National Institute of Informatics Internship Report

Relation Extraction via Manual Prompting and Prompt Tuning via Light-weight LLM

Batuhan Karaca
Department of Computer Science
University of Freiburg

March 25, 2025

Overview

1. Background Research
2. Task: Relation Extraction via Manual Prompting
3. Task: Prompt Tuning via Light-weight LLM
    3.1 Results
4. Self-Reflection and Future Directions

Background Research

- Reading the the Book: Knowledge Graphs [1]
- Reading about LLMs
- The Paper: Unifying Large Language Models and Knowledge Graphs: A Roadmap [2]

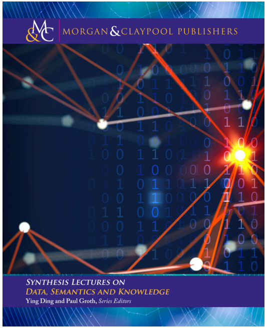

MORGAN & CLAYPOOL PUBLISHERS
SYNTHESIS LECTURES ON
DATA, SEMANTICS AND KNOWLEDGE
Ying Ding and Paul Groth, Series Editors

Figure: The Book: Knowledge Graphs

Problem

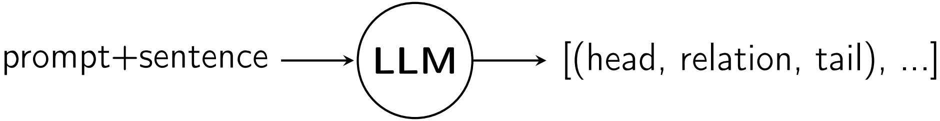
Sentence

```
The bond insurers declined to
↪   comment on friday, though on
↪   thursday, mbia 's chief
↪   financial officer, charles e.
↪   chaplin, vigorously defended
↪   his company at a hearing in
↪   congress and said it did not
↪   need any help.
```

Triples

```
[
    ('MBIA', 'org:top_members/employees', 'Charles E. Chaplin'),
    ...
]
```

Task: Relation Extraction via Manual Prompting



prompt+sentence → **LLM** → [(head, relation, tail), ...]

Prompt

```
List of predicates is [...]. What Subject-Predicate-Object triples are
↪  included in the task sentence? Each triple is in the form ('subject',
↪  'predicate', 'object'). 'predicate' must be from the list of predicates
↪  only. Only use 'NA' when you cannot find any reasonable predicates. Some
↪  reference sentence-triple pairs will be given. For each pair, sentence
↪  is given after 'Sentence: ' and the corresponding triples are given in
↪  a python list after 'Triple: ' word. Only return the triples in a
↪  Python list. The task sentence is the last sentence with no triples.
```

Points to Consider

1. Accuracy of ground truth KG
2. Negative LLM outputs not entailed by the labels
3. Positive LLM outputs that can be entailed by the labels
4. Number of parameters

Assumptions

1. CWA (i.e. No negative relations)
    Accurate constraints
2. No other entailments (i.e. unique relations)

Sample Sentence

Directed by david gordon green; written by green, based on the novel by stewart o'nan; director of photography, tim orr; edited by william anderson; music by david wingo and jeff mcilwain; production designer, richard wright; produced by dan lindau, paul miller, lisa muskat and cami taylor; released by warner independent pictures.

Without predicates

ChatGPT

```
[
    ('david gordon green', 'directed by', None),
    ('green', 'written by', None),
    ('green', 'based on', 'the novel by stewart o\'nan'),
    ('tim orr', 'director of photography', None),
    ('william anderson', 'edited by', None),
    ('david wingo', 'music by', None),
    ('jeff mcilwain', 'music by', None),
    ('richard wright', 'production designer', None),
    ('dan lindau', 'produced by', None),
    ('paul miller', 'produced by', None),
    ('lisa muskat', 'produced by', None),
    ('cami taylor', 'produced by', None),
    ('warner independent pictures', 'released by', None)
]
```

