Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text

By: Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen

Presented by: Jennifer Fang [Week 03]

Department of Computer Science: University of Virginia

@ https://qdata.github.io/deep2Read/

Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text

Goal: Develop a new model, called GRAFT-Net (Graphs of Relations Among Facts and Text Networks), for extracting answers from a question-specific subgraph containing text and KB entities and relations.

Key Terms

- **Question Answering (QA)**: finding answers to questions posed in natural language
  - **Current QA Approaches**: before, it required a pipeline of multiple ML modules
  - Now, the shift is toward training end-end deep NN models
  - These models only use a single info source; either KB or text
- **Late fusion:** take each source separately and aggregate their predictions after they're finished
- **Early fusion:** use 1 single model to extract answers from a *question subgraph* with both KB facts and text sentences

KB vs. Text Corpus

- **Criteria:** *coverage* of info and *difficulty* of extracting answers
- **Knowledge Base (KB)**: store of information; low coverage but are easier to extract answers from since they're constructed to be queried
  - $K = (V, E, R)$
  - V = set of entities
  - E = triplet of edges (s, r, o) to denote relation $r \in R$ that holds between $s, o \in V$
- **Text Corpus**: large text corpus' have high coverage, but the information is represented in many different text patterns; hard for models to generalize and extract information easily
  - Text corpus $D = \{d_1, . . . , d_{|D|}\}$ set of documents
  - Each $d_i = (w_1, . . . , w_{|di|})$ sequence of words

Entities, links, and questions

Entity linking system

- L = set of links (v, d_p) connecting an entity $v \in V$ with dp (a word @ position p)
- Ld = set of all entity links in document d

Natural language question

- $q = (w_1, . . . , w_{|q|})$ set of words
- Extract its answers $\{a\}_q$
- From $G = (K, D, L)$

Process

1. Extract subgraph G_q from G which contains the answer w/high probability
2. Use GRAFT-Net to learn node representations in G_q conditioned on q to classify each node as answer or not answer

Process 1 - question subgraph (G_q) retrieval

- Use 2 parallel pipelines
  - One over KB: returns a set of entities
  - One over text corpus D: returns set of documents
- The 2 are combined with entity links to produce a full-connected graph

1 - KB Retrieval

1. Entity linking on question q to produce a set of seed entities = S_q
2. Run PPR (Personalized PageRank) to identify other possible answer entities
3. Average word vectors to compute a relation vector v(r) from the surface form of the relation, question vector v(q) from the question’s words; cosine similarity between these edge weights
4. Retain top E entities v_1 through v_E by PPR score + edges between them to add to G_q

2 - Text Retrieval

Used Wikipedia as dataset + retrieved text @ sentence level

1. Retrieve top 5 most relevant Wikipedia articles: using weighted bag-of-words model from DrQA
2. Populate a Lucene index with sentences from the articles
3. Retrieve top ranking sentences $d_1, \dots, d_D$ based on the question words
4. Add retrieved documents + any entities linked to them to G_q

* Lucene is a full-text search library in Java

Final composition of G_q

$\mathbf{G_q} = (\mathbf{V_q}, \mathbf{E_q}, \mathbf{R^+})$

$\mathbf{V_q}$ = retrieved entities + documents = $\{v_1, \ldots, v_E\} \cup \{d_1, \ldots, d_D\}$

$\mathbf{E_q}$ = relations from K among these entities + entity-links between documents and entities

$\mathbf{E_q} = \{(s, o, r) \in E : s, o \in V_q, r \in R\} \cup \{(v, d_p, r_L) : (v, d_p) \in L_d, d \in V_q\}$

* $\mathbf{r_L}$ = special “linking” relation with $R^+ = R \cup \{r_L\}$

GRAFT-Nets

1. Label nodes in V_q: question q and answers $\{a_q\}$
  - $y_v = 1$ if $v \in \{a_q\}$
  - $y_v = 0$ otherwise for all $v \in V_q$

2. The task of QA becomes to:

- Perform binary classification over the nodes of graph G_q
- Use graph-propagation based models that learn node representations then perform classification of the nodes
- Those models follow standard gather-apply-scatter paradigm to learn the node representation with homogeneous updates, i.e. treating all neighbors equally.

