Natural Language Statistical Features of LSTM-Generated Texts

Marco Lippi, Marcelo A. Montemurro, Mirko Degli Esposti, and Giampaolo Cristadoro

Abstract— **Long short-term memory (LSTM) networks have recently shown remarkable performance in several tasks that are dealing with natural language generation, such as image captioning or poetry composition. Yet, only few works have analyzed text generated by LSTMs in order to quantitatively evaluate to which extent such artificial texts resemble those generated by humans. We compared the statistical structure of LSTM-generated language to that of written natural language, and to those produced by Markov models of various orders. In particular, we characterized the statistical structure of language by assessing word-frequency statistics, long-range correlations, and entropy measures. Our main finding is that while both LSTM- and Markov-generated texts can exhibit features similar to real ones in their word-frequency statistics and entropy measures, LSTM-texts are shown to reproduce long-range correlations at scales comparable to those found in natural language. Moreover, for LSTM networks, a temperature-like parameter controlling the generation process shows an optimal value—for which the produced texts are closest to real language—consistent across different statistical features investigated.**

Index Terms— **Authorship attribution, entropy, long short-term memory networks, long-range correlations, natural language generation (NLG).**

I. Introduction

BUILDING artificial systems that are capable of mimicking human creativity has long been one of the aims of artificial intelligence (AI) [1]. In this paper, we focus on the problem of natural language generation (NLG), which encompasses the capability of machines to synthesize text in a way that resembles spoken or written language typically employed by humans [2]. This research field has recently experienced a period of great excitement, mostly due to the huge development in the area of deep learning, whose methods and algorithms have certainly contributed to move significant steps forward [3].

Deep learning techniques have, in fact, produced stunning results in several different research fields and application domains, and one of the major successes has been that of generative models [4]. In the area of NLG, many studies have been dedicated to specific and focused applications, such as image and video captioning [5], [6], poem synthesis [7], or lyric generation [8]. In all these cases, the considered generated texts are relatively short (captions, poems, and lyrics) and correlations between words rarely span across several sentences. The scenario totally changes when we consider longer texts, such as novels. Natural language has been widely studied within this context, and notoriously it shows statistical properties in the distribution of terms, as well as long-range correlations between words [9]–[11]. In comparison to short texts such as captions, this is a much more challenging setting to imitate for machines.

In this paper, we aim to provide an extensive empirical evaluation of texts generated by long short-term memory (LSTM) networks, one of the most widely used deep learning models for NLG. Our goal is to quantitatively assess whether LSTM texts share some similarities with natural language that is commonly produced by humans. To this aim, we trained an LSTM network with a corpus that consists of a collection of novels by Charles Dickens. Such network is trained to predict the next character of a given text, and thus it can be employed to iteratively generate a document of any desired length. The setting was adopted in several works (see [12] and [13] and citations therein).

In our experimental framework, we evaluated several different aspects of machine-generated texts comparing them against the statistics of real language and Markov-generated samples. First, we analyzed fundamental linguistic properties typically shown by texts, such as Zipf's [14] and Heaps' [15] laws for words. Second, we studied whether the generated texts presented long-range correlations, which are commonly encountered in human-generated texts but difficult to reproduce for machines. As a third point, we compared the entropy of the generated texts with the one of the original corpus. We then moved our analysis to a higher level by carrying out a preliminary study looking at characteristics dealing with the style and quality of the generated texts: in particular, we analyzed the degree of creativity and the plagiarism of the artificial texts with respect to the data set on which the LSTM was trained, by looking at the longest common subsequences (LCSs). We also assessed whether an authorship

Manuscript received March 29, 2018; revised October 21, 2018; accepted December 25, 2018. (Marco Lippi and Marcelo A. Montemurro contributed equally to this work.) *(Corresponding author: Marco Lippi.)*

M. Lippi is with the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, 41121 Modena, Italy (e-mail: marco.lippi@unimore.it).

M. A. Montemurro is with the Division of Neuroscience and Experimental Psychology, The University of Manchester, Manchester M13 9PT, U.K. (e-mail: m.montemurro@manchester.ac.uk).

M. D. Esposti is with the Department of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy (e-mail: mirko.degliesposti@unibo.it).

G. Cristadoro is with the Department of Mathematics and Applications, University of Milano - Bicocca, 20126 Milan, Italy (e-mail: giampaolo.cristadoro@unimib.it).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2890970

attribution algorithm would capture some analogy between the generated text and the original one, in terms of authors' style.

Surprisingly, very few studies have been dedicated to a thorough analysis and to a quantitative evaluation of the similarities between texts created by machines and by humans. Karpathy *et al.* [13] also provide an experimental analysis of LSTM networks trained for character-by-character text generation, but they focused their study on a qualitative evaluation of the cell activations within the neural architecture: for example, they looked for open and closed parentheses or quotes that typically span a few tens or hundreds of characters. They claim that the LSTM model is capable of capturing long-range dependencies and is thus only supported by such qualitative evidence without giving a deep insight into the characteristics of the generated documents. Lin and Tegmark [16] compared natural language texts to those generated by Markov models and LSTMs, by exploiting metrics coming from information theory. Their analysis showed that LSTMs are capable of capturing correlations that Markov models instead fail to represent, yet the range of correlations they consider is still quite limited (up to 1000 characters). Conversely, Takahashi and Tanaka-Ishii [17] reported that LSTM language models have limitation in reproducing such long-range correlations if measured with a method based on clustering properties of rare words; note, however, that their analysis is still limited to a range of ~1000 words, and the corpus they employ for training is much smaller than the one used in our experiments. Instead, Ghodsi and DeNero [18] analyzed some statistical properties of text generated by a recurrent neural network language model (RNNLM), especially focusing on the length of sentences, the vocabulary distribution, and the distribution of specific grammatical elements, such as pronouns. In a complementary study, Lake and Baroni [19] focused on short-scale structures instead of the overall statistical properties of pseudotexts. They showed that recurrent neural networks (RNNs) are able to exploit systematic compositionality and thus also able to reproduce, for example, abstract grammatical generalizations.

The remainder of this paper is organized as follows. Section II will describe the LSTM model used in our experiments. Section III will present the statistical methods employed for the quantitative evaluation of the artificial text properties. Section IV will describe the corpora that are used to train our model, and Section V will report and discuss the experimental results. Section VI will finally conclude this paper, also pointing for future research directions.

II. Long Short-Term Memory Networks

Long short-term memory networks (LSTMs) are RNNs that have been first developed at the end of the 1990s, achieving remarkable results in applications dealing with input sequences [20]. Such model was specifically designed to address the issue of vanishing gradients that greatly limited the applicability of standard RNNs [21]. Within the "deep learning revolution" that AI has been undergoing in the past decade, LSTMs have regained popularity, being now widely used in several research and industrial applications, including automatic machine translation, speech recognition, and text-to-speech generation (see [22] and references therein).

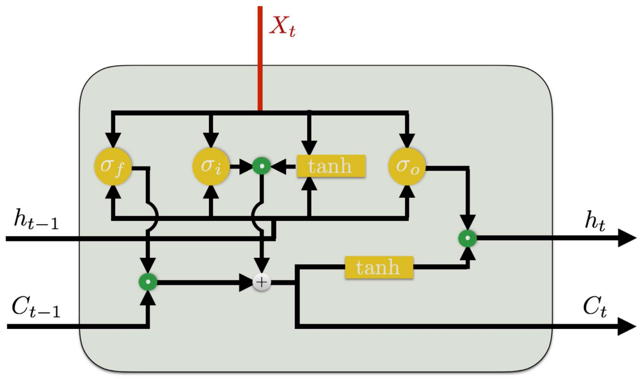

Fig. 1. LSTM cell. σ_f, σ_i, and σ_o are typically the sigmoid function. Recurrent connections h_t and C_t propagate information through time.

A. General Framework

RNNs allow to process sequences of arbitrary lengths, by exploiting L hidden layers h_t^ℓ, with $\ell = \{1, \ldots, L\}$, whose cells are functions not only of the layer input x_t but also of the hidden layer at the previous time step: $h_t^\ell = f(x_t, h_{t-1}^\ell)$. RNNs are typically trained with backpropagation through time (BPTT) [23], by unfolding the recurrent structure into a sort of temporal chain through which the gradient is propagated, up to a certain number K of time steps. Unfortunately, this method suffers from the well-known problem of vanishing or exploding gradients [20], [21], which makes plain RNNs scarcely used in practice. LSTMs overcome this issue by exploiting a more complex hidden cell, namely, a memory cell, and nonlinear-gating units, which control the information flows into and out of the cell.

