ICLR 2018 Reproducibility Challenge: Enhance Word Representation for Out-of-Vocabulary on Ubuntu Dialogue Corpus

Hugo Scurti
McGill University
(260743006)
hugo.scurti@mail.mcgill.ca

Isaac Sultan
McGill University
(260680295)
issac.sultan@mail.mcgill.ca

Alex Wong
McGill University
(260602944)
alexander.wong4@mail.mcgill.ca

***Abstract*— The paper reproduces a baseline finding of an ICLR 2018 submission. This baseline is an adaptation by the submission of Chen et al.'s Enhanced LSTM method to perform a language modelling task. The model is trained and evaluated on the Ubuntu Dialogue Corpus v2.0. Training time was the key bottleneck in reproducing the model. As such, the reproduction model was trained to 9,750 batch steps, lower than the 22,000 - 25,000 steps that the authors of the submission cited as providing the greatest Mean Reciprocal Rank (MRR). The reproduction produced an MRR of 73.3%, performing inferior to the ESIM baseline within the submission (MRR 80.2%). Therefore, it cannot concluded definitively if the performance findings stated in submission are valid.**

I. INTRODUCTION

Reproducibility is often described as one of the main principles of the scientific method. However, confidence in peer-reviewed scientific findings has been shaken as of late, by what is deemed a "reproducibility crisis" [1]. This issue can be seen, too, within the machine learning (ML) field [2]. Olorisade et al.'s paper notes that access to datasets and source code are noted as key obstacles to the reproducibility of ML experiments. Our paper aims to reproduce a baseline finding of the "Enhanced Word Representation for Out-Of-Vocabulary on Ubuntu Dialogue Corpus" paper submitted to the ICLR 2018 Conference [3]. We show that the performance obtained by its authors can be re-computed - an important step in assessing the overall validity of this paper.

The paper presents a method that improves performance when modelling dialogue systems that contain out-of-vocabulary words. We aimed to reproduce a result by implementing Chen et al.'s Enhanced LSTM Method (ESIM) [4] to model the Ubuntu Dialogue Corpus v2.0 discussed in the paper. Although our initial motivation was to reproduce a novel implementation presented, where a modified ESIM (ESIM^e) employs enhanced word embedding vectors (pre-trained word vectors from GloVe [5] combined with Word2Vec [6] generated on the corpus), we were limited by the lack of implementation detail within the paper. The ESIM had its performance measured by 1 in 10 Recall@k for R@1, R@2 and R@5, as well as Mean Reciprocal Rank (MRR) (these metrics are explained by Lowe et al. [7]). We demonstrate that while our implementation under-performs in terms of the aforementioned metrics, our results do show promise. Thus, we are of the opinion that the model can be reproduced to the same degree of performance if allowed further training.

II. SUMMARY OF PAPER

The paper presents a method that improves performance of language modelling on dialogue corpora. The ESIM, and the modifications made to it by the authors are presented. Next utterance selection is modelled by estimating the probability of selecting a response given a context. The findings of the paper are that the ESIM^e, as well as the use of a enhanced embeddings (a combination of pre-trained and trained word embeddings with character-composed embeddings), provides state-of-the-art performance for this task on the corpora in question.

A. Enhanced ESIM Description

The ESIM^e is represented visually in Fig. 1 and described as such. The word representation layer is composed of a vector space constructed from word-embedding and character-composed embedding. This is a mapping of each word in context and response. Word embedding vectors are generated on the training corpus using Word2Vec (d1 dimensions) and pre-trained word embedding vectors are sourced from GloVe (d2 dimensions). The vectors are then concatenated algorithmically to generate a dictionary of word embedding vectors with dimensionality d1 + d2. The character-composed embedding is generated by concatenating the final state vector of the forward and backward direction of bi-directional LSTM (BiLSTM) [8].

