Go For a Walk and Arrive at the Answer
Track 2 - Model Ablation Study
COMP 551-001

Kayla Branson
kayla.branson@mail.mcgill.ca
260629913

Ashique Hossain
ashique.hossain@mail.mcgill.ca
260572615

Samin Yeasar Arnob
samin.arnob@mail.mcgill.ca
260800927

I. INTRODUCTION

In this paper, we first provide an overview of the MINERVA model proposed by Das et. al in "Go for a Walk and Arrive at the Answer" [1]. We then show the results we obtained using their code and compare them to the results in the paper. For our ablation study, we explore various code modifications and see their effects on the final results.

II. OVERVIEW OF THE PAPER

Das et. al introduce a reinforcement learning agent called MINERVA, which is an intuitive acronym for "Meandering In Networks of Entities to Reach Verisimilar Answers." Given a knowledge graph of entities (nodes) and relations (edges), MINERVA learns paths in order to infer unobserved relationships between entities. An example of such a graph can be seen in Figure 1.
Mathematically:

$$
\begin{aligned}
e_1, e_2 &\in \mathcal{E} \\
r &\in \mathcal{R} \\
\mathcal{G} &= (V, E, \mathcal{R}) \\
V &= \mathcal{E} \\
E &\subseteq V \times R \times V
\end{aligned}
$$

where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and $\mathcal{G}$ is the knowledge graph. MINERVA learns relations in the form $(e_1, r, ?)$ - intuitively, given an entity e_1 and a relation r, it tries to find an entity e_2 that fulfills such a relation.

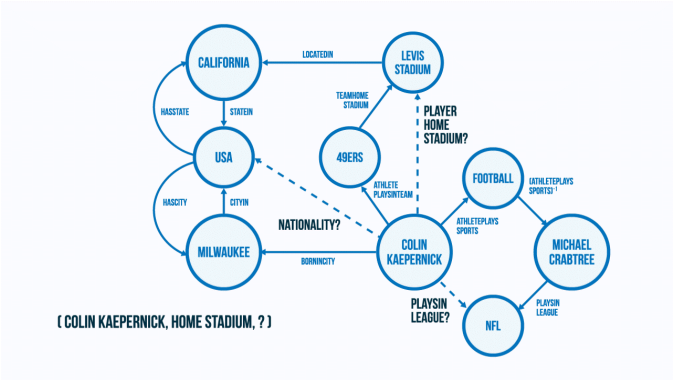

Fig. 1. An example of a knowledge graph. In this example, the solid edges are observed and the dotted edges are unobserved.

MINERVA contains a partially observed Markov Decision Process (MDP) on this knowledge graph. The MDP is a 5-tuple, $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \delta, R)$, wherein $\mathcal{S}$ = states, $\mathcal{O}$ = observations, $\mathcal{A}$ = actions, δ = transitions, R = rewards. The reward is +1 if the terminal state is correct, and 0 otherwise.

The MDP is solved with a policy network. The randomized non-stationary history-dependent policy is described by

$$\pi_\theta = (d_1, d_2, ...d_t) \tag{1}$$

where t represents a timestep. The observation/entity at timestep t is denoted by o_t and the action/relation is denoted by a_t. Within the policy network, an LSTM first encodes a history embedding:

$$h_t = \text{LSTM}(h_{t-1}, [a_{t-1}; o_t])$$

The output of the LSTM h_t is then fed into a Multilayer Perceptron (MLP) along with the current observation/entity o_t and the embedding for the query relation r_q.

$$d_t = \text{softmax}(A_t(W_2\text{ReLU}(W_1[h_t; o_t; r_q])))$$

d_t is a distribution from which the action at timestep t is sampled: $A_t \sim \text{Categorical}(d_t)$. The parameters of the LSTM, weights W_1 and W_2, their corresponding biases, and the embedding matrices form the parameters θ of the policy network.