Graph-propagation based model

$$h_v^{(l)} = \phi\left(h_v^{(l-1)}, \sum_{v' \in N_r(v)} h_{v'}^{(l-1)}\right)$$

- Initialize node representations $h_v^{(0)}$
- For $l = 1,...,L$ update $h_v^{(0)}$
- $N_r(v)$ = neighbors of v along incoming edges of type r
- Φ is a NN layer
- L = number of layers in the model; corresponds to the max length of the paths along which info should be propagated in the graph
- Once propagation is complete: use final layer representations $h_v^{(L)}$ to perform the desired task
- Desired task could be: link prediction in KB

Key differences in GRAFT-Net

1. G_q contains **heterogeneous nodes**: some correspond to KB entities (symbolic objects) and others represent textual documents (variable-length sequences of words)
2. Want to condition the representation of nodes on the natural language question q

Node Initialization

Nodes corresponding to entities initialized using fixed-size vectors $h_v^{(0)} = x_v \in \mathbf{R}^n$

X_v can be pre-trained or random KB embeddings; n = embedding size

Document is represented with variable length $H_d^{(l)} \in \mathbf{R}^{|d| \times n}$

Words = $(w_1, w_2, \ldots, w_{|d|})$ then its hidden representation $H_d^{(0)}$ = LSTM$(w_1, w_2, \ldots)$ LSTM = long short-term memory unit

p-th row of $H_d^{(l)}$

Embedding of p-th word in document d @ layer l as $H_{d,p}^{(l)}$

Heterogeneous Updates

Entities: M(v) = {(d,p)} = set of positions p in documents d that correspond to a mention of entity v

- Update for entity nodes = single-layer feed-forward network (FFN) over concatenation of 4 states

$$h_v^{(l)} = \text{FFN}\left(\begin{bmatrix} h_v^{(l-1)} \\ h_q^{(l-1)} \\ \sum_r \sum_{v' \in N_r(v)} \alpha_r^{v'} \psi_r(h_{v'}^{(l-1)}) \\ \sum_{(d,p) \in M(v)} H_{d,p}^{(l-1)} \end{bmatrix}\right)$$

$$h_v^{(l)} = \mathrm{FFN}\left(\begin{bmatrix} h_v^{(l-1)} \\ h_q^{(l-1)} \\ \sum_r \sum_{v' \in N_r(v)} \alpha_r^{v'} \psi_r(h_{v'}^{(l-1)}) \\ \sum_{(d,p) \in M(v)} H_{d,p}^{(l-1)} \end{bmatrix}\right)$$

1st 2 terms = entity and question representation from previous layer

3rd term = aggregation of states from entity neighbors of the current node $N_r(v)$

- After scaling with attention weight $\alpha_r^{v'}$
- After applying relation specific transformations ψ_r

4th term = aggregation of all the states of all tokens that correspond to mentions of the entity v among the documents in subgraph

Variables

$\alpha_r^{v'}$ = attention weight, calculated using question + relation embeddings

ψ_r = relation specific transformation

x_r = relation vector for r ∈ Rq; update along an edge is

$$\psi_r(h_{v'}^{(l-1)}) = pr_{v'}^{(l-1)} \mathrm{FFN}\left(x_r, h_{v'}^{(l-1)}\right).$$

$pr_{v'}^{(l-1)}$ = PageRank score used to control propagation of embeddings along paths starting @ seed nodes

Heterogeneous Updates

Documents: L(d,p) = set of all entities linked to word @ position p in document d

<u>Update in 2 steps</u>

1. Aggregate over the entity states coming in @ each position separately

   $$\tilde{H}_{d,p}^{(l)} = \mathrm{FFN}\left(H_{d,p}^{(l-1)}, \sum_{v \in L(d,p)} h_v^{(l-1)}\right)$$

   $h_v^{(l-1)}$ normalized by # outgoing edges @ v

2. Aggregate states within the document using an LSTM

   $$H_d^{(l)} = \mathrm{LSTM}(\tilde{H}_d^{(l)}).$$

Conditioning on the Question

Dependence on the question in 2 ways: attention over relations + personalized propagation

$q = w_1^q , . . . , w_{|q|}^q$ = words of the question

$$h_q^{(0)} = \mathrm{LSTM}(w_1^q, \ldots, w_{|q|}^q)_{|q|} \in \mathbb{R}^n,$$

In subsequent layers, $h_q^{(l)}$ = $\mathrm{FFN}\left(\sum_{v \in S_q} h_v^{(l)}\right)$

Attention over Relations

Attention weight computed using question + relation embeddings

$$\alpha_r^{v'} = \mathrm{softmax}(x_r^T h_q^{(l-1)}),$$

- Embeddings are propagated more along the edges that are *relevant to the question*

Directed Propagation

Many questions require *multi-hop reasoning*

Follows a path from seed node from question to the target answer node

Propagation starts @ seed entities S_q mentioned in question

PageRank scores $pr_v^{(l)}$; measure total weight of paths from seed entity to the current node

Directed Propagation

$$pr_v^{(0)} = \begin{cases} \frac{1}{|S_q|} & \text{if} \quad v \in S_q \\ 0 & \text{o.w.} \end{cases},$$

$$pr_v^{(l)} = (1-\lambda) pr_v^{(l-1)} + \lambda \sum_r \sum_{v' \in N_r(v)} \alpha_r^{v'} pr_{v'}^{(l-1)}$$