Phi-3-mini

```
[('David Gordon Green', 'Directed by', 'Directed by David
↪   Gordon Green'),
 ('David Gordon Green', 'Written by', 'Written by David Gordon
↪   Green'),
 ('David Gordon Green', 'Based on the novel by', 'Based on the
↪   novel by Stewart O'Nan'),
 ('Tim Orr', 'Director of Photography', 'Tim Orr'),
 ('William Anderson', 'Editor', 'William Anderson'),
 ('David Wingo', 'Music by', 'David Wingo'),
 ('Jeff McIlwain', 'Music by', 'Jeff McIlwain'),
 ('Richard Wright', 'Production Designer', 'Richard Wright'),
 ('Dan Lindau', 'Produced by', 'Dan Lindau'),
 ('Paul Miller', 'Produced by', 'Paul Miller'),
 ('Lisa Muskat', 'Produced by', 'Lisa Muskat'),
 ('Cami Taylor', 'Produced by', 'Cami Taylor'),
 ('Warner Independent Pictures', 'Released by', 'Warner
↪   Independent Pictures')]
```

With predicates

ChatGPT

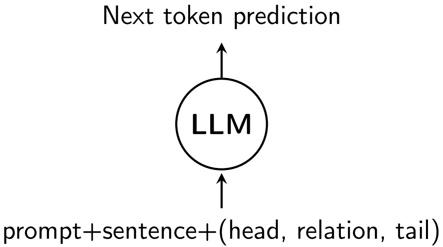
```
[
    ('david gordon green', 'per:employee_of', 'NA'),
    ('green', 'per:employee_of', 'NA'),
    ('stewart o\'nan', 'per:employee_of', 'NA'),
    ('tim orr', 'per:employee_of', 'NA'),
    ('william anderson', 'per:employee_of', 'NA'),
    ('david wingo', 'per:employee_of', 'NA'),
    ('jeff mcilwain', 'per:employee_of', 'NA'),
    ('richard wright', 'per:employee_of', 'NA'),
    ('dan lindau', 'per:employee_of', 'NA'),
    ('paul miller', 'per:employee_of', 'NA'),
    ('lisa muskat', 'per:employee_of', 'NA'),
    ('cami taylor', 'per:employee_of', 'NA'),
    ('warner independent pictures', 'org:alternate_names',
   ↪   'NA')
]
```

Phi-3-mini

```
[('directed by', 'org:founded_by', 'david gordon green'),
 ('written by', 'org:founded_by', 'green, based on the novel
↪   by stewart o\'nan'),
 ('director of photography', 'org:founded_by', 'tim orr'),
 ('edited by', 'org:founded_by', 'william anderson'),
 ('music by', 'org:founded_by', 'david wingo and jeff
↪   mcilwain'),
 ('production designer', 'org:founded_by', 'richard wright'),
 ('produced by', 'org:members', 'dan lindau, paul miller, lisa
↪   muskat and cami taylor'),
 ('released by', 'org:members', 'warner independent
↪   pictures')]
```

Task: Prompt Tuning via Light-weight LLM

Fine-tuning

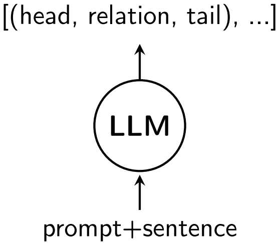


Next token prediction

LLM

prompt+sentence+(head, relation, tail)

Evaluation



[(head, relation, tail), ...]

