LOL: An Investigation into Cybernetic Humor, or: Can Machines Laugh?*

Davide Bacciu[1], Vincenzo Gervasi[1], and Giuseppe Prencipe[1]

1 **Dipartimento di Informatica, Università di Pisa, Pisa, Italy**
`davide.bacciu@unipi.it`
2 **Dipartimento di Informatica, Università di Pisa, Pisa, Italy**
`vincenzo.gervasi@unipi.it`
3 **Dipartimento di Informatica, Università di Pisa, Pisa, Italy**
`giuseppe.prencipe@unipi.it`

Abstract

We investigate literary theories of humour from a computational point of view. A corpus of approximately 11,000 jokes is used to train a neural network generating jokes; the state space of such network is then analyzed via appropriate discovery algorithms, and abstractions synthesized by the neural network are compared to those predicted by existing theories.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, I.2.6 Learning, I.5.4 Applications

Keywords and phrases deep learning; recurrent neural networks; dimensionality reduction algorithms

Digital Object Identifier 10.4230/LIPIcs.FUN.2016.3

1 Prologue

Tony Stark: Attitude control is a little sluggish above 15,000 meters, I'm guessing icing is the probable cause.

Jarvis: A very astute observation, sir. Perhaps, if you intend to visit other planets, we should improve the exosystems.

Tony Stark: When was it that I programmed into you such a poor sense of humor? I don't recall ever doing it.

Jarvis: Sir, in fact you did not. I learned it myself from monitoring your conversations. I would not judge it poor, it is rather on a par with your own. And now, with your permission, I will retire. Have to see a lady tonight.

Tony Stark: Again? Come on, Jarvis. You cannot *really* have an affair with Siri.

Tony Stark: Jarvis?

Tony Stark: JARVIS?

2 Introduction

The mechanisms of humour have been the subject of much study and investigation, starting with [1] and up to our days. Much of this work is based on *literary* theories, put forward by

* This work has been partially supported by the MIUR-SIR project LIST-IT (grant nr. RBSI14STDE).

some of the most eminent philosophers and thinkers of all times, or *medical* theories, investigating the impact of humor on brain activity or behaviour. Recent functional neuroimaging studies [4, 5], for instance, have investigated the process of comprehending and appreciating humor by examining functional activity in distinctive regions of brains stimulated by joke corpora. Yet, there is precious little work on the computational side, possibly due to the less hilarious nature of computer scientists as compared to men of letters and sawbones. In this paper, we set to investigate whether literary theories of humour can stand the test of algorithmic laughter. Or, in other words, we ask ourselves the vexed question: Can machines laugh?

We attempt to answer that question by testing whether an algorithm – namely, a neural network – can "understand" humour, and in particular whether it is possible to automatically identify abstractions that are predicted to be relevant by established literary theories about the mechanisms of humor. Notice that we do not focus here on distinguishing humorous from serious statements – a feat that is clearly way beyond the capabilities of the average human voter, not to mention the average machine – but rather on identifying the underlying mechanisms and triggers that are postulated to exist by literary theories, by verifying if similar mechanisms can be learned by machines.

3 Literary theories of humor

We have no hopes of surveying, in the limited space available, centuries of humor and laughter research, and will instead refer the reader to [2] for an extensive treatment. For the purpose of this work, we will focus on just three broad mechanisms, following [11], namely:

- *Release* theories consider humor as a psychological mechanism that triggers the suppression of inhibitions (as established by laws, social customs, etc.);
- *Incongruity* theories focus on the juxtaposition of two or more elements which are not normally associated;
- *Superiority* theories apply to the amusement that is provoked by depicting certain characters, considered of inferior social status, in ridicolous situations. These are at times called *Hostility* theories, when the jokes are particularly aggressive or belittling.

As machines are supposed to have relatively few inhibitions[1], we do not attempt to verify Release theories, and focus instead on Incongruity and (particularly) Superiority in our investigation.

Each of the mechanisms above can be exercised through a number of different *strategies*. Examples of such strategies include: the use of vulgar, pejorative or derogatory language; the use of exaggeration; the (ab)use of referential incongruity; the purposeful rejection of social norms and good manners; putting others down; using of allegories (e.g., animals in place of humans) to produce fantasy scenarios, and so on. We will not investigate specific strategies in detail here (although this would be an interesting extension for future work), but the reader may be able to recognize some of these strategies in the various examples that follow.

4 Deep Recurrent Neural Networks for character-based jokes learning

Recurrent neural networks (RNN) are a family of learning models capable of processing input information of variable-size under the form of sequential data. Generally speaking, a

[1] We expect inhibitions to become a major issue in the future [15], and postpone applications of our method to this case till that day.

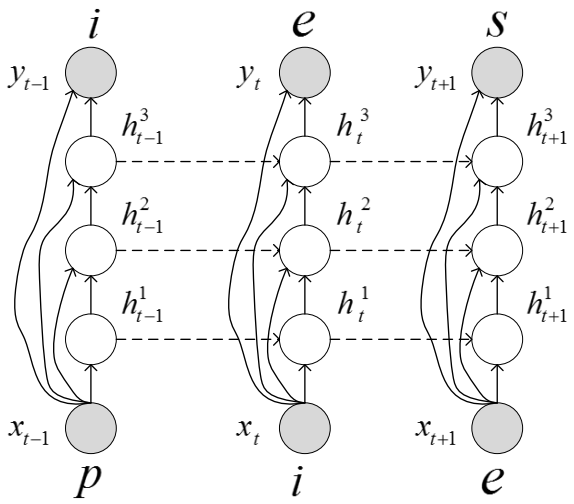

Figure 1 Unfolding over time of a deep recurrent neural network with 3 layers of hidden recursive neurons: x_t is the input at time t, h^i_t are the associated outputs of the hidden neurons in the i-th layer and y_t are the network predictions at time t. Recurrent connections are marked as dashed arrows and are unfolded over time (e.g. from time $t-1$ to time t. The graph represents a typical approach to next character prediction, where the network receives as input the current character of the sequence and should predict the following character.

