Naval Research Laboratory
Washington, DC 20375-5320

NRL/7340/MR—2024/5

Simulated Document Parsing Artificial Intelligence System

CHARLES NORSWORTHY

Center for Geospatial Sciences Branch
Ocean Sciences Division

April 25, 2023

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
25-04-2024	NRL Memorandum Report	5/23/23 – 9/22/23

4. TITLE AND SUBTITLE

Simulated Document Parsing Artificial Intelligence System

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Charles Norsworthy

5d. PROJECT NUMBER

5e. TASK NUMBER
BE031-03-42

5f. WORK UNIT NUMBER
1Y90

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
1005 Balch Boulevard
Stennis Space Center, MS 39529

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/7340/MR--2024/5

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
875 N Randolph St.
Arlington, VA 22217-1992

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper introduces an encoder-decoder network with attention that can be trained to convert a simulated document into a JSON string. In the documented experiment with this system, it achieved an accuracy of 99.99% on its simulated testing documents.

15. SUBJECT TERMS

Encoder-decoder network | Long short-term memory | LSTM | Bahdanau attention
Artificial intelligence system | Document parsing AI | Simulated document | KeyValuePair

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Charles Norsworthy
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	U	9	(228) 688-4323

This page intentionally left blank.

CONTENTS

This page intentionally left blank.

EXECUTIVE SUMMARY

This paper introduces an encoder-decoder network with attention that can be trained to convert a simulated document into a JSON string. In the documented experiment with this system, it achieved an accuracy of 99.99% on its simulated testing documents.

This page intentionally left blank.

SIMULATED DOCUMENT PARSING ARTIFICIAL INTELLIGENCE SYSTEM

1. INTRODUCTION

This report documents an artificial intelligence system that can convert a simulated document into a JSON [1] string. A document parsing AI system is different than a system that simply reads the text in a document. Instead, a document parsing AI system places the text that is present in a document into a structured JSON file. Although much work needs to be done with the system described in this report, significant progress was still made. This research is for the U.S. Naval Research Laboratory's Aero Document Data Extraction project, in which we need to parse documents from the Federal Aviation Administration [2] into JSON.

2. APPROACH

The training, validation, and testing data for this report's AI system is simulated, meaning it does not use data generated from actual documents. The data simulates an extremely small document with only text written inside it. Each simulated document will contain text with only the digits 0-9. These digits have only two characteristics: bold or non-bold. In these simulated documents, one string, in bold, is the key, and one or two strings, not in bold, are the values for the key. Each string in the data can only have 1-10 characters, and this number is selected randomly. Each digit in a string is also selected randomly.

These simulated documents simulate KeyValuePair objects described in "Synthetic Data Generation Project for a Document Parsing AI" [3]. These simulated documents simulate four value types of KeyValuePair objects: right_offset, left_under, right_offset_list, and left_under_list. The right_offset and left_under value types have only one string for the value. For this report, the right_offset_list, and left_under_list value types have only two strings for the value.

The value for a right_offset KeyValuePair "is one string to the right of the key" [3]. An example is shown in Figure 1.

Fig. 1

A parsed JSON representation of the document in Figure 1 is shown in Figure 2.

```
{"7353798":"28105"}
```

Fig. 2

The value for a left_under KeyValuePair "is one string under the key and aligned to the left of the key" [3]. An example is shown in Figure 3.

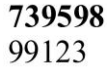

Fig. 3

A parsed JSON representation of the document in Figure 3 is shown in Figure 4.

Manuscript approved March 18, 2024.

Fig. 4

The value for a right_offset_list KeyValuePair "is a list of strings to the right of the key". "Each string in this list of strings is placed under the first string in the list and is aligned to the left of the first string in the list" [3]. An example is shown in Figure 5.

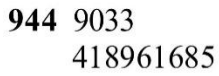

Fig. 5

A parsed JSON representation of the document in Figure 5 is shown in Figure 6.

Fig. 6

The value for a left_under_list KeyValuePair "is a list of strings aligned left under the key" [3]. An example is shown in Figure 7.

