COMP551-01 Final Project Report

T3-07: Byte-level Machine Reading across Morphologically Varied Languages

Team 1MorePlate

Yi Qiao Wang
yi.wang14@mail.mcgill.ca
260682080

Mathew Wright
mathew.wright@mail.mcgill.ca
260681105

Vince Porporino
vincenzo.porporino@mail.mcgill.ca
260681922

I. Introduction

One of the long-standing problems in the natural language understanding branch of artificial intelligence research is machine reading. This is the problem whereby machines are fed documents and answer questions about them.

Currently, the state of the art models for English machine reading tasks are end-to-end trained word-level recurrent neural networks. Word level models are especially suited for tackling English machine reading tasks due to the limited structural richness and complexity of the English language. Indeed, its relatively limited morphology allows machine learning algorithms to represent a broad category of word types while also maintaining reasonable model complexity [1].

Conversely, more morphologically rich languages like Russian and Turkish have many additional word types due to varied prefix and suffix constructions. As a consequence, even complex models employing large vocabularies struggle to provide extensive coverage. In addition, larger vocabularies lead to an increase in the number of parameters required in word embeddings while more word types results in fewer training examples available per word type.

In their paper "Byte-level Machine Reading across Morphologically Varied Languages", Kenter et al. claim that models utilizing byte-level encodings of natural language are advantageous for complex language tasks. Specifically, byte-level models are favorable because the byte representation provides a universal encoding format for all languages all the while keeping the vocabulary fixed at 256 tokens. This fixed vocabulary size delivers significantly reduced computational overhead. Aside from the improved efficiency, evaluation of byte-level models is less biased compared to their word-level equivalents since byte-level models remove the need for a morphology decomposition, use a predetermined vocabulary size, and contain fewer hyper-parameters to choose from.

The primary purpose of this report is to verify the main claim introduced by the authors. That is, representing inputs in the byte level is beneficial to performance in the machine reading task, particularly in morphologically rich languages. As an aside, we also aim to gauge and quantify the effort required to achieve the absolute performance as reported by the authors. To address these goals, we first outline and provide motivation for the aforementioned aspects of the paper which we intend to reproduce. Subsequently, details and considerations of our implementation methodology and experimental setup are provided. Finally, we present and analyze our findings by discussing whether or not the paper's claims are reproducible. In doing so, we detail the challenges faced along with suggested solutions.

II. Reproducibility Goals

The primary claim that we aim to reproduce is that byte-level encoder-decoder models perform better on morphologically rich languages when compared to their word-level counterparts. Due to our computational and resource constraints, the scope of our reproduction was narrowed down from that of the original paper. In particular, the languages over which these models were evaluated on was also narrowed down. Instead of training and evaluating models on English, Turkish and Russian, it was decided that Turkish would be the only language used. Since our intent is to reproduce the claim regarding the improvement of byte-level languages on morphologically rich languages, this decision is justified. From among the three languages with available datasets, Turkish is the most morphologically rich [1].

In addition to focusing our efforts on a single dataset, only a subset of the models presented by Kenter et al. were implemented and evaluated. From among the five different encoder-decoder models, only the multi-level recurrent neural network (RNN) and the bidirectional multi-level recurrent neural network (Bi-RNN) were chosen. The multi-level RNN was chosen because it is the top performer on the Turkish data, and the bidirectional RNN model was chosen to verify one of the paper's unexpected minor conclusions that bidirectional RNN do not yield better results than unidirectional models. To keep results comparable, we maintained the same baseline employed by Kenter et al.

Lastly, instead of training on the entirety of the available set of training examples, we chose to train on a subset of the data. This decision is again due to computational constraints. We would like to make the remark that training on a smaller set of data will evidently yield worse results when compared to training on the full data set, as was done in the original paper. However, this decision is justified in that we do not seek to improve upon or match the absolute performance of

the byte-models, but rather their relative improvement over their word-level analogues.

In addition to this primary goal, we seek to quantify the efforts required to reproduce the performance results Kenter et al. observed. Specifically, we evaluate the feasibility of achieving the reported absolute prediction performance, with respect to our particular set of computational constraints.

III. Implementation Methodology

The methodology used throughout the project as well as its justification is presented below.

In the implementation of the models and procedures described in subsequent sections, several popular software libraries were used. General purpose linear algebra operations were facilitated through the NumPy package. In addition to NumPy, the Python deep learning library PyTorch was used in the implementation of the various RNN models. Lastly, model prototyping, and testing was done on Google Colab, while training was done locally.

A. Datasets

The authors of the paper demonstrated their results on three different datasets.

