GENERATING CONFERENCE CALL-FOR-PAPERS USING STACKED LONG SHORT-TERM MEMORY NEURAL NETWORKS

Bálint Antal
1. Faculty of Informatics
University of Debrecen
Debrecen, POB 400, 4002, Hungary
2. CL-IC Technologies Ltd
4 Reed Close, Cambridge, CB2 9NX, United Kingdom
balint@cl-ic.com

Attila Csikász-Nagy
1. Randall Division of Cell and Molecular Biophysics
King's College London, United Kingdom
2. Faculty of Information Technology and Bionics
Pázmány Péter Catholic University, Budapest, Hungary
3. CL-IC Technologies Ltd
4 Reed Close, Cambridge, CB2 9NX, United Kingdom
attila@cl-ic.com

Rafael E. Carazo-Salas
1. School of Cellular and Molecular Medicine
University of Bristol
Biomedical Sciences Building, University Walk, Bristol BS8 1TD, United Kingdom
2. CL-IC Technologies Ltd
4 Reed Close, Cambridge, CB2 9NX, United Kingdom
rafael@cl-ic.com

ABSTRACT

In this paper, we describe a novel approach to generate conference call-for-papers using Natural Language Processing and Long Short-Term Memory network. The approach has been successfully evaluated on a publicly available dataset.

1 INTRODUCTION

Deep learning LeCun et al. (2015) techniques has been successfully applied to learn and generate sequence data Sutskever et al. (2014). In this paper, we present an approach to generate conference Call-for-papers (CFPs) using a stacked Long Short Term Memory (LSTM) network. Regardless of the hardship of the problem, we will show that the network is capable of predicting intelligible scientific keywords with relatively short training period. The paper is organized in the following way: Section 2 describes the approach in details. We show the quantitaive and qualitative results in Section 3. Finally, we draw conclusions in Section 4.

2 CALL-FOR-PAPERS GENERATION USING AN STACKED LSTM NETWORK

In this section, we describe the proposed approach in detail. First, we pre-process the raw text data. Then, we extract 10 topic models from one part of the dataset Blei (2012), which allows us to categorize the CFPs based on textual similarities. Then, we label the rest of the data with the topic models and train a Deep Neural Network for each topic. We use the trained models to generate texts from seed sentences.

2.1 METHODOLOGY

We have used the 2008, 2009 and 2010 versions of the WikiCFP database (http://www.wikicfp.com). Each dataset contains a large number of unprocessed and un-categorized scientific call-for-papers in an XML format. We have extracted the CFP descriptions from each data row. We have the 2008 data to create the topic models and we have used the 2009 and 2010 datasets as training data to the Deep Neural Network. We have used NLTK Bird & Klein (2009) for data pre-processing (tokenization, part-of-speech tagging, name entity recognition), Gensim Řehůřek & Sojka (2010) for topic modeling, Keras Chollet (2015), Theano Bergstra et al. (2010) and cuDNN Chetlur et al. (2014) for deep learning. For text generation, we have relied on the method presented in `https://github.com/fchollet/keras/blob/master/examples/lstm_text_generation.py`. We have run the experiments on COTS PC with a NVIDIA Titan X installed.

2.2 Data pre-processing

To prepare the data for topic model creation, we have tokenized the extracted 2008 CFPs. We have extracted the nouns from the tokens which were not named entities. We have removed the most frequent words as they were general conference-related terms (e.g. conference, submission, etc).

2.3 Topic model creation

To extract topic models form the data, we have used Latent Dirichlet Allocation Blei et al. (2003) on the preprocessed data. 10 models has been extracted from the data what we used to generate training data from the 2009 and 2010 datasets. Each CFP has been labeled with a topic number ($t = 0, \ldots, 9$) based on the probability estimates for each topic model. We have used 40 character long semi-redundant sequences created from the raw text to train a Deep Recurrent Neural Network.

2.4 Training

We have created a 3-layer stacked Long Short Term Memory networkSchmidhuber (1997). To avoid overfitting, after each LSTM-layer, we have included a Dropout layer Srivastava et al. (2014). Finally, we have mapped the learned representation to characters using a fully connected dense layer. The detailed architecture of the network can be seen in Table 1.

Table 1: The architecture of the Deep Recurrent Neural network. The network consists of a 3-layer stack of LSTM-Dropout layers and a dense layer.

Layer	**Shape**	**# Parameters**	**Activation**
LSTM	40×512	1347584	sigmoid
Dropout (0.2)	40×512	0	
LSTM	40×256	787456	sigmoid
Dropout (0.2)	40×256	0	
LSTM	128	197120	sigmoid
Dropout (0.2)	128	0	
Dense	# Characters	18705	softmax
Total parameters:		**2350865**	

For training we have used the AdaMax Kingma & Ba (2014) and categorical cross-entropy as a loss function. We have generated 10 models by running training a network for each topic model for 60 epochs.

2.5 Call-for-papers generation

We have used the trained model to predict CFPs in the following way: we have selected a random part from an existing CFP and used is a seed. Then we have predicted the next characters based on the probabilities assigned to the seed sentence.

3 Results

To evaluate the approach, we have generated 25 1000 character long CFP-excerpts and measured their similarity to their respective topic models using Latent Semantic Indexing Landauer (2006). Table 2 show the results obtained for each topic model.

Table 2: The corpus length, the number of distinct characters and the similarity score for the ten topic models.

TOPIC #	CORPUS LENGTH	# CHARS	SIMILARITY
0	4810101	150	0.400
1	7256126	148	0.897
2	1005447	145	0.864
3	1450533	147	0.821
4	7116410	148	0.822
5	1559038	123	0.899
6	661965	124	0.837
7	1926136	146	0.906
8	1635757	146	0.871
9	2692077	145	0.834

As it can be seen, 9 of the 10 generated text sets achieved high similarity with their topic models. However, generating topic 0 seems to less accurate, potentially because this topic contains the most special characters from the 10.

The following excerpt shows that the after 60 iteration, the model was able to predict computer-science and engineering related keywords successfully.

```
* computer science and technologies
    * software engineering
    * power electronics
    * computer science and systems
    * sensor networks and applications
    * and service oriented systems
    * applications of computer science
    * sensor networks and systems and multimedia systems and systems
    * service engineering
    * computer science and engineering
    * electronics and computer science
```

4 CONCLUSION AND FUTURE WORK

In this paper, we have presented an approach to call-for-paper generation using a stacked LSTM network and Natural Language Processing. The presented approach was evaluated on a publicly available dataset where it showed that intelligible scientific keywords were predicted. In the future, we would like to obtain data with CFP categories assigned and use the approach to predict discipline-related scientific keywords.

ACKNOWLEDGMENTS

Bálint Antal was supported by The János Bolyai Research Fellowship. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. Rafael E. Carazo Salas is supported by the University of Bristol

REFERENCES

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pp. 1–7, 2010.

Edward Loper Bird, Steven and Ewan Klein. *Natural Language Processing with Python*. OReilly Media Inc., 2009.

David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

François Chollet. Keras. https://github.com/fchollet/keras, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Thomas K Landauer. *Latent semantic analysis*. Wiley Online Library, 2006.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

Sepp Hochreiter; Jrgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 17351780, 1997.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.