Deep-speare: A Joint Neural Model of Poetic Language, Meter and Rhyme

Jey Han Lau[1,2], Trevor Cohn[2], Timothy Baldwin[2],
Julian Brooke[3], and Adam Hammond[4]

[1] IBM Research Australia

[2] School of CIS, The University of Melbourne

[3] University of British Columbia

[4] Dept of English, University of Toronto

27 March, 2019

Creativity

- Can machine learning models be creative?
- Can these models compose novel and interesting narrative?
- Creativity is a hallmark of intelligence — it often involves blending ideas from different domains.
- We focus on sonnet generation in this work.

Sonnets

Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date:

- A distinguishing feature of poetry is its *aesthetic forms*, e.g. rhyme and rhythm/meter.
- Rhyme: $\{$*day*, *May*$\}$; $\{$*temperate*, *date*$\}$.
- Stress (pentameter):

$$\begin{array}{cccccccccc} S^{-} & S^{+} & S^{-} & S^{+} & S^{-} & S^{+} & S^{-} & S^{+} & S^{-} & S^{+} \\ \textit{Shall} & \textit{I} & \multicolumn{2}{c}{\textit{compare}} & \textit{thee} & \textit{to} & \textit{a} & \multicolumn{2}{c}{\textit{summer's}} & \textit{day?} \end{array}$$

Modelling Approach

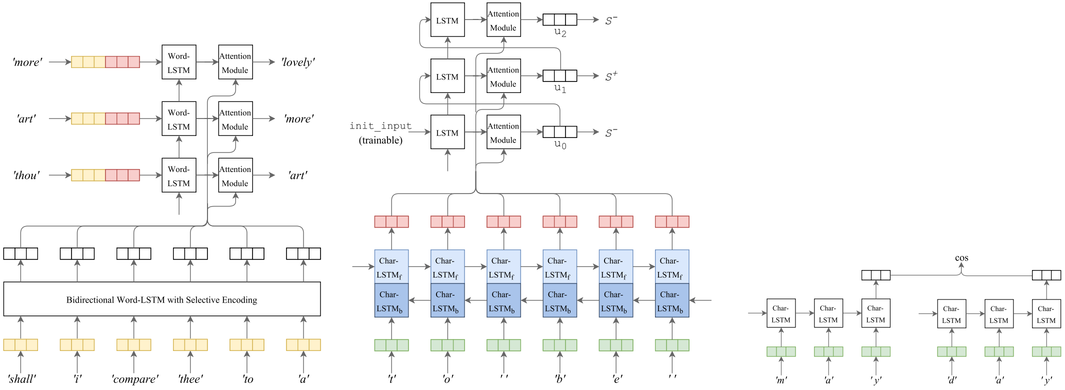
- We treat the task of poem generation as a constrained language modelling task.
- Given a rhyming scheme, each line follows a canonical meter and has a fixed number of stresses.
- We focus specifically on sonnets as it is a popular type of poetry (sufficient data) and has regular rhyming (ABAB, AABB or ABBA) and stress pattern (iambic pentameter).
- We train an unsupervised model of language, rhyme and meter on a corpus of sonnets.

Sonnet Corpus

- We first mine a generic poetry document collection from Project Gutenberg using GutenTag tool, based on its inbuilt poetry classifier.
- We then extract word and character statistics from Shakespeare's 154 sonnets.
- We use the statistics to filter out all non-sonnet poems, yielding our sonnet corpus.

Partition	**#Sonnets**	**#Words**
Train	2685	367K
Dev	335	46K
Test	335	46K

Model Architecture



(a) Language model

(b) Pentameter model

(c) Rhyme model

Language Model (LM)

- LM is a variant of an LSTM encoder–decoder model with attention.
- Encoder encodes preceding contexts, i.e. all sonnet lines before the current line.
- Decoder decodes one word at a time for the current line, while attending to the preceding context.
- Preceding context is filtered by a selective mechanism.
- Character encodings are incorporated for decoder input words.
- Input and output word embeddings are tied.

Selective Mechanism

- Not all words are equally useful in creating the sentence representation.
- Content words are likely to be more important for capturing its meaning.