The English dataset, called WikiReading, was taken from a paper by Hewlett et al. [2]. This dataset is comprised of examples where each entry is a *(document, property, value)* triple where *property* and *value* were taken from Wikidata and *document* contains information from Wikipedia. Due to its construction, the authors note that the WikiReading dataset is a particularly difficult dataset for machine reading tasks in the sense that the answer (*value*) of the query (*property*) is not necessarily present in the text (*document*) verbatim. Instead the computer must, in certain instances, infer the correct response to a given question.

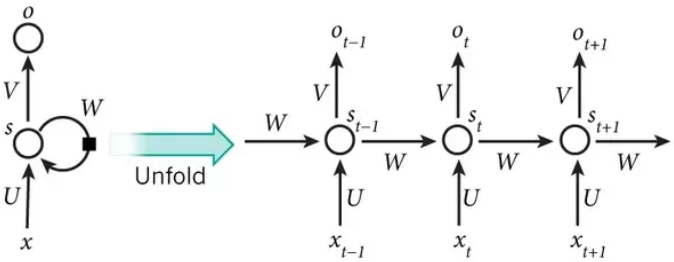
Besides WikiReading, Kenter et al. also contribute two novel datasets, one in Turkish and the other in Russian, constructed in the same way as WikiReading. It is on these datasets that the byte-models show a significant advantage and thus draw our attention. In our reproduction, we choose to only investigate the Turkish dataset as it is the language that is the most morphologically rich among the three [1].

B. Machine Learning Models

To better understand subsequent sections, a brief overview of the the algorithms pertaining to the machine learning models utilized in the machine reading task is provided.

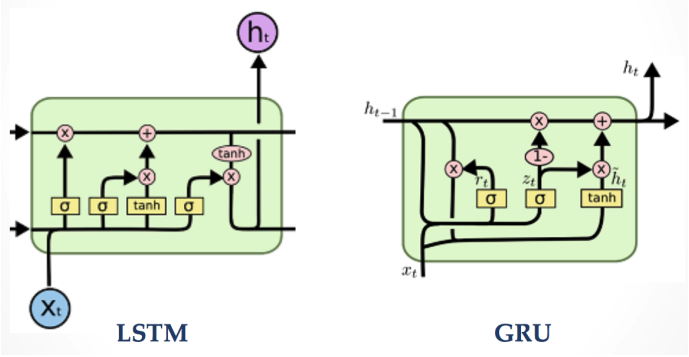
1) Recurrent Neural Networks: A recurrent neural network is a specialized form of the more general artificial neural network (ANN) whose architecture incorporates recurrent dependencies between hidden states. In particular, this property allows RNN models to model temporal relationships by storing relevant information in their hidden states, making them especially suitable for modeling and representing sequential data. Thus, they are a natural fit for tackling natural language problems, where inputs at a certain point in time are almost guaranteed to have been influenced by those that occur previously.

Fig. 1. RNN architecture and unrolled representation [3]



An important and relevant detail of the RNN model is the hidden or internal state (often called the RNN cell). This component of the RNN is what is responsible for acting as "memory", and is therefore the distinguishing mechanism that allows for an RNN to model temporal dependencies. Two types of RNN cells that are widely used in research and in practice is the long short-term memory (LSTM) cell and the gated recurrent unit (GRU). These units are both constructed of "gates" that can be viewed as traditional ANN neurons, each with its own activation function and associated weights. When combined together into units, these individual gates work together to provide the necessary mechanisms that allow for the modeling of temporal data.

Fig. 2. LSTM and GRU cells [4]



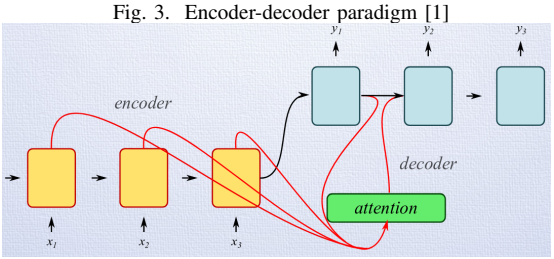
2) Encoder-decoder: Encoder-decoder (*enc-dec*) models are a class of machine learning models that specialize in sequence to sequence (*seq2seq*) learning. These models are composed of two RNNs trained in an end-to-end fashion. As their name suggests, one RNN acts as an encoder and the other a decoder.

At a high level, an *enc-dec* model takes in a sequence of text and runs it through the encoder RNN. It is not the output of this encoder RNN that is of interest however. The purpose of this preliminary RNN is to encode the input sequence, in our case the document and the query, into a representation that will be more useful than the original input itself.