neural network is a collection of basic processing units, the neurons, performing a weighted summation of their inputs followed by the application of a (often nonlinear) activation function. Such units are typically organized in a layered structure, starting with an input layer which serves to inject the input information into the network, followed by a variable number of layers of hidden units which serve to learn neural encodings of increasing complexity and terminated by a layer of output neurons which compute the learning model predictions. These units are interconnected through weighted connections which determine information exchange between neurons and layers. RNN are characterized by the presence of recurrent connections, that are links determining cycles in the network structure. The presence of such links endows the network with a memory of its past activations, allowing it to tackle learning tasks involving the processing of sequential information, such as a piece of text in natural language.

The actual structure of a RNN is determined by the unfolding over-time of the recurrent connections following the input sequence evolution over time [7]. In this paper, we focus on a deep recurrent architecture [9] entailing a variable number of recurrent layers whose structure unfolds over time and that are organized into a hierarchy, where deeper layers (i.e. closer to the output layer) are meant to extract higher level representations of the input information and sequence contexts of longer-distance. Figure 1 shows a basic view of a deep RNN with 3 recurrent layers unfolding over a text sequence: here $\mathbf{x}_t$ represents the current inputs at time t, while $\mathbf{h}^i_t$ is the vector holding the outputs of the hidden neurons of the i-th layer; finally, $\mathbf{y}_t$ are the network predictions. The unfolding highlights how the hidden state of the i-th layer depends on the current output of the preceding layer $\mathbf{h}^{i-1}_t$, on the current input $\mathbf{x}_t$ as well as on its output at time t, i.e. $\mathbf{h}^i_{t-1}$.

An hidden layer is typically realized by a collection of simple recurrent units which perform a weighted summation of their inputs followed by a nonlinear activation function,

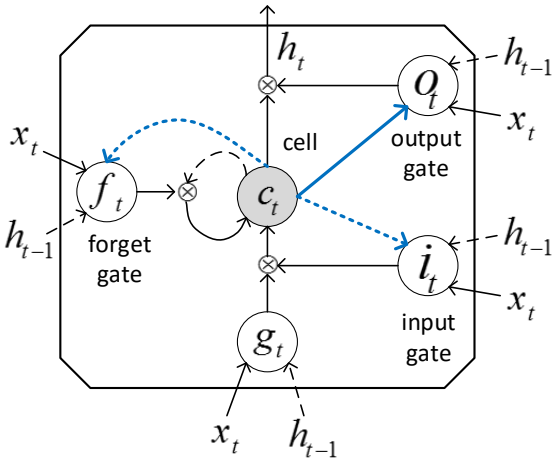

Figure 2 Memory cell of a Long Short Term Memory: recurrent connections are represented as dashed arrows, while continuous lines denote feedforward connections.

also referred to as squashing function, such as a logistic sigmoid. These RNN architectures are subject to a phenomenon, known as *vanishing gradient* [3], which limits their ability to capture long-term dependencies between elements of the input sequence. Simply put, the gradient of the error used to train the network parameters tends to be large for short term corrections while it annihilates on the long term. The Long Short Term Memory (LSTM) [12] is a RNN architecture that has been proposed to tackle the vanishing gradient problem and to learn sequential tasks with long term dependencies. The LSTM is based on a more articulated form of recurrent unit, known as the memory cell, whose structure is summarized in Figure 2. At the core of this structure sits the cell, that is a unit whose output c_t is computed as in the standard recurrent neuron by input summation followed by squashing. The role of this cell is, as in the simple recurrent neuron, to maintain a memory of the history of the input signals to the network. Differently form standard RNN, access to this memory cell is guarded by a number of gating units which regulate what inputs are allowed to influence the activation of the cell (input gate), when the memory cell is going to provide an output (output gate) and when the cell should reset its state (forget gate). Such gating units all receive inputs from feedforward and recurrent connections, performing the usual weighted summation followed by a sigmoid activation function, producing outputs i_t, o_t and $f_t \in [0, 1]$.

Several LSTM versions exists with different gating units and internal connectivity: in this paper, we refer to the, so called, Vanilla LSTM [10] in Figure 2 which, among the others, include *peephole* connections that allow the cell to control the activation of the gating units (thick blue arrows in Figure 2). Each unit in the memory cell is associated to a weight matrix of the coefficients of the weighted summation, that are the neural network parameters adapted through the learning process. Here, these are learned by the Back Propagation Through Time (BPTT) algorithm described in [10].

The learning task addressed in this paper is the one-character ahead prediction from a text sequence represented in Figure 1. The network architecture comprises multiple hidden recurrent layers of LSTM cells, while the current input character is represented by a 1-of-K encoding where $\mathbf{x}_t$ is a K-dimensional vector with the k-th entry set to one if the current

input is the k-th character of the alphabet. Similarly, the network is trained to output a K-dimensional vector $\mathbf{y}_t$ with the k' feature set to 1 if the next character is the k' element of the alphabet and it is 0 otherwise. In practice, the outputs of the network will be values close to 1 if the corresponding characters are likely to be the next ones in the sequence, otherwise we will expect them to be closer to 0. The network outputs can thus be transformed in an estimate of the posterior probability of the next character being the k-th (having observed current character x_t) by means of a softmax

$$P(y_t = k|x_t) = \frac{\exp y_t^k}{\sum_{k'=1}^{K} y_t^{k'}},$$

where y_t^k is the activation of the k-th output neuron at time time t (i.e. element k of vector $\mathbf{y}_t$). Following the generative scheme in [9], the most probable character y_t (according to the predicted posterior $P(y_t|x_t)$) can then be fed back as next input x_{t+1}, allowing a trained network to generate sequences of whatever length that respect the *linguistic* structure of the text used to train it. In Section 5, we apply an implementation of this character-level deep LSTM[2] to a dataset of English jokes with the intent of studying the neural representation emerging from training on such a corpus and if this can be related somehow with the linguistic theories of jokes. By approaching the problem from a character-based perspective, we believe that we strengthen the computational model's ability to extract a representation of those aspects of the joke literature that are associated with letter substitutions and consonances, such as with play-on-word humour. Such aspects would be lost, if adopting more monolithic approaches, e.g. representing text on a word level.