The context representation layer is where vector sequences are fed into the BiLSTM, where it learns to represent word and local sequence context. The attention matching layer then computes the similarity of hidden states between context and response to generate a co-attention matrix. Attended response vectors are generated for each word in context and the vector is concatenated with the difference and element-wise product. The matching aggregation layer aggregates response-aware context and response representation with a BiLSTM. Max pooling in combined with the concatenation of forward and backwards vector to form the final fixed vector v. Finally, v is fed into a fully-connected feedforward neural network in the prediction layer.

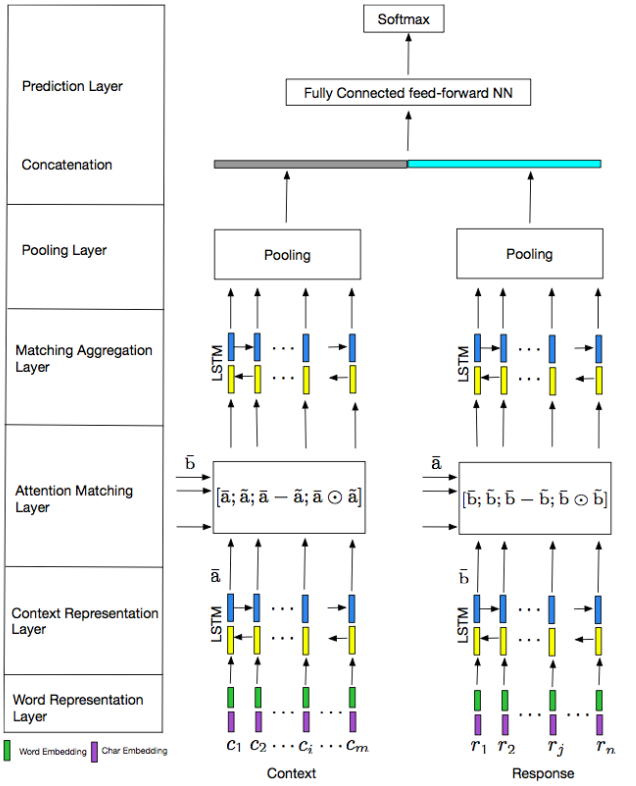

Fig. 1. High-level overview of ESIM^e sourced from reproduction paper. ESIM does not include the 'Char Embedding' in the Word Representation Layer (in purple).

B. Model Evaluation

The model is evaluated on the Ubuntu Dialogue Corpus v2.0. This is the largest unstructured multi-turns dialogue corpus, consisting of around one-million two-person conversations.

Model performance with the Ubuntu Corpus is measured with Recall@k, and Mean Reciprocal Rank (MRR), which demonstrate that the ESIM^e with enhanced embeddings displays the best performance. The model is also evaluated on the Douban Conversation Corpus [9], a Chinese language social networking conversation corpus. Model performance on this dataset was measured with Precision@1, Mean Average Precision (MAP), and MRR.

The model is further evaluated without the use of end-of-utterance and end-of-turn tags, resulting in poorer performance. This suggests that information regarding utterance and turn boundary structure can be harnessed in future models to further improve performance.

III. REPRODUCTION METHODOLOGY

As stated in the introduction, we aimed to replicate the ESIM that the paper presented as a baseline model. This model was created by Chen et al.'s aforementioned "Enhanced LSTM for Natural Language Inference" paper [4], and can be thought of as composed of two sequences of two parallel blocks of bidirectional LSTM layers, with an attention matching layer between the sequences, which is then followed by a pooling layer.

Only the Ubuntu Dialogue Corpus v2.0 was evaluated upon for the following reasons. Firstly, our limited knowledge in natural language inference restricted our understanding of the character-composed embedding layer used in the ESIM^e. Hence, we focused our attention onto the standard ESIM, which does not include this newly introduced feature. We also were hesitant to model the Douban Corpus as our insufficient understanding of Chinese language, that composes the corpus, could have hindered potential debugging efforts.