Basically, the LSTM cells are capable of maintaining their state over time, of forgetting what they have learned, and also of allowing novel information in. An example of such a cell is depicted in Fig. 1. The model is based on the concept of cell status at time t, namely, C_t, which depends on three gates: an input gate i_t that can let new information into the cell state, a forget gate f_t that can modulate how much information is forgotten from the previous state, and an output gate o_t that controls how much information is transferred to the upper layers. The following equations describe the behavior of an LSTM layer (we drop the layer index ℓ in order to simplify the notation):

$$f_t = \sigma_f(W_f x_t + U_f h_{t-1} + b_f) \tag{1}$$
$$i_t = \sigma_i(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$
$$o_t = \sigma_o(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{4}$$
$$h_t = o_t \odot \tanh(C_t) \tag{5}$$

where all σ_f, σ_i, and σ_o are typically the sigmoid function; $\odot$ indicates the Hadamard (or element-wise) product; and W, U, and b represent the model parameters that have to be learned. As a form of regularization, dropout is nowadays typically employed in deep neural networks [24]. Dropout simply consists of randomly dropping a percentage $(1-p)$ of connections between neurons during training while multiplying by p every weight in the network at testing time. In recurrent architectures, though, this general framework does not work well in practice, but dropout can still be successfully employed, if applied only to interlayer connections and not to recurrent ones [25]. This is how we employed dropout in our model.

B. LSTM for Text Generation

The most widely employed LSTM architecture for text generation is based on character-level sentence modeling. Basically, the input of the network consists of M characters that correspond to a fixed-size portion of text, whereas the number of output neurons is the total number S of possible symbols in the text, each neuron corresponding to one of such characters. The output layer consists of a softmax layer, so that each symbol has an associated probability, and all such probabilities sum up to 1. A hard way of generating texts is to pick the character with the highest probability as the next one in the sequence, and to feed it back to the network input. A soft alternative (the one used in practice) allows to sample the next character in the sequence from the probability distribution of the output cells. In this way, the output of the network is not deterministic, given the same initial input sequence. With such an iterated procedure, texts of any length can be generated. The hidden states of the cells keep track of the "memory" of the network, so as to exploit also information not directly encoded in the input any more. This generation phase can be controlled by a parameter T, usually named as temperature. Different temperature values can be tuned in order to obtain a smoother or sharper softmax distribution from which characters are generated. In particular, the final softmax layer in the LSTM computes the probability of each symbol j as follows:

$$P_j = \frac{\exp\left(\frac{y_j}{T}\right)}{\sum_{i=1}^{S}\exp\left(\frac{y_i}{T}\right)} \tag{6}$$

where y_i are the output values for each symbol, which are fed into softmax. Large T values lead toward a uniform distribution of symbol probabilities, whereas when T tends to zero, the distribution is skewed toward the most probable symbol.

Such a model is trained in a classic supervised learning setting where the input training corpus is fed to the network, using as target the true (known) next character, as it appears in the corpus. If the cell states are not reset as subsequent input windows are presented to the network, long-range dependencies can, in principle, be captured by the model.

Note that, in principle, the same task could be modeled at a word level, thus training the LSTM network to predict the next word in the text rather than the next character. Although this solution may appear to be a more appropriate way for modeling the problem, it has two main limitations: 1) the number of possible output classes of the network would become huge, being the total number of distinct words in the input corpus, leading to a much more difficult optimization problem, which would likely require a larger number of training examples and 2) the set of possible output words would be limited to those appearing in the training corpus, which could be enough in many cases, but would limit the creativity of the network.

III. Methods

We now present the methods we employ in order to quantitatively evaluate the characteristics of the artificially generated texts with respect to the original and human-generated texts.

A. Zipf's and Heaps' Laws for Words

Natural languages show remarkable statistical properties in their word statistics. The two best known examples are Zipf's [14] and Heaps' [15] laws in language, which refer to universal features related to word frequencies.

Zipf's law states that if the word frequencies of any sufficiently long text are arranged in the decreasing order, there is a power-law relationship between the frequency and the corresponding ranking order of each word. More explicitly, if we denote the rank of a word by r, the Zipf's law states the following relationship between the rank and the frequency of a word at that rank position $f(r)$, as follows:

$$f(r) = Ar^{-\beta}. \tag{7}$$

This relationship is roughly the same for all human languages, the exponent β taking values close to 1.

Heaps' law states that the number of the different words (i.e., the size of the vocabulary) after seeing t consecutive words in a text, obeys approximately the following relationship:

$$n(t) = Bt^{\nu} \tag{8}$$

with exponent ν typically taking values smaller than 1 [26].

B. Long-Range Correlations

Linguistic laws on the scaling of word frequencies, such as Zipf's and Heaps', do not reveal any statistical structure that depends directly on word order. Zipf's rank-frequency distribution would be unaltered after a random text shuffling, and similarly for the average behavior predicted by Heaps' law. Statistical measures that capture structure at the sequence level in texts involve correlations and spectral analysis [27].

Correlations in language are known to be of the power-law type [9]–[11], decaying as $C(\tau) \propto \tau^{-\gamma}$, where τ is the distance between symbols, for example, words or characters. It is possible to characterize the structure of long-range correlations using the method of detrended fluctuation analysis (DFA) [28], [29]. The first step in the method involves the mapping of the symbol sequence onto a numerical time series, by assigning a number to each basic symbol in the sequence. In order to preserve the maximum of structure from the text sources, we performed the mapping at character level—including all punctuation signs, numbers, capital letters, and accented forms. The procedure to assign a number to each character followed similar lines to the method employed in [10], where in the present case, each character is replaced by its rank. Thus, the most frequent character is assigned the number 1, the second most frequent is assigned the number 2, and so on. For a sequence of length N, the character at position t in the time series, with $t \in \mathbb{N}$, can be represented by the number $x(t)$, and the following random walk can be constructed:

$$X(t) = \sum_{i=1}^{t}(x(i) - \langle x \rangle) \tag{9}$$

where $\langle x \rangle$ represents the mean of the times series. The time series $X(t)$ is then split into windows of length L, and in each of those windows, the corresponding stretch of the series is fitted by a straight line by means of least squares. These linear fits represent the local trend within each of the windows of length L. The sequence of length L trends can be concatenated in a piecewise manner defining a piecewise linear function $Y_L(t)$. Then, we compute the average fluctuations at scale L, which is the deviation from the trend, defined as follows:

$$F(L) = \left(\frac{1}{N}\sum_{t=1}^{N}(X(t) - Y_L(t))^2\right)^{\frac{1}{2}}. \tag{10}$$

The nature of the correlations present in the original time series can be evaluated by observing the dependence of $F(L)$ on L. In particular, the growth of fluctuation with the scale L will be given as $F(L) \sim L^{\alpha}$, with $0 < \alpha < 1$. In the case of an uncorrelated or short-range correlated time series x_t, we have $F(L) \sim L^{(1/2)}$. However, in the presence of long-range correlations of the power-law type in the original time series x_t, the fluctuation exponent α will differ from 1/2 [29], with $\alpha > 1/2$ for persistent (positive) correlations and $\alpha < 1/2$ for antipersistent (negative) correlations.

C. Entropy and KL-Divergence Estimation

The entropy of a symbolic sequence can be interpreted as a quantification of the degree of predictability in the sequence. A high level of predictability of consecutive values in a sequence implies a low level of surprise about future symbols, which is linked to a low entropy. On the other hand, a sequence with a high degree of randomness will be characterized by a high level of surprise in the identity of future symbols, and result in a high value for the entropy. Therefore, the determination of the entropy of language serves as a quantification of its degree of order. Early attempts to determine the entropy of language were based on the close link between entropy and predictability [30]. However, the estimation of the entropy from long sequences of written text, requires the estimation of block probabilities, which poses a serious computational challenge, due to the presence of long-range correlations in language. The required sample size needed for an accurate estimation of the block probabilities grows exponentially with the length of the block, thus quickly rendering any amount of available text insufficient. This difficulty can be overcome through the link between entropy and predictability mentioned above. The degree of predictability in a sequence determines how much it can be compressed by a lossless compression method. Sequences with high predictability can be compressed more than sequences with a higher degree of randomness. In particular, it can be shown that under general assumptions of stationarity and ergodicity, the entropy rate of a stochastic source is a lower bound to the length per symbol of any encoding of it [31]. Hence, the entropy of symbolic sequences can be estimated by means of efficient lossless compression algorithms [32]–[34]. We estimated the entropy of long character sequences using an implementation of the algorithm proposed by Lempel and Ziv [32], [35], [36], which relies on the estimation of redundancy by looking for matches between future and past substrings in a symbolic sequence. Implementations of entropy estimation algorithms based on these methods have proved to work well for symbolic sequences even in the presence of long-range correlations as those found in language [34], [37].

