PATTERN AS A FOREIGN LANGUAGE

TTIC 31210 PROJECT REPORT
SPRING 2017

Hao Jiang
The University of Chicago

ABSTRACT

The recurrent neural network (RNN) based encoder-decoder architecture has been widely used for various sequence-to-sequence translation tasks such as natural language translation and grammar inference, and has achieved significant success on these tasks. In this project, we attempt to use this architecture to infer common patterns from multiple inputs, which is a crucial task for information extraction and management. This leads to two new types of tasks: combinations and summarization. In combination task, we train the encoder-decoder with known patterns and attempt to use it to recognize the combination of these patterns. In summarization task, we define some rules to infer a common pattern from multiple records and use encoder-decoder to learn these rules. Preliminary results show that existing architecture does NOT fit these applications and new architectures are required.

1 INTRODUCTION

In this project, we explore the possibility of using RNN-based encoder-decoder architecture to do pattern extraction. Pattern extraction infers common patterns from multiple records, and extract subcomponents from the records accordingly. Figure 1 demonstrates several lines of Java application logs and the pattern inferred from them. Pattern extraction allows in-depth understanding of the data's nature, enabling more efficient data compression and accurate data analysis. Previous methods of pattern extraction developed by Fisher et al. (2008) use a rule-based method to iteratively extract common words from records, which is inefficient when dealing with large dataset. In addition, this method does not learn from past dataset to speed up future processing. We plan to use RNN to address these problems.

Recurrent neural network(RNN) based encoder-decoder architecture has recently been widely adopted in various tasks such as neural machine translation (Hermann & Blunsom (2014); Cho et al. (2014)), image captioning (Karpathy & Li (2015) and grammar inference (Vinyals et al. (2015)). These tasks can all be viewed as a translation between source and target domains using encoding-decoding process. First, an encoder is employed to convert the input to a single vector, which is supposed to contain a summary of the input. With that encoded state as input, a decoder is then used to generate an output belonging to the destination domain from the encoded result. The entire encoder-decoder model is trained on source-target pairs to maximize the probability of correct translation.

A potential challenge of using RNN encoder for pattern extraction is constructing an efficient training set. Unlike in the case of natural language and grammar, the pattern does not have a closed, well-defined domain. The vocabularies of pattern can be arbitrary combinations of alphabet, numbers and symbols. There's also no "grammar" governing these vocabularies. Thus the attempt to construct a complete training set that covers all possible patterns is infeasible. Instead, we try to attack the problem from different directions.

In this project, we experiment with two approaches. First, we attempt to imitate human's ability to recognize some common pattern, e.g., date, time and ip address. We train the encoder-decoder model with these common patterns, and explore the model's ability to recognize combination of these patterns. This method will allow the model to remember some patterns and recognize them when later encounter these pattern again. Second, we develop an architecture allowing multiple

Log Data	14:23:01.045 [main] DEBUG o.h.d.s.DefaultService - Synchronizing 14:23:48.656 [Thread] DEBUG o.h.d.storage.StorageService - Persisting Data 14:24:05.656 [monitor] WARN o.h.d.storage.StorageService - Invalid Input
Pattern	Timestamp [Thread Name] Level Source - Content

Figure 1: Application Log and extracted Pattern

inputs to be encoded into a single state, and use this model to train on a dataset containing inputs and patterns inferred from them, in order to evaluate the ability of RNN-based encoder-decoder on directly inferring patterns from multiple inputs. In the experiment, we notice that none of these methods give a satisfied result on the topic, which implies that traditional RNN-based encoder-decoder architecture is not suitable for such kind of tasks and new architecture is needed.

The remainder of the paper is organized as following. Section 2 introduce the pattern extraction problem and RNN-based encoder-decoder structure, including previous works. Section 3 describes the method we experiment with in this report. Section 4 demonstrates the experiment result and Section 5 conclude our finding.

2 Background

2.1 Pattern Extraction

Most data management systems are designed to process organized, structural data. However, many real-world datasets that contain valuable information do not belong to this category. Examples include system logs, documents and image files. To manage these datasets with existing dbms, one important step is to organize these non-structual or semi-structual datasets into structual format. However, this task is challenging as these datasets often come missing documentation or with incomplete descriptions. Thus automatic inferrence of structure hidden in dataset is crucial to efficient processing of ad hoc data. This task is called pattern extraction.