A high-level overview of our methodology:

- Downloading and generating the dataset
- Preprocessing the dataset into the appropriate input format
- Filter unused tokens from the pre-trained word embeddings
- Adapting the ESIM source code
- Evaluation of the results

The Ubuntu Dialogue Corpus v2.0 was downloaded using the provided script that downloads a `.zip` file, extracts it, and samples random sentences to generate a `.csv` file. Default parameters were used, with a random seed of '1234', in order to generate an identical dataset as the paper that we we are replicating [3]. The script was called three times with different parameters, in order to generate training, test and validation sets.

The resulting test and validation `.csv` files must be pre-processed to mirror the format of the training file. These files were organised so that each line is composed of a context, the single positive response and 9 negative responses. We split each line so that each row in the resulting data-frame contains a unique response string. This arranges the data into <context, response, label> triples, with label as boolean value that flags a positive response.

The dataset must be tokenized in order for a Word2Vec embedding vector to be trained. The Stanford CoreNLP library contains a PTB Tokenizer that produces tokens that conform to the rules of The Penn Treebank [10]. Given that one row could contain more than one sentence, the result outputted by the Stanford CoreNLP server must match with the input rows fed passed into the call. Therefore, we called the API for one input row at a time (i.e. a context or hypothesis document) and processed the result by disregarding the sentence separation and only using the returned tokens. The tokens returned were additionally lemmatized. The resulting training dataset would consist of 3 columns for each sample: the first column contained an array of token corresponding to the context, the second column contained an array of token corresponding to the hypothesis, and the third column was the value indicating whether it is the correct response. The object was stored into a `.pickle` dump file, so that it could loaded to memory quickly in later usage.

While processing the tokens returned by the CoreNLP server calls, distinct tokens were stored in a dictionary with their frequency associated with it. A set of tokens were then used from this dictionary to filter the pre-trained GloVe

TABLE I

SPECIFICATIONS OF ESIM PARAMETERS USED IN REPRODUCTION

Parameter	Value	Source
Optimization Algorithm	ADAM	Paper in Question
Initial Learning Rate	0.001	Paper in Question
Learning Rate Decay	Exponential	Paper in Question
Batch Size	128	OpenReview
Matching Layer Hidden Units	200	Paper in Question
Prediction Layer Hidden Units	256	Paper
Early Stopping	Yes	Williams et al.
Gradient Clipping	No	OpenReview
Dropout	No	OpenReview
Embedding Dimension	300	Paper in Question
Sequence Length	100	HS, IS & AW*

*Refers to the authors of this reproducibility paper

vector and remove all words that did not appear in the training corpus. We thought this step would be beneficial since the unzipped glove dataset (glove.42B.300d.txt) is 4.67 GB large, which would take a considerable amount of memory simply to load it. The filtered GloVe dataset is 440 MB large and contains roughly 9% of the original GloVe dataset. Through this process, we confirmed the authors observation that only 22% of the 823,057 Ubuntu tokens occur in the pre-built GloVe word vectors, and that our reproduction produced the same dataset.

In training the character embeddings using Word2Vec, we used all the default hyperparameters, and trained each context/response as distinct inputs such that each context/response pair takes one line in the input data file.

Source code of the ESIM implemented by Williams et al. [11] was used as a base. This model implements the ESIM, and was chosen over Chen et al's implementation due to its implementation using TensorFlow, a deep learning library with greater ease-of-use [12] for us than Theano [13]. Williams et al. source code is licensed under an Apache license, which we complied with by including a copy of the license within our own distribution and providing clear attribution to The Apache Software Foundation.

We aimed to replicate the hyperparameters used by the paper as faithfully as possible. ADAM optimization algorithm [14] was used for training. A learning rate of 0.001 with exponential decay was used. The batch size was 128. The number of hidden units of BiLSTM for word embeddings in both context representation and matching aggregation layers was 200. In the prediction layer, we used ReLu activation instead of tanh activation. We did not use dropout and regularization. We contacted the papers authors and who informed us that no patience hyperparameter for early stopping and no gradient clipping hyperparameter were used. Table I shows the parameters and hyperparameters used in our implementation, the value of these hyperparameters and where we found these values.

