Computing Semantic Roles Using ANN's with External Memory

Christopher Towne
New College of Florida
Email: Christopher.Towne14@ncf.edu

Jugal Kalita
University of Colorado, Colorado Springs
Email: jkalita@uccs.edu

***Abstract*—Building on recent improvements in neural based Semantic Role Labeling (SRL) and on some recent improvements in neural architectures, in this paper we demonstrate a new memory network based model for SRL. Unlike previous neural architectures for the task of SRL, our model's architecture has the addition of long term memory capabilities and recall. In order to do so we had to remove the bidirectional capabilities of our controller and weight the cost function of the model in order to make it small and sensitive enough to be trained at all. However despite the working model, due to some of the weaknesses of the kind of memory network that we used, the Differntiable Neural Computer (DNC), we are currently unable to demonstrate fully trained results.**

***Keywords*—Semantic Role Labeling, Frame Semantic Parsing, FrameNet, Differntiable Neural Computers**

I. Introduction

Semantic Role Labeling (SRL) is a natural language processing task that involves the figuring out the different semantic roles that different words play in a sentence. In all frameworks of SRL, those roles are defined in reference to something, either some concept or some verb. In the form of SRL that this paper is involved in, frame semantic parsing, the roles that words play are tied to the different semantic frames that lie within a given sentence. With the same word being able to belong to multiple frames without any problem. Taken from the FrameNet project [1] , each semantic frame represents a kind of event, relation, or entity and their constituent parts.

As an example, if one takes the standard sentence: ”Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo” and if one were to attempt to parse it in the task of frame semantic parsing, the task would be to note that the lexical units, the words that invoke the frame itself, in this case ”Manipulate_into_doing”, are the 5th and 6th buffalo(s), that role of Manipulator falls on the 2nd buffalo in the frame invoked by the 5th buffalo and falls on the 4th buffalo in the frame invoked by the 6th buffalo, and that the role of Victim falls upon the 2nd buffalo in the frame invoked by the 5th buffalo and falls upon the 8th buffalo in the frame invoked by the 6th buffalo.

Our approach to this task is a neural based approach similar to those done by Collobert et al. [2], Zhou and Xu [3], Swayamdipta et al. [4], and He et al. [5] . However the difference in our approach is the addition of a new recurrent neural based technique that we have added to our model. As neural techniques have gotten better and added into new SRL models, the capabilities of those models have also increased with them. As such into our model we utilized the recently publish technology of Differntiable Neural Computers (DNC) [6] in order to amplify the capabilities of our model.

Unfortunately, like all recently created technologies, while the DNC system has fixed a number of the kinks that its predecessor, the Neural Turing Machine [7], had, it still has a number of issue yet to be worked out. Namely problems with efficiency of memory access because of the DNC architecture's use of soft attention. There are multiple different lines of SRL setups that have been worked on throughout out the years. The framework that surrounds neural SRl attempts on datasets of the PropBank annotations, have developed a standard setup, however the framework for neural architectures that work on the FrameNet dataset are much more recent. As such we have chosen to follow the lead of Swayamdipta et al. [4] and focus our work on tackling the problem of argument identification in FrameNet. Our decision was mostly a practical one as a result of the aforementioned inherent problems that DNC setups have with efficiency and more namely with the lengths of time spent in training.

A. Related work

The first work and the pioneering work on the automation of SRL was done by Gildea and Jurafsky [8]. For a more recent overview of the task, Màrquez et al. [9] have a detailed introduction.

The first work on the subject of neural SRL was done by Collobert et al. [2]. In their paper they built a few end-to-end neural networks which all undertook a few separate natural language processing tasks, including the PropBank version of SRL, with the same instances of time delayed convolutional neural networks; yet, while the system was competitive within the three other natural language processing tasks, their system had some difficulty handling automated SRL.

That difficulty with SRL was later addressed by Zhou and Xu [3] who made improvements over the Collobert et al.'s convolutional approach by utilized the memory capacities of recurrent neural networks with a deep bidirectional LSTM network in order to have the neural network learn and remember the long term dependency's through out the sentences. The addition of short term memory allowed for a neural approach that was capable of rivaling the other, feature engineered, methods in SRL.

Recently He et al. [5] modified and improved Zhou and Xu's architecture by adding to it highway connections, recurrent dropout, decoding with BIO constraints, and ensemble methods among other things. They improved upon Zhou and Xu's result, and creating the current state of the art system in the PropBank framework.

Also recently Swayamdipta et al. [4] applied segmental RNNs with a softmax-margin cost function to frame semantic parsing and also achieved state of the art performance in FrameNet's subsection of SRL.