- Reuse the attention weights to ensure that nodes along *relevant* paths to the question receive high weight
- PageRank score used as a scaling factor when propagating embeddings along edges

Directed Propagation

- For l = 1, PageRank score = 0 for all entities except seed entities Propagate outwards from those nodes

- For l = 2, it’s non-zero for seed entities and their 1-hop neighbors -> only propagate along these edges

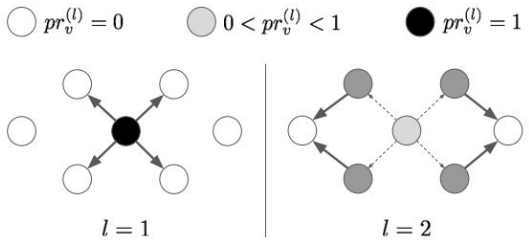

Figure 3: Directed propagation of embeddings in GRAFT-Net. A scalar *PageRank* score $pr_v^{(l)}$ is maintained for each node v across layers, which spreads out from the seed node. Embeddings are only propagated from nodes with $pr_v^{(l)} > 0$.

Answer selection

Final representations $h_v^{(L)} \in \mathbf{R}^n$; used for binary classification to select answers

$$\Pr\left(v \in \{a\}_q | \mathcal{G}_q, q\right) = \sigma(w^T h_v^{(L)} + b),$$

σ is the sigmoid function

- Training uses binary cross-entropy loss over these probabilities

Regularization via Fact Dropout

Want: model to learn a robust classifier

How: fact-dropout aka randomly drop edges from the graph during training (with probability p_0)

Usually easier to extract answers from KB than from documents, so model tends to rely on KB more

Experiment Setup

1. **Datasets**: WikiMovies-10K and WebQuestionsSP

2. **Compared models:** KV-KB [Key Value Memory Networks model; only KB], KV-EF [same model with access to text as well], GN-KB [GRAFT-Net model; no text], GN-LF [late fusion version of GRAFT-Net; did 1 with only text and 1 with only KB], GN-EF [**main model**, early fusion], GN-EF+LF [ensemble over GN-EF and GN-LF models]

Results

1. Best performance was GN-EF+LF aka the ensemble of early and late fusion
2. Adding text adds a great benefit to performance
3. That benefit diminishes as KB completeness reaches 100%

Model	Text Only	KB + Text			
		10 %	30%	50%	100%
WikiMovies-10K					
KV-KB	–	15.8 / 9.8	44.7 / 30.4	63.8 / 46.4	94.3 / 76.1
KV-EF	50.4 / 40.9	53.6 / 44.0	60.6 / 48.1	75.3 / 59.1	93.8 / 81.4
GN-KB	–	19.7 / 17.3	48.4 / 37.1	67.7 / 58.1	**97.0 / 97.6**
GN-LF	**73.2 / 64.0**	74.5 / 65.4	78.7 / 68.5	83.3 / 74.2	96.5 / 92.0
GN-EF		75.4 / 66.3	82.6 / 71.3	87.6 / 76.2	96.9 / 94.1
GN-EF+LF		**79.0 / 66.7**	**84.6 / 74.2**	**88.4 / 78.6**	**96.8 / 97.3**
WebQuestionsSP					
KV-KB	–	12.5 / 4.3	25.8 / 13.8	33.3 / 21.3	46.7 / 38.6
KV-EF	23.2 / 13.0	24.6 / 14.4	27.0 / 17.7	32.5 / 23.6	40.5 / 30.9
GN-KB	–	15.5 / 6.5	34.9 / 20.4	47.7 / 34.3	66.7 / 62.4
GN-LF	**25.3 / 15.3**	29.8 / 17.0	39.1 / 25.9	46.2 / 35.6	65.4 / 56.8
GN-EF		31.5 / 17.7	40.7 / 25.2	49.9 / 34.7	67.8 / 60.4
GN-EF+LF		**33.3 / 19.3**	**42.5 / 26.7**	**52.3 / 37.4**	**68.7 / 62.3**

Effect of Novel Ideas

1. **Heterogeneous Updates**
   - Tested a non-heterogeneous version as well
   - Cannot disambiguate different entities mentioned in the same document
   - Non-heterogeneous is consistently worse than the heterogeneous
2. **Conditioning on the Question**
   - Both directed propagation method and attention over relations led to better performance
3. **Fact Dropout**
   - Moderate levels improve performance (~0.2)

Conclusion

GRAFT-Net classifies nodes in subgraphs with both KB entities and text documents

Achieves performance competitive to state-of-the-art methods; outperforms baselines when using text + incomplete KB

Future Work

1. Extend GRAFT-Nets to pick spans of text as answers, not just entities
2. Improve the subgraph retrieval process