Neuro-Symbolic Entropy Regularization (Supplementary material)

Kareem Ahmed[1] **Eric Wang**[1] **Kai-Wei Chang**[1] **Guy Van den Broeck**[1]

[1]Computer Science Department, University of California Los Angeles, USA

1 ADDITIONAL EXPERIMENTAL DETAILS

All of our models were implemented in PyTorch [Paszke et al., 2019]. Our code makes use of the PySDD to compile the constraints. In all the experiments, we performed a grid search on the coefficent of the constraint loss as well as the the coefficient of the entropy loss in the range $[1 \times 10^0, 1 \times 10^{-1}, 5 \times 10^{-1}, 1 \times 10^{-2}, 5 \times 10^{-2}, 1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-4}, 5 \times 10^{-4}]$. That is in addition to searching over other hyperparameters that will be listed, and vary by experiment. All of our constraints are included with our code.

1.1 ENTITY-RELATION EXTRACTION

We begin by testing our research questions in the semi-supervised setting. Here the model is presented with only a portion of the labeled training set, with the rest used exclusively in an unsupervised manner by the respective approach.

We make use of the natural ontology of entity types and their relations present when dealing with relational data. This defines a set of relations and their permissible argument types. As is with all of our constraints, we express the aforementioned ontology in the language of Boolean logic.

Our approach to recognizing the named entities and their pairwise relations is most similar to Zhong and Chen [2020]. Contextual embeddings are first procured, using the $\text{BERT}_{\text{BASE}}$ model from the Hugging Face Transformers library [1], for every token in the sentence. These are then fed into a named entity recognition module that outputs a vector of per-class probability for every entity. A classifier then classifies the concatenated contextual embeddings and entity predictions into a relation.

We employ two entity-relation extraction datasets, the Automatic Content Extraction (ACE) 2005 [Walker et al., 2006] and SciERC datasets [Luan et al., 2018]. ACE05 defines an ontology over 7 entities and 18 relations from mixed-genre text, whereas SciERC defines 6 entity types with 7 possible relation between them and includes annotations for scientific entities and there relations, assimilated from 12 AI conference/workshop proceedings. We report the percentage of coherent predictions: data points for which the predicted entity types, as well as the relations are correct.

Constraint The ACE05 specification lists all permissible relations and their arguments the conjunction of which represent our constraint. Unlike ACE05, SciERC does not specify an ontology of entities and their permissible relations. Therefore, our constraint is determined through procuring the set of all possible relation-subject-object triples in the training set, and applying a threshold to eliminate all noisy labelings in the training set. The script for extracting such a cosntraint is provided with our code.

We used SGD as the optimizer with an initial learning rate of 1.0, which was annealed by a decay rate of 0.9 for every 10 epochs that there is no improvement on the validation set. Every model was allowed to train for 100 epochs, with early stopping if progress is not made for 20 epochs.

1.2 PREDICTING SIMPLE PATHS

For this task, our aim is to find the shortest path in a graph, or more specifically a 4-by-4 grid, $G = (V, E)$ with uniform edge weights. Our input is a binary vector of length $|V| + |E|$, with the first $|V|$ variables indicating the source and destination, and the next $|E|$ variables encoding a subgraph $G' \subseteq G$. Each label is a binary vector of length $|E|$ encoding the shortest *simple* path in G', a requirement that we enforce through our constraint. We follow the algorithm proposed by Nishino et al. [2017] to generate a constraint for each simple path in the grid, conjoined with indicators specifying the corresponding source-destination pair. Our constraint is then the disjunction of all such conjunctions.

[1]https://github.com/huggingface/transformers

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

To generate the data, we begin by randomly removing one third of the edges in the graph G, resulting in a subgraph, G'. Subsequently, we filter out connected components in G' with fewer than 5 nodes to reduce degenerate cases. We then sample a source and destination node uniformly at random. The latter constitutes a single data point. We generate a dataset of 1600 examples, with a $60/20/20$ train/validation/test split. We keep all the hyperparameters provided by Xu et al. [2018] fixed, employing a 5-layer MLP as our baseline, with 50 hidden units per layer, and the Adam optimizer with a learning rate of 1×10^{-3}.

1.3 PREFERENCE LEARNING

We also consider the task of preference learning. Given the user's ranking of a subset of elements, we wish to predict the user's preferences over the remaining elements of the set. We encode an ordering over n items as a binary matrix X_{ij}, where for each $i, j \in 1, \ldots, n$, X_{ij} denotes that item i is at position j. Our constraint α requires that the network's output be a valid total ordering.

We use preference ranking data over 10 types of sushi for $5,000$ individuals, taken from PREFLIB [Mattei and Walsh, 2013], split 60/20/20. Our inputs consist of the user's preference over 6 sushi types, with the model tasked to predict the user's preference, a *strict* total order, over the remaining 4. We keep all the hyperparameters provided by Xu et al. [2018] fixed, employing a 3-layer MLP as our baseline, with 25 hidden units per layer, and the Adam optimizer with a learning rate of 1×10^{-3}.

1.4 WARCRAFT SHORTEST PATH

Following [Pogančić et al., 2020], our training set consists of $10,000$ terrain maps curated using Warcraft II tileset. Each map encodes an underlying grid of dimension 12×12, where each vertex is assigned a cost depending on the type of terrain it represents (e.g. earth has lower cost than water). The shortest (minimum cost) path between the top left and bottom right vertices is encoded as an indicator matrix, and serves as label. Presented with an image of a terrain map, a convolutional neural network – following [Pogančić et al., 2020], we use ResNet18 [He et al., 2016] – outputs a 12×12 binary matrix indicating the vertices that constitute the minimum cost path.

We keep all the hyperparameters provided by [Pogančić et al., 2020] fixed , using an Adam optimizer with a learning rate of 5×10^{-4}. To obtain the constraint, we compiled a constraint for a 6×6 grid, that was applied 9 times, to each overlapping region of the 12×12 grid.