II. Task Overview - Frame Semantic Parsing

FrameNet's style of annotating semantic roles is based around the concept of semantic frames. More specifically FrameNet maintains a list of semantic frames, also referred to as just frames, and for each of those frames FrameNet also maintains a list of lexical units, words that can invoke those frames, and frame elements, the roles or arguments that make up the context of each frame.

Within the annotations of FrameNet, annotated sentences are marked with targets, the instances of lexical units that are invoking frames in those instances, and the target's invoked frame. And for each of those target and frame pairs, phrases or clusters of words that serve as frame elements are mark with the identity of the roles they play.

The task in frame semantic parsing is the task of identifying the location and identities of frame elements that belong to each frame in a sentence, however the details of the task varies. Depending, some attempts of the task take a more in depth route and not only identify the frame elements, but also identify the frames and targets themselves. That would have been the preferred route of this work, however due do the limits of speed of training a DNC model we did not take that route and instead chose to focus on frame element, argument, prediction.

III. Model Summary

The model that we demonstrate in this paper is a instance of a Differntiable Neural Computers (DNC) [6]. DNCs are a recent kind of neural network addon that were built upon the advancements of recurrent neural networks, except that they take memory a bit further and add to a given network the additional capabilities of long term memory storage. A DNC is not a neural network architecture on its own, but is composed of an external memory matrix, which is gated by a number of accessing functions, and coupled with a normal neural network that controls access to the external memory matrix, all of which is differntiable.

A. Controller Network

The controller of a DNC, or the neural network that is coupled with the memory matrix, can be any neural network architecture as the only modification that the DNC does to its controller is to change the size of the input into its controller. At the beginning of each timestep, before the controller is run, the previous output of the DNC's memory matrix is append to the input. And after that run of that timestep a portion of the controller's output is split off into different vectors that are passed through the gates of the memory accessing function in order to dictate the internal on going of the otherwise external memory.

[*] Taken without permission from Deepmind's website (deepmind.com)

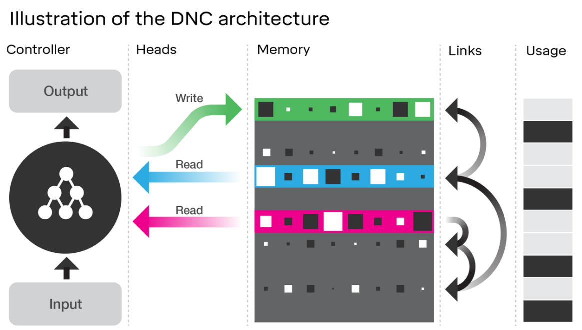


Fig. 1: An example of the framework of the DNC architecture [*].

The model that we used to control the memory matrix is LSTM stack of 8, where each of those 8 were created with a hidden size of 300. Around each of the LSTM cells in that stack highway connections [10], a form of gating that allows input to either go through the neural network layer or pass around the layer unchanged, are wrapped around them. In addition, around the 6 LSTM cells in the middle of the stack, dropout [11] around their output has been implemented.

B. Input and Output Setup

In our setup there are three main feature in the input, the untouched words, the lemmas of those words, and the frame and target identification. Each of those feature is placed into an embedding and trained. The initial embedding for words and lemma are taken from pretrained GloVe [12] embeddings on 840 billion tokens. Any word or lemma that was not found in GloVe was given an randomly initialized vector for as unk symbol. The embeddings for frames were randomly initialized. In addition to those core input features we also gave the DNC parts of speech, dependency, and syntactic parsings as input features as well. Those feature were not embedded.

Most other neural models for SRL utilize bidirectional recurrent neural networks. However because of the addition of the memory matrix and the soft attention used to access it, DNCs do not scale well and creating a bidirectional network with DNC's was infeasible with our computational limitations. As a work around to that issue we decided to experiment with instead giving the network its input twice: once to analysis the input, and once again to have the network give its predictions. As such all of the input sequence is doubled except for a binary sequence used to indicate to the model the valid and unmasked output locations in the sequence, which is only on during the second iteration of the input sequence.

C. Cost Function

The output of our model was a sequence of vectors, where each element in the vector represented a frame element id. However at most or really all timesteps in the sequence most of the frame element output markers would be marked off. As such every every output node tended toward zero using an unweighted cost function. We used two additional weightings to reorient the loss.

The first weight was a basic weighting to balance out the classes (in this case the frame element identities), and the second was a simple penalty on the loss of the output nodes in the timesteps in the sequence where a false negative had occurred. The weight we settled on for that penalty was a times 13. Some alternate weight schemes had been try, namely lowering the cost for all nodes where the label was null. However that proved to be too finicky to find a hyperparameter weight value and the scheme was switched to penalizing false negatives.