The aim of MINERVA is to find the parameters θ that maximize the expected reward for the policy network defined in Equation 1:

$$J(\theta) = \mathbb{E}_{(e_1,r,e_2)\sim D}\mathbb{E}_{A_1,..,A_{T-1}\sim\pi_\Theta}[R(S_T) \mid S_1 = (e_1, e_1, r, e_2)]$$

We assume that there is a true underlying distribution $(e_1, r, e_2) \sim D$ and use REINFORCE [2] to solve the optimization problem.

III. REPRODUCING THE RESULTS

Das et. al's paper was subject to controversy surrounding the legitimacy of results when it was under review for the ICLR 2018 conference[1]. We were curious as to whether the controversy was deserved. In table I we present the results we obtained for the S1, S2, and S3 variations of the Countries dataset. In table II we explore the metrics on all

[1]The full discussion thread: `https://openreview.net/forum?id=Syg-YfWCW`

other datasets. While some of the results we achieved came close to those presented in the paper, many are alarmingly different. The results for the Kinship and NELL-995 datasets in particular are strikingly different.

Dataset	Reported AUC-PR	Obtained AUC-PR
Countries S1	1.00 ± 0.00	1.00
Countries S2	92.36 ± 2.41	91.67
Countries S3	95.10 ± 1.20	93.75

TABLE I

COMPARISON OF THE RESULTS REPORTED IN [1] AND THE BEST RESULTS WE OBTAINED IN OUR TESTING FOR THE COUNTRIES DATASETS

Dataset	Metric	Paper Results	Our Results
Kinship	Hits@1	0.605	0.479
	Hits@3	0.812	0.764
	Hits@10	0.924	0.932
	MRR	0.720	0.640
UMLS	Hits@1	0.728	0.714
	Hits@3	0.900	0.897
	Hits@10	0.968	0.921
	MRR	0.825	0.807
WN18RR	Hits@1	0.413	0.328
	Hits@3	0.456	0.440
	Hits@10	0.513	0.512
	MRR	0.448	0.395
FB15K-237	Hits@1	0.217	0.227
	Hits@3	0.329	0.296
	Hits@10	0.456	0.386
	MRR	0.293	0.278
NELL-995	Hits@1	0.663	0.463
	Hits@3	0.773	0.534
	Hits@10	0.831	0.573
	MRR	0.725	0.504

TABLE II

COMPARISON OF THE RESULTS REPORTED IN [1] AND THE BEST RESULTS WE OBTAINED IN OUR TESTING FOR MEDIUM/LARGE DATASETS

A. Code Modifications

The authors provided the best hyperparameter values for β (entropy regularization constant) and λ (moving average constant for REINFORCE baseline) and Path Length. They gave embedding dimensions and did not share some other important information such as the number of iterations and batch size, and whether or not they used/trained entity embeddings.

The default number of iterations in their code was set to 1000, and the models were selected based on best validation performance (which was only tested every 100 iterations). For small datasets, it does not make sense to validate the data so infrequently - in fact, we found that the best models were well within the first 100 iterations for the Countries datasets, as seen in Figure 2. We modified the parameters, adding one (–eval_every) that allows for more frequent testing with the validation set.

In the authors' code, the best model was selected based on the best Hits@10 performance - this was not explicitly explained in the paper, but we kept it for consistency. If, however, at iteration x, Hits@10=h and MRR=m_x, and h is the maximum Hits@10 value, the model from iteration x would be output as the "optimal" model, even if $\exists$ an iteration $y > x$ s.t. Hits@10=h and the MRR $m_y > m_x$. We changed the code so that MRR was checked after Hits@10.

