A Long Short-Term Memory Framework for Predicting Humor in Dialogues

Dario Bertero and **Pascale Fung**
Human Language Technology Center
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
dbertero@connect.ust.hk, pascale@ece.ust.hk

Abstract

We propose a first-ever attempt to employ a Long Short-Term memory based framework to predict humor in dialogues. We analyze data from a popular TV-sitcom, whose canned laughters give an indication of when the audience would react. We model the setup-punchline relation of conversational humor with a Long Short-Term Memory, with utterance encodings obtained from a Convolutional Neural Network. Out neural network framework is able to improve the F-score of 8% over a Conditional Random Field baseline. We show how the LSTM effectively models the setup-punchline relation reducing the number of false positives and increasing the recall. We aim to employ our humor prediction model to build effective empathetic machine able to understand jokes.

1 Introduction

There has been many recent attempts to detect and understand humor, irony and sarcasm from sentences, usually taken from Twitter (Reyes et al., 2013; Barbieri and Saggion, 2014; Riloff et al., 2013; Joshi et al., 2015), customer reviews (Reyes and Rosso, 2012) or generic canned jokes (Yang et al., 2015). Bamman and Smith (2015) and Karoui et al. (2015) included the surrounding context.

Our work has a different focus from the above. We analyze transcripts of funny dialogues, a genre somehow neglected but important for human-robot interaction. Laughter is the natural reaction of people to a verbal or textual humorous stimulus. We want to predict when the audience would laugh. Compared to a typical canned joke or a sarcastic Tweet, a dialog utterance is perceived as funny only in relation to the dialog context and the past history. In a spontaneous setting a funny dialog is usually built through a setup which prepares the audience to receive the humorous discourse stimuli, followed by a punchline which releases the tension and triggers the laughter reaction (Attardo, 1997; Taylor and Mazlack, 2005). Automatic understanding of a humorous dialog is a first step to build an effective empathetic machine fully able to react to the user's humor and to other discourse stimuli. We are ultimately interested in developing robots that can bond with humans better (Devillers et al., 2015).

As a source of situational humor we study a popular TV sitcom: "The Big Bang Theory". The domain of sitcoms is of interest as it provides a full dialog setting, together with an indication of when the audience is expected to laugh, given by the background canned laughters. An example of dialog from this sitcom, as well as of the setup-punchline schema, is shown below (punchlines in bold):

PENNY: *Okay, Sheldon, what can I get you?*
SHELDON: *Alcohol.*
PENNY: *Could you be a little more specific?*
SHELDON: ***Ethyl alcohol.*** **LAUGH** ***Forty milliliters.*** **LAUGH**
PENNY: *I'm sorry, honey, I don't know milliliters.*
SHELDON: ***Ah. Blame President James Jimmy Carter.*** **LAUGH** ***He started America on a path to the metric system***

but then just gave up. **LAUGH**

The utterances before the punchline are the setup. Without them, the punchlines may not be perceived as humorous (the last utterance, out of context, may be a political complaint), only with proper setup a laughter would be triggered. The humorous intent is also strengthen by the fact the dialog takes place in a bar (evident from the previous and following utterances), where a request of 40 ml of "Ethyl Alcohol" is unusual and weird.

Our previous attempts on the same corpus (Bertero and Fung, 2016b; Bertero and Fung, 2016a) showed that using a bag-of-ngram representation over a sliding window or a simple RNN to capture the contextual information of the setup was not ideal. For this reason we propose a method based on a Long Short-Term Memory network (Hochreiter and Schmidhuber, 1997), where we encode each sentence through a Convolutional Neural Network (Collobert et al., 2011). LSTM is successfully used in context-dependent sequential classification tasks such as speech recognition (Graves et al., 2013), dependency parsing (Dyer et al., 2015) and conversation modelling (Shang et al., 2015). This is also to our knowledge the first-ever attempt that a LSTM is applied to humor response prediction or general humor detection tasks.

2 Methodology

We employ a supervised classification method to detect when punchlines occur. The bulk of our classifier is made of a concatenation of a Convolutional Neural Network (Collobert et al., 2011) to encode each utterance, followed by a Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) to model the sequential context of the dialog. Before the output softmax layer we add a vector of higher level syntactic, structural and sentiment features. A framework diagram is shown in Figure 1.