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$$
$$\overline{\mathbf{h}} = [\overrightarrow{\mathbf{h}}_C; \overleftarrow{\mathbf{h}}_1]$$
$$\mathbf{h}'_i = \mathbf{h}_i \odot \sigma(\mathbf{W}_a \mathbf{h}_i + \mathbf{U}_a \overline{\mathbf{h}} + \mathbf{b}_a)$$

- $h_i =$ encoder's hidden step at timestep i
- $C =$ total number of words in the sentence

Pentameter Model (PM)

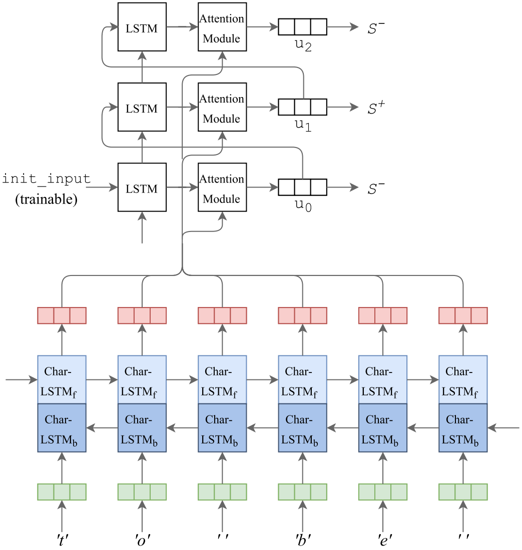
- PM is designed to capture the alternating stress pattern.
- Given a sonnet line, PM learns to attend to the appropriate characters to predict the 10 binary stress symbols sequentially.

T	Attention	Prediction
0	*Shall I compare thee to a summer's day?*	S^-
1	*Shall I compare thee to a summer's day?*	S^+
2	*Shall I compare thee to a summer's day?*	S^-
3	*Shall I compare thee to a summer's day?*	S^+
	...	
8	*Shall I compare thee to a summer's day?*	S^-
9	*Shall I compare thee to a summer's day?*	S^+

Pentameter Model (PM)

- PM fashioned as an encoder–decoder model.
- Encoder encodes the characters of a sonnet line.
- Decoder attends to the character encodings to predict the stresses.
- Decoder states are not used in prediction.
- Attention networks focus on characters whose position is monotonically increasing.
- In addition to cross-entropy loss, PM is regularised further with two auxiliary objectives that penalise repetition and low coverage.

Pentameter Model (PM)



Pentameter Model Formulation

- **Input**: a sentence in characters (list of letters)
- **Encoder**: bidir character LSTM to produce character encodings: $\mathbf{u}_j = [\vec{\mathbf{u}}_j; \overleftarrow{\mathbf{u}}_j]$
- **Decoder**: unidir LSTM: $\mathbf{g}_t = \text{LSTM}(\mathbf{u}^*_{t-1}, \mathbf{g}_{t-1})$
- $\mathbf{u}^*_{t-1}$ = weighted sum of character encodings from previous time step
- **Output**: $P(S^-) = \sigma(\mathbf{W}_e \mathbf{u}^*_t + b_e)$
- We do not include $\mathbf{g}_t$ here, as the model will quickly learn that it can ignore $\mathbf{u}^*_t$ in the prediction.

Attention Networks

- Two forms of attention: position attention and character attention

- μ_t $(0 - 1)$, the mean position of attention:

$$
\begin{aligned}
\mu'_t &= \sigma(\mathbf{v}_c^\mathsf{T} \tanh(\mathbf{W}_c \mathbf{g}_t + \mathbf{U}_c \mu_{t-1} + \mathbf{b}_c)) \\
\mu_t &= \min(\mu'_t + \mu_{t-1}, 1.0) \\
\overline{\mu}_t &= M \times \mu_t
\end{aligned}
$$

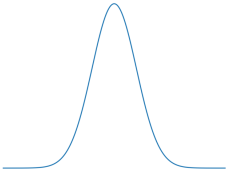
  where $M =$ number of characters in the input line.

- Probability for each character position:

$$p_j^t = \exp\left(\frac{-(j - \overline{\mu}_t)^2}{2T^2}\right)$$

  where standard deviation T is a hyper-parameter.

