Knowledge Graph Generation From Text

Igor Melnyk, Pierre Dognin, Payel Das

IBM Research

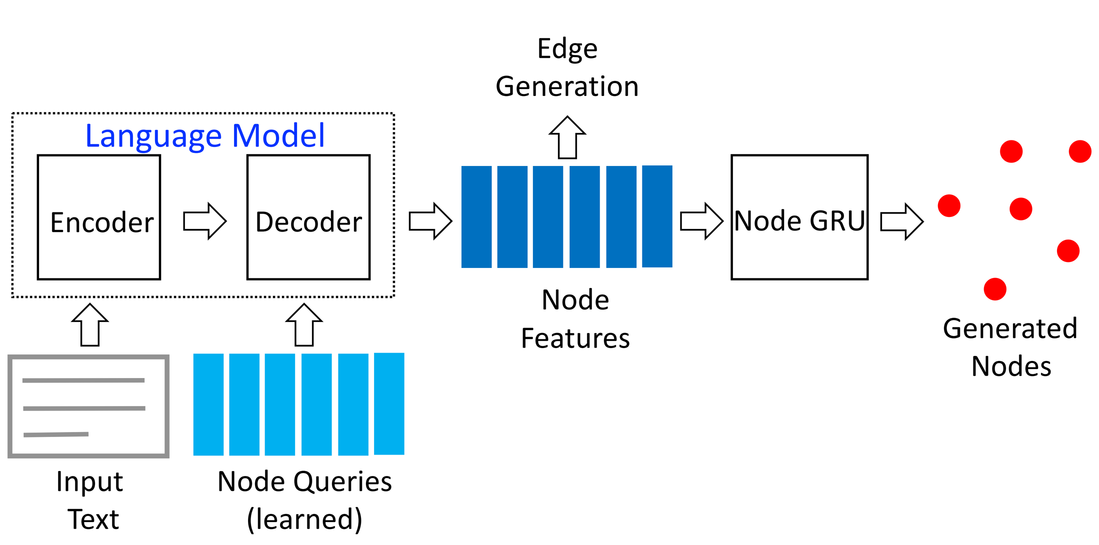
IBM Research

EMNLP 2022

Introduction

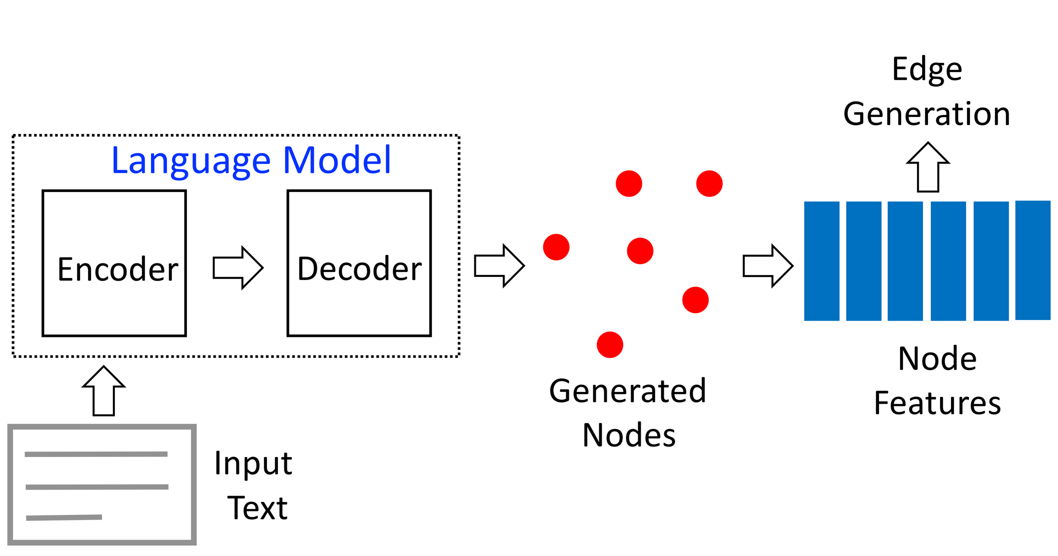
- Automatic Knowledge Graph (KG) Construction
  - Convert text corpora into structured and compressed graph representation
  - Used in many downstream applications:
    - Reasoning, decision making, question answering
- Challenges
  - Non-unique graph representation
  - Complex node and edge structure
  - Large output space
  - Lack of architectures specialized for graph-structured output
  - Limited parallel training data
- Proposed approach - Grapher
  - Given input text, split graph generation in two steps
  - Frist, using pretrained language model, generate nodes
  - Second, using obtained node information, generate edges