2.1 Convolutional Neural Network for each utterance

The first stage of our classifier is represented by a Convolutional Neural Network (Collobert et al., 2011). Low-level, high-dimensional input feature vectors are fed into a first embedding layer to obtain a low dimensional dense vector. A sliding window is

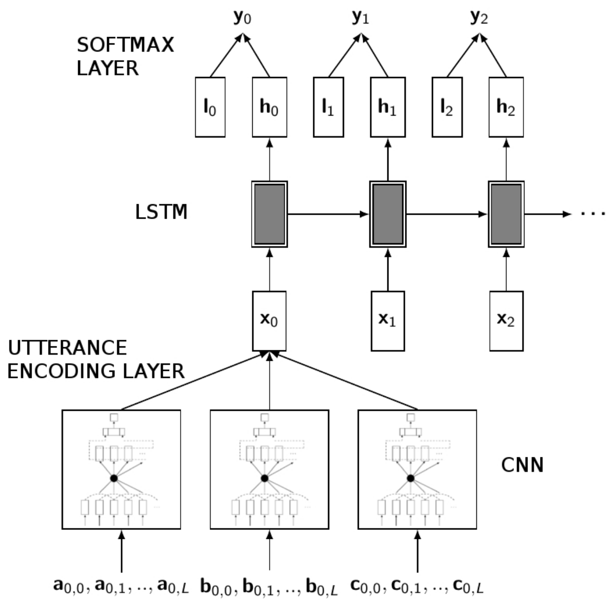

Figure 1: Framework diagram. $\mathbf{a}_t$, $\mathbf{b}_t$ and $\mathbf{c}_t$ are the CNN three input features (words, word2vec and character trigrams). $\mathbf{l}_t$ are the high level feature vectors, and $\mathbf{y}_t$ the outputs for each utterance.

then moved on these vectors and another layer is applied to each group of token vectors, in order to capture the local context of each token. A max-pooling operation is then applied to extract the most salient features of all the tokens into a single vector for the whole utterance. An additional layer is used to generalize and distribute each feature to its full range before obtaining the final utterance vector.

In our task we use three input features:

1. Word tokens: each utterance token is represented as a one-hot vector. This feature models how much each word is likely to trigger humor in the specific corpus.

2. Character trigrams: each token is represented as a bag-of-character-trigrams vector. The feature models the role of the word signifier and removes the influence of the word stems.

3. Word2Vec: we extract for each token a word vector from word2vec (Mikolov et al., 2013), trained on the text9 Wikipedia corpus[1]. This representation models the general semantic

[1]Extension of the text8 corpus, obtained from http://mattmahoney.net/dc/textdata

meanings, and matches words that do not appear to others similar in meaning.

The convolution and max-pooling operation is applied individually to each feature, and the three vectors obtained are then concatenated together and fed to the final sentence encoding layer, which combines all the contributions.

2.2 Long/Short Term Memory for the utterance sequence

The LSTM is an improvement over the Recurrent Neural Network aimed to improve its memory capabilities. In a standard RNN the hidden memory layer is updated through a function of the input and the hidden layer at the previous time instant:

$$\mathbf{h}_t = \tanh(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}) \qquad (1)$$

where $\mathbf{x}$ is the network input and $\mathbf{b}$ the bias term. This kind of connection is not very effective to maintain the information stored for long time instants, as well as it does not allow to forget unneeded information between two time steps. The LSTM enhances the RNN with a series of three multiplicative gates. The structure is the following:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{i_x} \mathbf{x}_t + \mathbf{W}_{i_h} \mathbf{h}_{t-1} + \mathbf{b}_i) \qquad (2)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_{f_x} \mathbf{x}_t + \mathbf{W}_{f_h} \mathbf{h}_{t-1} + \mathbf{b}_f) \qquad (3)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{o_x} \mathbf{x}_t + \mathbf{W}_{o_h} \mathbf{h}_{t-1} + \mathbf{b}_o) \qquad (4)$$
$$\mathbf{s}_t = \tanh(\mathbf{W}_{s_x} \mathbf{x}_t + \mathbf{W}_{s_h} \mathbf{h}_{t-1} + \mathbf{b}_h) \qquad (5)$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{s}_t \qquad (6)$$
$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \odot \mathbf{o}_t \qquad (7)$$