The Kullback–Leibler (KL) divergence $D(P|Q)$ is a measure of relative entropy between two probability distributions P and Q [31]. When $P \equiv Q$ then the KL divergence is zero, but it takes positive values when $P \neq Q$. It can be shown that $D(P|Q)$ is a measure of the extra numbers of bits that are required to encode typical sequences with the distribution P, when using a code based on Q [31]. This interpretation suggests that $D(P|Q)$ can also be estimated using compression algorithms in which one signal is compressed using past sequences in the second signal. More specifically, the KL divergence $D(P|Q)$ can be written as [31]

$$D(P|Q) = H_P(Q) - H(P) \tag{11}$$

where $H(P)$ is the entropy of the distribution P and $H_P(Q)$ is the cross entropy between P and Q. Let us assume that two text sequences produced by the stochastic source with the probability distribution P are represented by $X = \{x_t\}_{t=1}^{N}$, and those produced by Q as $Z = \{z_t\}_{t=1}^{N}$. Then, for notational succinctness, let us write the information quantities explicitly in terms of the generated sequences, therefore, representing the KL divergence between P and Q as $D(X|Z)$. With this notation, we have $D(X|Z) = H_X(Z) - H(X)$. A compression-based algorithm proposed in [38] permits to compute the cross entropy $H_X(Z)$ based on the symbolic sequences X and Z. Then, the KL divergence is obtained by subtracting the entropy of the sequence X from the cross entropy $H_X(Y)$. The KL divergence is a nonsymmetric quantity, and in order to have a distance-like measure between character sequences X and Y, we defined a symmetrized divergence as $D_s(X, Y) = (D(X|Y) + D(Y|X))/2$.

Another measure that is strictly related to entropy, and that is widely used for the evaluation of artificial texts, is perplexity [39], which can be computed as the geometric mean of the inverse probability for each predicted word in the document [40], [41], where probabilities are typically estimated on a language model trained on a larger corpus.

D. Creativity and Authorship Attribution

Providing a quantitative method that is able to address the creativity of a given algorithm for artificial texts generation is a complex task. In this paper, we consider two distinct but strictly intertwined aspects. We aim to measure at what extent the algorithm is capable of capturing the stylistic traits of a given author while at the same time avoiding to perform just a plagiarism of the training corpus.

Measuring the LCS is one of the simplest ways to implement the idea of quantitatively measuring plagiarism: given the kth character x_k of the artificial text, we denote by L_k the length of the longest subsequence starting at x_k that is also contained in (thus, plagiarized from) the training corpus. Different statistics of the set of all L_k can be used to quantify how various algorithms are able to reproduce some stylist traits of a given corpus while generating innovative texts, not written before. Here, we adopt the simplest one and consider the maximum over the whole artificial text: $\bar{L} = \max_k L_k$ (see [42] and [43]).

We now move our analysis to a higher level by exploring how artificial texts resemble the style of the training author. Stylistic traits are supposed to reflect subtle choices of the author in terms of vocabulary, syntactic constructions, and structural composition, among others. As such, a comprehensive quantification of the style of an author is out of reach. On the other hand, a very simple feature such as the frequency distribution of n-gram of letters has been successfully selected as a key ingredient in some of the most effective approaches to authorship attribution [44], [45].

We use one of the state-of-the-art algorithms to test the automatic attribution of the author of our artificial texts. The implemented method is, in fact, one of the two methods that have been successfully used for the attribution of Antonio Gramsci's papers [44]. Essentially, each method defines a kind of similarity distance between texts. Let us briefly describe the first method used here (see [45] for further details). The method is based on (characters) n-grams and it is probably one of the simplest possible measures on a text: after a first experiment based on bigram frequencies by Bennett [46]. Kešelj *et al.* [47] published a paper in which n-gram frequencies were used to define a similarity distance between texts (see also [48]). The similarity distance was introduced and discussed in [44]: we call ω an arbitrary n-gram, and we denote $f_X(\omega)$ and $f_Y(\omega)$ as the relative frequencies with which ω occurs in text X and Y. $D_n(X)$ is the n-gram dictionary of text X, that is, the set of all n-grams that have nonzero frequency in X (similarly for Y) and we define what we will call the n-gram distance between text X and text Y as[1]

$$d_n(X, Y) := \frac{1}{|Z|} \sum_{\omega \in D_n(X) \cup D_n(Y)} \left(\frac{f_X(\omega) - f_Y(\omega)}{f_X(\omega) + f_Y(\omega)} \right)^2 \quad (12)$$

where the denominator $|Z| = |D_n(X)| + |D_n(Y)|$ is the sum of the number of different n-grams in the two dictionaries while the inner sum is taken over all different n-grams occurring in the two texts.

Suppose the goal is to decide whether a given document X has been authored by author A or not. The approach adopted in [44] consists of first collecting m documents from author A and m documents from an author B (or, in general, from more authors). The distance of the candidate text X to these documents is then used, with the help of a simple probabilistic method, to produce a similarity index $I(X)$, $-1 \le I(X) \le 1$ (see [44] for more details). Values of the index close to 1 (respectively, -1) reveal a very strong attribution to author A (respectively, B), whereas values close to 0 indicate a very weak attribution (see [44] and [45] for more details).

IV. Corpora

Our aim was to train an LSTM network on a large corpus obtained from a single author in order to also perform the experiments on authorship attribution and to assess whether the model was capable of capturing some stylistic features of its "teacher." We employed the works by Charles Dickens, since he was a very prolific author whose bibliography is freely available in plain text format at ProjectGutenberg.[2] We collected a total of 18 works by Charles Dickens, which resulted in a corpus of over 24 million characters.[3]

For the authorship attribution experiments, we also collected a smaller corpus of texts, some still from Charles Dickens, and some others from a different author. Clearly, these additional texts from Dickens needed to be disjoint from those in the larger corpus, on which the LSTM network had to be trained.[4] Regarding non-Dickens documents, we collected texts by Robert Louis Stevenson, who was also a prolific author of the 19th century.[5] For this second corpus, we collected 30 documents both for Dickens and Stevenson, each consisting of 10 000 characters.

V. Experiments

The experiments with LSTMs were run using the torch-rnn package.[6] We trained an LSTM network with two layers and 1024 cells in each layer. As customary in text-generation experiments with LSTMs [13], the training set was split into chunks of length equal to K characters. In this way, gradients in truncated backpropagation are propagated up to K steps back, but the status of each LSTM cell is not reset after each example, so that longer range correlations can, in principle, be learned. We first present results obtained using $K = 100$, later investigating the impact of such hyperparameter on the considered evaluation metrics. To avoid overfitting, a dropout equal to 0.7 was applied, and a validation set (4% of the whole corpus) was exploited to monitor the loss function during training.

[1]To be more precise, d_n is a pseudodistance because it does not satisfy the triangle inequality and it is not even positive definite: two texts X, Y can be at distance $d_n(X, Y) = 0$ without being the same.

[2]http://gutenberg.org

[3]We used the following works: A Tale of *Two Cities*, textitDavid Copperfield, textitOliver Twist, textitBleak House, textitGreat Expectations, textitThe Life and Adventures of Nicholas Nickleby, textitThe Old Curiosity Shop, textitThe Pickwick Papers, textitDombey and Son, textitLittle Dorrit, textitLife and Adventures of Martin Chuzzlewit, textitOur Mutual Friend, textitBarnaby Rudge, textitA Christmas Carol, textitThe Uncommercial Traveller, textitHard Times, textitLetters, textitA Child's History of England.

[4]We used excerpts from the following works: *Signal-Man*, textitA Christmas Tree, textitThe Poor Relation's Story, textitThe Schoolboy's Story, textitHunted Down, textitPictures From Italy, textitThe Chimes, textitThe Haunted Man and the Ghost's Bargain, textitTom Tiddler's Ground, textitThe Wreck of the Golden Mary.

[5]We used excerpts from the following works: *Treasure Island*, *The Strange Case of Doctor Jekyll and Mister Hyde*, *Kidnapped*, *The Black Arrow*.