5 Learning from jokes

We can now apply the theory presented above to our goal, namely: testing in an objective, computational way whether the phenomena predicted by the literary theory on jokes can be automatically identified in real jokes.

5.1 Experimental setup

We consider a corpus of 10942 English jokes collected by [8] for the purpose of assessing a joke retrieval system and, in particular, the ability in recognizing if different texts are essentially "telling" the same joke with different words. The original dataset[3] has been annotated by the authors of [8] using 10 semantic categories (animal, number, color, organization, currency, person, location, time/date, music and vehicle), which have been stripped off in our version of the dataset. For the purpose of our analysis, it is interesting to note that the dataset is quite heterogeneous, containing jokes of different length and rhetorical structures. For instance, it includes a wide selection of short question-answering jokes, stereotyped situations (e.g. number of professionals required to change a lightbulb) and recurring characters (e.g. pirates, stupid Mama's, doctors). The dataset has been formatted by introducing a newpage (formfeed) symbol at the end of each joke to support the RNN in learning joke separation: note that the full corpus appears to the network as a unique sequence of concatenated jokes.

[2] Source code available on Github: `https://github.com/karpathy/char-rnn`
[3] `https://people.cs.umass.edu/~lfriedl/code/JokesCorpus-txt.tgz`

We have trained the character-level LSTM described in Section 4 using, mostly, the standard hyperparameter configuration bundled with the software. The objective of this analysis, in fact, is not the optimization of the predictive performance of the model but rather an explorative analysis of the learned neural encoding of jokes. In this respect, we have explored some alternative network configurations which might had an effect on the development of different neural representations. In particular, we have considered network configurations comprising 3 or 4 layers of LSTM cells to see if a deeper network was capable of capturing more complex contexts and concepts in the additional layer. Similarly, we have experimented by varying the length of the sequential context that was allowed to influence parameter learning, controlled by a number of hyperparameters such as the length of network unfolding in BPTT learning. Training of the different network configurations has been performed using 95% of the available data as training set and the remaining 5% as a validation set to select the best network for analysis. The number of training epochs (i.e. how many times the learning model "sees" the full training data) has been set to 50 and jokes were randomly shuffled in order to prevent formation of any artificial bias resulting from a fixed joke order (e.g. the fact that a question-answering joke is typically followed by another of the same type). The network ultimately selected for the following analysis is the one achieving the lowest validation error and comprises 3 hidden layers of 512 LSTM neurons, for a total of 560K parameters.

5.2 Jokes generation

A trained neural network can be used for recognizing patterns as well as for generating sequences that present the same patterns as in the training set. Hence, our LSTM network can be used to *generate* new jokes. This is obtained by *priming* the network (i.e., by providing a prefix sequence, which may even be just random) and then sampling the most probable next character from the network. This character is then added to the sequence, provided as new input, and the process is iterated thus generating whole jokes.

We must admit that our network would not be particularly popular at a party (except, maybe, if the party was thrown by computer scientists). Here is an example of generated joke:[4]

Q: What do you call a car that feels married? A: A cat that is a beer!

It is remarkable that the network has learned enough of the English language and of the structure of many jokes, to be able to sample correct sentences and joke-forms. However, it clearly lacks sufficient understanding of the world at large to handle paradoxes, or to inject hostility or incongruence in the generated text while walking the thin line that separates "nonsense" humor from "no sense" gibberish. It is not our goal in this paper to create a joke generator. Rather, the network's handling of recurring roles, dialogue, rhetorical structure and climax, deserve a more in-depth analysis.

5.3 Analysis and visualization of the neural encoding

The memory cells of a trained LSTM network develop an internal representation of the most salient sequential features in the input data. They tune as detectors of certain patterns that are used by the network to generate text streams that are coherent in style and structure

[4] The authors have suffered through many hundreds of generated jokes in the course of the experiments leading to the present work. None of them was particularly funny.

with the input jokes. In this sense, we expect such neurons to, somehow, encode key aspect of the jokes prosody, linguistic entities and rhetorical elements. Neural networks are often accused of developing obscure internal representation of the input information, that are only useful for the purpose of producing network prediction. We claim that the internal state of a network can provide interesting insights into the data if investigated with approaches from exploratory data analysis. For this purpose, we have collected the activation of the memory cells of the trained LSTM for a random sample of 2482 jokes in the dataset. Each character of the jokes is transformed into 3 vectors of 512 features corresponding to the activation of the memory cells in the 3 LSTM layers. Each layer can be analysed and processed independently as they encode sequential information at different resolution. In practice, the joke samples considered in this analysis are transformed into 423K vectors of neuron activations for each LSTM layer.