When training the model, performance metrics were printed every 50 steps. Accuracy and cost over the validation set and over a subset of the training set were employed to evaluate the training of the model. We did not evaluate over the whole training set since this is significantly larger and would greatly slower the training time. A checkpoint of the model was also stored at every 50 steps, so that training could be paused and resumed, allowing parallel evaluation of the performance of the true metrics needed to compare with the paper's model.

TABLE II

COMPARISON OF PAPER ESIM TO OUR REPRODUCTION RESULTS

Model	R@1	R@2	R@5	MRR
ESIM	0.696	0.820	0.954	0.802
Our Reproduction	0.603	0.741	0.921	0.733

We implemented algorithms to evaluate R@k and MRR. The output of the ESIM was a one-dimensional vector with a probability for each each response. This was cast as a numpy array and then joined as a column of a data-frame, with the other columns as context, response and label. The pandas library [15] was used as it allows fast sorting by column. The data-frame is first sorted by prediction and then by context. This now grants the most likely responses for each context in descending order. Then, by evaluating these 10 responses at a time, the desired metrics could measured.

IV. EMPIRICAL RESULTS

Given the complexity of the model and the size of the dataset, the training time was much longer than expected. We were able to train the model to c. 9,750 batch iterations. Our best model performed with an MRR of 0.733, whereas the baseline ESIM model stated in the paper performed with an MRR of 0.802 (Table. II). By looking solely at our best model, we can see that we were not able to reproduce the results shown in the paper. However, it is important to note that our results were achieved on a batch step of 9,750, whereas the author stated that their best results were achieved at batch steps between 22,000 and 25,000.

We also saved results from the metrics used while training to try to see a trend during training. Unfortunately, we only started logging theses values at a batch step of 3,500. If logging had started from time 0, a more classical learning curve would have been observed. The accuracy on the validation set fluctuates over time to a greater extent than the accuracy on the training set. However, we can see that both accuracies are still close and trend upwards, indicating that the model could have been trained further before the training accuracy differs from the validation accuracy. Since the validation accuracy never diverges, the model never reaches a point of overtraining. Therefore, given more iterations, we could have trained the model longer and achieve a better accuracy.

As for the loss metric, at the end of training we can see that the training set starts to have a slightly higher loss than the validation set. It could be argued that one reason for such behaviour is that the datasets have a different class distribution. Indeed, in the validation set, since we have 9 distractors for each correct response, the data representing correct responses accounts for 10% of the dataset, whereas the training set contains a perfectly balanced distribution between correct responses and distractors.

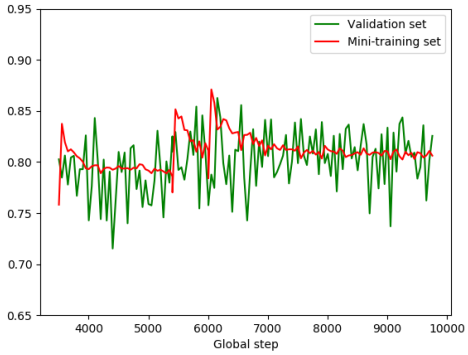

Fig. 2. Accuracy during training.

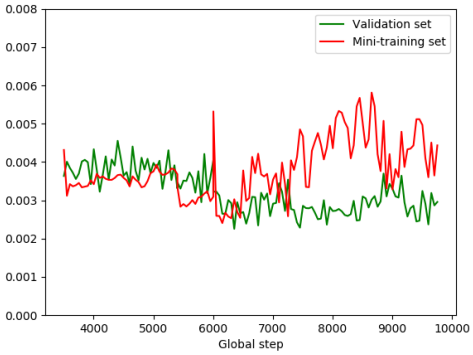

Fig. 3. Loss during training.