[6]https://github.com/jcjohnson/torch-rnn

TABLE I
SOME SAMPLES GENERATED BY LSTM, AS A FUNCTION OF THE TEMPERATURE T

T	LSTM generated text
0.1	I had no doubt that I had no doubt I had no sooner said that I had no sooner said that I had no sooner said that I was a stranger to me to see him
0.3	I will not see him as I have been the family of my heart and expense, and I should have seen him so soon.
0.5	'I am sure I think it is,' said the doctor, looking at him at his heart on the window, and set him there for a short time, 'that I shall find the girl there.
0.6	The old man entered the other side, and then ascended the key on his shoulder. 'I think I have no doubt, sir,' replied the woman.
0.7	'I can refer to the world,' said Mr. Tupman, suspiciously, 'that the same brother was on the provoked passage.'
0.8	You look at me, you know I dont think it should be what I have understood. I know what she has of no confidence, and I have first cheered you.
0.9	And so the position and paper escaped you by the Major, with the neck, by my own evening, that there was a shadow of it.
1.0	This interference of point that was always large in the evening, which was, and used to save the crooked way, and could leave the undertakers upon it.
1.2	Softly. They rose and who faithfully as stalled off, and his distrust in the tip of which power it was.
1.5	As if us, she Immoodnished Mrs Jipe Town horsemaking, like the nights foldans with mid-yoUge false. Half-up?
2.0	Connursing, visibly; brassbling, on what cohn; orPertixwerkliss issuin'p). haf-pihy; and he-carse masls anycori; nod me: full, nor cur your two yellmoteg'
3.0	'Siceday; Quaky ok, ,-GNIRRZVRIIMoklheHw, eab-mo,'_ ventvedes.r.' Egg_iglazzro!wM. 'Sav nam. Ebb-_Edjaevi."

TABLE II
SOME SAMPLES GENERATED WITH MARKOV MODELS OF INCREASING ORDER

order	Markov generated text
2	'And Mr Butentime ther foreweemair Masperf torto lit, 'It make to to yesee!" shis thed to goin blie, thave com of a dess at's mand havestroult frot own: ady. Saint
4	But the Nation, and you in looking at all,' said Mrs. Kenwigs's hospite as I have gover with now, 'and the eyes. Perhaps, and even to help, I am sure loving her,
6	Sam eyed Oliver, with some to her motion of the ladies, which at no immense mob gathering which the after all the adorned to nod it, Sir!' said Fledgeby
8	The foot of the medical manner, "Jenny saw, asserting to the large majority, just outside as he had hardly achieved to be induced the room.
10	I endeavour to get so precious contents, handed downstairs with pleasure the great Constitutions. It happened to him like a man, and the fallacy of human being accepted lover of the tide
12	Mr Witherden the notary, too, regarded him; with what seemed to bear reference to the friar, taking this, as it served to divert his attention was diverted by any artifice.

To compare the statistics of the LSTM texts with that of other nontrivial models, we used a plain m-order Markov process as a baseline. A family of m-order Markov models were trained from our corpus, with m taking values 2, 4, 6, 8, 10, and 12. The models were used to generate artificial texts starting from a seed taken from the original corpus.

Tables I and II show some examples of texts generated with different temperatures from the LSTM, and by different Markov models, respectively. The whole set of generated samples is publicly available at http://agentgroup.unimore.it/Lippi/generated_samples_Dickens.zip.

A. Zipf's and Heaps' Laws

As a first test comparing the statistics of the LSTM-generated texts to other models capable of rendering stochastic versions of texts, we looked and the distribution of words frequencies. We first evaluated the Zipf's law, by measuring the relationship between the rank in the set of words, ordered by frequency, and word frequency itself. Fig. 2(a) shows the rank-frequency distributions of words in the LSTM texts for temperatures in the range 0.1–3.0. The plot also shows, in black, the result for the original Dickens corpus. We fitted the value of the exponent within the region between ranks 10^2 and 10^3 to determine more clearly the behavior with temperature, which showed a clear power-law behavior consistently across all but the two extreme temperatures. Fig. 2(b) shows the resulting value of the Zipf's exponent β as a function of temperature in the range 0.3–2.0. LSTM texts generated with low temperatures have a frequency rank distribution, which decays faster with rank. On the contrary, for higher temperatures, the distribution flattens showing a smaller exponent β. The dashed line in the figure shows the value of the exponent estimated from Dickens' corpus, which intercepts the LSTM results at a temperature approximately between 0.8 and 0.9.

A similar analysis was performed on the Markov-generated texts. Fig. 2(c) shows the rank-frequency distribution for texts generated with Markov models. Interestingly, there is little variation of the distribution as a function of Markov order. This is also corroborated by the estimation of the

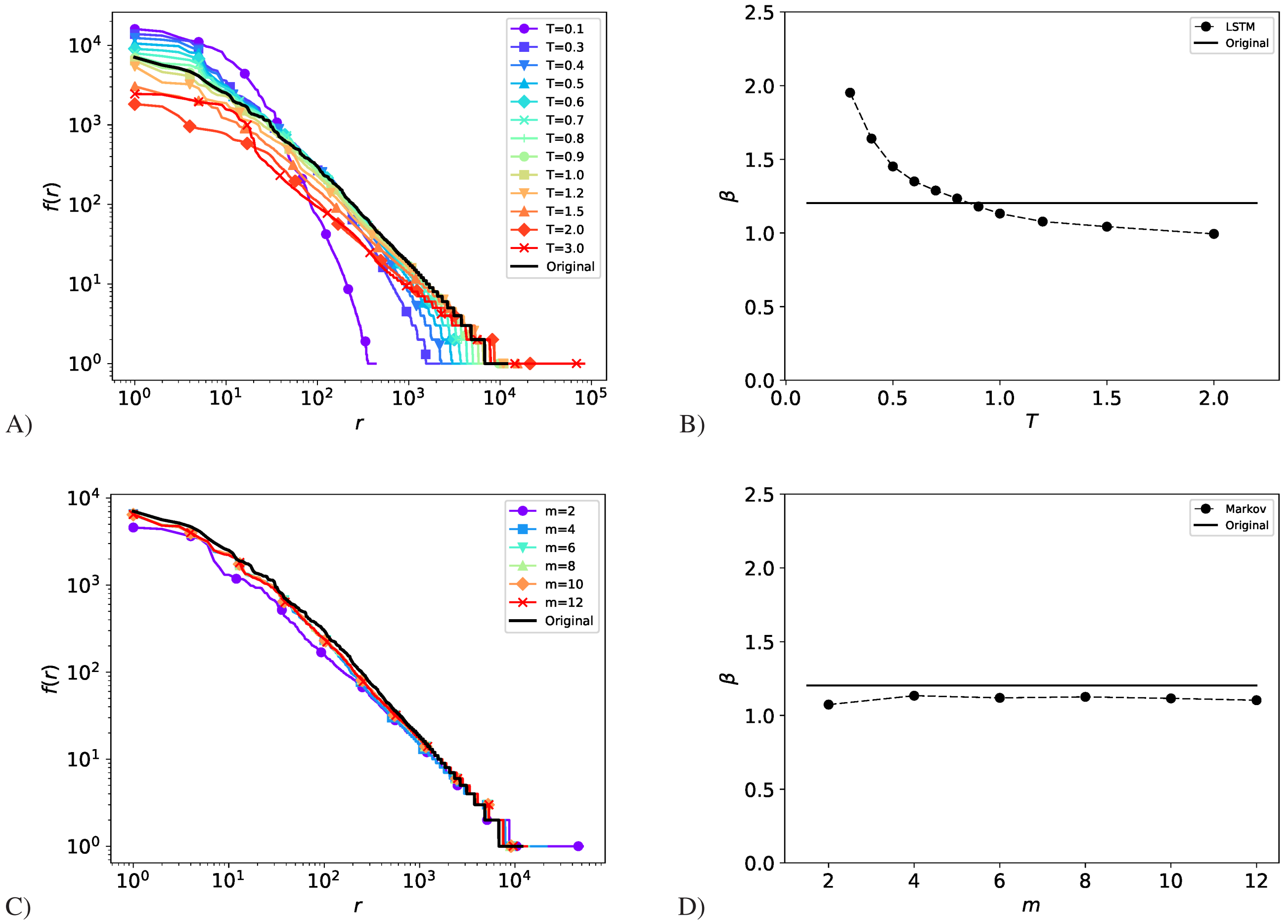

Fig. 2. Zipf's law. (a)–(c) Zipf's rank-frequency distribution in LSTM (respectively, Markov) texts, with temperature T in the range 0.1——3.0 (respectively, order m in the range 2——12). (b)–(d) Exponent β measured in the region between ranks 10^2 and 10^3, as a function of T (LSTM) or m (Markov), with dashed line corresponding to the exponent measured in the original corpus. For readability, (a)–(c) do not show markers for every point. Figure best seen in colors.

exponent [Fig. 2(d)] that shows the exponent of Markov texts with a value slightly below that of the original source.