Concretely, it is the hidden states of the encoder RNN that are used as the initial hidden states of the decoder RNN. Once the decoder RNN has the hidden states of the encoder,

it "decodes" the information encoded previously to produce an output, in our case the answer to the posed question. This type of model is trained end-to-end. This signifies that both the encoder and decoder are viewed as one in the sense that backpropagation is computed all the way from the output of the decoder to the input of the encoder.

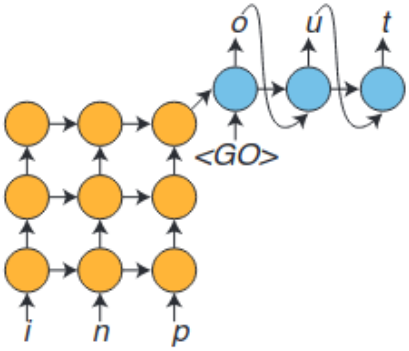
Fig. 3. Encoder-decoder paradigm [1]



3) Multi-level and Bidirectional Recurrent Neural Networks: The multi-level RNN is a generalization of the single layer RNN. Like the multi-level ANN, this added model complexity allows the algorithm more power by providing additional parameters and by allowing the model to learn more complex and varied transformations.

The bidirectional multi-level RNN is an even further generalized version of the multi-level RNN. Its main feature is that it "encodes" the input sequence in both directions, thereby increasing the data available to the encoder. This increase in available information is done by feeding the input sequence to two separate RNNs, one of which is fed the reversed sequence. Then, the hidden states of both RNNs are combined and used as the initial hidden state of the decoder. By including the reversed sequence, it is possible for one state to not only depend on what occurs in previous time steps, but to also peer into future time steps, should it be beneficial.

Fig. 4. Multi-layer RNN [1]



C. Experimental Setup

The experimental setup we used closely followed that detailed by Kenter et al. This is a natural decision, as we did not want to introduce bias in our own results by altering the setting in which the original results were obtained.

1) Encoders: The multi-level RNN encoder uses multiple stacked recurrent cells instead of only one, with each level

Fig. 5. Bidirectional RNN [1]

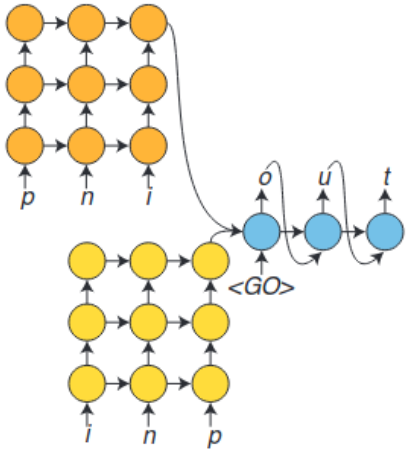


TABLE I
TUNED HYPER-PARAMETERS AND CONSIDERED VALUES

Embedding size	128, 256
Internal state size	256, 512
Number of stacked RNN cells	1, 2
RNN cell type	GRU, LSTM
Learning rate	10^{-5}, 10^{-4}, 10^{-3}

maintaining its own set of weights. The bidirectional encoder employs two multi-level RNNs, where the final output state is a concatenation of the final states of each individual encoder projected down to that of a single encoder. Model selection via hyper-parameter tuning was also done. The values for these hyper-parameters are the same as those considered by Kenter et al. This information is given in table I.

To evaluate the byte-level models, a baseline encoder comprising of a word-level encoder-decoder RNN was implemented. The baseline's hyper-parameters were not tuned, but rather taken from the word-level model that produced the highest accuracy on the English WikiReading dataset as reported by Hewlett et al. [2]. This decision was made in order to remain consistent with the original paper and to achieve comparable results. Like Kenter et al. the embeddings were not pre-trained in the baseline word model. The specific parameters are given in table II.

2) Decoder: Each of our *seq2seq* models are trained end-to-end with the same decoder for both byte-level models. This decoder is a single hidden layer RNN utilizing softmax activation and computing cross-entropy loss at each time step. In addition, the decoder applies an attention mechanism to

TABLE II
WORD LEVEL BASELINE HYPER-PARAMETER VALUES

Embedding size	300
Internal state size	1024
Vocabulary size	100000
RNN cell type	LSTM
Number of stacked RNN cells	1
Learning rate	10^{-2}

all hidden states of the encoder. The decoder's recurrent cell matched that used by the encoder.