Position Attention



Input	S	h	a	l	l		I		c	o	m	p	a	r	e
$\mathbf{p^t}$	0.0	0.0	0.0	0.0	0.0		0.0		0.6	1.0	0.6	0.1	0.0	0.0	0.0

Aggregate Attention

$$
\begin{aligned}
\mathbf{u}'_j &= p^t_j \mathbf{u}_j \\
d^t_j &= \mathbf{v}_d^{\mathsf{T}} \tanh(\mathbf{W}_d \mathbf{u}'_j + \mathbf{U}_d \mathbf{g}_t + \mathbf{b}_d) \\
\mathbf{f}^t &= \operatorname{softmax}(\mathbf{d}^t + \log \mathbf{p}^t) \\
\mathbf{u}^*_t &= \sum_j f^t_j \mathbf{u}_j
\end{aligned}
$$

- $\mathbf{d}^t$ = character attention;
- $\mathbf{p}^t$ = position attention.

Additional Regularisation

- Repeat loss penalises the model when it attends to previously attended characters:

$$\mathcal{L}_{rep} = \sum_t \sum_j \min(f_j^t, \sum_{t=1}^{t-1} f_j^t)$$

- Coverage loss penalises the model when vowels are ignored:

$$\mathcal{L}_{cov} = \sum_{j \in V} \text{ReLU}(C - \sum_{t=1}^{10} f_j^t)$$

C = minimum attention threshold; V = set of positions containing vowels.

- Total pentameter model loss: $\mathcal{L}_{pm} = \mathcal{L}_{ent} + \alpha\mathcal{L}_{rep} + \beta\mathcal{L}_{cov}$

Rhyme Model

- We learn rhyme in an unsupervised fashion for 2 reasons:
  - Extendable to other languages that don't have pronunciation dictionaries;
  - The language of our sonnets is not Modern English, so contemporary pronunciation dictionaries may not be accurate.
- Assumption: rhyme exists in a quatrain.
- Feed sentence-ending word pairs as input to the rhyme model and train it to separate rhyming word pairs from non-rhyming ones.

Rhyme Model

Shall I compare thee to a summer's day? $\overline{\mathbf{u}}_t$
Thou art more lovely and more temperate: $\overline{\mathbf{u}}_r$
Rough winds do shake the darling buds of May, $\overline{\mathbf{u}}_{r+1}$
And summer's lease hath all too short a date: $\overline{\mathbf{u}}_{r+2}$

$$Q = \{\cos(\overline{\mathbf{u}}_t, \overline{\mathbf{u}}_r), \cos(\overline{\mathbf{u}}_t, \overline{\mathbf{u}}_{r+1}), \cos(\overline{\mathbf{u}}_t, \overline{\mathbf{u}}_{r+2})\}$$
$$\mathcal{L}_{rm} = \max(0, \delta - \mathrm{top}(Q, 1) + \mathrm{top}(Q, 2))$$

- $\mathrm{top}(Q, k)$ returns the k-th largest element in Q.
- Intuitively the model is trained to learn a sufficient margin that separates the best pair from **all others**, with the second-best being used to quantify **all others**.

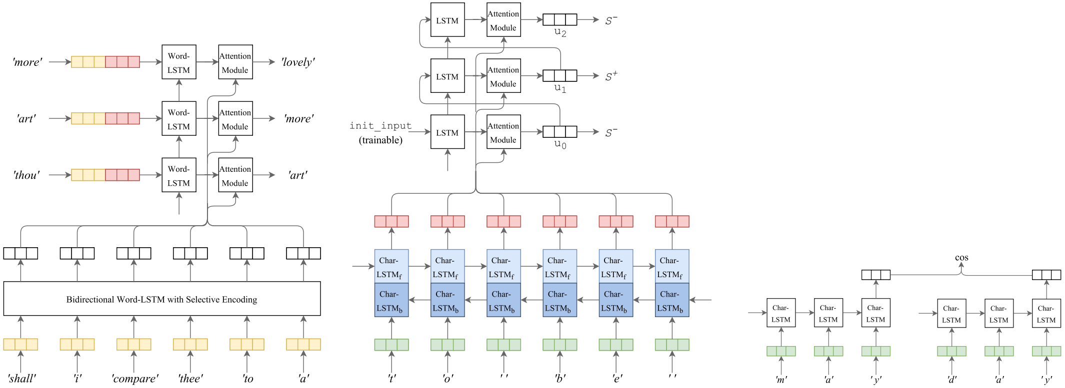
Joint Training

- All components trained together by treating each component as a sub-task in a multi-task learning setting.