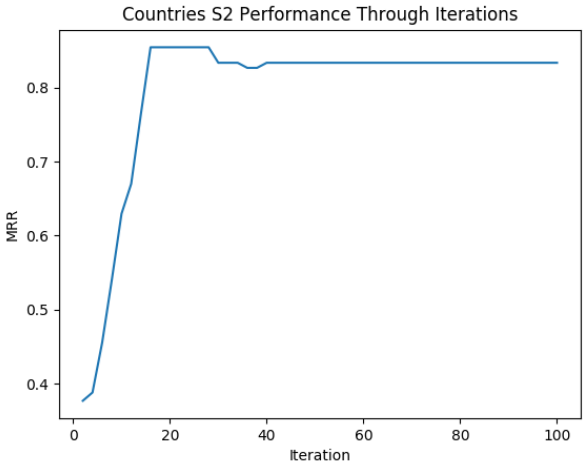

Fig. 2. The best model for Countries S2 was at iteration 16, and performance dropped by the 100th iteration.

Another aspect of the code that we changed was the ability to change the activation functions for the MLPs. The authors of the paper hardcoded ReLU activation functions for both the hidden and output layers of the MLP. They did not explain why they chose ReLU, so we wanted to see how different activation functions affected the model's performance.

IV. EXPERIMENTS

A. Dataset Performance

When we initially read Das et. al's paper, we were struck by MINERVA's performance on the Countries Dataset. The Countries dataset contains regions, subregions, and countries, and the queries are in the form locatedIn(c, ?). The relations between entities are either neighbourOf(c1, c2) or locatedIn(c, r), where r is a region. The authors expressed that "MINERVA significantly outperforms all other methods on the hardest task (S3)" [1]. Most other models did better on S2 than S3, and if S3 was supposed to be the "hardest" task, why did MINERVA falter on S2? The tasks are as follows [3]:

- S1: All relations locatedIn(c, r) where c is a test country and r is a region are removed from the knowledge base.
- S2: In addition to S1, all relations locatedIn(c, s) are removed where c is a test country and s is a subregion.
- S3: In addition to S2, all relations locatedIn(c, r) where r is a region and c is a training country that has a test or validation set country as a neighbor are also removed.

A clearer description of the tasks still does not clarify why MINERVA does not perform as well on S2. After delving deeper into the data, we were able to figure out why this

occurs. First, the test set is incredibly small; it only contains 25 relations. Furthermore, the tasks themselves are different. The task for S1 and S3 is to determine a region for a country (i.e. locatedIn(Bulgaria, Europe)). The task for S2 is to determine a subregion (i.e. locatedIn(Bulgaria, Eastern Europe)). Lastly, the dataset is flawed. One of the "incorrect" query answers on S2 was locatedIn(Zimbabwe, Southern Africa). The "correct" answer was Eastern Africa; however, Zimbabwe **is** in Southern Africa [4]. Although the authors pride themselves on outperforming all other models for the S3 task, the Countries dataset is so small and flawed that it hardly makes sense to use for anything other than a proof of concept in early stages of model development.

B. Comparing Loss and Reward

1) Background and Hypothesis: In our initial exploration of the model, we observed that the training loss on certain datasets reached a minimum after around 100 iterations, but increased after. This unexpected behaviour prompted us to further investigate how the loss plays in to the model's performance. We decided to compare the loss to the empirical reward on the training batch, which is what the algorithm attempts to maximize.

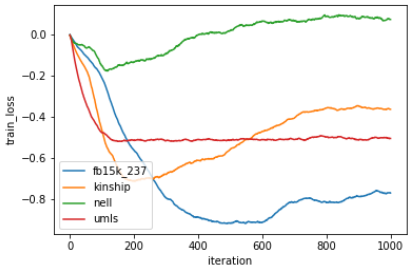

Fig. 3. Training loss of various models with increasing # of iterations

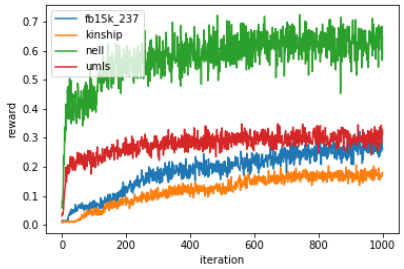

Fig. 4. Reward for the same models with increasing # of iterations

2) Observations: For comparing loss and reward, we used FB15K-237, Kinship,NELL-995,UMLS datasets. We trained the model for 1000 iterations. Due to computation limitations we choose to analyze only on four datasets and for no other specific reason.

