modelling similarly to Misztal and Indurkhya (2014), or by adapting the system to generate poetry with an inherent degree of a specific sentiment, not just a general negative or positive value.

Acknowledgments

Thanks to Trond Aalberg and the AI Master students at NTNU's Department of Computer Science for discussions, insights and helpful feedback.

Thanks also to the people who participated in the experiments that were conducted and to Tikhonov and Yamshchikov for providing the data set used.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

John Benhart, Tianlin Duan, Peter Hase, Liuyi Zhu, and Cynthia Rudin. 2018. Shall I compare thee to a machine-written sonnet? An approach to algorithmic sonnet generation. *CoRR*, abs/1811.05067.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Margaret M. Bradley and Peter J. Lang. 1999. Affective norms for English words (ANEW): Instruction manual and affective ratings. Technical Report C-1, Center for Research in Psychophysiology, University of Florida, Gainesville, FL, USA.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-FACE poetry generation. In *Proceedings of the 3rd International Conference on Computational Creativity*, pages 95–102, Dublin, Ireland.

Tim Van de Cruys. 2020. Automatic poetry generation from prosaic text. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2471–2480, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Pablo Gervás. 2000. WASP: Evaluation of different strategies for the automatic generation of Spanish verse. In *Proceedings of the AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, pages 93–100, University of Birmingham, England.

Pablo Gervás. 2001. An Expert System for the Composition of Formal Spanish Poetry. In *Applications and Innovations in Intelligent Systems VIII*, pages 19–32, London, England. Springer.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, BC, Canada. Association for Computational Linguistics.

Hugo Gonçalo Oliveira. 2012. PoeTryMe: A versatile platform for poetry generation. In *Proceedings of the Workshop on Computational Creativity, Concept Invention, and General Intelligence*, pages 21–26, Montpellier, France. Publications of the Institute of Cognitive Science.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

C.J. Hutto and Eric Gilbert. 2015. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014*, pages 82–91, Ann Arbor, MI, USA. AAAI Press.

Robert P. Levy. 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*, Vancouver, BC, Canada.

Hisar Maruli Manurung. 2004. *An evolutionary algorithm approach to poetry generation*. Ph.D. thesis, Institute for Communicating and Collaborative Systems, School of Informatics, College of Science and Engineering, University of Edinburgh, Edinburgh, Scotland.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Joanna Misztal and Bipin Indurkhya. 2014. Poetry generation system with an emotional personality. In *Proceedings of the 5th International Conference on Computational Creativity*, pages 72–81, Ljubljana, Slovenia.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Carlo Strapparava and Alessandro Valitutti. 2004. WordNet-Affect: an affective extension of WordNet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1083–1086, Lisbon, Portugal.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

Aleksey Tikhonov and Ivan Yamshchikov. 2018a. Sounds Wilde. phonetically extended embeddings for author-stylized poetry generation. In *Proceedings of the 15th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 117–124, Brussels, Belgium. Association for Computational Linguistics.

Alexey Tikhonov and Ivan P. Yamshchikov. 2018b. Guess who? Multilingual approach for the automated generation of author-stylized poetry. In *2018 IEEE Spoken Language Technology Workshop*, pages 787–794, Athens, Greece. IEEE.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016a. Chinese song iambics generation with neural attention-based model. *CoRR*, abs/1604.06274.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016b. Chinese poetry generation with planning based neural network. *CoRR*, abs/1610.09889.

Rui Yan. 2016. I, Poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2238–2244. AAAI Press.

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2016. Generating Chinese classical poems with RNN encoder-decoder. *CoRR*, abs/1604.01537.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Wenhao Li. 2018. Automatic Poetry Generation with Mutual Reinforcement Learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3143–3153, Brussels, Belgium. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative Chinese poetry generation using neural memory. *CoRR*, abs/1705.03773.

Xingxing Zhang and Mirella Lapata. 2014. Chinese Poetry Generation with Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar. Association for Computational Linguistics.