where $\odot$ is the element-wise product. Each gate factor is able to let through or suppress a specific update contribution, thus allowing a selective information retaining. The input gate $\mathbf{i}$ is applied to the cell input $\mathbf{s}$, the forget gate $\mathbf{f}$ to the cell value at the previous time step $\mathbf{c}_{t-1}$, and the output gate $\mathbf{o}$ to the cell output for the current time instant $\mathbf{h}_t$. In this way a cell value can be retained for multiple time steps when $\mathbf{i} = 0$, ignored in the output when $\mathbf{o} = 0$, and forgotten when $\mathbf{f} = 0$.

As dialog utterances are sequential, we feed all utterance vectors of a sitcom scene in sequence into a Long Short-Term Memory block to incorporate contextual information. The memory unit of the LSTM keeps track of the context in each scene, and mimics human memory to accumulate the setup that may trigger a punchline.

Before the output we incorporate a set of high level features from our previous work (Bertero and Fung, 2016b) and past literature (Reyes et al., 2013; Barbieri and Saggion, 2014). They include:

- Structural features: average word length, sentence length, difference in sentence length with the five previous utterances.
- Part of speech proportion: noun, verbs, adjectives and adverbs.
- Antonyms: proportion of antonym words with the previous utterance (from WordNet (Miller, 1995)).
- Sentiment: positive, negative and average sentiment score among all words (from SentiWordNet (Esuli and Sebastiani, 2006)).
- Speaker and turn: speaker character identity and utterance position in the turn (beginning, middle, end, isolated).
- Speaking rate: time duration of the utterance from the subtitle files, divided by the sentence length.

All these features are concatenated to the LSTM output, and a softmax layer is applied to get the final output probabilities.

3 Experiments

3.1 Corpus

We built a corpus from the popular TV-sitcom "The Big Bang Theory", seasons 1 to 6. We downloaded the subtitle files (annotated with the timestamps of each utterance) and the scripts[2], used to segment all the episodes into scenes and get the speaker identity of each utterance. We extracted the audio track of each episode in order to retrieve the canned laughters timestamps, with a vocal removal tool followed by a silence/sound detector. We then annotated each utterance as a punchline in case it was followed by a laughter within $1s$, assuming that utterances not

[2]From bigbangtrans.wordpress.com

Classifier and features	Accuracy	Precision	Recall	F-score
CRF n-grams	61.8	56.8	45.1	50.2
CRF language features	67.8	**67.5**	47.8	56.0
CRF n-grams + language features	65.9	61.2	55.3	58.1
LSTM	63.1	56.7	58.7	57.6
LSTM + high level features	**70.0**	66.7	**59.4**	**62.9**

Table 1: Results, percentage.

Encoding stage	A	P	R	F1
CNN	70.0	66.7	59.4	**62.9**
LSTM	68.4	66.2	53.4	59.1

Table 2: Comparison between a CNN and a LSTM sentence encoding input.

followed by a laughter would be the setup for the punchline.

We obtained a total of 135 episodes, 1589 overall scenes, 42.8% of punchlines, and an average interval between two punchlines of 2.2 utterances. We built a training set of 80% of the overall episodes, and a development and test set of 10% each. The episodes were drawn from all the seasons with the same proportion. The total number of utterances is 35865 for the training set, 3904 for the development set and 3903 for the test set.

3.2 Experimental setup and baseline

In the neural network we set the size to 100 for all the hidden layers of the CNN and the LSTM, and 5 to the convolutional window. We applied a dropout regularization layer (Srivastava et al., 2014) after the output of the LSTM, and L2 regularization on the softmax output layer. The network was trained with standard backpropagation, using each scene as a training unit. The development set was used to tune the hyperparameters, and to determine the early stopping condition. When the error on the development set began to increase for the first time we kept training only the final softmax layer, this improved the overall results. The neural network was implemented with THEANO toolkit (Bergstra et al., 2010). We ran experiments with and without the extra high-level feature vector.