The neural encoding vectors are high-dimensional data that can be analyzed and visualized on bi-dimensional maps by resorting to dimensionality reduction algorithms. Principal Component Analysis (PCA) is a popular approach to dimensionality reduction which targets the identification of the direction of maximum variance in the data, referred to as principal components. We have applied PCA to the neural encoding datasets described above, focusing on the first 2–3 principal components since we target visualization of the high dimensional encodings on the screen space. Nonetheless, an analysis of the PCA results shows that the first 2–3 principal components (PC) are sufficient to capture 95% of the variance in the data, confirming our intuition that a couple of components already provide a good insight into the original neural encodings. To make the analysis more robust with respect to the choice of the dimensionality reduction algorithm, we have replicated PCA analysis using the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [14]. This is a non-parametric embedding technique for dimensionality reduction which has state-of-the-art-performance in numerous high-dimensional data visualization tasks. Here, we use an efficient t-SNE version [13] exploiting variants of the Barnes-Hut and dual-tree algorithm to approximate the gradient used for learning of the embeddings, which has a complexity of $O(N \log N)$ rather than $O(N^2)$ in the original t-SNE (where N is the dataset size).

5.4 Endings and climax

The 3 LSTM layers encode information of increasing complexity. The first layer operates at the level of character aggregation and is thus of reduced interest for the purpose of this analysis. Figure 3 shows the projection on the first two PCs of the states in the second and third hidden layer: the light-gray area is the result of the projection of the characters in the two datasets, whereas the red circles denote the positions where the form-feed joke separator character is projected. The yellow crosses, instead, highlight the points where the final character of all jokes is projected, i.e. two consecutive newline characters after the form feed separator.

Figure 3 shows that the joke separator and joke ending projections are spread all over the map when considering the activations of the second layer, whereas at the level of the third layer it emerges an organization of the state space where the joke separator/ending tends to be projected in specialized and contiguous areas of the map. In other words, the network seems to realize that a joke is coming to an end by the activations shifting towards the top-left area of the map. In particular, the joke ending seems to be encoded in a very tightly focused area of the bottommost plot. The form-feed separator, on the other hand, occupies a larger area of the map with several outliers, which can be noted in the central area of the plot. Each of these outliers corresponds to a joke; the fact that they are outliers suggests

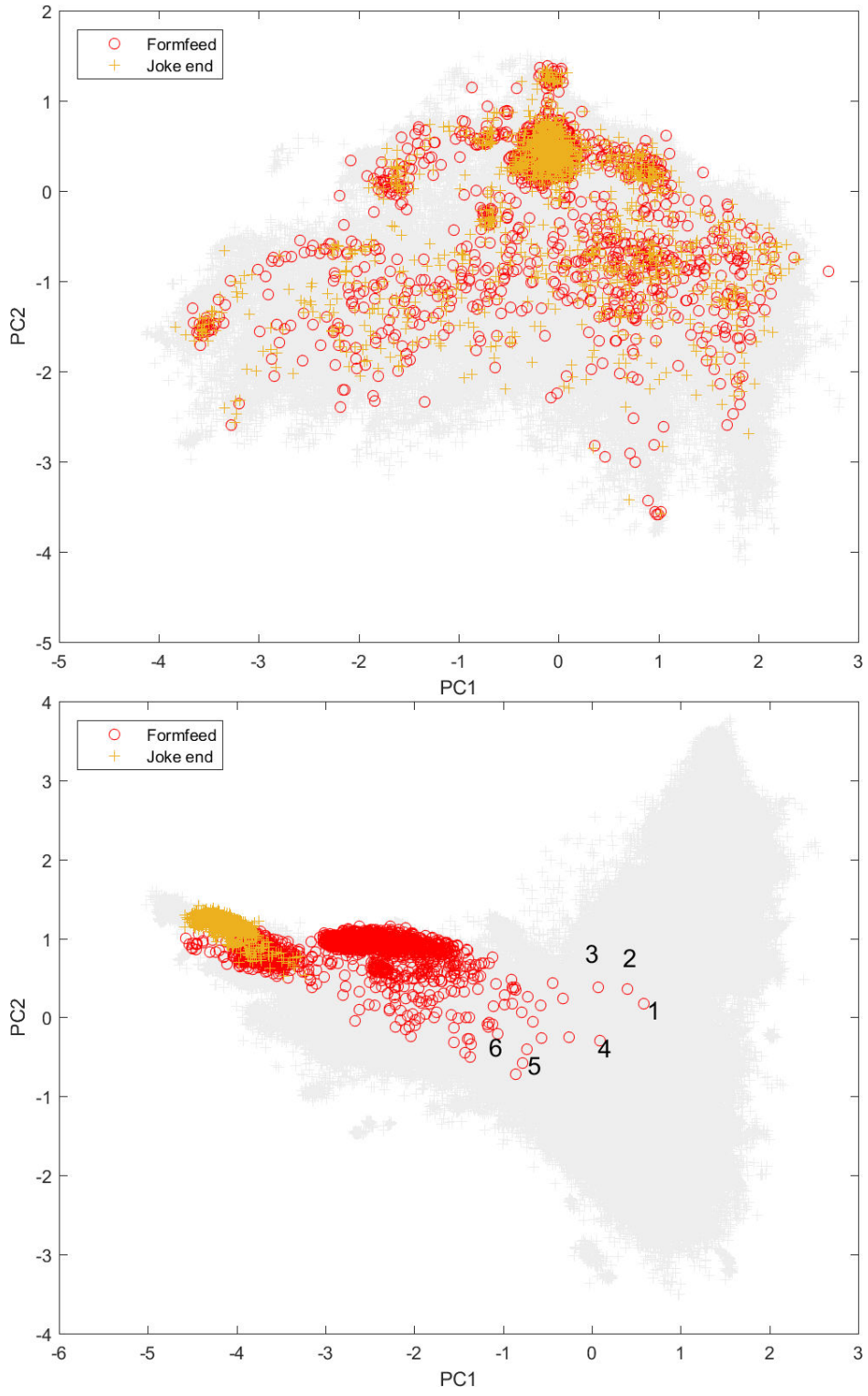

Figure 3 Projection on the first two principal components of the neuron activations for the second and third LSTM layers, respectively on the top and bottom plots. The light-gray area denotes the projection of all characters in the dataset: coloured placeholders identify the points where the joke ending and form-feed separator are projected. Numbers identify examples of joke outliers discussed in Section 5.4.