In order to assess the validity of Heaps' law for the artificial language, we computed the statistics of the vocabulary growth for the LSTM and Markov texts. Fig. 3(a) shows the results for the LSTM texts for a range of temperatures from 0.1 to 3.0. For comparison, the curve for the original text is shown in black. Fig. 2(b) represents the results of fitting the power-law behavior in the region for $t > 1000$, using the same range of temperatures shown for the β exponent in Fig. 2(b) The dashed line, representing results for the original text, cuts the LSTM data at a point between temperatures 0.8 and 0.9.

B. Long-Range Correlations

Beyond the statistics of word frequencies, natural texts show correlations that span hundreds or thousands of characters, showing power-law decay. Although a direct measure of these correlations is possible, in principle, more efficient methods are based on spectral [27] or fluctuation [28], [29] properties of the sequences. In particular, we tested the LSTM-generated texts using DFA [28], [29] applied to linguistic character sequences. By means of DFA, it is possible to estimate such exponent α, by fitting a power law to the dependence of the fluctuations $F(L)$ with the scale L. We estimated the dependence of $F(L)$ with the scale L for LSTM texts generated at different temperatures and for the Markov-generated texts for different orders. Fig. 4(a) shows the normalized fluctuations $F_n(L)$ as a function of the scale L for LSTM-generated texts together with the results for the original Dickens' corpus and a shuffled realization of it. In all cases, there are signs of power-law behavior for a wide range of scales, with the exception of the two extreme temperatures (0.1 and 3.0). Although the data corresponding to the shuffled text seems to have a less steep slope than the LSTM samples, a more quantitative analysis is required to compare the extent of correlations. Fig. 4(b) shows the estimations of the exponent α obtained by fitting a power law to the data in (a) in the region spanned by scales $L = 10^2 \dots 10^4$. For comparison, the full black line corresponds to the result for the original Dickens' text while the dashed line is the result obtained from the shuffled text. The full black circles are the average of the estimation of the exponent from the 10 available samples, whereas small squares show the result for each sample. It is in the region of temperatures between 0.5 and 1.0 that the exponent α is close to the value for the original text. Yet, although the dispersion of individual sample measures is very broadly spread for

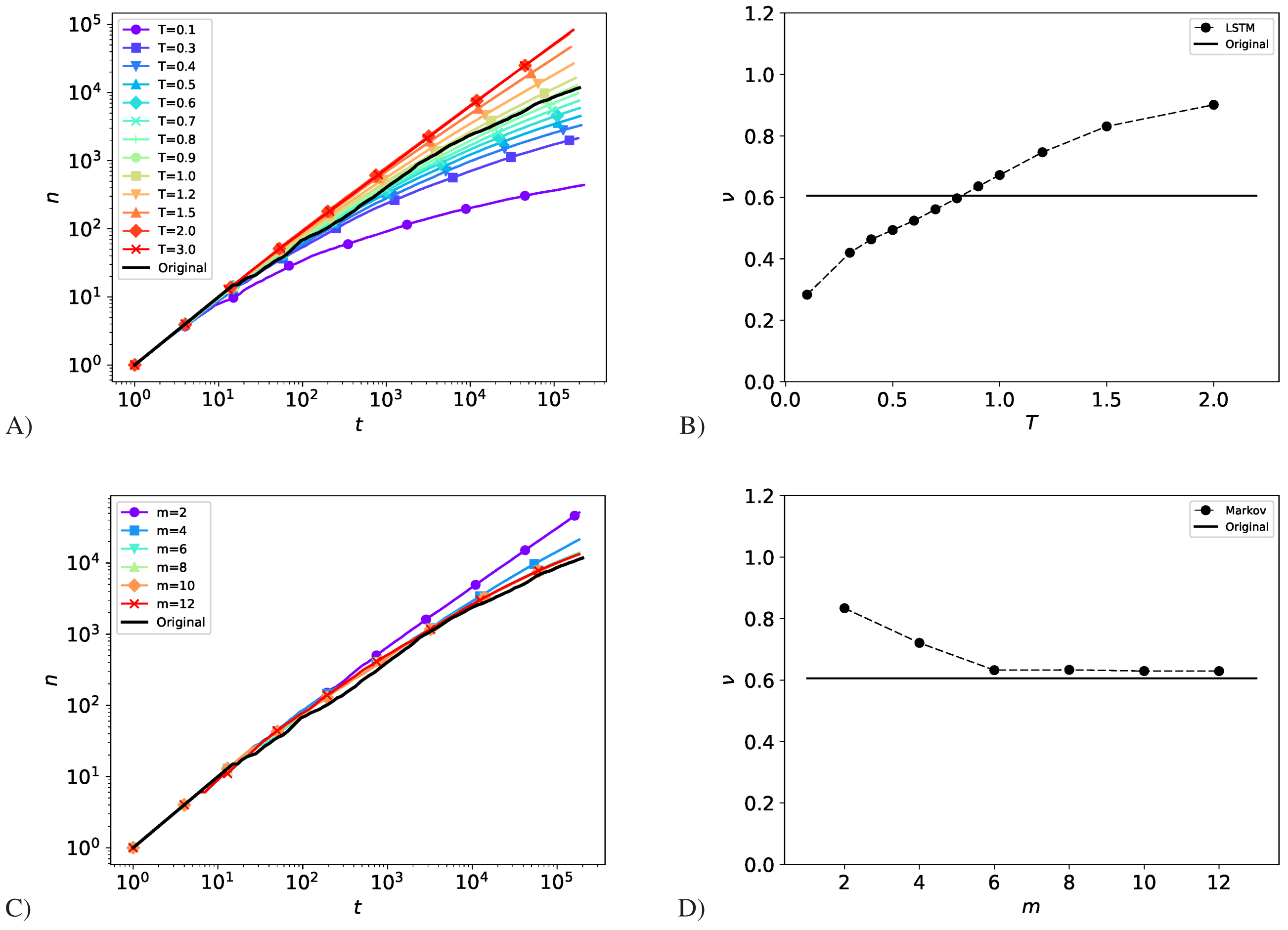

Fig. 3. Heaps' law. (a)–(c) Heaps' growth curve for the LSTM (respectively, Markov) texts, for temperatures in the range 0.1−−3.0 (respectively, order in range 2−−12). (b)–(d) Exponent ν measured in the $t > 1000$ region of the Heaps' curve for LSTM (respectively, Markov) texts as a function of the temperature (respectively, order m). In panels (b) and (d), the dashed line corresponds to the exponent for the Dickens' corpus. Figure best seen in colors.

temperatures close to 0.5, they are tightly clustered around the mean for $T = 1.0$. A similar analysis was done on the Markov-generated texts. Fig. 4(c) shows the result of the DFA for Markov texts of orders 2–12, and a comparison to the original and shuffled texts. Clearly, all the Markov-generated texts have a structural correlation that resembles more the shuffled text than the original one. This trend is corroborated in Fig. 4(d), showing the estimated exponent α as a function of Markov order. In all cases, such value is very close to that obtained by the shuffled text.

C. Entropy

Our final test to probe the statistical structure of the LSTM texts consisted in the estimation of entropy measures. The first estimation corresponded to a symmetrized KL divergence (see Section III) between the LSTM for different temperatures and original text, estimated by means of compression algorithms that are sensitive to the long-range structure of the signal. Fig. 5(a) shows the divergence as a function of T. There is a well-defined minimum at $T \sim 1$, indicating at that temperature the structure of the LSTM text is closest to the original. Fig. 5(c) shows a similar plot for Markov texts, which also approximate the structure of the original texts asymptotically for larger orders. Although the previous analyses showed the behavior of a distancelike quantity, the entropy is a more direct estimation of the statistical structure of a signal. Fig. 5(b) and (d) shows the value of the entropy for the LSTM and Markov texts as a function of the temperature and order, respectively. In both cases, the solid line corresponds to the value of the entropy of the original text. For LSTM, lower T values produce texts with small entropy, thus easier to compress, since they show repetitive patterns. As T grows, entropy also grows, with the texts becoming the most similar to the original around $T \sim 0.9$ (in line with our analyses) and far more disordered for larger T values. Markov texts, instead, show a monotonic approximation to the entropy of the original text, slightly above even for higher orders.

