**1. Introduction**

-

**2. Related Work**

- Scene Graph Generation model은 크게 two-stage model과 one-stage model로 분류할 수 있음

- two-stage model은 object detection model과 relation prediction model을 순차적으로 따로 학습하며, object detector에서 출력한 N개 object 간의 모든 object pair를 relation prediction model에 입력해 relation을 예측하는 방식임

- 높은 relation extraction 성능을 보이지만, 연산량이 많아지며 부정확한 object detection이 다음 과정까지 전파되는 문제가 있음

- 이를 개선하기 위해 제안된 one-stage model은 세 가지로 분류할 수 있으며, 최근에는 대부분 DETR과 같은 one-stage object detection model을 활용함

- Object-triplet detection model은 object detector 위에, subject-predicate-object로 구성된 triplet query로부터 relation을 예측하는 triplet predictor를 추가한 구조임

- 이 경우 triplet predictor가 object detector로부터 얻는 사전 정보가 없기 때문에, detected object로부터 직접 정보를 추출하는 module을 포함하다 보니 복잡한 구조를 갖게 됨

- Triplet detection model은 object detector 없이 triplet detector만으로 이루어져 있음

- triplet을 직접 찾기 때문에, relation을 갖지 않는 object는 놓쳐 complete scene graph가 아닌 sub-graph를 감지하는 것에 집중하게 되는 단점이 있음

- Relation extraction model은 object detector와 triple query를 사용하지 않는 작은 relation predictor로 이루어져 있으며, object detector에서 relation 관련 정보를 추출해 사용함

- Relationformer는 transformer decoder에 relation token을 추가해 global information을 추출한 후, object query pair와 relation token의 final hidden representation을 결합해 relation predictor에 입력함

- 본 논문에서는 final hidden representation 외에 detector의 self-attention layer를 활용하고, multi-task learning에 도움을 주는 training technique 또한 제안함

**3. Method**

**3.1. Preliminaries**

- EGTR model은 one-stage object detector DETR을 활용한 구조를 가짐

- DETR은 우선, input image를 CNN에 입력해 (C,Hf,Wf)크기의 feature를 얻고, 1x1 conv과 flatten을 통해 (dmodel,Hf\*Wf) 크기의 input sequence를 만들어 줌

- input sequence는 transformer encoder의 input으로 사용하고, decoder는 N개의 object query를 input으로 가짐

- 그 후, decoder의 반복적인 self-attention, cross-attention layer를 거치며 각 object query가 object candidate의 feature를 학습하게 됨

- 최종적으로 object detection head에서 마지막 layer의 object query를 입력 받아, fc layer를 지나 category label과 box coordinate를 예측함

- object query와 ground truth object 간의 bipartite matching cost를 minimize하는 최적의 permutation을 찾아, object detection loss를 계산하게 됨

**3.2. EGTR**

- EGTR model은 DETR decoder의 self-attention weight를 이용해 relation extraction을 수행함

- attention query와 key를 각각 subject와 object로 생각했을 때, 각 layer의 NxN 크기의 self-attention weight가 predicate 정보를 담고 있다고 볼 수 있음

- 기존의 dot-product attention 대신, query와 key 각각에 linear weight를 곱해준 후 pairwise concatenation을 적용해, 정보를 최대한 보존한 (N,N,2dmodel) 크기의 relation representation R을 얻음

- object query의 last layer representation에도 똑같이 적용해 줌

- 추가적으로, linear layer와 sigmoid를 거쳐 (N,N,1) 크기의 gate value g를 얻고, 모든 layer의 g\*R을 더한 gated summation을 구함

- 이후, MLP와 sigmoid를 거쳐 (N,N,Cp) 크기의 relation graph Ghat을 출력하고, 여기서 Cp는 predicate category를 의미함

**3.3. Learning and Inference**

- EGTR 학습에는 multi-task learning이 사용되며, object detection, relation extraction, connectivity prediction 총 세 개 task 각각에 해당하는 세 개의 loss를 합쳐 최종 loss가 됨