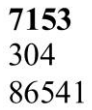

Fig. 7

A parsed JSON representation of the document in Figure 7 is shown in Figure 8.

Fig. 8

Note that the training, validation, and testing data used by the simulated document parsing AI are not real documents such as the documents shown above.

A digit in a simulated document is stored in memory using one-hot encoding [4] with a cardinality of 47. A series of 4 one-hot encoded vectors describe a single digit, encoding what the digit is (0-9), the digit's x coordinate, the digit's y coordinate, and whether the digit is bold. Consider a document with 30 digits, corresponding to a simulated document with the greatest number of digits possible for this report. This document's input encoding will be a list of 30 * 4 = 120 one-hot encoded vectors of length 47. The output of the network can be written as a string of characters, which, if the network produces a correct output, will be a JSON string.

The network that is trained is an encoder-decoder network [5] with an attention [6] layer. The network uses Long Short-Term Memory [7] layers and each LSTM layer has 50 units. The encoder is composed of two bidirectional [8] LSTMs. The decoder is composed of four LSTMs. The attention layer is an additive attention layer, also known as Bahdanau attention [6]. This model contains some of the recommendations from an article entitled "How to Configure an encoder-decoder Model for Neural

Machine Translation" [9]. The model is implemented using the Keras library [10], a library which was also used to produce the complete network summary shown below in Figure 9.

```
Model: "KeyValuePairParser"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 network_in (InputLayer)        [(None, None, 47)]   0           []

 dec_in (InputLayer)            [(None, None, 200)]  0           []

 bdr_enc_1 (Bidirectional)      [(None, None, 100),  39200       ['network_in[0][0]']
                                 (None, 50),
                                 (None, 50),
                                 (None, 50),
                                 (None, 50)]

 dec_lstm_1 (LSTM)              [(None, None, 50),   50200       ['dec_in[0][0]',
                                 (None, 50),                      'bdr_enc_1[0][1]',
                                 (None, 50)]                      'bdr_enc_1[0][2]']

 dec_lstm_2 (LSTM)              [(None, None, 50),   20200       ['dec_lstm_1[0][0]',
                                 (None, 50),                      'bdr_enc_1[0][3]',
                                 (None, 50)]                      'bdr_enc_1[0][4]']

 bdr_enc_2 (Bidirectional)      [(None, None, 100),  60400       ['bdr_enc_1[0][0]']
                                 (None, 50),
                                 (None, 50),
                                 (None, 50),
                                 (None, 50)]

 dec_lstm_3 (LSTM)              [(None, None, 50),   20200       ['dec_lstm_2[0][0]',
                                 (None, 50),                      'bdr_enc_2[0][1]',
                                 (None, 50)]                      'bdr_enc_2[0][2]']

 dec_lstm_4 (LSTM)              [(None, None, 50),   20200       ['dec_lstm_3[0][0]',
                                 (None, 50),                      'bdr_enc_2[0][3]',
                                 (None, 50)]                      'bdr_enc_2[0][4]']

 dec1-4out (Concatenate)        (None, None, 200)    0           ['dec_lstm_1[0][0]',
                                                                  'dec_lstm_2[0][0]',
                                                                  'dec_lstm_3[0][0]',
                                                                  'dec_lstm_4[0][0]']

 enc1-2out (Concatenate)        (None, None, 200)    0           ['bdr_enc_1[0][0]',
                                                                  'bdr_enc_2[0][0]']

 attn_layer (AdditiveAttention)  (None, None, 200)   200         ['dec1-4out[0][0]',
                                                                  'enc1-2out[0][0]']

 dec-out_attn-out (Concatenate)  (None, None, 400)   0           ['dec1-4out[0][0]',
                                                                  'attn_layer[0][0]']

 dense (Dense)                  (None, None, 19)     7619        ['dec-out attn-out[0][0]']

==================================================================================================
Total params: 218,219
Trainable params: 218,219
Non-trainable params: 0
__________________________________________________________________________________________________
```