3) Training and Evaluation: As in the authors' experiment each query sequence and document sentence in the byte-level model has its length limited to 50 bytes and 400 bytes respectively. Such a truncation is valid and justified by computational limitations as well as the fact that questions do not often exceed this length [1]. The response's length was also capped at 50 bytes. An interesting observation made by Kenter et al. is that the 400 byte document length limitation is sufficient for answering most questions. They hypothesize that this is due to Wikipedia's particular writing structure. In particular, it is often the case that the first paragraph of a Wikipedia is indeed very dense and informative. In the word-level baseline, each document is limited to 60 words. No restrictions were placed on the query and response lengths.

The hyper-parameter tuning methodology we follow is described as follows. As the original paper did not outline any particular methods, we first conducted a coarse, high-level, search over the hyper-parameter space to identify promising configurations. Once this subset of better performing models was found, a more thorough optimization was undertaken.

Specifically, we employed a process of elimination involving multiple steps. Each step in the process involved removing the worst performing hyper-parameter combinations from consideration and repeating optimization on the remaining models using larger datasets. This greedy approach is used due to computation restrictions. Indeed, tuning over all configurations on large amounts of data is not feasible given our available resources.

All models were optimized with respect to Adam. This choice of optimization strategy was used to remain consistent with the original paper. In addition, Adam is also suitable for the RNN model as it maintains individual learning rates for each parameter that depends on the most recent k magnitudes of the weight gradient. This allows it to help address the vanishing and exploding gradient problem commonly observed in the RNN algorithm.

For the purpose of evaluation, only predicted answers that were identical to the true answer were considered correct. That is, no normalization or stemming, aside from dates, were considered.

IV. Results

Unfortunately, we were not able to achieve meaningful predictions on test data. We hypothesize that this is mainly attributed to the amount of data that the models were trained on. In the original paper, Kenter et al. were able to train on multiple hundreds of thousands of examples, with the WikiReading dataset comprising of sixteen million examples. This scale was simply not attainable with our given resources. An example of the poor predictive performance is given in table III.

While we were not able to achieve any form of predictive accuracy, we were able to collect noteworthy results about the loss during both the training and testing processes. We observed that the average test loss per batch was higher in the byte-level models compared to the baseline. As in Kenter et al., it was noted that the bidirectional RNN did not perform better than the multi-layer RNN at minimizing training loss.

TABLE III
Prediction example of byte-level multi-layer RNN

Prediction	Target
iiiiiiikiiTieeeeiiiiiiiiiiiiiiiiiiiieiiiiiiiiiei	Krdistan Eyaleti

In terms of cross-entropy loss on test data, we see that the average loss per test batch is indeed lowest in the byte-level models compared to the word-level baseline. Furthermore, the previously observation regarding the loss difference between the multi-layer and bidirectional RNN models appears again on testing data. Again, it appears that the bidirectional RNN does not improve upon the loss of the multi-layer RNN.

Fig. 6. Average training batch loss per epoch

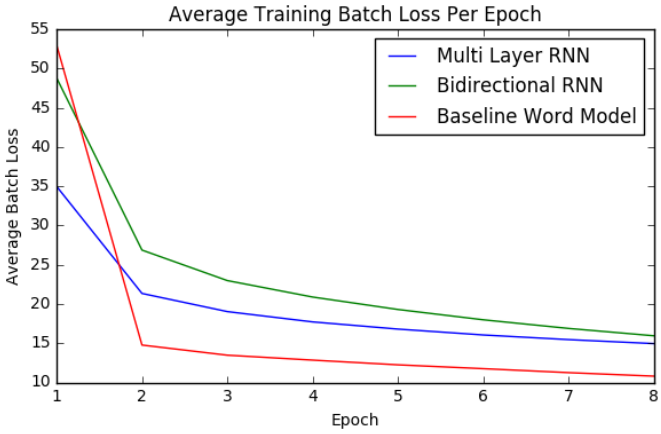
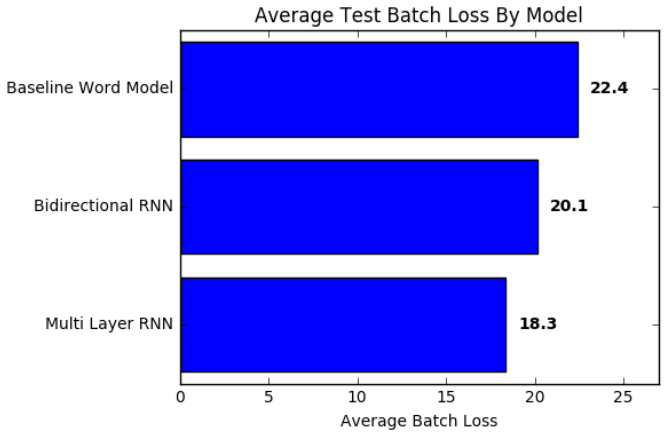


Fig. 7. Average test batch loss by model



V. Discussion

While we were not capable of achieving the desired predictive accuracy that was originally intended and thus unable to directly reproduce the results presented in the original paper, there are still a few takeaways worth mentioning.