IV. Differentiable Neural Computers

Differentiable Neural Computers are in the most recent iteration of the attempt to augment the already impressive short term memory capabilities of recurrent neural network with a form of long term memory storage. Like other kinds of memory networks, DNCs have a long term component; in their case, their external memory matrix. Unlike the memory in something like an LSTM, where memory is split up and distributed throughout the network, the memory in a DNC's memory matrices is external and explicitly accessed.

To access its memory matrix, the DNC utilizes soft attention. The DNC forms read, write, and erase vectors for both keys and strength. One of the interesting properties of the DNC is it weighs its reading of the matrix by both an explicit vector and by a content weighed vector. For a memory matrix $M_t \in \mathbb{R}^{NxW}$, the read vectors, $[r_t^1, .., r_t^R]$ where R is the number of read head, are defined as $r_t^i = M_t^\intercal w_t^{r,i}$, where $w_t^{r,i}$ is the weighting of that row in the matrix for that head. Defining the read weights, w^r, themselves is a bit more complicated, but in short it is the sum of temporal weighting (the time of writing) and the content bases weight, which itself is a cosine distance based metric multiplied by the explicitly passed in weight for the strength of reading at that row location.

Writing in a DNC system has two parts, erasing and writing. More specifically it is defined as $M_{t+1} = M_t \circ (E - w_t^w e_t^\intercal) + w_t^w v_t$. Where $\circ$ denotes an element wise multiplication, E is a matrix of ones, w^w is the write weight, $e \in [0,1]^W$ is the erase vector, and lastly v is the write vector, the actual value to write. Once again the complicated part is w^w, the writing weight. In short, there is a gate that gates the decision to write, which gates another gate that decides whether to use a dynamic allocation based writing system or a content similarity based writing system.

In short the DNC does three main things: it takes read arguments and returns sums of the rows of the memory matrix, weighted by both explicitly passed weights and content similarity based weights. It takes a collection of write arguments and can write to dynamically selected locations or to content similar locations. And it is differentiable. Once it finishes all those operations, the read vectors are passed through a weight matrix and added to the controller output. In addition the read vector are append to the input of the next time step.

V. Experiments

A. Dataset

In our experiments we used data taken from the 1.5 version of the FrameNet database. More specifically we used the same training, development, and testing sets as Das and Smith [13]. In addition the non-core features and the lemmas mention in the Input and Output Setup subsection were all taken from prior processing done by Das and Smith on the splits. For some of the data multiple frames with the same target and frame identities were given, with differing only on the frame elements addressed. Those frames were merged into one frame and treated as such in training and testing. In addition one frame was thrown out of the training set, the frame 'Test35', as it could not be located within FrameNet itself and did not appear in either the development or testing sets.

Furthermore, because FrameNet is setup so that each of its frames treat their frame elements as separate roles, it is difficult to compress the number of possible frame elements as they are unorganized. As such we compressed them by name; so two frame elements both called Time in two different frames would be assumed to represent the same semantic role within their frames an were given one identity within the model.

B. Experimental Setup

As noted above, for practical reasons with the DNC we have chosen to focus on frame element/argument identification within this paper. A model for identifying frame identifications was created, where the prediction of that model could be piped into the model for frame element identification; however since training that model on top of the argument identification model was infeasible and because of the complication it caused with optimizing hyperparameters it was shelved. In addition the because of the DNC's slowness in gradient decent we were unable to have enough time to run more than about 14 epochs.

C. Evaluation

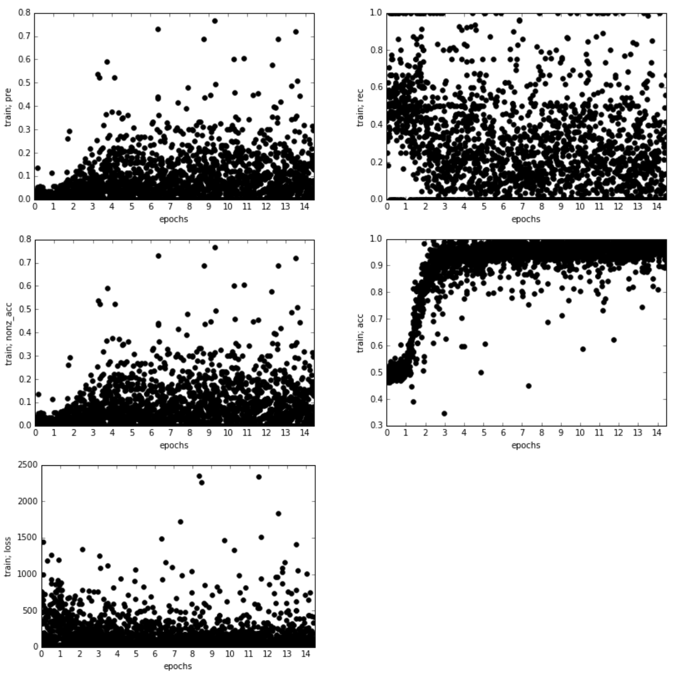
To evaluate, we used a modified version of the standard evaluation script from SemEval '07, where point score for frame evaluations was set to zero. However we are not an expert in Perl so that may have broken it, but it likely does as it is supposed to. That Perl script takes in XML documents to compare, as such we we developed a pipeline to convert the output of the neural network into XML. To do so we masked out any frame elements that could not occur with the frame that the model was given. We then rounded the output of the neural network, grouped the output by continuous streams of the same predicted frame element, and converted those streams into two start and end tags. The SemEval '07 script gives three metrics: recall, precision, and the F score.