- Although the components (LM, PM and RM) appear to be disjointed, shared parameters allow the components to mutually influence each other during training.

- If each component is trained separately, PM performs poorly.

Model Architecture



(a) Language model (b) Pentameter model (c) Rhyme model

Generation Procedure

- We focus on quatrain generation (i.e. 4 lines of poetry).
- Words are sampled one word at a time, using a temperature between 0.6 to 0.8.
- To enforce rhyme, at the start of generation we randomly select a rhyming scheme, and re-sample sentence-ending words as necessary.
- We use the cosine similarity score predicted by the rhyme model to judge whether a pair of words rhymes.
- To enforce iambic pentameter, we generate 10 candidate sentences for each quatrain sentence.
- We then sample one sentence from the 10 candidates, weighted by their $\mathcal{L}_{pm}$ scores.

Evaluation: Language Model

- `LM`: Vanilla LSTM language model;
- `LM`*: LSTM language model that incorporates character encodings;
- `LM`**: LSTM language model that incorporates both character encodings and preceding context;
- `LM**-C`: Similar to `LM`**, but preceding context is encoded using convolutional networks;
- `LM**+PM+RM`: the full model.

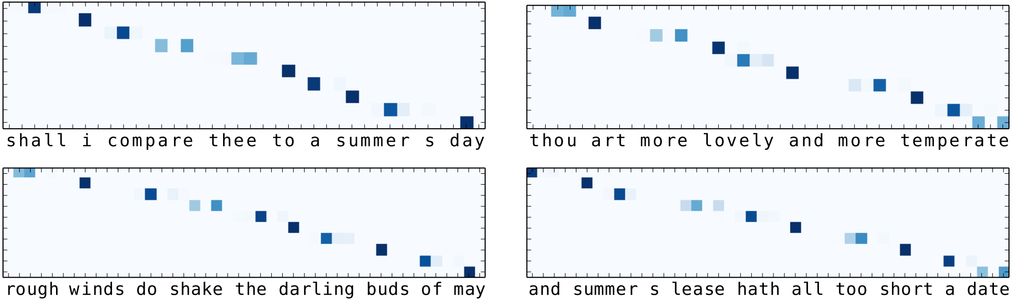
	LM	LM*	LM**	LM**-C	LM**+PM+RM
Perplexity	90.13	84.23	80.41	83.68	80.22

Evaluation: Pentameter Model

- We use attention weights to predict stress patterns for words in the test data.
- Compare them against stress patterns defined in the CMU pronunciation dictionary.
- To extract stress pattern for a word, we iterate through the pentameter (10 steps), and append the appropriate stress to the word if any of its characters receives an attention $\geqslant 0.20$.
- `Stress-BL`: pretrained weighted finite state transducer model (Hopkins and Kiela (2017)).

	`Stress-BL`	`LM**+PM+RM`
Accuracy	0.80	0.74

Qualitative Evaluation



- Informal inspection reveals a number of mistakes are due to dictionary errors.
- Attention heatmap results are encouraging.
- *lovely*: second stress focuses on character *e* than *y*.

Evaluation: Rhyme Model

- Gold Standard: A word pair is judged to rhyme is the pronunciation of their last syllables (based on CMU) match.
- `LM**+PM+RM`: Rhyme if predicted cosine similarity $\geqslant 0.80$.
- `Rhyme-BL`: Rhyme if their last vowel and succeeding consonant sequences match.
- `Rhyme-EM`: Rhyme if rhyming strength predicted by a pre-trained rhyme model (Reddy and Knight (2011)) $\geqslant 0.02$.

	`Stress-BL`	`Rhyme-EM`	`LM**+PM+RM`
F1	0.74	0.71	0.91

Evaluation: Crowdworkers

- Crowdworkers are presented with a pair of poems (one machine-generated and one human-written), and asked to guess which is the human-written one.

- `LM`: vanilla LSTM language model;

- `LM**`: LSTM language model that incorporates both character encodings and preceding context;

- `LM**+PM+RM`: the full model, with joint training of the language, pentameter and rhyme models.