- object detection loss는 기존 DETR과 같은 loss를 사용하며, 나머지 2개 loss는 새로 제안함

- relation extraction loss는 binary cross-entropy를 사용하며, object detection에서 구한 permutation을 predicted graph Ghat에 적용한 Ghatprime과 one-hot encoding한 ground truth graph G 간의 차이로 계산하게 됨

- 하지만 gt graph G를 그대로 사용하면 매우 sparse하기 때문에, 그림과 같이 값이 1인 GT region, 행과 열 둘 다 gt object에 해당하지만 relation이 없어 값이 0인 negative region, 그 외의 non-matching region, 총 3개 region마다 다른 방식으로 학습함

- GT region의 경우, adaptive smoothing을 적용해 학습함

- 학습 초반에는 object detection 성능이 낮아 object representation이 부족하고, 각 object candidate마다 편차가 있으므로, GT region의 predicate을 1로 예측하지 않고, uncertainty u를 고려해 (1-ui)(1-uj)를 Gijk 값으로 사용함

- 이 때 각 object candidate의 uncertainty u는 왼쪽 식과 같이 정의하며, costi는 matching cost, costmin은 perfectly matching될 때의 minimum cost, alpha는 minimum uncertainty를 나타내는 hyperparameter가 됨

- adaptive smoothing을 통해 학습 초반에는 model이 object detection에 집중하게 되고, 시간이 지나면서 multi-task learning이 가능해짐

- negative region과 non-matching region의 경우, graph의 대부분을 차지하기 때문에 sparsity를 다루기 위해 sampling을 함

- hard negative mining을 적용해 Ghatprime의 predicted relation score가 높은 순으로 정렬하고, 정해진 개수의 sample을 선택해 학습함

- 마지막으로 connectivity prediction loss를 구하기 위해, relation graph를 얻을 때와 비슷한 과정을 거쳐, (N,N,1) 크기의 connectivity graph Ehat을 얻게 됨

- connectivity graph는 Graph-RCNN처럼 object pair만으로 relatedness를 binary하게 예측하며, relation extraction의 auxiliary task 역할을 함

- Inference 시엔 두 object의 class score 곱에 predicate score 를 곱해 triplet score를 계산하고, 추가적으로 connectivity score를 곱해 relation 가능성이 낮은 object pair를 제거해 줌

**4. Experiments**

- EGTR을 다른 one-stage SGG model들과 비교한 결과, efficiency 측면에서 가장 적은 parameter와 가장 높은 inference speed를 나타냈고, 가장 높은 object detection 성능을 보였음

- triplet detection 성능의 경우, SOTA method인 SSR-CNN을 넘지는 못했지만, recall, mean recall 을 각각 최대화하는 logit adjustment parameter를 적용했을 때 이와 유사한 성능을 보였음

- relation extractor source에 대한 ablation 결과, 각 layer의 hidden state보다 query, key를 사용하는 게 triplet detection에 효과적임을 알 수 있음

- 하지만 마지막 layer의 hidden state의 경우 모든 layer의 query, key에 상응하는 효과를 보이므로, object의 context representation을 가장 많이 포함함

- adaptive smoothing, connectivity prediction, sampling의 3가지 proposed technique 모두 성능 향상에 도움을 주었음

- hard negative sampling과 hard non-matching sampling은 각각 recall과 mean recall 성능을 높였음

- hard negative는 tail predicate class에 대한 negative를 샘플링하고, hard non-matching은 gt object와 predicate 분포가 유사한 object에 대한 negative를 샘플링하기 때문에 생기는 결과로 보임

- relation을 갖는 object에만 집중하는 triplet detection-based model들과 비교했을 때, EGTR은relation이 없는 object에 대한 detection 성능이 월등히 높은 것으로 보아, complete scene graph 생성 능력을 갖고 있다는 것을 알 수 있음

-