Few-Shot Relation Extraction

Testing various meta-learning algorithms for Relation Extraction

Humans are capable of recognizing new object classes from just a few examples. In contrast, traditional machine learning techniques typically require large datasets to achieve comparable results. In recent years, substantial effort in computer vision has been dedicated to addressing few-shot learning tasks. This approach, known as **Few-Shot Learning (FSL)**, seeks to overcome the challenges of data scarcity without sacrificing model performance.

Despite the strides made in image processing, the application of FSL in NLP remains largely uncharted territory, with only a handful of datasets available. This could be attributed to the capabilities of models like GPT-4, hailed as a few-shot learner due to their ability to acquire new information through minimal or no training data [1]. Nevertheless, our research explores the effectiveness of popular FSL algorithms in the context of relation extraction.

FewRel Dataset

The Few-Shot Relation Classification Dataset (FewRel) [2] Dataset consists of 70,000 sentences on 100 relations derived from Wikipedia (each relation has ~700 sentences). Relation extraction is a NLP task that involves identifying semantic relationships between entities within a text. For example, given sentences and their relations:

Sentence	**Relation**
(1) London is the capital of the U.K. (2) Washington is the capital of the U.S.A.	*capital_of*
(1) Newton served as the president of the Royal Society. (2) Leibniz was a member of the Prussian Academy of Sciences.	*member_of*
(1) Samuel Langhorne Clemens, better known by his pen name Mark Twain, was an American writer. (2) Alexei Maximovich Peshkov, primarily known as Maxim Gorky, was a Russian and Soviet writer.	*birth*

Where blue is the head entity and red is the tail entity[1].

Dataset Features

The dataset is stored as a JSON, for example:

Relation P931: *place served by transport hub* [3]

```
{
  "tokens": [
    "Merpati", "flight", "106", "departed", "Jakarta",
    "(", "CGK", ")", "on", "a", "domestic", "flight",
    "To", "Tanjung", "Pandan", "(", "TJQ",
    ")", "."
  ],
  "h": ["tjq", "Q1331049", [[16]]],
  "t": ["tanjung pandan","Q3056359", [[13, 14]]
  ]
}
```

[1] This is similar to how a KB might store information.

We're interested in the following features: The head entity here is *TJQ*, the tail entity is *Tanjung Pandan*, and the relation is "*place served by transport hub*." P931[2], Q1331049, and Q3056359 are unique identifiers from Wikidata. 16, 13, and 14 and the positions of the tokens in the text. We require the head and tail entities for masking, and the positions for positional encodings. The relation is the target variable we'll try to predict.

Pre-processing

Given sample data as follows:

```
"P177": [
    {
      "tokens": ["Charlie", "visits", "Paris", "from", "London."],
      "h": ["Charlie", "Q4", [[0]]],
      "t": ["Paris", "Q5", [[2]]]
    }
```

1. We'll first convert our tokens to lowercase to reduce our vocabulary set.
2. Then we'll create a word vector either through pre-trained embeddings such as BERT or GloVe or from scratch as follows:

```
[
  {"word": "charlie", "vec": [0.3, 0.3]},
  {"word": "visits", "vec": [0.5, 0.2]},
  {"word": "paris", "vec": [0.4, 0.4]},
  {"word": "from", "vec": [0.3, 0.2]},
  {"word": "london", "vec": [0.2, 0.5]}
]
```

3. The word vector will:
    a. Map each word to an ID based on the provided word vectors.
    b. Be a matrix of word vectors where each row corresponds to a word and its pre-trained embedding.
    c. Also handle unknown words by assigning them a special "UNK" vector and using "BLANK" to fill in shorter sentences.
4. We'll then map each instance to its relation ID as a dictionary where
    a. each key is a relation ID, and each value is a list containing two integers.
    b. These integers define the starting and ending indices in the processed data arrays where instances of the corresponding relation are stored.
    c. For example, ['P177'] = [100, 120] means an instance of relation 'P177' is located in the index 100:120 in the processed data array.
5. We'll then create a mask using the positional encodings of the head and tail entities.
    a. For example, in the above sentence, the head is 0 and the tail is 2.
    b. Our mask might look like: [1, 2, 2, 3, 3]
    c. Where 1 is used for tokens before the head entity, 2 for tokens between the head and the tail entities, and 3 for tokens after the tail entity.
    d. The purpose of such masking is to provide the model with additional contextual clues about the positional relevance of each part of the sentence concerning the entities involved.
6. We'll finally feed the data into our meta-learning model as a support and query set.

[2] For example, P931 can be found on Property talk:P931 - Wikidata.

Models and Evaluation

- We will run 5-way 1-shot, 5-way 5-shot, 10-way 1-shot, and 10-way 5-shot settings.
    - *N*-way *K*-Shot: Where *N* is the number of classes in the support set and *K* is the number of instances in each class.
    - Our evaluation metric would be accuracy[3].
- We'll be working with three models:
    - k-NNs: This will be our baseline. We will use pre-trained word embedding and perform distance matching for k-shots.
    - Prototypical Networks [4]: classifies new instances based on their distance to prototype representations of each class.
    - MAML [5]: Quickly adapt to new tasks with minimal training data by finding a set of initial model parameters specifically optimized for fast learning on new tasks.
- For both ProtoNets and MAML, we'll use a CNN backbone of the same size. We will test various word embeddings to see what works best for our approach.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., ... & McGrew, B. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
2. Han, X., Zhu, H., Yu, P., Wang, Z., Yao, Y., Liu, Z., & Sun, M. (2018). FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. *arXiv preprint arXiv:1810.10147*.
3. https://github.com/thunlp/FewRel/
4. Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in neural information processing systems, 30*.
5. Finn, C., Abbeel, P., & Levine, S. (2017, July). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126-1135). PMLR.

[3] Accuracy is a standard measure in FewRel. Comparing our implementation with the original paper [2] would help us understand if our code is buggy.