To complete the analysis, we also computed the perplexity of LSTM-generated texts, using the `KenLM` library.[7] To do this, we learned a bigram language model on the original Dickens' corpus and then we computed the perplexity both for the artificial texts, and for the off-sample Dickens' documents in the test corpus used for authorship attribution. Results are perfectly in line with those obtained with entropy computation

[7]https://github.com/kpu/kenlm

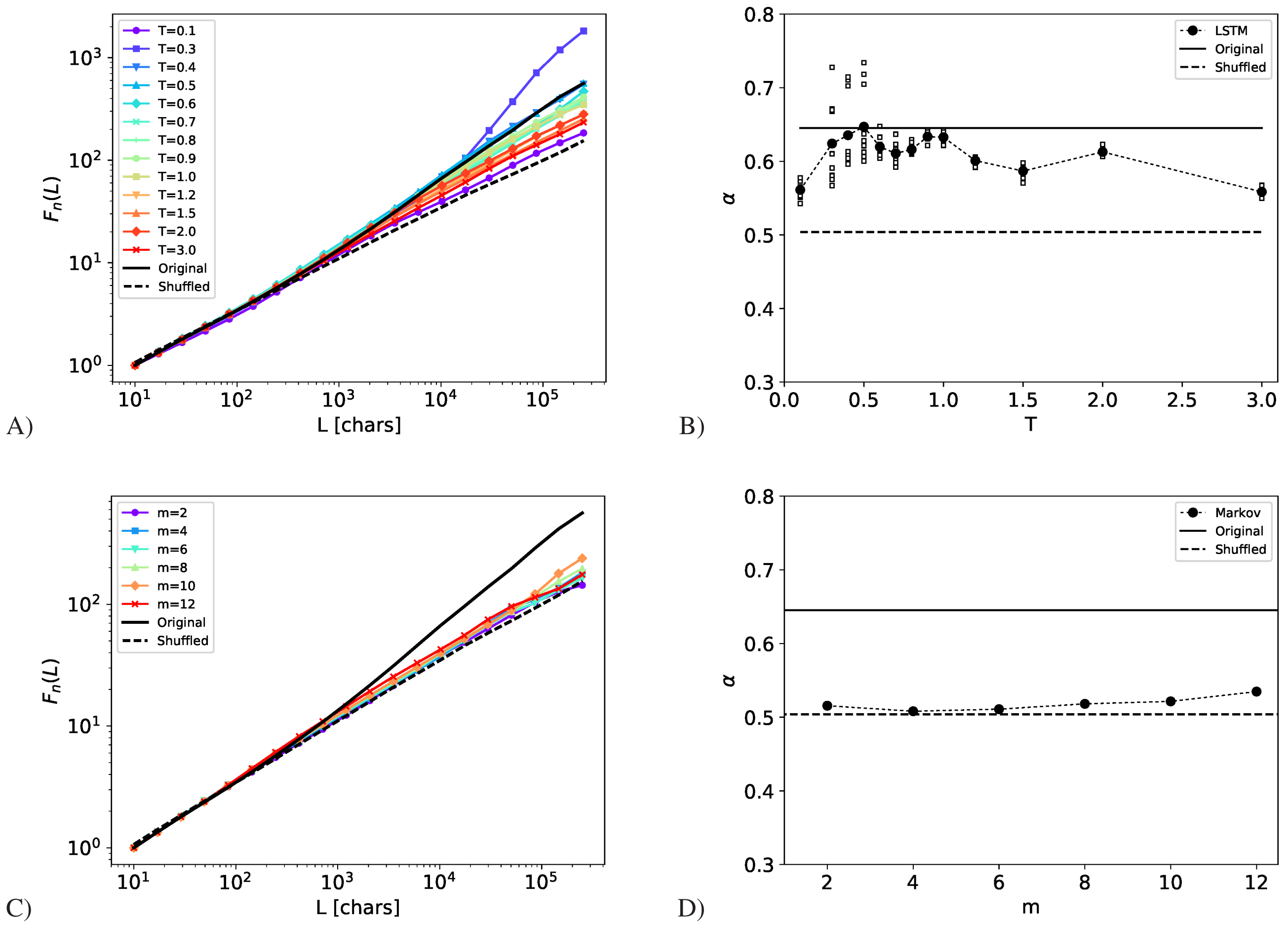

Fig. 4. Long-range correlations. (a)–(c) Normalized fluctuations $F_n(L)$ as a function of the scale L for LSTM (respectively, Markov) texts. (b)–(d) Fluctuation exponent α obtained by fitting a power law the LSTM and Markov data in (a) and (c), in the range $L = [10^2, 10^4]$. Black circles: mean over 10 samples. Small empty squares: estimations for individual samples. Full (dashed) black line: real (shuffled) text.

via compression, with values of temperature around 0.9–1.0 producing texts that are the most similar to the original.

D. Creativity and Authorship Attribution

To assess the degree of plagiarism (and, thus, of creativity) for our models, we measured the length of the LCS between the artificial texts and the training corpus. For Markov texts, not surprisingly the length of the LCS rapidly grows with the order of the model, from a value of 19 for order 2, up to 73 and 125 for orders 10 and 12, respectively. For LSTMs, the length of the LCS instead results to be lower, with values comprised between 40 and 60 when T is in the range between 0.4 and 1.2. For smaller and larger temperatures, the LCS is clearly shorter, as the generated texts have a less realistic structure, thus it is less probable to encounter patterns that are identical to the original. As a comparison with what we could call a sort of self-plagiarism, encountered in a subset of the training corpus, we also computed the length of the LCS of the novel "Oliver Twist" with respect to five other novels by Dickens (*David Copperfield*, *A Tale of Two Cities*, *Bleak House*, *Great Expectations*, and *Hard Times*): the LCS in this case resulted to be 45 characters long.

For the experiments on authorship attribution, we used the Dickens and Stevenson corpora described in Section IV. For a fixed temperature of the generated text, attribution is performed by averaging over 10 samples of artificial texts, using n-grams from $n = 1$ up to $n = 12$. To assess the validity of the attribution algorithm, we compare the attribution of the generated texts with that of real texts. In particular, for each of the 30 Dickens and Stevenson documents, we employ the remaining 29 documents of each group to perform attribution to either author. Fig. 6 shows the value of the index $I(X)$ defined in Section III-D as a function of n-grams (on the x-axis) and either temperature for LSTM (a) or order for Markov texts (b), respectively. The attribution of real texts performs extremely well, with a maximum for $n = 6$ and $n = 10$ for Dickens and Stevenson, respectively. For LSTM, although the texts are indeed always correctly attributed to Dickens, it is interesting to note how the best results are again achieved for values of T around 0.8–1.0, whereas for lower temperatures, the attribution with larger n-grams drops, as the generated texts tend to reproduce periodic (hence, less realistic) patterns. Not surprisingly, Markov texts with $m \geq 6$ are also well attributed to Dickens, due to the high degree of plagiarism described in Section V-D.