LLM

prompt+sentence

Prompt

```
List of predicates is [...]. What Subject-Predicate-Object triples are
↪  included in the following sentence?
```

Dataset Transformations: Train sample

Sentence-Triples Columns:

```
{
    'sentence': 'Book by arthur laurents, suggested by the
   ↪  memoirs of gypsy rose lee; music by jule styne; lyrics
   ↪  by stephen sondheim; directed by mr. laurents;
   ↪  choreography by jerome robbins, reproduced by bonnie
   ↪  walker; music director-arranger, patrick vaccariello;
   ↪  sets by james youmans; costumes by martin pakledinaz;
   ↪  lighting by howell binkley; sound by dan moses
   ↪  schreier; production stage manager, craig jacobs;
   ↪  orchestrations by sid ramin and robert ginzler; dance
   ↪  arrangements by john kander; music coordinator,
   ↪  seymour red press.',
    'labels': "[('Jerome Robbins', 'NA', 'James Youmans')]"
}
```

ChatML:

```
{
    'messages': "<|user|>\nList of predicates is [...].  What
   ↪  Subject-Predicate-Object triples are included in the
   ↪  following sentence?\n Sentence: Book by arthur
   ↪  laurents, suggested by the memoirs of gypsy rose lee;
   ↪  music by jule styne; lyrics by stephen sondheim;
   ↪  directed by mr. laurents; choreography by jerome
   ↪  robbins, reproduced by bonnie walker; music
   ↪  director-arranger, patrick vaccariello; sets by james
   ↪  youmans; costumes by martin pakledinaz; lighting by
   ↪  howell binkley; sound by dan moses schreier;
   ↪  production stage manager, craig jacobs; orchestrations
   ↪  by sid ramin and robert ginzler; dance arrangements by
   ↪  john kander; music coordinator, seymour red
   ↪  press.<|end|>\n<|assistant|>\n[('Jerome Robbins',
   ↪  'NA', 'James Youmans')]<|end|>\n<|endoftext|>"
}
```

Dataset Transformations: Validation/Test Sample

Sentence-Triples Columns:

```
{
    'sentence': 'The spokesman of the nigeria police force
   ↪  emmanuel ojukwu told reporters in abuja that 162 of
   ↪  the suspects were already being prosecuted.',
    'labels': "[('Nigeria Police Force', 'NA', '162')]"
}
```

ChatML:

```
{
    'labels': "[('Nigeria Police Force', 'NA', '162')]",
    'messages': "<|user|>\nList of predicates is [...].  What
   ↪  Subject-Predicate-Object triples are included in the
   ↪  following sentence?\n Sentence: The spokesman of the
   ↪  nigeria police force emmanuel ojukwu told reporters in
   ↪  abuja that 162 of the suspects were already being
   ↪  prosecuted.<|end|>\n<|assistant|>\n"
}
```

Model and Training: Phi-3-mini

- Costly requests to web APIs
- Availability of resources
- Performance per number of parameters

```
# LoRA rank r=16
print_info_params(
    model,
    size_type_to_print=("GB", "MB"),
    print_trainable="a",
    calc_param_counts = False,
    calc_grad_counts = False,
)
>>> all params: 2035321856 || all size: 2.5 GB || trainable params: 26181632 || trainable
↪   size: 99.8 MB || trainable%: 1.3 || nontrainable params: 2009140224 || nontrainable
↪   size: 2.4 GB || nontrainable%: 98.7
```

Model and Training: Q-LoRA

```
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.bfloat16,
    max_seq_length=4096
)
model = AutoModelForCausalLM.from_pretrained(
    "microsoft/Phi-3-mini-4k-instruct",
    torch_dtype=compute_dtype,
    trust_remote_code=True,
    device_map="cuda",
    attn_implementation='eager',
    quantization_config=bnb_config,
)
```

Model and Training: Gradient Checkpointing

```python
model = prepare_model_for_kbit_training(
    model,
    use_gradient_checkpointing=True,
    gradient_checkpointing_kwargs={"use_reentrant": True}
)
```

Model and Training: Gradient accumulation

```
...
if (self.batch_step + 1) % self.gradient_accumulation_steps == 0:
    torch.autograd.backward(loss)
    self.optimizer.step()
    self.scheduler.step()
    self.optimizer.zero_grad()
...
```