Evaluation: Crowdworkers (2)

Model	Accuracy
LM	0.742
LM**	0.672
LM**+PM+RM	0.532
LM**+RM	0.532

- Accuracy improves LM $<$ LM** $<$ LM**+PM+RM, indicating generated quatrains are less distinguishable.
- Are workers judging poems using just rhyme?
- Test with LM**+RM reveals that's the case.
- Meter/stress is largely ignored by laypersons in poetry evaluation.

Evaluation: Expert

Model	Meter	Rhyme	Read.	Emotion
LM	4.00±0.73	1.57 ±0.67	2.77 ±0.67	2.73 ±0.51
LM^{**}	4.07 ±1.03	1.53 ±0.88	3.10 ±1.04	2.93±0.93
LM^{**}+PM+RM	4.10 ±0.91	4.43±0.56	2.70±0.69	2.90±0.79
Human	3.87 ±1.12	4.10 ±1.35	4.80±0.48	4.37 ±0.71

- A literature expert is asked to judge poems on the quality of meter, rhyme, readability and emotion.
- Full model has the highest meter and rhyme ratings, even higher than human, reflecting that poets regularly break rules.
- Despite excellent form, machine-generated poems are easily distinguished due to lower emotional impact and readability.
- Vanilla language model (LM) captures meter surprisingly well.

Summary

- We introduce a joint neural model that learns language, rhyme and stress in an unsupervised fashion.
- We encode assumptions we have about the rhyme and stress in the architecture of the network.
- Model can be adapted to poetry in other languages.
- We assess the quality of generated poems using judgements from crowdworkers and a literature expert.
- Our results suggest future research should look beyond forms, towards the substance of good poetry.
- Code and data: `https://github.com/jhlau/deepspeare`

Creativity?

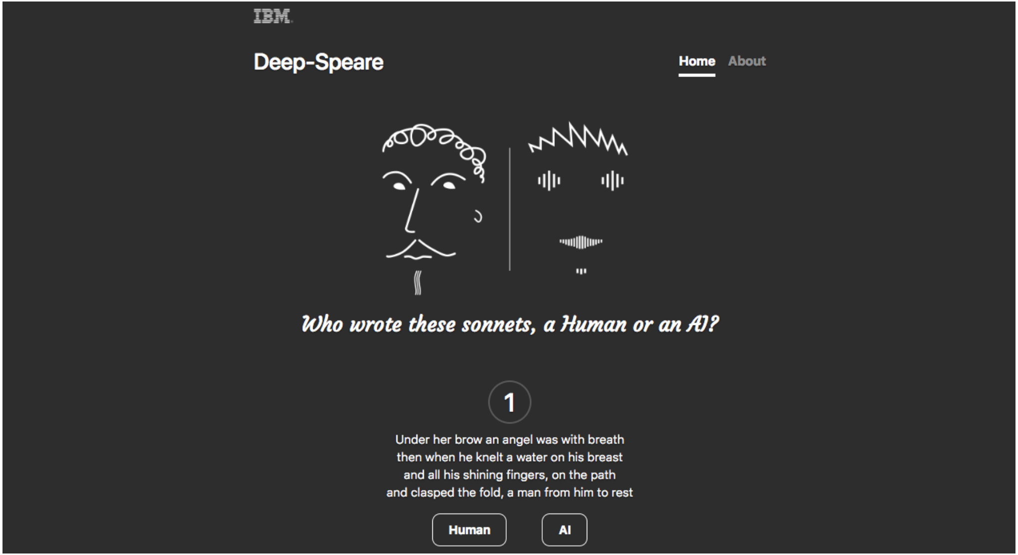
Can machine learning models be creative?
Can these models compose novel and interesting narrative?

- To a certain extent, yes.
- The model is not just copying words from training data.
- It composes new narratives that seem like... poetry.

“Untitled”

in darkness to behold him, with a light
and him was filled with terror on my breast
and saw its brazen ruler of the night
but, lo! it was a monarch of the rest

Demo


IBM
Deep-Speare
Home
About
Who wrote these sonnets, a Human or an AI?
1
Under her brow an angel was with breath
then when he knelt a water on his breast
and all his shining fingers, on the path
and clasped the fold, a man from him to rest
Human
AI

Questions?