that the network was not expecting the jokes to end at that point of the sentence. In order to understand what characteristics of the jokes make them outliers, we have extracted a sample of them, identified by numbers in the bottommost plot of Figure 3. Sample 1 corresponds to the following joke

> The Boston taxi driver backed into the stationary fruit stall and within seconds he had a cop beside him. "Name?" "Brendan O'Connor." "Same as mine. Where are you from?" "County Cork." "Same as me ....." The policeman paused with his pen in the air. "Hold on a moment and I'll come back and talk about the old county. I want to say something to this fella that ran into the back of your cab." <

whose outlier nature might be the result of the joke ending with HTML tags which have not being removed by the authors of the benchmark. The neighboring jokes, i.e. i2 and 3 on the map, are very short ones:

> Q: What do monsters make with cars? A: Traffic Jam

> Q: What hotel do vampires prefer? A: The Coff Inn

confirming the fact that the network might not have understood that the joke has ended. Another set of outlier jokes puzzles the network due to the last sentence being an attribution statement, such as in the following joke (4 on the map):

> There are three kinds of lies: lies, damned lies, and statistics. Attributed by Mark Twain to Benjamin Disraeli

Then, there appears to be other jokes, marked as 5 on the map, that are outliers despite their style and structure being coherent with many other jokes in the corpus:

> Q: Why did Helen Keller have yellow fingers? A: from whispering sweet-nothings in her boyfriend's ear

> Whats a blondes favorite nursury rhyme? humpme dumpme

> Q: What's this (slowly waving fingers)? A: Helen Keller moaning

Finally, another outlier type seems to be associated with short fact-like jokes such as the one marked as 6 on the map:

> A nuclear war can ruin your whole day.

In general terms, we can say that a portion of the state-space of the higher level neurons of the network becomes responsive to *normal* joke conclusion. On the other hand, a piece of text which terminates unexpectedly with respect to the learned joke styles would result in it being mapped to a different region of the state-space, such as the outliers above. One may also argue that such outliers correspond to jokes that are not very much funny or, at least, not enough to induce the network in understanding that they have reached the climax.

5.5 The *Sexist* neuron (and its fellows *Dark* and *Racist*)

The analysis of the state-space maps can also provide a useful insight into the semantic concepts and linguistic structures that might emerge in the network as the result of training. For instance, we are interested in understanding if the network can learn from scratch a neural representation of prototypical aspects of the joke literature, such as recurring themes,

characters and rhetorical constructs. Figure 4 shows how the network encodes a specific subset of words appearing in the joke corpus: again, no clear structure emerges at the level of the second LSTM layer (topmost plot). On the other hand, the third layer has learned an internal representation where these words are grouped together in a specific area of the map (see bottommost plot). In other words, these terms produce very similar neural activations in the third layer. By taking a closer look at the word list, one can note that these terms can be associated with a specific female character, characterized by being good-looking and possibly a bit stupid. In a sense, the network seems to have learned neurons responding to *sexist* humour.

The state-space map obtained by the t-SNE algorithm shows a similar organization: see the topmost plot in Figure 5 showing the t-SNE map for the activations of the third LSTM layer. Terms having a sexist interpretation are projected close on the map, whereas words associated with different interpretation of the female role are plotted in other areas. For instance, a neutral term such as *girl* activates a different set of neurons with respect to those responding to sexist humor. Similarly, words related with the role of a woman within a family activate a set of neurons that is responsive also to terms indicating male family members. The bottommost plot in Figure 5 provides a clearer characterization of this latter area of the map. In particular, this seems to represent those neurons that respond to prototypical characters of the joke narration, such as *bartenders*, *engineers* and, of course, family members.

The same organization of the state space can be identified in the PCA map in the topmost plot of Figure 6. The same plot highlights an additional group of interesting joke characters that can be associated with *black humour*, for which the network has developed a very specific neural representation (see the cluster of red crosses in the plot). Another popular theme in joke literature, deals with humour associated with a country-based characterization of the persona. The bottommost plot in Figure 6 shows that the network has again conceptualized words pertaining with nationalities developing two distinct areas of the state-space devoted to two different groups of words (here we show only results for t-SNE due to the lack of space, but the same division has been found in PCA). The presence of two clusters in state-space might depend on different morphological features (i.e., nationality adjectives ending in -sh vs. -an); we might hypothesize that the two clusters would be joined in a single "nationality" concept on a deeper network.

5.6 Tracing the fun

As a joke is sequentially presented to the network (character by character), the network's internal state evolves, and the "current position" of the network moves in the high-dimensional space that represents the internal state. We can plot a trace of such movements to inspect not only the kind of abstractions that the network has synthesized, as done in previous sections, but also how a joke manages expectations and surprises while heading towards its climax.