Node Generation using Query Nodes



- Query Nodes
  - Decoder receives as input learnable node queries (embedding matrix)
  - Disable causal masking in PLM to attend to all queries
  - Read-off node features directly as decoder output
  - Use GRU head to generate final node output
  - Permutation-invariance of the nodes
    - Target-align nodes using bipartite matching
    - Use cross-entropy as the matching cost

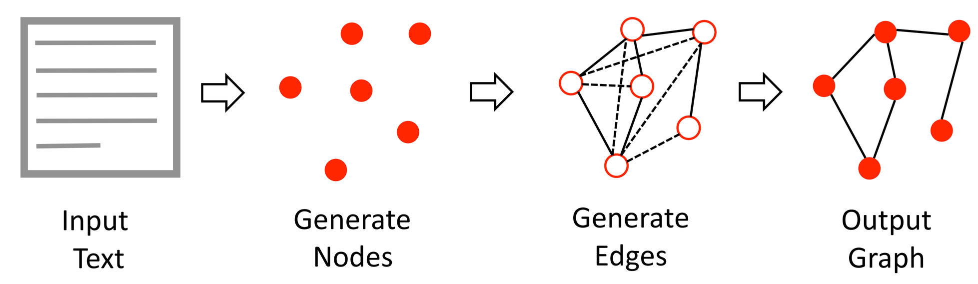
Node Generation using Text Nodes



- Text Nodes
  - Fine-tune PLM to translate text to a sequence of nodes
    - <PAD> NODE1 <NODE_SEP> NODE2 <NODE_SEP> NODE3 </S>
  - Use <NODE_SEP> to delineate generated node boundaries and get features
  - Extracted node features are sent to Edge construction model

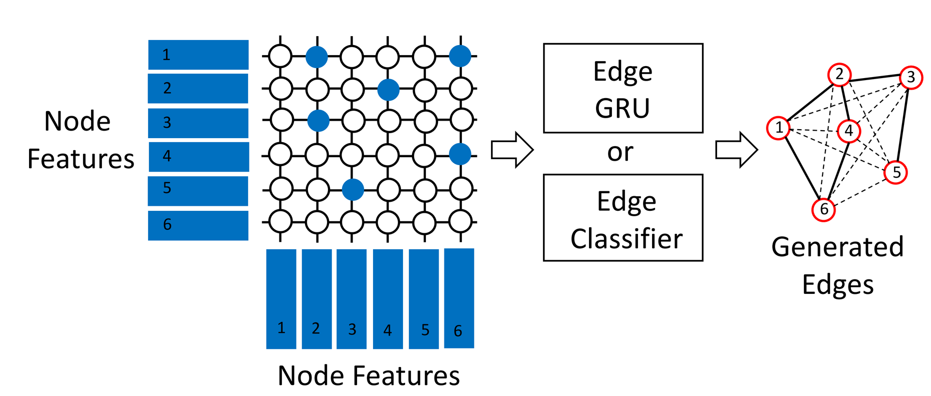
Paper: arxiv.org/abs/2211.10511

System Overview



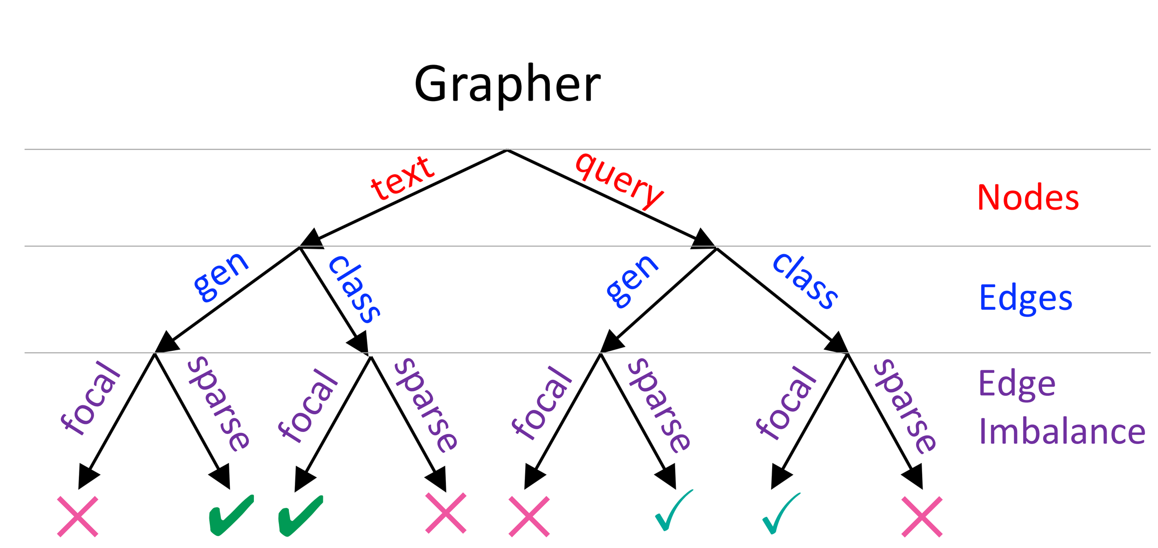
- Main Properties
  - Use of pretrained language model (PLM) for node extraction
  - Efficient partitioning of graph construction in two stages
  - Avoids inefficient graph linearization
  - Generates each node and edge only once
  - Can represent graph entities by any words or set of words
  - Entire system is end-to-end trainable