D. Results

We will use one other papers in frame semantic parsing as a base line to compare our work; the work of Swayamdipta et al. [4]. While there is other work in argument identification in SRL, there are few other papers the look at argument identification on its own with the FrameNet database. Most other argument identification systems have in the past focused on the PropBank styled datasets.

	P	R	F
Swayamdipta et al.	71.7	66.3	68.9
DNC Model	0.00034	0.00297	0.00062

In addition, below we provide graphs of our DNC's performance on the training set in different metrics:



As a note, the precision and recall scores in the graphs above are not measure in the same way that the precision and recall scores are generated by the evaluation script. The script generates its scores by comparing the nodes of trees generated from the XML, while the scores in the graphs are generated by directly comparing the predicted sequence to the gold standard label. Furthermore, because the script is tree based it is possible that the script looks for more precision in the predicted frame elements that our training metrics, like exact matching of frame element locations or similar.

E. Analysis

Unfortunately, the main conclusion that we are able to draw from this work is that the addition of the DNC's memory matrix and operations makes a model really slow. It took two and a half days to train the model over 14 epoch. To reach a decent amount of epoch, say 500, it would take 3 months. As a result of of low amount of training, there are few solid conclusions that we are capable of drawing. We had intended to run some comparisons of the metrics in terms of the commonous of frame elements, the distance between targets and frame elements, and the length of the sentence; however, with the low accuracy in the XML document, they all drop to zero, rendering it pointless.

VI. Future Work

In this paper we have described a working model that comes with long term memory capabilities, however, unfortunately, describing is the main thing we have done. Upgrading our DNC model into a fully functional SRL model would likely take a good bit more work. There is not much of a way to know with the number of epochs trained over how our setup is. As such there are a few directions that we can take our work.

The most straight forward path would to be to just run it for a few months and get the proper result, however doing so might be ridiculous depending on resources. Another path would be to shrink the model in either the controller size or in the size of the memory matrix and to streamline the training process, however if we did such we would never learn the true capabilities of the larger DNC models. One last option would be to, potentially, drop the soft attention of the DNC model and attempt to train them using reinforcement learning methods [14] or evolutionary methods [15].

VII. Acknowledgments

This work was supported by the REU program at the University of Colorado, Colorado Springs, which is supported by the NSF grants #1659788 and #1359275 and by professors Kalita, Ventura, and Boult.

References

[1] C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The berkeley framenet project," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, Association for Computational Linguistics, 1998, pp. 86–90.

[2] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[3] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural networks.," in *Association for Computational Linguistics*, (Beijing, China), ACL, Jul. 2015, pp. 1127–1137.

[4] S. Swayamdipta, S. Thomson, C. Dyer, and N. A. Smith, "Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold," *arXiv preprint arXiv:1706.09528*, 2017.

[5] L. He, K. Lee, M. Lewis, and L. Zettlemoyer, "Deep semantic role labeling: What works and what's next," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2017.

[6] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[7] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[8] D. Gildea and D. Jurafsky, "Automatic labeling of semantic roles," *Computational linguistics*, vol. 28, no. 3, pp. 245–288, 2002.

[9] L. Màrquez, X. Carreras, K. C. Litkowski, and S. Stevenson, "Semantic role labeling: An introduction to the special issue," *Computational linguistics*, vol. 34, no. 2, pp. 145–159, 2008.

[10] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[11] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[12] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[13] D. Das and N. A. Smith, "Graph-based lexicon expansion with sparsity-inducing penalties," in *Proceedings of the 2012 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*, Association for Computational Linguistics, 2012, pp. 677–687.

[14] W. Zaremba and I. Sutskever, "Reinforcement learning neural turing machines," *arXiv preprint arXiv:1505.00521*, vol. 419, 2015.

[15] R. B. Greve, E. J. Jacobsen, and S. Risi, "Evolving neural turing machines for reward-based learning," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, ACM, 2016, pp. 117–124.