Project Status Report - Group 23

Abhinav Gupta (agupta67@usc.edu)
Akshita Kapur (kapuraks@usc.edu)
Dhruvam Zaveri (dzaveri@usc.edu)
Hrishikesh Thakur (hthakur@usc.edu)
Shreyas Shrawage (shrawage@usc.edu)

1 Tasks that have been performed by you

1.1 Extended Literature Survey

Few-shot learning (Wang Y. and M., 2020) utilizes transfer learning and meta-learning to generalize from limited data, crucial for enhancing pre-trained language models' logical reasoning. The Differentiable Symbolic Reasoning (DSR) framework combines pre-trained LMs and symbolic reasoning, demonstrating over 20% accuracy improvement on deductive reasoning benchmarks compared to traditional methods.

Transfer learning (Pan S. J., 2010), discussed by Pan and Yang, enhances learning in a target domain by leveraging knowledge from a related source domain. The survey classifies it into inductive, transductive, and unsupervised settings, examining various approaches based on the type of knowledge transferred.

The paper introduces Scallop, a language blending deep learning and logical reasoning for neurosymbolic applications (Li et al., 2023). It offers flexible symbolic representation, declarative logic programming, and automatic differentiable reasoning. Scallop can augment language models' logical reasoning, improving accuracy, efficiency, interpretability, and generalizability over traditional methods.

1.2 Data Acquisition

In our research, we encountered the challenge of obtaining a specific dataset not found in reviewed papers. After a comprehensive search online, we located the dataset on Hugging Face. This discovery significantly aided our analysis phases and ensured research integrity. The CLUTRR dataset assesses natural language understanding systems' systematic generalization and inductive reasoning across 14 configurations.

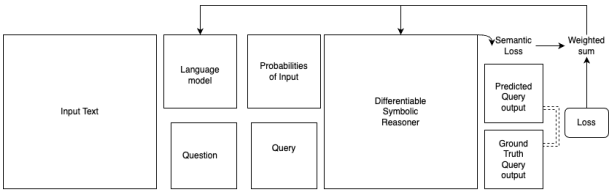

Figure 1

1.3 Novelty Implementation

The paper uses DSR-LM and transfer learning to train the model on kinship classification that falls under the task of logical reasoning. In our project, we are trying to recreate the implementation of this model using parts of the existing codebase. In our efforts to experiment with the real-world performance of this model, we are working towards building separate parallel models to implement the use of LLaMA, OpenAI GPT, and Bard API. Once these models are fine-tuned, we will use the same testing method over the DBpedia-INF curated subset of data. Through this methodology, we will be able to compare and analyze the performance of the Differentiable Symbolic Reasoning framework. The current model uses an LM for NER tagging, a pre-trained fine-tuned model to form word-embeddings and the RoBERTa classifier for relationship prediction. Refer figure 1 for graphical representation.

In our enhanced model architecture, we propose the introduction of an additional Dynamic Semantic Refinement (DSR) layer. This novel layer is specifically designed to refine the probabilistic outputs generated by the initial DSR layer. The refined probabilities are subsequently utilized to ascertain the likelihoods associated with the predicted queries. To accommodate this architectural modification, we have updated the loss function to incorporate a weighted sum that amalgamates the prediction loss with the semantic losses derived

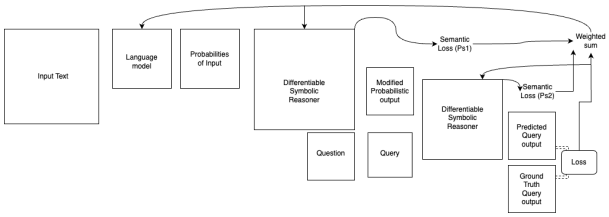

Figure 2

$$J(\theta_1, \phi_1, \theta_2, \phi_2) = \frac{1}{|D|} \sum_{(x,q,y)\in D} w_1 L(P_{\phi_2}(M_{\theta_2}(M_{\theta_1}(x)), q), y) + w_2 P_{s1}(M_{\theta_1}(x), \phi_1) + w_3 P_{s2}(M_{\theta_2}(M_{\theta_1}(x)), \phi_2)$$

Figure 3

from both DSR layers. This revised approach aims to bolster the model's semantic understanding and predictive accuracy by integrating an additional layer of probabilistic refinement. Refer figure 3 for loss function.

1 and 2 are the parameters for the first and second DSR layers, respectively. 1 and 2are the weights for the rules inferred in the first and second DSR layers, respectively. Ps1 and Ps2 are the semantic loss functions for the first and second DSR layers, respectively. w1, w2, w3 are the weights that control the importance of the prediction loss and the semantic losses in the overall loss function. Refer figure 2 for graphical representation.

2 Risks and challenges that you think you need to address by the project deadline

2.1 System Requirements and Installation of Scallopy

Further analysis of the codebase and a deeper understanding of the libraries used in the implementation of the model helped us understand the scale. The training of this model used very heavy hardware resources that lie outside our reach as students. It requires two 20-core Intel Xeon CPUs, four GeForce RTX 2080 Ti GPUs, and 768 GB RAM. So, our initial plans of modifying the system architecture with adding additional layers and re-training the model would be close to impossible. Another significant challenge we are facing pertains to the installation of Scallopy. It is a crucial library used in the implementation. However Scallopy is compatible only with Linux environments. Our team is using Windows and MacOS. Hence, implementing that again presents a substantial obstacle.

3 Your plan to mitigate the risks and address the challenges

3.1 System Requirement and Scallopy Installation

We tried installing different VirtualBox environments to work with Linux. However, the system capacity with VirtualBox restricts complete usage of CPU, GPU, and RAM of the system. We actively searched for alternative libraries or tools that are compatible with Windows or MacOS environments. We tried to leverage Oracle Virtual Box to install Ubuntu, providing a dedicated Linux platform for executing Scallopy and meeting system requirements. We also explored the use of Amazon Web Services (AWS) EC2 instances to access the cloud based computation power and linux-based computing resources. And finally, we have also requested access to the USC HPC system for running the model as a solution to the current issue.

3.2 Novelty Implementation

The problem with installing Scallopy and training the model on a personal system is restricting us from working on the project. We have spent hours trying to fix the situation by reducing the dataset size, finding alternatives for Scallopy. There are parts of the model that we have implemented that do not require the use of this library. However, without this crucial element, the entire architecture crashes.

Therefore, we have decided to create our scaled-down version of the DSR-LM model. We are taking inspiration from the model presented in our reference paper. However, we are using architectural layers and libraries more compatible with our systems. This way, we have better control over each layer, allowing us to modify each component individually without worrying about hurting the existing system.

4 Individual Contributions

4.1 Literature-Review Based Contribution

Abhinav Gupta and Akshita Kapur

- Executed a comprehensive survey of existing literature.
- Analyzed findings to guide the novelty.
- Summarized key points to steer the project's trajectory.

4.2 Data acquisition and analysis based contribution

Hrishikesh Thakur and Dhruvam Zaveri

- Managed the procurement of relevant datasets.
- Conducted exploratory analysis and preprocessing.

4.3 Coding and Implementation based contribution

Hrishikesh Thakur and Shreyas Shrawage

- Determined technical requirements for the system.
- Established the foundational code and system setup.
- Ensured the integration of necessary software.

References

Ziyang Li, Jiani Huang, and Mayur Naik. 2023. Scallop: A language for neurosymbolic programming. *Proc. ACM Program. Lang.*, 7(PLDI).

Yang Q. Pan S. J. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*.

Kwok J. T. Wang Y., Yao Q. and Ni L. M. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*.