The primary challenge we faced was the amount of training data required for meaningful predictions. However, we believe that this is not due to a shortcoming on the authors' part but is due to the complexity and difficulty of the machine reading problem. This difficulty is only exacerbated by our constrained resources. One of the benefits of the byte-level model, as investigated by Kenter et al. is the performance increase associated with using a smaller, fixed vocabulary size of 256. Due to the comparatively large vocabulary sizes, training the word-level model took significantly more resources than the byte-level model. The word-level model needed as much as 70 gigabytes of memory, split between RAM and swap space, to train with a batch size of 50 documents. The byte-level model only required significantly less, using at most 5 gigabytes of memory to train using the same batch size.

That said, we hypothesize that given ample computation, the original paper's findings are reproducible. This is because we did notice improvements in the cross-entropy loss when utilizing the byte-level models. Moreover, the behavior and relative performance of the three models we considered in our reproduction matched that reported by Kenter et al. exactly. Their principle conclusion was that byte-level models perform better than word-level ones [1], which is what we observe in both the training loss and the testing loss. Additionally, one of their unexpected, yet remarkable, results is that, contrary to previous research, bidirectional RNNs do not outperform unidirectional RNNs [1]. Once again, this is present in our findings, albeit on loss rather than $F1$-measure.

VI. Conclusion

The purpose of our investigation was two-fold. First, we set out to verify the claim made by Kenter et al. that byte-level models provided an advantage over their word-level counterparts in machine reading tasks on morphologically rich languages. In this vein, we succeeded in showing a similar, but not identical result using two popular machine learning models on a Turkish language dataset. Indeed, we were able to observe that the cross-entropy loss of the byte-level model decreased much faster than that of the word-level model on the same data. However, we were not able to reproduce the absolute performance achieved by the original authors. This is largely due to the the fact that we were computationally limited. After building our prediction models based on the specifications outlined in the paper and training them on the Turkish dataset, we conclude that the performance obtained by the authors is not reproducible in cases using a small subset of the data and limited computational resources.

Although we were unable to replicate the original paper's claims, we can confirm one of the proposed benefits of the byte-level model; its computational advantage over word-level models. Not only did the byte-level models converge faster, allowing them to be trained on less data, they also imposed a drastically reduced memory overhead. This leads us to believe that a byte-level approach would be in fact be advantageous in the machine reading task on languages with rich morphology.

VII. Statement of Contributions

All team members were involved in defining the scope of the problem, developing the goals and methodology. Yi Qiao implemented the machine learning models while also producing the written report. Mathew and Vince were responsible for training the models and running the experiments. Mathew also performed the data processing and results analysis, while Vince completed the executive summary and contributed in the editing of both reports.

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

References

[1] T. Kenter, L. Jones and D. Hewlett, *A Framework for Designing the Architectures of Deep Convolutional Neural Networks*, in Proceedings of the The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), 2018. [Online]. Available: http://www.tomkenter.nl/pdf/kenter_byte-level_2018.pdf[Accessed April 21, 2018].

[2] D. Hewlett, A. Lacoste, L. Polosukhin, A. Fandrianto, J. Han, M. Kelcey and D.Berthelot, *WIKIREADING: A novel large-scale language understanding task over wikipedia*, in ACL, 2016.

[3] Y. LeCun, Y. Bengio and G. Hinton, A. Fandrianto, J. Han, M. Kelcey and D.Berthelot, *Deep Learning*, in Nature vol. 512, pp. 436-444, 2015. [Online]. Available: https://www.researchgate.net/profile/Y_Bengio/publication/277411157_Deep_Learning/links/55e0cdf908ae2fac471ccf0f/Deep-Learning.pdf?origin=publication_detail[Accessed April 21, 2018].

[4] J. Sun, *Recurrent Neural Networks*. [Online]. Available: http://www.sunlab.org/teaching/cse6250/fall2222/lab/dl-rnn[Accessed April 21, 2018].

[5] O. Mogren, *Recent Advances in Neural Machine Translation*. [Online]. Available: http://mogren.one/talks/2016/09/29/nmt.html[Accessed April 21, 2018].