Figure 7 shows two such traces, where for clarity we have plotted only the points corresponding to full words on a 2-dimensional space as given by our PCA; the red circle indicates the beginning of the joke, whereas the red cross indicates its ending. In most jokes, it seems that there are three main *attractors*: short non-function words (1 or 2 characters such as "a") in the top-left quadrant; longer non-function words (such as "why" or "what") in the top-right quadrant, and function words (such as names or verbs) in the lower-right quadrants (with a more marked vertical diffusion along the right edge). Hence, the trajectory of a joke often describes linear or triangular shapes between these attractors, as shown

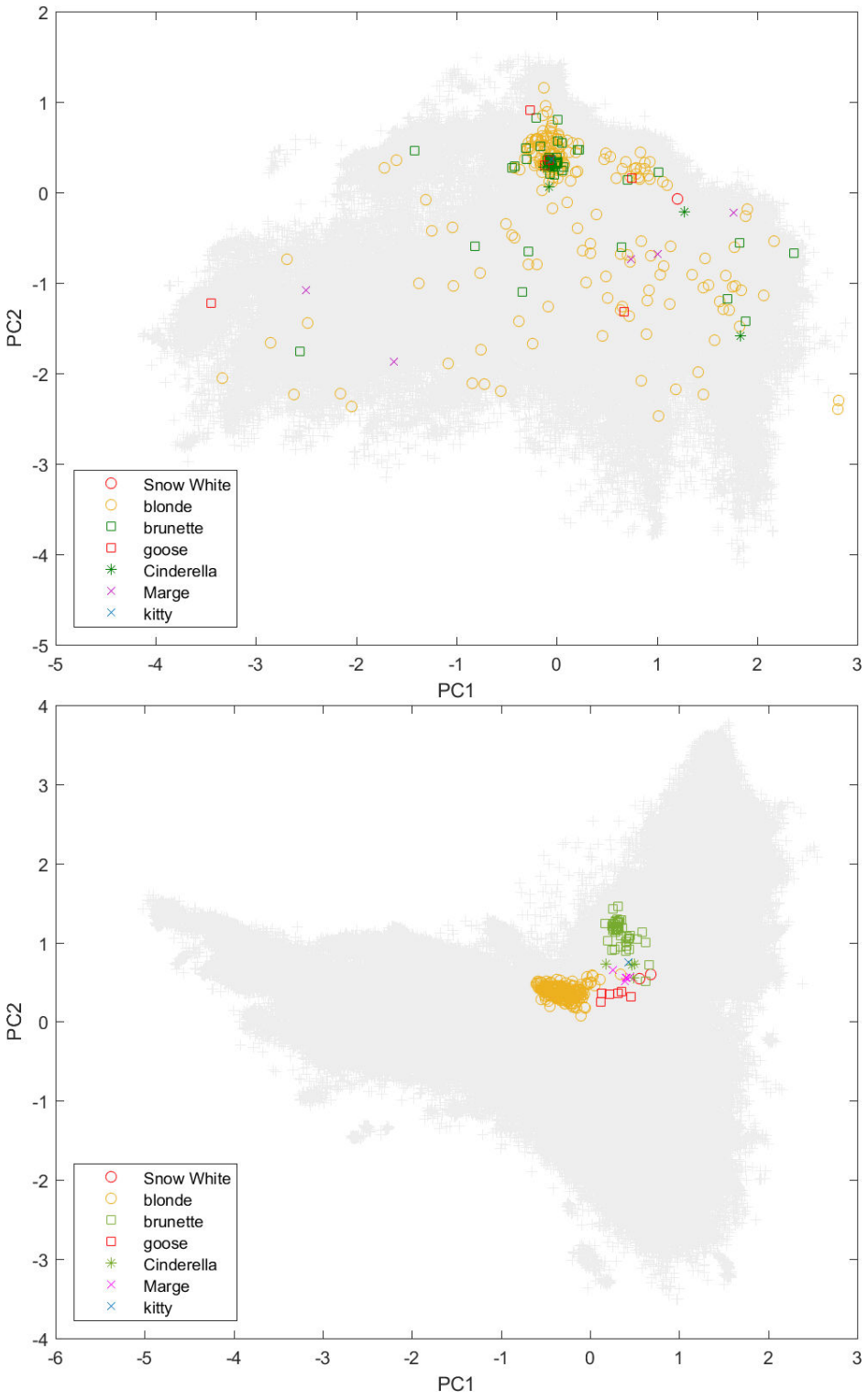

Figure 4 PCA projection on the second (top) and third (bottom) LSTM layers for selected words related to *sexist* humour.

