1 Introduction

Previous works (Zhang et al., 2018) shows that structured features like dependency parse trees can improve results of downstream tasks such as relation extraction. Ideally, when feature extraction technique is reliable, we just set the tasks as a sequence and designing a multi-stage model. However, this is not the case for many internal tasks. For example, in relation extraction, we may expand the neighborhood of two entities in knowledge graph and use the connecting graph as GNN feature for relation extraction. But the number of nodes in the connecting neighbor is too huge to be directly used as graph connecting feature. In this case, we may want to put "attention mechanism" on the neighboring graph and select a small subset of nodes.

It is desirable to make such pipeline a end-to-end model to avoid the stacking errors. However, since this intermediate output is discrete, the graph structure itself is not differentiable with graph parameters. And that’s what we aim to tackle in this project.

2 Dataset

We use Stanford TACRED dataset (Zhang et al., 2017) as the test stand of our relation extraction task. TACRED is a large-scale relation extraction dataset with 106,264 examples built over newswire and web text from the corpus used in the yearly TAC Knowledge Base Population (TAC KBP) challenges. Examples in TACRED cover 41 relation types as used in the TAC KBP challenges (e.g., per:schools_attended and org:members) or are labeled as no_relation if no defined relation is held.

We may use knowledge graph for structured features, therefore we need entity linking. we use AIDA CoNLL-YAGO dataset to train entity linker.

Right now, entity linking is considered as a fixed part in our pipeline. We trained the state-of-the-art algorithm (Kolitsas et al., 2018) in CoNLL-YAGO dataset to link entities of TACRED sentences on YAGO. We do so because we found that TACRED is initially extracted from a cold-start knowledge base population task so there is no anchor entries with regard to existing knowledge graphs.

3 Baseline Methods

(Zeng et al., 2014) is a well known CNN-based method for relation extraction. (Zhang et al., 2017) is the state-of-the-art method without modeling any graph structures. Training GCN on fixed dependency parse trees by using a dependency parser, (Zhang et al., 2018) achieved state-of-the art result by combining GCN features with original LSTM features. By changing the GCN interaction, (Wu et al., 2019) improved the result for a little margin.

4 Current Progress

There are two main questions for our project: (1) what is the proper way to learn a structure extractor and a classifier in an end-to-end way? (2) what kind of graph structure is helpful with relation extraction.

4.1 End-to-End Training of Classifier and Structured Feature Representation

For the first question, we investigated several papers (Peng et al., 2018; Niculae et al., 2018b) tackling the non-differentiable problem of structured features.

Suppose we have a input sentence $x$ and we want to estimate its target label $y$. In the middle we want use some structured feature $z$, such as parse tree, that is generated by a scoring function...
$S$ parametrized by $\theta$:

$$\hat{z} = \arg \max_{z \in Z} S(x, z; \theta)$$

where $Z$ is the set of a valid structures. Finally, we predict $y$ with another scoring model with parameter $\xi$:

$$\hat{y} = \arg \max_{y \in Y} f(x, \hat{z}, y; \xi)$$

where $Y$ is the space of labels. Our training task is to minimize the labeling error $L(y, \hat{y})$. Fixing $\hat{z}$, it is easy to calculate the gradient $\frac{\partial L}{\partial \xi}$. However, since $\hat{z}$ is an argmax function over $x$, $\hat{z}$ appears as a piecewise constant function, which means its gradient $\frac{\partial \hat{z}}{\partial \xi}$ is either 0 or undefined. There are two ways to alleviate this problem.

The first method aims to modify the structure prediction model, i.e. argmax, with a "softer" one. The method is called SparseMAP (Niculae et al., 2018a). In this work, the structure predictor is modified to output a sparse distribution over structures $Z$ instead of a single $\hat{z}$. By doing so the final loss function is the expectation on such distribution, so the loss function becomes differentiable w.r.t. $\theta$.

The other line of work borrows the idea from straight-through estimator (Bengio et al., 2013). They first relax the space of $Z$ to a vector space to allow numerical gradient descent. However, training $\theta$ just like training a dependency parser: it requires a valid ground truth in $Z$. Therefore, after doing gradient update we will need to use a projection algorithm to project the updated $\hat{z} - \alpha \frac{\partial L}{\partial \xi}$ to space $Z$. To train the parameter $\theta$, we just use the projected $\hat{z}'$ as the ground truth input. The advantage of this type of work is that it doesn’t require any modification to the structure predictor, which in some cases need a lot of ad-hoc tricks to do so. This is the line of work we are implementing right now.

There is also another line of work to marginalize $z$ using dynamic programming methods like Forward-Backward Algorithm (Kim et al., 2017). However, since we are modeling with GCN we don’t consider this line of work.

### 4.2 Types of Features We Consider

Right now we consider 3 types of structured features.

1. **Knowledge graph neighbourhood**: it is union of neighbour graph of the two entities. However, if we consider 2-hop neighbours for the two entities, the size of the graph will already be forbiddingly large. Therefore, we perform two types of constraints on the graph, which ensures sparsity and still preserves connection. One way to perform such operation is to use min-cost max-flow. We can model the cost of each edge by a knowledge graph model (which serves as a scoring function in the section above) and then the min-cost max-flow graph is the most related interaction information between the source and target entities.

2. **Parse Trees**: dependency parsing, or other structured parsing techniques are considered “solved” fields in NLP community. It may not be very desirable to train such reliable part of the model end-to-end with the final task. However, in (Peng et al., 2018) the work indeed shows some improvement by jointly training a neural dependency parser and a sentiment analysis model.

3. **Graph of Context / Sliding Window**: we can simply build a fully-connected graph by using the text snippets for each nodes. We can model their interaction by neural networks. To improve the interpretability of the graph, we could perform some sparsity constraints on the graph.

### References