As a baseline for comparison we used an implementation of the Conditional Random Field (Lafferty et al., 2001) from CRFSuite (Okazaki, 2007), with L2 regularization. We ran experiments using the same high level feature vector added at the end of the neural network, 1-2-3gram features of a window made by the utterance and the four previous, and the two feature sets combined. We also compared the overall system where we replace the CNN with an LSTM sentence encoder (Li et al., 2015), where we kept the same input features.

3.3 Results and discussion

Results of our system and our baselines are shown in table 1. The LSTM with the aid of the high level feature vector generally outperformed all the CRF baselines with the highest accuracy of 70.0% and the highest F-score of 62.9%. The biggest improvement of the LSTM is the improvement of the recall without affecting too much the precision. Lexical features given by n-gram from a context window are very useful to recognize more punchlines in our baseline experiment, but they also yield many false positives, when the same n-gram is used in different contexts. A CNN-LSTM network seems to overcome this issue as the CNN stage is better in modeling the lexical and semantic content of the utterance, as the LSTM allows to put each utterance in relation with the past context, filtering out many false positives from wrong contexts.

The choice of the CNN is further justified by the results obtained from the comparison between the CNN and the LSTM sentence encoding input, shown in table 2. The CNN is more effective, obtaining a recall of 10% higher and 6% more in F-score. The CNN is a simpler model that might benefit more of a small-size corpus. It also required a much shorter training time compared to the LSTM. We may consider in the future to use more data, and try other sentence input encoders, including deeper or bi-directional LSTMs, to find the most effective one.

Predicting humor response from the canned laughters is a particularly challenging task. In some cases canned laughters are inserted by the show producers with the purpose of solicit response to weak jokes, where otherwise people would not laugh. The audience must also be kept constantly amused, extra canned laughters may help in scenes where fewer jokes are used.

4 Conclusion and future work

We proposed a Long Short-Term Memory based framework to predict punchlines in a humorous dialog. We showed that our neural network is particularly effective in increasing the F-score to 62.9% over a Conditional Random Field baseline of 58.1%. We furthermore showed that the LSTM is more effective in obtaining an higher recall with fewer false positives compared to simple n-gram shifting context window features.

As future work we plan to use a virtual agent system to collect a set of human-robot humorous interactions, and adapt our model to predict humor from them.

Acknowledgments

This work was partially funded by the Hong Kong Phd Fellowship Scheme, and partially by grant #16214415 of the Hong Kong Research Grants Council.

References

Salvatore Attardo. 1997. The semantic foundations of cognitive theories of humor. *Humor-International Journal of Humor Research*, (10):395–420.

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *Ninth International AAAI Conference on Web and Social Media*.

Francesco Barbieri and Horacio Saggion. 2014. Modelling irony in twitter. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 56–64.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

Dario Bertero and Pascale Fung. 2016a. Deep learning of audio and language features for humor prediction. *International Conference on Language Resources and Evaluation (LREC) 2016*.

Dario Bertero and Pascale Fung. 2016b. Predicting humor response in dialogues from TV sitcoms. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Laurence Devillers, Sophie Rosset, Guillaume Dubuisson Duplessis, Mohamed A Sehili, Lucile Béchade, Agnes Delaborde, Clément Gossart, Vincent Letard, Fan Yang, Yucel Yemez, et al. 2015. Multimodal data collection of human-robot humorous interactions in the joker project. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 348–354. IEEE.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July. Association for Computational Linguistics.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 757–762.

Jihen Karoui, Benamara Farah, Véronique Moriceau, Nathalie Aussenac-Gilles, and Lamia Hadrich-Belguith. 2015. Towards a contextual pragmatic model to detect irony in tweets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint*

Conference on Natural Language Processing (Volume 2: Short Papers), pages 644–650, Beijing, China, July. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

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

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

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs). *URL http://www.chokkan.org/software/crfsuite*.

Antonio Reyes and Paolo Rosso. 2012. Making objective decisions from subjective data: Detecting irony in customer reviews. *Decision Support Systems*, 53(4):754–760.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Language Resources and Evaluation*, 47(1):239–268.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *EMNLP*, pages 704–714.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Julia Taylor and Lawrence Mazlack. 2005. Toward computational recognition of humorous intent. In *Proceedings of Cognitive Science Conference*, pages 2166–2171.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376, Lisbon, Portugal, September. Association for Computational Linguistics.