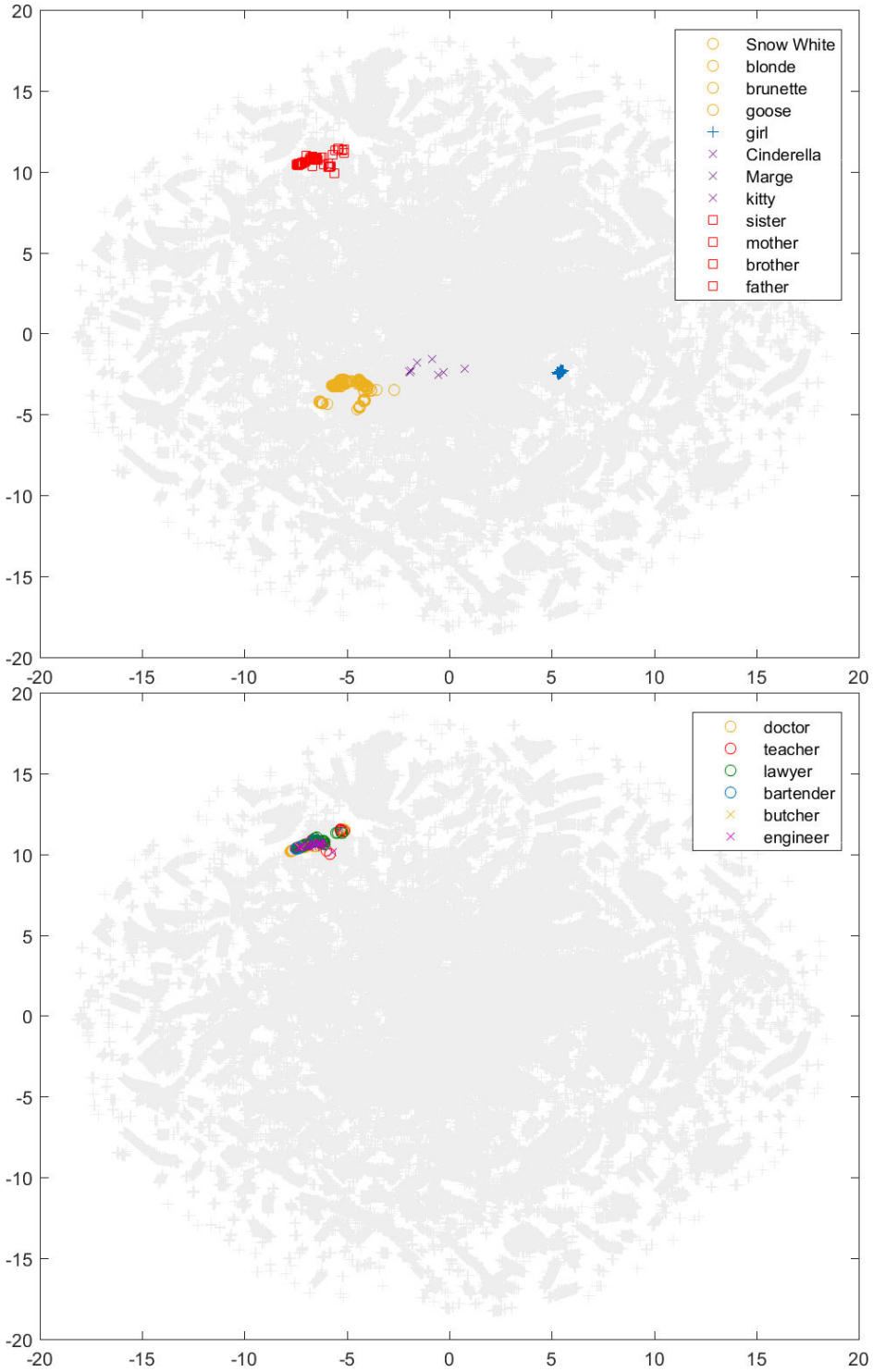

Figure 5 t-SNE embedding for selected words related to *sexist* humour (top) and for prototypical joke characters (bottom).

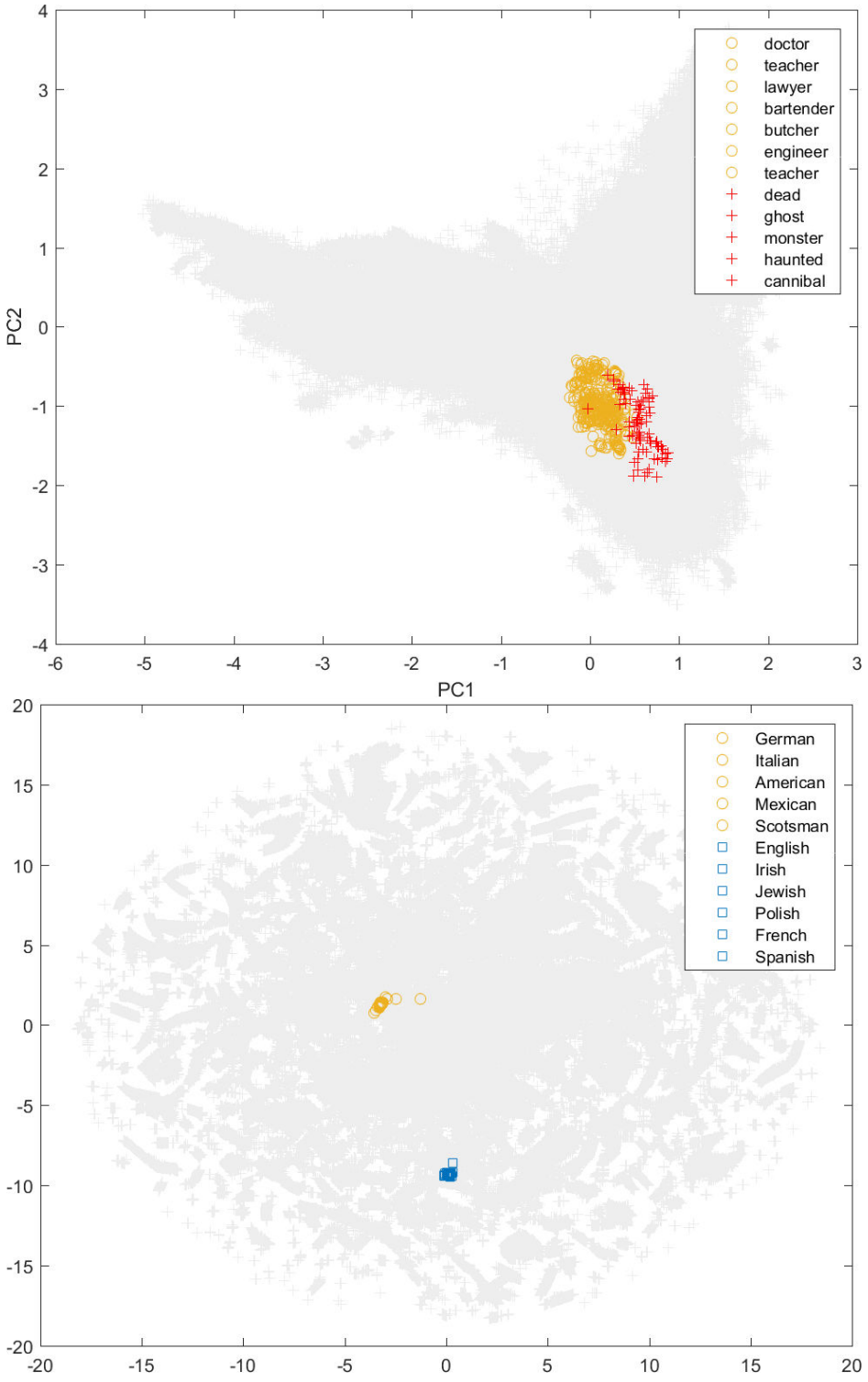

Figure 6 PCA projection for prototypical joke characters (top) and t-SNE embedding for words from jokes with a country-based characterization (bottom).