Edge Generation



- Edge Generation
  - Given a pair of node features, decide existence of an edge
  - First option: GRU-based edge generation
    - Able to construct any edge sequence
    - Risk of not matching target edge token sequence exactly
  - Second option: Classifier-based edge construction
    - More efficient and accurate if edge set is fixed
    - Can misclassify, if limited coverage of possible edges
  - Issue: Imbalanced Edge Distribution
    - Number of actual edges is small and <NO_EDGE> is large
    - Makes training harder
    - Two solutions:
      - Use Focal loss instead of Cross Entropy loss
      - Use Sparse Adjacency Matrix, re-balancing the classes

Summary of Architectural Choices



Experiments

- WEBNLG 2020 – small dataset
  - Text Nodes and Class Edges performs the best
  - GRU-based decoding is a bit less accurate than Class Edges
  - Query-based node generation is behind

	M.	F1	Prec.	Rec.
Amazon AI	E	0.689	0.689	0.690
	P	0.696	0.696	0.698
	S	0.686	0.686	0.687
BT5	E	0.682	0.670	0.701
	P	0.713	0.700	0.736
	S	0.675	0.663	0.695
CycleGT	E	0.342	0.338	0.349
	P	0.360	0.355	0.372
	S	0.309	0.306	0.315
Stanford OIE	E	0.158	0.154	0.164
	P	0.200	0.194	0.211
	S	0.127	0.125	0.130
ReGen	E	**0.723**	**0.714**	**0.738**
	P	**0.767**	**0.755**	**0.788**
	S	**0.720**	**0.713**	**0.735**
Grapher Query Nodes Gen Edges	E	0.395	0.391	0.400
	P	0.325	0.318	0.337
	S	0.289	0.285	0.294
Grapher Query Nodes Class Edges	E	0.466	0.463	0.469
	P	0.360	0.356	0.368
	S	0.347	0.345	0.351
Grapher Text Nodes Gen Edges	E	0.683	0.675	0.695
	P	0.713	0.702	0.730
	S	0.681	0.673	0.693
Grapher Text Nodes Class Edges	E	**0.722**	**0.715**	**0.733**
	P	**0.750**	**0.741**	**0.765**
	S	**0.719**	**0.712**	**0.730**

- TEKGEN – large dataset
  - GRU-based decoding performs similar or better than classification edge head
  - Using more training data makes GRU-based edge decoder more accurate
  - Text Nodes outperforms the query-based system

	M.	F1	Prec.	Rec.
ReGen	E	0.623	0.610	0.647
Grapher Query Nodes Gen Edges	E	0.386	0.361	0.430
	P	0.438	0.405	0.496
	S	0.386	0.361	0.430
Grapher Query Nodes Class Edges	E	0.361	0.338	0.401
	P	0.408	0.378	0.463
	S	0.360	0.337	0.401
Grapher Text Nodes Gen Edges	E	**0.707**	**0.693**	**0.730**
	P	**0.741**	**0.723**	**0.771**
	S	**0.706**	**0.692**	**0.729**
Grapher Text Nodes Class Edges	E	0.700	0.686	0.722
	P	0.735	0.717	0.764
	S	0.700	0.685	0.721

- NYT – small dataset
  - Text nodes and generation edges perform the best
  - More training data enables GRU edge decoder becomes more accurate
  - Text Nodes outperforms the query-based system

	M.	F1	Prec.	Rec.
T5 + Linearized Graph	E	0.832	0.831	0.834
	P	0.834	0.832	0.837
	S	0.824	0.822	0.826
Grapher Text Nodes Gen Edges	E	**0.918**	**0.917**	**0.920**
	P	**0.919**	**0.918**	**0.921**
	S	**0.913**	**0.911**	**0.914**
Grapher Text Nodes Class Edges	E	0.870	0.867	0.872
	P	0.871	0.869	0.874
	S	0.860	0.858	0.862

Code: github.com/IBM/Grapher