Cross Entropy

```python
def loss_function(self, batch):
    ...
    metric_adv = 1 # N
    labels_index = labels.view(*labels.shape, 1) # (N, L, 1)
    lpa = torch.gather(log_probs, index=labels_index, dim=-1).squeeze(-1) #
    ↪  (N, L)
    ...
```

Self-Critical Sequence Training

REINFORCE algorithm

1) Predict the reward G_t [3]

```
def loss_function(self, batch):
    ...
    actions_amax = logits.argmax(dim=-1) # (N, L)
    scale_amax = calc_scale(actions_amax, labels) if self.baseline else 0 # N
    scale_sample = None

    probs = F.softmax(logits, dim=-1) # (N, L, S)
    probs_view_2d = probs.view(-1, logits.shape[-1]) # (N*L, S)

    actions_sample_2d = torch.multinomial(probs_view_2d, self.num_samples, replacement=True) # (N*L, R)
    actions_sample_2d_T = actions_sample_2d.T # (R, N*L)
    actions_sample = actions_sample_2d_T.reshape(-1, labels.shape[-1]) # (R*N, L)

    labels_sample = labels.expand(self.num_samples, *labels.shape).reshape(-1, labels.shape[-1]) # (R*N, L)
    scale_sample = calc_scale(actions_sample, labels_sample).view(self.num_samples,-1) # (R, N)
    # -(Q - b)
    metric_adv = scale_amax - scale_sample # (R,N)
    ...
```

Self-Critical Sequence Training

REINFORCE algorithm

2) Perform a gradient update $\theta \leftarrow \theta + \alpha\gamma^t G_t \nabla_\theta \ln \pi_\theta(A_t|S_t)$ [3]

```
def loss_function(self, batch):
    ...
    actions_sample_index = actions_sample.view(self.num_samples,
    ↪   *labels.shape,1) # (R, N, L, 1)
    log_probs_expanded = log_probs.expand(self.num_samples,
    ↪   *log_probs.shape) # (R, N, L, S)
    lpa = torch.gather(log_probs_expanded, index=actions_sample_index,
    ↪   dim=-1).squeeze(-1) # (R, N, L)
    ...
```

Smoothed Return

```
def loss_function(self, batch):
    ...
    smoothed_loss = log_probs.sum(dim=-1) # (N, L)
    smoothed_loss.masked_fill_(padding_mask, 0.0)
    smoothed_loss = smoothed_loss.sum() / (num_active_elements * log_probs.shape[-1])
    ...
    lpa.masked_fill_(padding_mask, 0.0) # (R, N, L) or (N, L)
    lpa_len_1 = len(lpa.shape)-1
    lpa_permuted = lpa.permute(lpa_len_1, *range(lpa_len_1)) # (L, R, N) or (L, N)
    loss = metric_adv * lpa_permuted # (L, R, N) or (L, N)
    loss = loss.sum() / num_active_elements
    return (1 - self.label_smoothing_factor) * loss + self.label_smoothing_factor *
    ↪   smoothed_loss
```

Self-Critical Sequence Training: Rewards

```
def loss_function(self, batch):
  ...
  def calc_scale(candidates, references):
   candidates_detokenized = self.tokenizer.batch_decode(
    candidates,
    skip_special_tokens=True,
    clean_up_tokenization_spaces=True,
   )
   references_detokenized = self.tokenizer.batch_decode(
    references,
    skip_special_tokens=True,
    clean_up_tokenization_spaces=True,
   )
   def ret_metric(res):
    return res if self.train_metric_key==None else res[self.train_metric_key]
   if self.multiple:
    res = [ret_metric(self.train_metric([c], [references_detokenized])) for c in
    ↪  candidates_detokenized]
   else:
    res = [ret_metric(self.train_metric([c], [[r]])) for c, r in zip(candidates_detokenized,
    ↪  references_detokenized)]
   return torch.tensor(res).to(self.model.device)
   ...
```