3) Analysis: It is clear that while the loss initially drops but then gradually increases for certain models, the reward has a trend of increasing with the iterations. The loss at each iteration or batch is calculated according to the following formula:

$$L_i = 0.98 * L_{i-1} + 0.02 * L_B$$

where L_i is the current training loss, L_{i-1} is the previous batch's training loss, and L_B is the batch training loss calculated according to the REINFORCE algorithm. The training loss is initialized to zero, and we observe an initial drop in the loss for about 50 iterations. This corresponds roughly to the initial increase in reward in figure 3. However, it appears that depending on the dataset, the loss either converges right away (umls) or takes over 800 iterations before it appears to converge. The reward, meanwhile, varies between consecutive batches, but has a local average that is increasing. This suggests that while the loss is used to update the weights of the network, it does not have a direct correspondence with the rewards, which is what the algorithm is actually maximizing.

C. Changing the Activation Functions

1) Motivation and Hypothesis: The original authors chose the ReLU activation function for the hidden layers of their MLP, without providing much justification for their choice. We thought it would be appropriate to explore other activation functions, and investigate whether or not their choice of ReLU was optimal. We anticipate that the choice of activation function will not significantly affect the results, unless an issue such as vanishing or exploding gradients arises. However, we anticipate that other functions, such as the exponential linear unit (ELU) will result in faster learning, allowing it to reach a higher score at an earlier iteration than ReLU. This is due to the ELU bringing the mean unit activations closer to zero, effectively acting as a form of normalization, which is known to improve learning time and provide better results than ReLU, while still solving the vanishing gradient problem.[5]

2) Observations: All tests were done on the **person-borninlocation** dataset, a subset of NELL-995.

Activation Function	Iterations	Hits@1	Hits@10	MRR
cReLU	100	0.5699	0.6788	0.609
	300	0.5233	0.7202	0.6031
ELU	100	0.5803	0.6632	0.6095
	300	0.5855	0.6632	0.6217
ReLU	100	0.5959	0.6736	0.6191
	300	0.513	0.6943	0.586
ReLU6	100	0.5751	0.6995	0.6161
	300	0.4197	0.7513	0.5299
Sigmoid	100	0.5648	0.6736	0.6002
	300	0.4456	0.7047	0.5602
Softplus	100	0.5751	0.6736	0.6114
	300	0.4093	0.7358	0.5423
Tanh	100	0.5803	0.6528	0.6019
	300	0.3834	0.772	0.5409

TABLE III
COMPARISON OF HITS@1, HITS@10, AND MRR FOR THE PERSONBORNINLOCATION SUBSET OF NELL-995, USING DIFFERENT ACTIVATION FUNCTIONS.

As demonstrated in Table III, the ReLU activation function provides the highest MRR score at 100 iterations out of

all choices. However, after 300 iterations, the ReLU MRR decreases, and is lower than both the cReLU and ELU functions. In fact, at 300 iterations, the ELU function has a higher score than ReLU at 100 and at 300 iterations.

3) Analysis: While the ELU function was able to obtain a higher MRR score by the 300$^{\text{th}}$ iteration, it performed more poorly at the 100$^{\text{th}}$, which was not in accordance with our hypothesis that ELU would learn more quickly, therefore providing better results then ReLU at an earlier iteration.

However, it may be argued based on our observations that the ReLU is not necessarily the best function for every dataset. The fact that the performance of the ReLU function decreases as the number of iterations increases implies that the ReLU function may cause overfitting of the training data. In fact, all functions other than ELU demonstrated some degree of overfitting. Of those functions, the cReLU suffered a decrease in MRR of 0.006, and ReLU decreased by 0.033, with all others showing a significantly higher decrease, i.e. a higher degree of overfitting.

4) Limitations: Unfortunately, due to limited computational resources, we were unable to test this suite of activation functions on multiple datasets. It would have been valuable to look at the differences between each function over different datasets, as we would be able to more definitively determine whether or not ReLU was the correct choice for activation function on this model. Furthermore, it would have been valuable to use combinations of different activation functions in the model, as is done in many advanced neural networks.