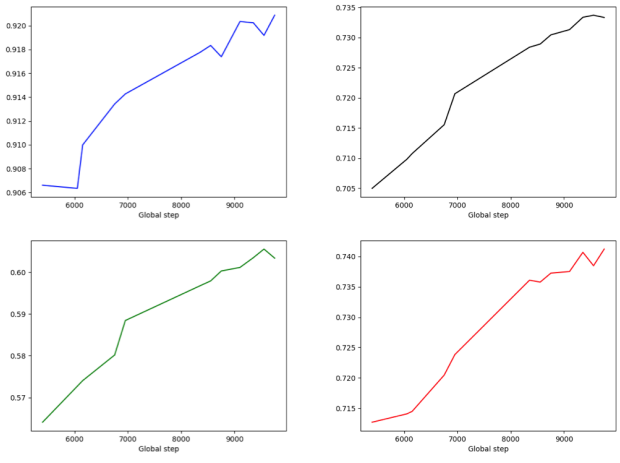

Fig. 4. Reproduced ESIM Performance Metrics: R@1 (bottom left), R@2 (bottom right), R@5 (top left), MRR (top right).

The final metrics used to evaluate our reproduction against the paper were applied on the model, at spaced intervals. Evenly distributed results were observed across batches between 5,300 and 9,750. As shown in Fig 4, all curves demonstrate increasing performance over time in a linear manner. If we compare these metrics with the one used while training, we can see that the accuracy and the loss metrics are not representative of the results that we are getting with the Recalls and MRR metrics. As an extension to our reproducibility efforts, evaluating the model with the Recalls and MRR metrics while training would be more beneficial than evaluating the accuracy and the loss. These results would be logged at every 50 batch steps, as we have done with the other metrics to get a clearer idea of when the model is overtrained.

As a result, given that we see an exclusively positive linear trend in the Recall and MRR graphs, we can say that if our model had been trained further (perhaps to 22,000-25,000 iterations) we would achieve greater performances that would be similar to the results of the ESIM, as stated in the paper.

V. DISCUSSION

The Ubuntu Corpus v2.0 corpus was very accessible, with the source code required to generate it hosted on Github. In order to download the dataset, we executed `create_ubuntu_dataset.py` 3 times (once for each dataset). One issue faced when generating this caused inclusion of special characters within the corpus. This caused a read function called by `create_ubuntu_dataset.py` to crash. This bug was fixed by modifying the source code to read in binary mode. Pre-preprocessing the dataset in the same manner as the paper was implementable too, using the PTB Tokenizer included in the Stanford CoreNLP library [16]. This operation was slow as we could not pass multiple sentences at a time, splitting by the new-line character, as raw text often included these characters within a single sample. Another factor slowing this operation was the design decision to implement the tokenizer in Python instead of Java. This necessitated an API call to an instance of the CoreNLP Java server with each new line rather than a more streamlined process in Java that the Stanford CoreNLP library was designed for.

We were not able to access the source code of the paper's authors due to issues regarding their employer's open source policy. Therefore, in order to replicate the findings of the paper, we based our model on another paper's implementation of the ESIM model, that we found on Github. This model was implemented in Python using the TensorFlow library version 1.4.1 (tensorflow-gpu) [12], which differs from the version 1.1.0 that was used by the paper that we replicated. The model required some changes regarding its output and the hyperparameters used.

The majority of hyperparameters required for a true replication were specified within the paper. All other hyperparameters were obtained by contacting the authors via OpenReview. The authors replied quickly and comprehensively within our first two interactions within the comments section

of OpenReview. The authors informed us that the default random seed was used to generate the dataset ('1234'). As for hyperparameters, no patience hyperparameter and no gradient clipping hyperparameter were used. We were not able to implement the model with a sequence length of 180, due to RAM constraints. Therefore, we reduced the sequence length size to 100. This change in the implementation could definitely be a factor in terms of model performance, given that contexts from the Ubuntu Dialog Corpus can often reach a word length higher than 100.

The source code was adapted to suit the output required for the next response modelling task. Rather than outputting a discrete value, a continuous output in [0, 1] was required, that could correspond to the probabilities that we would then rank per context. Thus, we changed the softmax activation functions units in the output layer to sigmoid units.

