since 1995

Data Science Natural Language Processing NLP Papers Summary

Day 129: NLP Papers Summary – Neural Approaches To Conversational AI – KB-QA (Neural Methods)

By Ryan 8th May 2020 No Comments

Conversational QA system has become very popular in recent years and many techniques have been explored to build a good reliable QA system. Most QA systems allow users to either query from a knowledge base (KB-QA) or from a collection of documents (text-QA). We will first review the KB-QA space, both symbolic and neural methods and then the text-QA space.

In terms of KB-QA space, we went through the symbolic methods yesterday and will c rough th . We will review different neural methods that been

Loading [MathJax]/extensions/MathZoom.js

proposed to address the issues faced in symbolic methods. We will also be discussing multi-turn and conversational KB-QA agents which are not as well-studied.

Embedding-based methods

To counter the paraphrasing issue, we can use embeddings to encode all the entities and relations in a KB. This maps different paraphrased questions with similar semantics closely in the neural space. Most KB embedding models are developed to tackle the KB completion task, which neglects the search complexity problem since you only need to predict whether an unseen triplet holds or not.

Bilinear model is one of the basic KB embedding models where we have an embedding vector for each entity and a matrix for each relation in the KB. The model would then use these to score how likely a triplet holds. This can be extended to answer queries that involves multiple steps. For example, to answer "Where did Tad Lincoln's parents live?", we would need to have an initial anchor entity s (Tad Lincoln) and all the relevant sequence of relations (parents, location). These sequence of relations can be represented by combining individual matrix relation and form a relation embeddings pathway. The score for a particular path query can then be compute as follows:

$$score(q,t) = x_s^T W_{r1} \dots W_{rk} x_t$$

Multi-step Reasoning on KB

Most often, a query might require multi-step reasoning to deduce the answer. To do multi-step reasoning, for each relation, we are interested in subsequent relational paths that are connected and relevant. Given the example below, to determine the citizenship of Obama, we would need to deduce the answer from paths that contain relevant entities and relations. For example, the triplet (Obama, born-in, Hawaii) and (Hawaii, part-of, USA) allows us to connect Obama's citizenship to be USA, using the born-in and part-of relations.

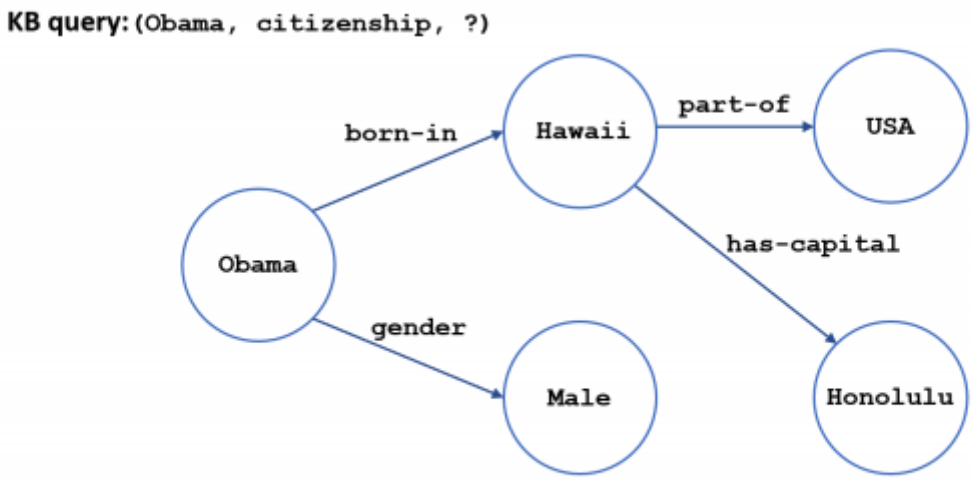


Figure 3.2: An example of knowledge base reasoning (KBR). We want to identify the answer node USA for a KB query (Obama, citizenship, ?). Figure adapted from Shen et al. (2018).

There are three categories of knowledge base reasoning (KBR) methods:

1. Symbolic methods
2. Neural methods
3. Reinforcement learning based methods (not covered in this blog post)

SYMBOLIC METHODS

Path Ranking algorithm (PRA) is one of the main symbolic methods for learning relational paths in KBs. PRA uses random walks to perform multiple depth-first search to find relational paths. Therefore, given a query (s, r, ?), PRA will use a set of relational paths for r relation to traverse the KB and score each candidate answer t using a linear model. The disadvantage of PRA is that it does not capture semantic similarities between relations and so PRA can easily produce millions of distinct paths, making it inefficient to query and hard to scale.