D. Disabling History Encoding

The authors themselves did an ablation study on the effectiveness of remembering path history, in which the agent chose the next action without the history h_t. On the Kinship dataset, they reported a 27% decrease in Hits@1 performance and a 13% decrease in Hits@10 performance. When we attempted the same ablation, our Hits@1 score decreased 76% and Hits@10 score decreased 41%.

Kinship was one of the datasets that had the largest discrepancy between the reported MRR and our obtained MRR (as seen in table II), but this alone does not explain why our removal of h_t impacted the results more severely than Das et. al's removal. It could be that the configurations of the other parameters that we used depended on the presence of h_t more than theirs did. They did not provide their code for this ablation, so we don't know exactly the reason for this major discrepancy that actually works in the authors' favour - the history embedding proves to be incredibly important piece of the model.

E. Simplifying MLP and LSTM Architecture

1) Motivation and Hypothesis: Das et. al cite an embedding dimension of 200 and hidden layer size of 400 for all of their experiments. The authors' code did not successfully test on models with embedding size 200 and hidden layer size 400 for many of datasets, including UMLs and Kinship. [2] As a result, we modified the architecture while attempting to reproduce their results. The authors did not justify the reason for their original architecture, so we explored simpler architectures to observe how the results differ. We compared MRR across different LSTM embedding dimension sizes and MLP hidden layer sizes for the Kinship and UMLS datasets.

2) Observations: We report the best scores after up to 300 iterations on Kinship and UMLS, using a simple architecture for the networks.

Dataset	Dimension Size	Hits@1	Hit@10	MRR
Kinship	Embedding 50 Hidden 100	0.4227	0.892	0.5749
	Embedding 50 Hidden 50	0.4786	0.9320	0.6395
UMLS	Embedding 50 Hidden 100	0.6233	0.9017	0.7536
	Embedding 50 Hidden 50	0.6944	0.9289	0.7901

TABLE IV
COMPARISON OF HITS@1, HITS@10, AND MRR WHEN CHANGING EMBEDDING DIMENSION SIZE AND HIDDEN LAYER SIZE OF THE MLP FOR KINSHIP AND UMLS DATASETS.

3) Analysis: With smaller architectures, MINERVA appeared to achieve nearly the same MRR scores as our original reproduced results. By reducing the hidden layer size, we were able to increase performance at earlier iterations, converging to the optimal parameters much faster than with a larger architecture. While it seems that a larger and more complex architecture may allow the model to ultimately perform better, smaller architectures are still viable and have a lower computational cost.

F. Increasing path length

1) Background and Hypothesis: The maximum path length in MINERVA is a hyperparameter that limits the maximum path length allowed in a proposed solution to a query. The original authors limited the maximum path length to 2-3 for most datasets, and 1 for WIKIMOVIES set. We wanted to explore results for greater path lengths, as some queries may not be able to be resolved with such short paths. In order to train MINERVA, the authors modified the original datasets to include certain additional relations:

1) No-op relations: Following this relation in the KB will cause the agent to remain at the same state.
2) Inverse relations: For each relation (E_1, R_1, E_2) in the original dataset, the authors added an inverse relation (E_2, R_1^{-1}, E_1).

The purpose of these additions is to allow the agent greater flexibility in how it arrives at an answer. No-op relations allow the agent to remain at a solution state if it arrives at an answer before the maximum path length is reaches. The inverse relations allow an agent to backtrack if it makes a mistake, such that a poor choice of action early on in the path does not render the entire attempt at a solution fruitless.

[2] After training the model for hours, an attempt at a test would result in a segmentation fault. In python.