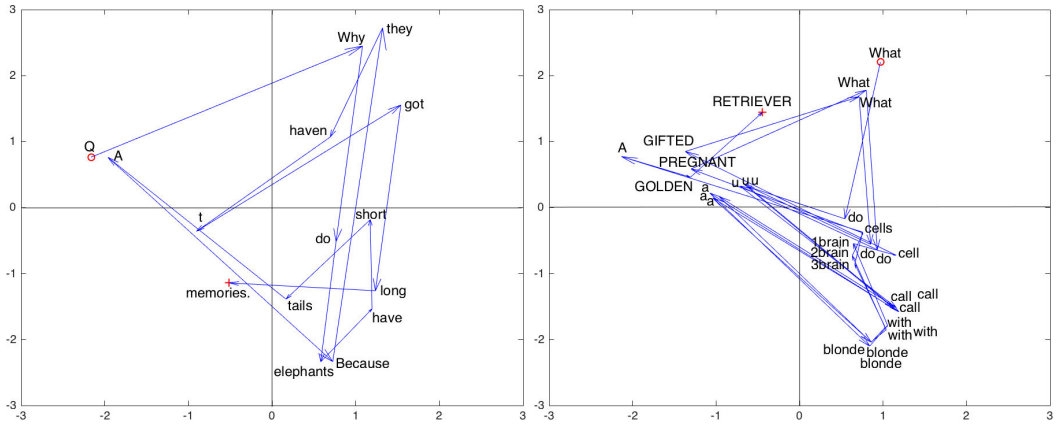

Figure 7 Climax trajectories for a single-laugh joke (left) and a three-laughs one (right).

in Figure 7(left). There are however other common moves: for example, the climax of a joke (i.e., where the reader or listener is expected to smile or laugh) is often mapped to a significant skew towards the left (negative x coordinates in Figure 7). In many cases, the cue word for the climax is in fact the left-most point among function words. The joke that generates the traces in Figure 7(left) is:

> Q: Why do elephants have short tails? A: Because they haven't got long memories.

Since many jokes reach their climax at the end, gradually building tension until the final release, it might be difficult to distinguish climax from termination. However, our data set included a few multiple-release jokes, and in analyzing the corresponding traces the same phenomenon could be observed for each of the climaxes. As an example, Figure 7 right shows the trace for the following joke:[5]

> What do u call a blonde with 1 brain cell? GIFTED! What do u call a blonde with 2 brain cells? PREGNANT! What do u call a blonde with 3 brain cells? A GOLDEN RETRIEVER!

6 Conclusions

In this work, we have reported on our results with a *computational* approach to literary theories of humor. We trained a particular form of neural network on a corpus of jokes, and "dissected" the artificial mind so generated by analyzing its contents. Our results confirm that certain typical humorous constructions, such as those postulated by Superiority or Incongruity theories, really exist in our corpus, and are sufficiently objective that the learning algorithm underlying the workings of a neural network can identify them.

What is even more striking, it appears that the neural network seems to have conceptualized some key aspects, characters and themes of humour literature by simply looking at joke texts character by character. The network is, in fact, oblivious of the structure of

[5] It is worth to remark that the joke was on a single line, as reported verbatim, so the presence of new-line characters is ruled out as an explanation for the detection of climax. Also, while in this particular example the comic answers are in all capitals, other jokes using lower case exhibit the same behaviour, so all-caps is also ruled out as a marker.

the grammar or even the structure of legal words in the language. Nevertheless, it is able to construct a representation of the world where a *blonde* and a *goose* are associated to a common interpretation, that is radically different from the representation associated to *girl* or to words identifying other animals. In a sense, we have seen how teaching a network on joke literature makes it quite prone to develop a *sexist* and *racist* view of the world, apparently confirming the very old idea that our language helps shaping the way we see the world.

References

1. Aristotle. *Poetics*, volume two: On Comedy. IV century BCE. Last surviving copy reportedly lost in the fire of the Abbey's Library in 1327, according to [6].
2. Salvatore Attardo. *Linguistic Theories of Humor*. De Gruyter Mouton, Berlin, 1994.
3. Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.
4. Yu-Chen Chan. Emotional structure of jokes: A corpus-based investigation. *Bio-medical materials and engineering*, 24(6):3083–3090, 2014.
5. Yu-Chen Chan and Joseph P Lavallee. Temporo-parietal and fronto-parietal lobe contributions to theory of mind and executive control: an fMRI study of verbal jokes. *Frontiers in psychology*, 6, 2015.
6. Umberto Eco. *Il nome della rosa*. Bompiani, 1980.
7. Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. *Neural Networks, IEEE Transactions on*, 9(5):768–786, 1998.
8. Lisa Friedland and James Allan. Joke retrieval: Recognizing the same joke told differently. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM'08, pages 883–892, New York, NY, USA, 2008. ACM.
9. Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL: `http://arxiv.org/abs/1308.0850`.
10. Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.
11. Ulrich Günther. *What is in a laugh? Humour, jokes and laughter in the conversational corpus of the BNC*. PhD thesis, Albert-Ludwigs-Universität, Freiburg, 2003.
12. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
13. Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.
14. Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.
15. Wendel Wallach and Colin Allen. *Moral Machines: Teaching Robots Right from Wrong*. Oxford University Press, November 2008.