Fig. 9

Before training the network, the training and validation data are generated. The training data has 20,000 simulated documents created for each of the four KeyValuePair value types, equaling 20,000 * 4 = 80,000 simulated documents in total. The validation data has 4,000 simulated documents created for each of the four KeyValuePair value types, equaling 4,000 * 4 = 16,000 simulated documents in total. After training, an additional 4,000 simulated documents for each KeyValuePair value type are newly generated, equaling 4,000 * 4 = 16,000 simulated documents. This is to test the network. The code uses an exact accuracy metric for testing. With this metric, if the network has one character in its output string that is incorrect for a given input, then the network has 0% accuracy for that simulated document. Then, an additional 10 simulated documents for each KeyValuePair value type are newly generated, equaling 10 *

4 = 40 simulated documents. The network evaluates these as input and the code writes output from the network into a string so the developer can see which, if any, written outputs are incorrect and how they are incorrect.

3. EXPERIMENT

The network was trained for 50 epochs with a batch size of 16 and a learning rate of 0.0015. The network was trained using the Adam optimizer [11] with categorical cross-entropy loss [12]. Although the system was trained for 50 epochs, the code saves the network weights with the best validation loss. The network's highest accuracy on the training data during training was 99.94%. The network's highest validation score during training was 100.00%. After training, the accuracy of the saved network using the exact accuracy metric on the testing data was 99.99%.

4. CONCLUSION

This paper shows it may be possible for an encoder-decoder network with attention to parse real documents that each contain a KeyValuePair object, since it was able to parse simulated documents that each simulate a KeyValuePair object.

5. FUTURE WORK

Each simulated document in the training, validation, and testing data is randomly generated, and there is a chance that there are duplicate documents. This will need to be fixed in future work. This data is also not generated from real documents, so there may be missing nuance with the positions of characters in a document for a given font. Moving forward, I need to instead generate real PDF [13] documents and use Apache PDFBox® [14] to read the generated PDFs and get the characters, their positions, and whether or not they are bold. Then I could augment the data by moving around all the characters. I need to include letters (a-z, A-Z) in addition to digits (0-9) in the documents. The system may benefit from documents with more than 10 characters in each string. There are more KeyValuePair value types than just the four presented here. Thus, I may need to work on parsing additional KeyValuePair value types. Also, when parsing KeyValuePair value types with more than one string for the value, the network needs to be able to parse KeyValuePair objects with more than two value strings. I also need to work on parsing more objects than just KeyValuePair objects, such as Table [3] objects. I may also train a Transformer [15] network on simulated and real documents.

REFERENCES

1. JSON. (n.d.). *Introducing JSON.* https://www.json.org/json-en.html

2. U.S. Department Of Transportation. (n.d.). *Transmittal Letters.* Federal Aviation Administration. https://www.faa.gov/air_traffic/flight_info/aeronav/aero_data/Transmittal_Letters/

3. Norsworthy, C. (2022, December 13). *Synthetic Data Generation Project for a Document Parsing AI*. Defense Technical Information Center. https://apps.dtic.mil/sti/pdfs/AD1187927.pdf

4. Brownlee, J. (2020, June 12). *Ordinal and One-Hot Encodings for Categorical Data*. Machine Learning Mastery. https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/

5. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

6. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

7. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

8. Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11), 2673-2681.

9. Brownlee, J. (2019, August 7). *How to Configure an Encoder-Decoder Model for Neural Machine Translation.* Machine Learning Mastery. https://machinelearningmastery.com/configure-encoder-decoder-model-neural-machine-translation/

10. Keras. (n.d.). *Keras.* Keras. https://keras.io/

11. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

12. Keras. (n.d.). *Probabilistic losses.* Keras. https://keras.io/api/losses/probabilistic_losses/#categoricalcrossentropy-class

13. Adobe. (2023). *Everything you need to know about the PDF.* Adobe. https://www.adobe.com/acrobat/about-adobe-pdf.html

14. The Apache Software Foundation. (2023). *Apache PDFBox® - A Java PDF Library.* PDFBox®. https://pdfbox.apache.org/

15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.