As a result of these mechanisms provided to the agent, we suspect that if the agent was able to find a solution with a low path length, it will be able to arrive at the same solution with a longer path length as well, by padding the end of the path with no-ops. With longer paths being allowed, there is less pressure on the agent to make correct moves at every step, as incorrect moves can be reversed by following the inverse relations, and the longer path allows for more of these mistakes to be made while still arriving at the correct solution and eventually gaining the reward. This serves as a mechanism to allow the agent to explore more, potentially learning more information about the environment while still being able to exploit the rewards.

However, given that MINERVA was still able to perform well in the original paper with low path lengths suggests that most queries can in fact be answered with short paths. We suspect that increasing the path length will allow the model to answer queries which have longer minimum-length solution paths, while not decreasing accuracy on queries with shorter minimum-length solution paths. This should lead to an overall increase in the MRR of the model.

2) Observations: We performed this experiment the **organizationhiredperson** subset of **NELL-995**, varying the maximum path length from 1 to 10.

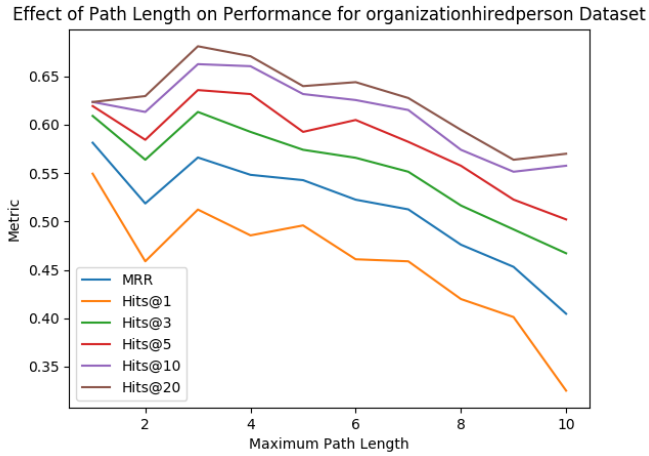

Fig. 5.

As clearly shown in Figure 4, increasing the path length beyond 3 decreased the performance of the model for all scores other than Hits@10 and Hits@20. The latter two scores showed a slight increase at a maximum length of 10, compared to a length of 9. However, there was still an overall decrease between the maximum scores observed at length 10 versus a length of 3.

3) Analysis: The fact that scores tend to decrease overall with an increase in path length implies that the agent is unable to recover from mistakes made when it is allowed more steps in its solution. This means that the policy is unable to suggest no-ops at the appropriate time, or that it is unable to recover from mistaken actions taken at an earlier step. The freedom to make mistakes in the path may actually serve as noise to the model, reducing its ability to discern (on queries with low solution lengths) the desirable sequence of no-ops from taking a different route and backtracking to fix mistakes.

4) Limitations: As each dataset needed to be trained and tested 10 times in order to provide a single set of observations for this experiment, it was extremely computationally expensive to run. As such, we decided to limit this experiment to one dataset. It would have been valuable to explore further datasets, as they may show differing behaviour at longer path lengths.

G. Comparing MAP scores to MRR

1) Background: Two commonly used measures for determining the effectiveness of a query-answering model are mean reciprocal rank (MRR) over the various queries, and mean average precision (MAP) over various relations.

Mean reciprocal rank is calculated by taking the reciprocal of the rank of the correct solution on each query, and averaging that quantity over all queries. For example, if the proposed answers to a query are ranked from most likely to least, and the position Q of the correct answer is $Q = 5$, then the reciprocal rank is $1/5$. Taking the mean of this quantity over the queries in the set will give the MRR for a set.

Mean average precision, however, takes the ranked answers, and calculates the precision of each correct answer over the answers ranked more highly than it. For example, if we get answers for some query, ranked best to worst, as follows:

$$1, 0, 0, 1, 0, 1$$

Where 1 represents a correct answer, and 0 represents an incorrect answer, then we will achieve precisions on each answer as follows:

$$1/1, 0/2, 0/3, 2/4, 0/5, 3/6$$

Taking the average over the correct answers, i.e. entries 1, 4, and 6, we get a MAP score of 0.667.