The paper does not detail the computing infrastructure that was used. For our implementation we used a Google Cloud Engine (GCE) instance that runs Ubuntu 16.06.3 LTS on a 2.3GHz Intel Xeon E5 v3 (Haswell) platform and employs one Nvidia Tesla K80 GPU. The CUDA toolkit [17] was installed in order for Tensorflow to benefit from GPU acceleration. As stated earlier, the author mentioned the greatest performance (measured by MRR) on the validation set when training was seen at around 22,000 - 25,000 batch steps. We were only able to train the model to 9,750 batch steps given our implementation architecture. This took to 65 hours. It is very likely that the authors trained their model using multiple GPU units, which would be unfeasible for us given the costs on a virtual machine. GCE prices a similar architecture, adjusted to include 4 GPU units, at $1,508.58 per month.

Overall, we found the paper to be clear and concise. However, we found it difficult to implement the authors' enhancements to the ESIM model. In particular, the training of character-composed embeddings is briefly described only as the concatenation of final state vectors at the BiLSTM. Further communication with the authors did not clarify enough for our purposes how exactly these embeddings are concatenated with word embeddings within the model. Only the ESIM was reproduced, as a result of this lack of clarity.

VI. CONCLUSION

While our implementation of the ESIM displayed a lower performance than that of the paper, we were limited by hardware constraints that forced us to train the model with less iterations, and a smaller sequence length. However, we were pleased to find that the model can be re-computed. The open-source nature of the code used in generating the corpus, pre-processing it to a dataset and training the model, combined with our fruitful clarification conversations with the papers authors aided our reproduction. For future studies, implementing the model within TensorFlow, or a similar high performance deep learning library, to harness a multi-GPU setup could confer a model that validates the results of the paper in question.

STATEMENT OF CONTRIBUTIONS

We hereby state that all the work presented in this report is that of the authors.

Hugo implemented the preprocessing methods, implemented the ESIM model, generated the graphs and the results, and helped write the Empirical Results section of the paper.

Isaac managed the project, wrote the majority of the report and reviewed it, implemented the evaluation metrics and helped implement the ESIM model.

Alex helped on the preprocessing methods, implemented the evaluation metrics, experimented on word embedding generations, reviewed the report and exported the final report in the correct format.

REFERENCES

[1] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, May 2015.

[2] B. K. Olorisade, P. Brereton, and P. Andras, "Reproducibility in machine learning-based studies: An example of text mining," 2017.

[3] Anonymous, "Enhance word representation for out-of-vocabulary on ubuntu dialogue corpus," *International Conference on Learning Representations*, 2018.

[4] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced lstm for natural language inference," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, (Vancouver), ACL, July 2017.

[5] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[7] R. Lowe, N. Pow, I. Serban, and J. Pineau, "The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems," *arXiv preprint arXiv:1506.08909*, 2015.

[8] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[9] Y. Wu, W. Wu, M. Zhou, and Z. Li, "Sequential match network: A new architecture for multi-turn response selection in retrieval-based chatbots," *CoRR*, vol. abs/1612.01627, 2016.

[10] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[11] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[13] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermüller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. B. Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. de Brébisson, O. Breuleux, P. L. Carrier, K. Cho, J. Chorowski, P. F. Christiano, T. Cooijmans, M. Côté, M. Côté, A. C. Courville, Y. N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. E. Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. J. Goodfellow, M. Graham, Ç. Gülçehre, P. Hamel, I. Harlouchet, J. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov, V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrançois, S. Lemieux, N. Léonard,

Z. Lin, J. A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P. Manzagol, O. Mastropietro, R. McGibbon, R. Memisevic, B. van Merriënboer, V. Michalski, M. Mirza, A. Orlandi, C. J. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. Schlüter, J. Schulman, G. Schwartz, I. V. Serban, D. Serdyuk, S. Shabanian, É. Simon, S. Spieckermann, S. R. Subramanyam, J. Sygnowski, J. Tanguay, G. van Tulder, J. P. Turian, S. Urban, P. Vincent, F. Visin, H. de Vries, D. Warde-Farley, D. J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, and Y. Zhang, "Theano: A python framework for fast computation of mathematical expressions," *CoRR*, vol. abs/1605.02688, 2016.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[15] W. McKinney, "pandas: a foundational python library for data analysis and statistics,"

[16] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit.," 2014.