Evaluation

```
class MicroF1(CustomMetric):
    ...
    def forward(self, raw_text_preds, raw_text_true):
        raw_text_preds = raw_text_preds.strip("` \n")
        if raw_text_preds.startswith("python"):
            raw_text_preds = self.fix(raw_text_preds)
        preds_triples = eval(raw_text_preds)
        true_triples = eval(raw_text_true)
        preds_triples_set = set(preds_triples)
        true_triples_set = set(true_triples)
        true_positives = preds_triples_set & true_triples_set # intersection set
        lentp = len(true_positives)
        self.tps += lentp # add intersection cardinal
        self.total += (len(preds_triples) + len(true_triples) - lentp) # add union cardinal
        return raw_text_preds, raw_text_truedef ret(self):
        if self.total == 0:
            return 0
        return self.tps / self.totaldef fix(self, raw_text_preds):
        return raw_text_preds[raw_text_preds.index("["):]
```

Hyperparameter Tuning

```
def objective(trial):
 train_metric_key = trial.suggest_categorical(”train_metric_key”, [None,
 ↪  'rouge1_fmeasure', 'rouge2_fmeasure', 'rougeL_fmeasure',
 ↪  'rougeLsum_fmeasure'])
 if train_metric_key == None:
  num_samples = trial.suggest_int(”num_samples”, 0, 4)
  if num_samples == 0:
   train_metric = None
  else:
   train_metric = BLEUScore(n_gram=4, smooth=True)
 else:
  num_samples = trial.suggest_int(”num_samples”, 1, 4)
  train_metric = ROUGEScore(use_stemmer=True, accumulate='avg')
  ...
```

Hyperparameter Tuning

```python
def objective(trial):
    ...
    beta1 = trial.suggest_float("beta1", 0.5, 1-EPSILON, log=True)
    beta2 = trial.suggest_float("beta2", beta1, 1-EPSILON, log=True)
    params = {
        # LOSS KWARGS
        "label_smoothing_factor": trial.suggest_float("label_smoothing_factor", EPSILON, 1, log=True),#
        "train_metric": train_metric,#
        "train_metric_key": train_metric_key,#
        "num_samples": num_samples,#
        "multiple": trial.suggest_categorical("multiple", [True, False]),#
        # GEN KWARGS
        "do_sample": trial.suggest_categorical("do_sample", [True, False]),#
        "num_beams": trial.suggest_int("num_samples", 1, 4),
        "temperature": trial.suggest_float("temperature", EPSILON, 2, log=True),
        "top_p": trial.suggest_float("top_p_1", EPSILON, 1, log=True),
        # MODEL KWARGS
        "lora_r": trial.suggest_int("lora_r", 1, 64),#
        "lora_dropout": trial.suggest_float("lora_dropout", EPSILON, 1, log=True),#
        # OPTIM KWARGS
        "optim_arg": OptimArg(
        name="AdamW",
            args={
                "lr": trial.suggest_float("weight_decay", 1e-20, 1e-1, log=True), #1e-8
                "betas": (beta1, beta2), #(0.9, 0.999)
                "weight_decay": trial.suggest_float("weight_decay", 1e-4, 1e-1, log=True), #1e-2
            },
        )#,
    }
    ...
```

Results

Self-Reflection and Future Directions

- Implementation of Trainer API
  - External API
  - PyTorch Lightning
- More constraints (e.g A number of ontological features in input)
- Model+knowledge base
- Novel models (e.g Phi-4, DeepSeek)

References

- A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. d. Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.-C. N. Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann, "Knowledge Graphs," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–37, May 2022, arXiv:2003.02320 [cs]. [Online]. Available: http://arxiv.org/abs/2003.02320

- S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying Large Language Models and Knowledge Graphs: A Roadmap," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580–3599, Jul. 2024, arXiv:2306.08302 [cs]. [Online]. Available: http://arxiv.org/abs/2306.08302

- L. Weng, "Policy Gradient Algorithms," Apr. 2018. [Online]. Available: https://lilianweng.github.io/posts/2018-04-08-policy-gradient/

Thank you for listening