In the case of the MAP scores calculated by the authors, they take the set of relations, and look at the queries over each relation. For each query, they determine whether or not the prediction by MINERVA is correct, and sort the predictions based on the score (MRR) reported for that prediction. Then, the average precision calculation proceeds as normal, giving an average precision score for the relation. This AP score can be interpreted as being proportional to how valuable a prediction for that particular relation is, so a higher score (AP is close to 1.0) implies that the model is successful in identifying the most likely paths (answers) given that particular relation. By taking the mean of the AP scores over all types of relations, we arrive at the MAP.

As such, the MAP score measures the quality of the highest scoring answers, whereas MRR only gives a score based on the position of the correct answer. Because MRR is calculated over queries, whereas MAP is calculated over relations, they are not directly comparable.

2) Observations: In attempting to reproduce the paper's data, we observed that while our MRR scores were lower than those of the authors, the MAP scores did not display a large difference. The following MAP scores are based on relation-subsets of the NELL-995 dataset, and the reported MRR score from the paper is that of the overall NELL-995 dataset.

Relation	Metric	Paper Results	Our Results
AHS	MAP	0.895	0.8845
	MRR	0.725	0.7732
APFT	MAP	0.824	0.8238
	MRR	0.725	0.5433
APiL	MAP	0.970	0.9649
	MRR	0.725	0.5818
APS	MAP	0.985	0.9855
	MRR	0.725	0.4681
OHQiC	MAP	0.946	0.9335
	MRR	0.725	0.7689
OHP	MAP	0.851	0.8734
	MRR	0.725	0.5812
PBiL	MAP	0.793	0.7773
	MRR	0.725	0.5890
TPS	MAP	0.846	0.8294
	MRR	0.725	0.4220
WF	MAP	0.825	0.7989
	MRR	0.725	0.6095

TABLE V

COMPARISON OF NELL-995 MRR AND VARIOUS RELATION MAP SCORES PROVIDED BY [1] AND THE MRR AND MAP SCORES WE OBTAINED FOR THE SAME RELATIONS

3) Analysis: As shown in Table V, our MAP scores are generally within ± 0.03 of those reported in the original paper, however, our MRR scores vary from between 0.4220 on the relation TeamPlaysSport, to 0.7689 on OrganizationHeadquarteredInCity. The paper's reported MRR for the overall NELL-995 dataset is 0.725, and ours is 0.504. Only one of our subsets has an MRR higher than that of the original paper's NELL-995 score, with the rest being significantly lower.

The difference in MRR implies that in our results, the correct answer to the query has a much lower rank than what the original authors found. However, the similarity between the MAP scores suggests that while the scores are lower overall, the top scoring answers in each relation still tend to be correct.

V. CONCLUSION

Our results indicate that the performance of MINERVA is not quite as strong as the authors of the original paper suggested, even after re-tuning hyperparameters. We found that many of the optimal model settings were in fact suboptimal, or lead to segfaults in TensorFlow. We have found that the authors' choice of ReLU as an activation function may not be optimal in all cases, and that their architecture may be somewhat over-complicated for certain datasets. However, it is clear that the design of the model, especially the use of two networks and encoding the history, cannot be significantly simplified, as ablating the history encoding or the neural network architecture both decrease the effectiveness of MINERVA.

REFERENCES

[1] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Syg-YfWCW

[2] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992696

[3] T. Rocktäschel and S. Riedel, "End-to-end differentiable proving," *CoRR*, vol. abs/1705.11040, 2017. [Online]. Available: http://arxiv.org/abs/1705.11040

[4] e. a. Clyde William Sanger, Kenneth Ingham, "Zimbabwe," 2018.

[5] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *CoRR*, vol. abs/1511.07289, 2015. [Online]. Available: http://arxiv.org/abs/1511.07289

APPENDIX I
CONFIGURATIONS FOR BEST SCORES

Below are the configurations with which we achieved the best results, as outlined in Tables I and II. The number of LSTM layers for all models is 3, except for NELL-995, where it is 4. The number of iterations for convergence is unknown for WN18RR, as these scores came from the authors' saved model.