The state-of-art research regarding pattern extraction from textual data is described in Fisher et al. (2008). The authors defined a domain language to describe pattern for non-structual data, and proposed a rule-based algorithm to discover common patterns from a list of textual records. The algorithm first locate frequently appeared symbols (for example, the ":" in a timestamp record or "." in IP address) from these records, and use them to split the records into smaller group of pieces. The process is repeated untill no common symbol can be found. The algorithm considers each group of pieces as a union of tokens, and apply various rules to abstract the structure. Based on the idea, software systems such as PADS Project (2017); Cloudera (2017) are built and put into practice.

2.2 RNN based encoder-decoder architecture

Traditional feed-forward neural network can only process input of fixed size, which makes it infeasible for variable length input such as textual and speech data. Recurrent neural network (Rumelhart et al. (1988); Werbos (1990)) overcomes this limitation by concatenating multiple neural networks sharing same parameters together, with each network corresponds to a single input in the input sequence. However, in this setting, RNN will always output a sequence that has the same length of the input. To perform translation between sequences of different length, Cho et al. (2014) proposed a RNN-based encoder-decoder architecture, in which a RNN is used to encode input sequence as a fixed-length vector (the encoded state), and another RNN is used to expand this encoded state into the output sequence. This structure is shown in Figure 2.

Sutskever et al. (2014) adopts Long Short-Term Memory (LSTM, Hochreiter & Schmidhuber (1997)) in this architecture for machine translation. Later, variants of this network including bi-directional LSTM and Attention (Bahdanau et al. (2015)) are also proposed for an improved accuracy. In Vinyals et al. (2015), the authors demonstrated that LSTM-based auto encoders can be used to infer tree-like structures such as grammars from sequential input, which also inspires us to propose the idea described in this report.

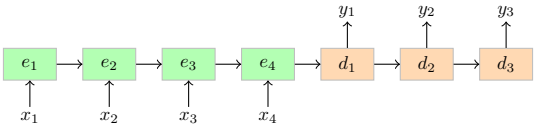

Figure 2: Recurrent Neural Network based Encoder-Decoder

3 Pattern Extraction with RNN

In this report, we experiment with two methods that utilizes LSTM-base encoder-decoder for pattern extraction task.

3.1 Domain Language for Pattern Definition

We use a simple domain language which is similar to Fisher et al. (2008) to define valid patterns. The context-free grammar of this language is shown as following.

$$
\begin{aligned}
&pattern \rightarrow union \mid seq \mid term \\
&union \rightarrow \text{<UNION>}\ (pattern\ \text{<SEP>})*\ pattern\ \text{</UNION>} \\
&seq \rightarrow (pattern\ \text{<SEP>})*\ pattern \\
&term \rightarrow \text{<NUM>} \\
&term \rightarrow \text{<WORD>} \\
&term \rightarrow \{all\ valid\ symbols\ in\ input\}
\end{aligned}
$$

The terminal symbols include all symbols appears in the input, as well as abstract terminals "<NUM>" and "<WORD>", which represent arbitrary consecutive digits and alphabetic characters correspondingly. A union is a collection of patterns allowing exactly one child to appear. It is an equivalence to "|" in regular expression. A sequence is a list of patterns that appear consecutively.

3.2 Pattern Combination

In pattern combination, we train a model on short, simple patterns and test the model on the ability to recognize the combination of these patterns. In training set, we use patterns containing two or three terms listed below. These patterns contains only two abstract terms and a symbol "-"

<WORD> <NUM>
<WORD> – <WORD>
<WORD> – <NUM>
<NUM> – <NUM>

We then use the trained model to test a composite pattern

<WORD> – <WORD> <NUM> – <NUM>

We choose the training pattern and the test pattern so that all transitions in the composite pattern can be found in the training set. Intuitively, this will allow the model to learn these transitions and infer the composite pattern from test input.