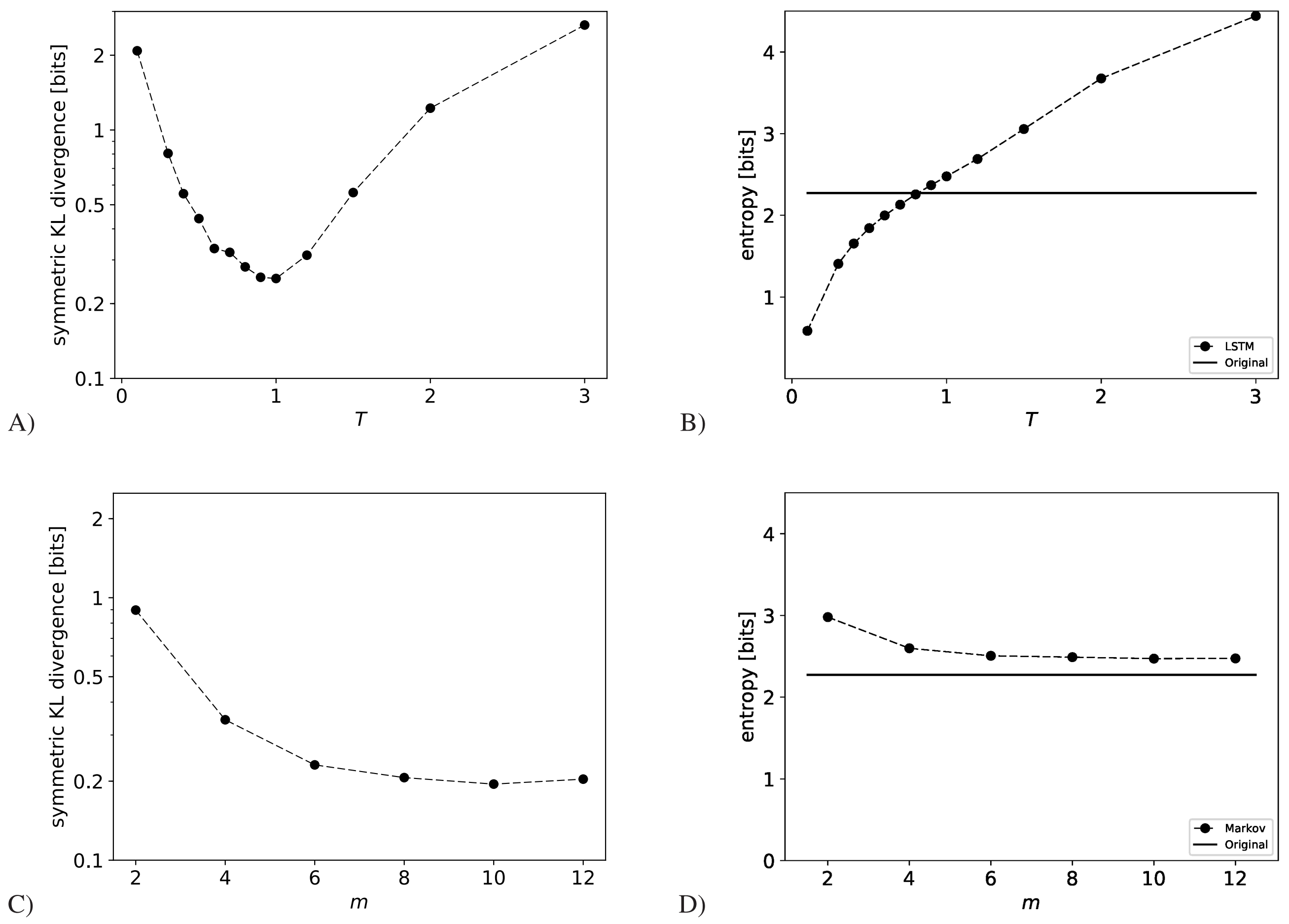

Fig. 5. Divergence and entropy. Symmetrized KL divergence and entropy, as a function of temperature for the LSTM texts [(a) and (b)], and as a function of the order for the Markov-generated texts [(c) and (d)]. In (b) and (D), the solid line represents the entropy of the original text.

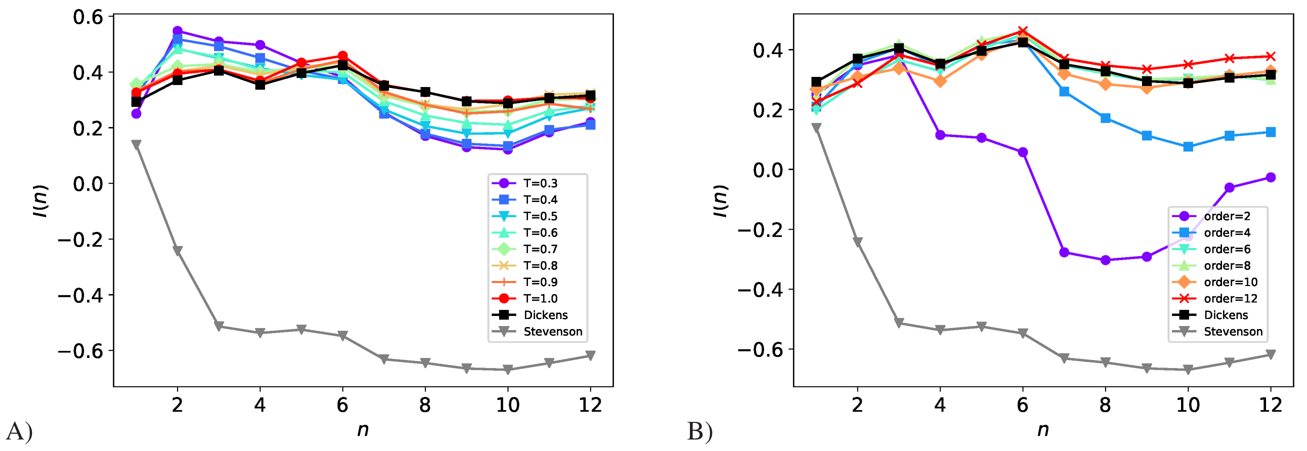

Fig. 6. Authorship attribution for (a) LSTM and (b) Markov texts. We show the attribution index $I(X)$ for a text X as a function of n-grams and as a function of T (for LSTM) and m (for Markov models). Indices closer to 1 (respectively, -1) indicate stronger attributions to Dickens (respectively, Stevenson).

E. Impact of K Hyperparameter

The results presented so far consistently show that there is a narrow range of temperatures around $T = 1$ for which the texts generated by LSTM are the most similar to the original, in terms of all the considered metrics. We finally investigated the impact of K hyperparameter on the generated texts. Given that K is the number of backward gradient propagation steps in the BPTT algorithm, it directly affects the way in which long-range information is propagated through the cells. Therefore, we considered two additional LSTM networks, trained with $K = 10$ and $K = 1000$, respectively, and identical to the first network for all the other settings. For all the considered metrics apart from long-range correlations, results are not significantly different from those obtained for the network

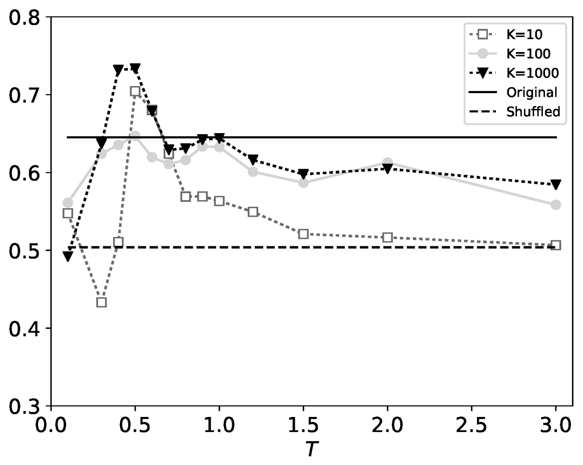

Fig. 7. Exponent of DFA as a function of T, for different values of K.

trained with $K = 100$ (see supplementary material for details). Long-range correlations are instead more significantly affected by the variation in K. In Fig. 7, we observe two different phenomena: 1) a peak for low-level temperatures, likely due to spurious, periodic patterns, that is visible also for $K = 100$ (also note the large variance across samples shown in Fig. 4) and 2) a dramatic drop in the exponent α for $K = 10$ starting from $T = 0.7$, whereas for $K = 1000$, the exponent is much closer to that of real text.

VI. Conclusion

In this paper, we investigated at what degree texts generated by an LSTM network resemble texts generated by humans. To this aim, we presented an extensive experimental evaluation where we compared several characteristics of artificial and original texts, starting from statistical properties typically shown by natural language, such as the distribution of word frequencies and long-range correlations, up to higher-level analyses, such as the attribution of authorship. This paper shows that LSTM-generated texts share key statistical features with natural language. In particular, the experimental results highlight the crucial role of the temperature parameter in producing texts that resemble those created by humans in their statistical structure, with an optimal range of temperatures, around $T = 1$, that induce the highest degree of similarity. Interestingly, we also illustrated how a network trained on a single-author corpus can produce texts that are attributed to that author, according to authorship attribution algorithms.

The presented study suggests many interesting research directions which we plan to investigate in future works.

First, we aim to compare the semantic information of original and artificial texts. It is clear from the samples shown in the experimental section that LSTM texts are still far from human-generated texts in terms of meaningfulness, although showing similar statistical properties. It is thus possible that certain correlates of semantic information, such as burstiness and clustering of keywords, are reflected in LSTM texts. Given that even the origin of long-range correlations in natural language is still debated, our work paves the way to deeper future investigations in this direction.

Second, we aim to extend the analysis of these statistical properties of the LSTM texts to different languages, in order to assess whether there are some languages that are easier or more difficult to reproduce for a machine.

Finally, we aim to extend the study on authorship attribution and plagiarism, for which in this paper we only presented preliminary results. For that purpose, we plan to employ a larger corpus, in order to compare several authors, genres and languages, and to test different algorithms, for instance, those based on trainable machine learning systems [49], [50].