Dataset	β	λ	**Hidden Size**	**Embedding Size**	**Iterations**
Countries S1	0.01	0.1	25	25	1
Countries S2	0.02	0.1	100	10	16
Countries S3	0.01	0.1	400	200	1
Kinship	0.05	0.1	50	50	1500
UMLS	0.05	0.05	100	200	1100
WN18RR	0.05	0.05	50	50	-
FB15K-237	0.02	0.05	50	50	1000
NELL-995	0.06	0	400	200	1800

APPENDIX II
EXECUTIVE SUMMARY

A. Introduction

In this paper, we provide a summary of our findings from an ablation study on the MINERVA model proposed by Das et. al in "Go for a Walk and Arrive at the Answer"[1]. MINERVA is a reinforcement learning-based model which answers queries on a knowledge graph. Given a knowledge graph of entities (nodes) and relations (edges), MINERVA learns to solve queries by finding unlabelled relations between entities.

B. Summary

1) Reproduction of Results: In attempting to recreate the original paper's results, we found that MINERVA did not perform as strongly as reported. NELL-995 showed the most significant change, a decrease in MRR of 30% from 0.725 to 0.504. UMLS showed a similar performance to the original results, decreasing by around 1%, while other datasets showed between a 6-12% decrease.

2) Changing the Activation Functions: The original authors chose the ReLU activation function for the hidden layers of their MLP, without providing much justification for their choice. In order to evaluate their choice and the importance of ReLU units to the model, we tested a variety of alternatives, including cReLU, ELU, sigmoid and tanh functions, on the personborninlocation dataset, a subset of NELL-995.

We found that after 100 iterations, the ReLU function beat out all the others, with an test-set MRR of 0.6191, closely followed by ReLU6 at 0.6161 and softplus at 0.6114. By the 300^{th} iteration, ELU and cReLU, with MRR scores of 0.6217 and 0.6031 respectively, had overtaken ReLU which had a score of 0.5860.

It is clear that on this dataset, ELU was able to outperform ReLU when used in the hidden layers of the MLP policy network, after 300 iterations. The MRR of ELU at 300 iterations was greater than that of any other activation function in any iteration. This implies some degree of overfitting when training the network using ReLU activation, and all others except ELU. We believe that for this reason, care should be taken when choosing activation functions, and that the choice may need to be tuned for different datasets.

3) Disabling History Encoding: The LSTM network takes the history of entities and relations in a path, as well as other factors, as input, and feeds its output to the MLP policy network.

The authors themselves did an ablation study on the effectiveness of remembering path history, in which the agent chose the next action based on only local information (current state), without the history h_t. On the Kinship dataset, they reported a 27% decrease in Hits@1 performance and a 13% decrease in Hits@10 performance.

When we attempted the same ablation, our Hits@1 score decreased 76% and Hits@10 score decreased 41%. It is unknown why the difference in performance we observed was so much more drastic, even with In any case, it is clear that without the encoded history, MINERVA does a much poorer job of predicting answers, and this appears to be a key aspect of the model.

4) Simplifying MLP and LSTM Architecture: Das et. al cite an embedding dimension of 200 and hidden layer size of 400 for all of their experiments, although we encountered segfaults using their architecture. The authors did not justify their choices, so we explored simpler architectures to observe how the results differ.

With an embedding size of 50, and hidden layer size of 100, we only saw a 1-2% decrease in MRR. While the larger architecture is ultimately better, smaller architectures are still viable and have a lower computational cost, and as such should be considered in future applications of MINERVA.

C. Conclusion

Our results indicate that the performance of MINERVA is not quite as strong as the authors of the original paper suggested, even after re-tuning hyperparameters. We have found that the authors' choice of ReLU as an activation function may not be optimal in all cases. However, it is clear that the design of the model cannot be significantly simplified, as ablating the history encoding or the neural network architecture both significantly decrease the effectiveness of MINERVA.