[17] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, pp. 40–53, Mar. 2008.

APPENDIX
PUBLIC REVIEW POSTED ON OPENREVIEW

Introduction

As participants in the 2018 ICLR Reproducibility Challenge, we aimed to reproduce the findings of this paper. The paper presents a methodology to improve performance on modelling dialogue systems that contain out-of-vocabulary words. Ultimately, we implemented Chen et al.'s Enhanced LSTM Method (ESIM) to model the Ubuntu Dialogue Corpus v2.0 as presented in the paper.

Overall, we found the paper to be clear and concise. However, we found it difficult to implement the authors enhancements to the ESIM model. In particular, the training of character-composed embeddings is briefly described only as the concatenation of final state vectors at the BiLSTM. Further communication with the authors did not clarify enough for our purposes how exactly these embeddings are concatenated with word embeddings within the model. For this reason, as stated above, we decided only to replicate the ESIM.

Reproduction

Downloading and preprocessing the Ubuntu Dialogue Corpus and pre-trained GloVe was a simple matter of following the procedure specified in the paper. Using publicly available datasets definitely facilitated reproducibility. In generating the dataset, all default parameters were used, with a random seed of 1234 that the authors provided upon inquiry.

The generated data was modified into formats appropriate for Word2Vec and ESIM inputs. The dataset was tokenized using the publicly available Stanford CoreNLP library PTB Tokenizer, and then lemmatized. We then used a stored set of distinct tokens to filter the pre-trained GloVe vector, removing all words that do not appear in the training corpus. We thought this step would be beneficial since the unzipped glove dataset (glove.42B.300d.txt) is 4.67 GB large, which would take a considerable amount of memory simply to load it. The filtered GloVe dataset takes about 440 MB and contains roughly 9% of the original GloVe dataset. Through this process, we confirmed the authors observation that only 22% of the 823,057 Ubuntu tokens occur in the pre-built GloVe word vectors, and that our reproduction produced the same dataset.

To reproduce the baseline ESIM model, we were not able to access the source code of the papers authors due to issues regarding their employers open source policy. Instead, we implemented the ESIM using source code of an implementation by Williams et al., that was found on GitHub. We followed all hyperparameters specifications possible, and when particular hyperparameters were not provided, we consulted the authors who provided further detail. Specifically, these hyperparameters were patience, and gradient clipping threshold, and max epochs. Of these, the authors stated that patience and gradient clipping were not used, and that training usually achieved the best performance (MRR) on the validation set at around 22,000 - 25,000 batch steps. In general, the authors replied quickly and comprehensively to our enquiries within the comments section of OpenReview, which contributed positively to the reproducibility of the paper.

When training the model, performance metrics were printed every 50 steps. Accuracy and cost over the validation set and over a subset of the training set were employed to evaluate the training of the model. We did not evaluate over the whole training set since this is significantly larger and would greatly slower the training time. We implemented our own algorithms to evaluate R@k and MRR.

The paper does not detail the computing infrastructure that was used. For our implementation we used an Google Cloud Engine instance (full technical specifications in the linked report).

We were only able to train the model to 9,750 steps given our implementation architecture. This took 65 hours to train. This is considerably less than the 22,000-25,000 steps described by the authors as providing the best results. It is very likely that the authors trained their model using multiple GPU units, which would be unfeasible for us given the cost of GPUs on a virtual machine.

Results and Conclusion

Given our limited computation power, we were able to train our reproduction model to the same level of performance as described by the authors. We observed a MRR of 0.733, lower than the MRR of 0.802 seen in the paper in question. However, on account of the methodology followed in our investigation, we can confirm that the model is recomputable. It seems highly likely that the papers results are valid, and would have been observed by us had our model been trained with more iterations.

The full report by H. Scurti, I. Sultan, and A. Wong is contained in the folder "report" in the repository linked below:
`https://bitbucket.org/hugoscurti/comp551_f17_a4/src/`