3.3 Direct Pattern Inference

In direct pattern inference, we need to accept multiple inputs and generate one output. We expand LSTM-based encoder-decoder architecture to 2-dimension by designate one LSTM encoder to each input. The encoded results from all encoder are then averaged and fed into the decoder for decoding. As in all RNN architectures, all encoders share the same parameters. The architecture is demonstrated in Figure 3.

The training set and test set are comprised of groups of input-pattern pair. Each group contains 10 input strings and one pattern string. Each input string is a sequence of 3 characters (header) and 6 numbers, e.g., ABC123456. In some groups, all inputs share the same header, assume it is XYZ.

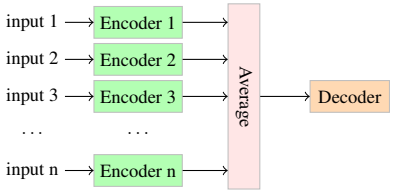

Figure 3: Direct Pattern Inference with RNN

The model is supposed to infer a pattern "XYZ <NUM>". In some groups, each input has distinct header, and the model is supposed to infer pattern "<WORD><NUM>"

4 Experiment

We generate random string and numbers following the patterns described in previous section as training and test set. All experiments are conducted with bi-directional LSTM with hidden dimension 200 (100 for each direction). We use Adam for parameter update and set the learning rate to 0.001.

4.1 Pattern Combination

We train the model on dataset containing only simple patterns, and test the model on composite patterns. Not surprisingly, the model can recognize all simple patterns and quickly achieve a training accuracy of 100%. However, the test accuracy is always around 52%. Considering the small vocabulary we maintain in this experiment, this result is almost worthless. To inspect the problem, we generate decoded result for some inputs. For input string like "ABC-DEF32-324", expected output is "<WORD>-<WORD ><NUM >-<NUM>", while our model only generate "<WORD>-<NUM >". A possible reason for such error is that LSTM model can only recognize those simple patterns when they appear at the beginning of the sentence, but failed to recognize them when they appear in the middle.

4.2 Direct Pattern Inference

We train the model on randomly generated training set of 10000 group and test set of 1000 groups, with half of the group containing identical headers and the other half distinct headers. To our surprise, the training does not converge at all and the test accuracy is always below 5%. It turns out that a simple average of encoded state from separated input failed to capture a good summary of the input group.

5 Conclusion

In this report, we demonstrate our attempts of using LSTM-based encoder-decoder model to pattern extraction from textual data. Our preliminary results shows that the traditional architecture that are widely adopted by previous researchers is not a good fit for pattern extraction task. We will continue work on building new architecture for this task in the future research.

References

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *Proceedings of the Third International Conference on Learning Representations (ICLR 2015)*, abs/1409.0473, 2015.

Cho, Kyunghyun, van Merrienboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1179.

Cloudera. RecordBreaker, 2017. URL https://www.cloudera.com.

Fisher, Kathleen, Walker, David, Zhu, Kenny Q., and White, Peter. From dirt to shovels: Fully automatic tool generation from ad hoc data. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '08, pp. 421–434, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-689-9. doi: 10.1145/1328438.1328488. URL http://doi.acm.org/10.1145/1328438.1328488.

Hermann, Karl Moritz and Blunsom, Phil. A simple model for learning multilingual compositional semantics. *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, 2014.

Hochreiter, Sepp and Schmidhuber, Jrgen. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Karpathy, Andrej and Li, Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 3128–3137, 2015. doi: 10.1109/CVPR.2015.7298932. URL https://doi.org/10.1109/CVPR.2015.7298932.

PADS Project. Pads project, 2017. URL http://www.padsproj.org.

Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pp. 696–699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. URL http://dl.acm.org/citation.cfm?id=65669.104451.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pp. 3104–3112, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969173.

Vinyals, Oriol, Kaiser, Lukasz, Koo, Terry, Petrov, Slav, Sutskever, Ilya, and Hinton, Geoffrey E. Grammar as a foreign language. *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.

Werbos, Paul J. Backpropagation through time: What it does and how to do it. In *Proceedings of the IEEE 78(10)*, pp. 1550 – 1560, 1990.