Acknowledgment

The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

References

[1] M. A. Boden, “Creativity and artificial intelligence,” *Artif. Intell.*, vol. 103, nos. 1–2, pp. 347–356, 1998.
[2] E. Reiter, R. Dale, and Z. Feng, *Building Natural Language Generation Systems*, vol. 33. Cambridge, MA, USA: MIT Press, 2000.
[3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, May 2015.
[4] Y. Bengio, “Learning deep architectures for AI,” *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[5] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3128–3137.
[6] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, “Sequence to sequence–video to text,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4534–4542.
[7] X. Zhang and M. Lapata, “Chinese poetry generation with recurrent neural networks,” in *Proc. EMNLP*, 2014, pp. 670–680.
[8] P. Potash, A. Romanov, and A. Rumshisky, “GhostWriter: Using an LSTM for automatic rap lyric generation,” in *Proc. EMNLP*, 2015, pp. 1919–1924.
[9] W. Ebeling and A. Neiman, “Long-range correlations between letters and sentences in texts,” *Phys. A, Stat. Mech. Appl.*, vol. 215, no. 3, pp. 233–241, 1995.
[10] M. A. Montemurro and P. Pury, “Long-range fractal correlations in literary corpora,” *Fractals*, vol. 10, no. 4, pp. 451–461, 2002.
[11] E. G. Altmann, G. Cristadoro, and M. D. Esposti, “On the origin of long-range correlations in texts,” *Proc. Nat. Acad. Sci.*, vol. 109, no. 29, pp. 11582–11587, 2012.
[12] A. Graves. (Aug. 2013). “Generating sequences with recurrent neural networks.” [Online]. Available: https://arxiv.org/abs/1308.0850
[13] A. Karpathy, J. Johnson, and L. Fei-Fei. (Jun. 2015). “Visualizing and understanding recurrent networks.” [Online]. Available: https://arxiv.org/abs/1506.02078
[14] G. K. Zipf, *The Psycho-Biology of Language—An Introduction to Dynamic Philology*. Boston, MA, USA: Houghton Mifflin Company, 1935.
[15] H. S. Heaps, *Information Retrieval, Computational and Theoretical Aspects*. New York, NY, USA: Academic Press, 1978.
[16] H. W. Lin and M. Tegmark, “Critical behavior in physics and probabilistic formal languages,” *Entropy*, vol. 19, no. 7, p. 299, 2017.
[17] S. Takahashi and K. Tanaka-Ishii, “Do neural nets learn statistical laws behind natural language?” *PLoS One*, vol. 12, no. 12, 2017, Art. no. e0189326.
[18] A. Ghodsi and J. DeNero, “An analysis of the ability of statistical language models to capture the structural properties of language,” in *Proc. 9th Int. Natural Lang. Gener. Conf.*, 2016, p. 227.
[19] B. M. Lake and M. Baroni. (2017). “Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks.” [Online]. Available: https://arxiv.org/abs/1711.00350
[20] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[21] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[22] K. Greff, R. K. Srivastava, and J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[23] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[25] W. Zaremba, I. Sutskever, and O. Vinyals. (Sep. 2014). "Recurrent neural network regularization." [Online]. Available: https://arxiv.org/abs/1409.2329

[26] M. Gerlach and E. G. Altmann, "Stochastic model for the vocabulary growth in natural languages," *Phys. Rev. X*, vol. 3, no. 2, 2013, Art. no. 021006.

[27] R. F. Voss and J. Clarke, "1/f noise in music and speech," *Nature*, vol. 258, no. 5533, pp. 317–318, 1975.

[28] C.-K. Peng *et al.*, "Long-range correlations in nucleotide sequences," *Nature*, vol. 356, no. 6365, pp. 168–170, 1992.

[29] S. V. Buldyrev *et al.*, "Long-range correlation properties of coding and noncoding DNA sequences: GenBank analysis," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 51, no. 5, p. 5084, 1995.

[30] C. E. Shannon, "Prediction and entropy of printed English," *Bell System Tech. J., The*, vol. 30, no. 1, pp. 50–64, Jan. 1951.

[31] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley, 2006.

[32] A. Lempel and J. Ziv, "On the complexity of finite sequences," *IEEE Trans. Inf. Theory*, vol. 22, no. 1, pp. 75–81, Jan. 1976.

[33] A. D. Wyner and J. Ziv, "Some asymptotic properties of the entropy of a stationary ergodic data source with applications to data compression," *IEEE Trans. Inf. Theory*, vol. 35, no. 6, pp. 1250–1258, Nov. 1989.

[34] T. Schürmann and P. Grassberger, "Entropy estimation of symbol sequences," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 6, no. 3, pp. 414–427, 1996.

[35] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theory*, vol. IT-23, no. 3, pp. 337–343, May 1977.

[36] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, Sep. 1978.

[37] I. Kontoyiannis, P. H. Algoet, YU. M. Suhov, and A. J. Wyner, "Nonparametric entropy estimation for stationary processes and random fields, with applications to English text," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 1319–1327, May 1998.

[38] J. Ziv and N. Merhav, "A measure of relative entropy between individual sequences with application to universal classification," *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1270–1279, Jul. 1993.

[39] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker, "Perplexity—A measure of the difficulty of speech recognition tasks," *J. Acoust. Soc. Amer.*, vol. 62, no. S1, p. S63, 1977.

[40] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.

[41] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3156–3164.

[42] A. Papadopoulos, F. Pachet, and P. Roy, "Generating non-plagiaristic Markov sequences with max order sampling," in *Creativity and Universality in Language*. Cham, Switzerland: Springer, 2016, pp. 85–103.

[43] F. R. P. Pachet and A. Papadopoulos, *Constrained Markov Sequence Generation: Applications to Music and Text* (Computational Synthesis and Creative Systems). Cham, Switzerland: Springer, 2018.

[44] C. Basile, D. Benedetto, E. Caglioti, and M. Degli Esposti, "An example of mathematical authorship attribution," *J. Math. Phys.*, vol. 49, no. 12, 2008, Art. no. 125211.

[45] D. Benedetto, M. D. Esposti, and G. Maspero, "The puzzle of Basil's Epistula 38: A mathematical approach to a philological problem," *J. Quant. Linguistics*, vol. 20, no. 4, pp. 267–287, 2013.

[46] W. R. Bennett, *Scientific and Engineering Problem-Solving With the Computer*. Upper Saddle River, NJ, USA: Prentice-Hall, 1976.

[47] V. Kešelj, F. Peng, N. Cercone, and C. Thomas, "N-gram-based author profiles for authorship attribution," in *Proc. PACLING*, Halifax, NS, Canada, Aug. 2003, pp. 255–264.

[48] R. Clement and D. Sharp, "Ngram and Bayesian classification of documents for topic and authorship," *Literary Linguistic Comput.*, vol. 18, no. 4, pp. 423–447, 2003.

[49] A. Neme, J. R. G. Pulido, A. Muñoz, and S. Hernández, and T. Dey, "Stylistics analysis and authorship attribution algorithms based on self-organizing maps," *Neurocomputing*, vol. 147, pp. 147–159, Jan. 2015.

[50] M. L. Jockers and D. M. Witten, "A comparative study of machine learning methods for authorship attribution," *Literary Linguistic Comput.*, vol. 25, no. 2, pp. 215–223, 2010.

Marco Lippi received the Ph.D. degree in computer and automation engineering from the University of Florence, Florence, Italy, in 2010.

He was a Post-Doctoral Fellow at the University of Florence, the University of Siena, Siena, Italy, and the University of Bologna, Bologna, Italy. He was a Visiting Scholar with Pierre and Marie Curie University (UPMC), Paris, France. He is currently an Assistant Professor with the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Modena, Italy, where he is involved in machine learning and artificial intelligence, with applications in bioinformatics, law, game playing, natural language processing, and argumentation mining.

Marcelo A. Montemurro received the Ph.D. degree in theoretical physics from the National University of Córdoba, Cordoba, Argentina, in 2002.

He then moved to Italy to work at the International Centre for Theoretical Physics in Trieste, Italy, with a fellowship from UNESCO. He was with the International Center for Theoretical Physics, Trieste, Italy. In 2004, he was with The University of Manchester, Manchester, U.K., where he was a Lecturer and held a number of fellowships. His current research interests include the statistical physics of complex systems and computational neuroscience.

Mirko Degli Esposti received the Degree in physics from the University of Bologna, Bologna, Italy, in 1988, and the Ph.D. degree in mathematics from Penn State University, State College, PA, USA, in 1994.

He is currently a Full Professor of mathematical physics with the Computer Science Department, University of Bologna. His current research interests include applications of the theory of dynamical systems and of information theory to life science and human science, especially human language.

Giampaolo Cristadoro received the Ph.D. degree in theoretical physics from the University of Insubria, Como, Italy, and the Paul Sabatier University, Toulouse, France, in 2005.

He was a Post-Doctoral Fellow with the Max Planck Institute for the Physics of Complex Systems, Dresden, Germany, with the Center for Nonlinear and Complex Systems, Como, Italy, and with the Department of Mathematics, University of Bologna, Bologna, Italy, where he was later appointed as an Assistant Professor and an Associate Professor. He is currently an Associate Professor of mathematical physics with the Department of Mathematics and Applications, University of Milano-Bicocca, Milan, Italy. His current research interests include dynamical systems, probability and information theory, with applications to life science and natural language.