NEURAL METHODS

The overall architecture typically consists of 3 components:

1. Encoder and Decoder
2. Shared Memory
3. Controller

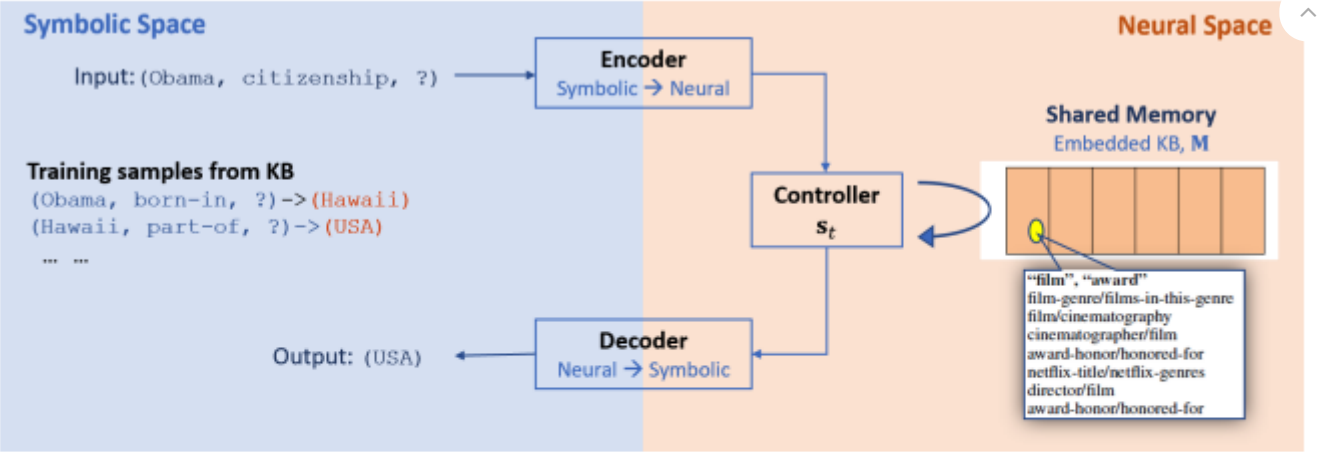


Figure 3.3: An overview of the neural methods for KBR (Shen et al., 2017a; Yang et al., 2017a). The KB is embedded in neural space as matrix **M** that is learned to store compactly the connections between related triples (e.g., the relations that are semantically similar are stored as a cluster). The controller is designed to adaptively produce lookup sequences in **M** and decide when to stop, and the encoder and decoder are responsible for the mapping between the symbolic and neural spaces.

The encoder maps the source entity and relation into their respective embedding vectors and concatenate them to form the initial hidden vector for the controller. The decoder takes in the controller hidden state at time t to generate an output vector, which can be map onto the symbolic space to generate the final output answer.

The shared memory M consists of a list of vectors that are randomly initialised and updated through training. Each vector in M represents a cluster of relations or entities and the distance between these vectors capture the semantic relatedness of these vectors. This way, the relation vector “citizenship” might be closely mapped to the relation vector “born-in”, allowing the model to draw up connections.</p.

The controller takes in an initial hidden state from the encoder and uses attention mechanism to iteratively update the hidden state until it decides to terminate the reasoning process, in which case the decoder is called to generate the final output. This reasoning process can be formulated as a markov decision processes (MDP) as the number of iteration steps during the reasoning process is not given by the training data but decided by the controller on the spot.

One of the main limitations of neural methods is that it is difficult to interpret the results and the underlying logic that the model uses to derive the final answer.

C Loading [MathJax]/extensions/MathZoom.js

All the methods described so far are based on single-turn conversational agents. Howeve is unlikely that the user can compose all the information required into a single query and so multi-turn conversational KB-QA agents are needed to interactively communicate with the users and continuously updating the query.

As shown in the figure below, on top of the semantic parser and KBR engine, conversational KB-QA agent is also equipped with a Dialogue Manager (DM), which tracks the dialogue state and use that to determine what question to ask the user next to help them navigate the KB effectively. There are 4 components shown below. Firstly, the belief tracker identify user intents, extracting relatable attributes, and tracking the dialogue state. Secondly, the soft-KB lookup behaves like the other KB-QA models above except the input data is based on the dialogue history and not just the user utterance. Thirdly, the beliefs summary summarises the vector state. Lastly, the dialogue policy network determines the next action based on the dialogue state.

DATASETS FOR CONVERSATIONAL KB-QA AGENTS

1. *Sequential Question Answering (SQA)*. Crowd-sourced workers are required to understand the original question intent and compose a sequence of related questions that lead to the final intent
2. *Complex Sequence Question Answering (CSQA)*. CSQA combines two sub-tasks of a) answering factoid questions through reasoning over KB and b) learning to converse through a sequence of coherent QA pairs

Source: https://arxiv.org/pdf/1809.08267.pdf

Ryan

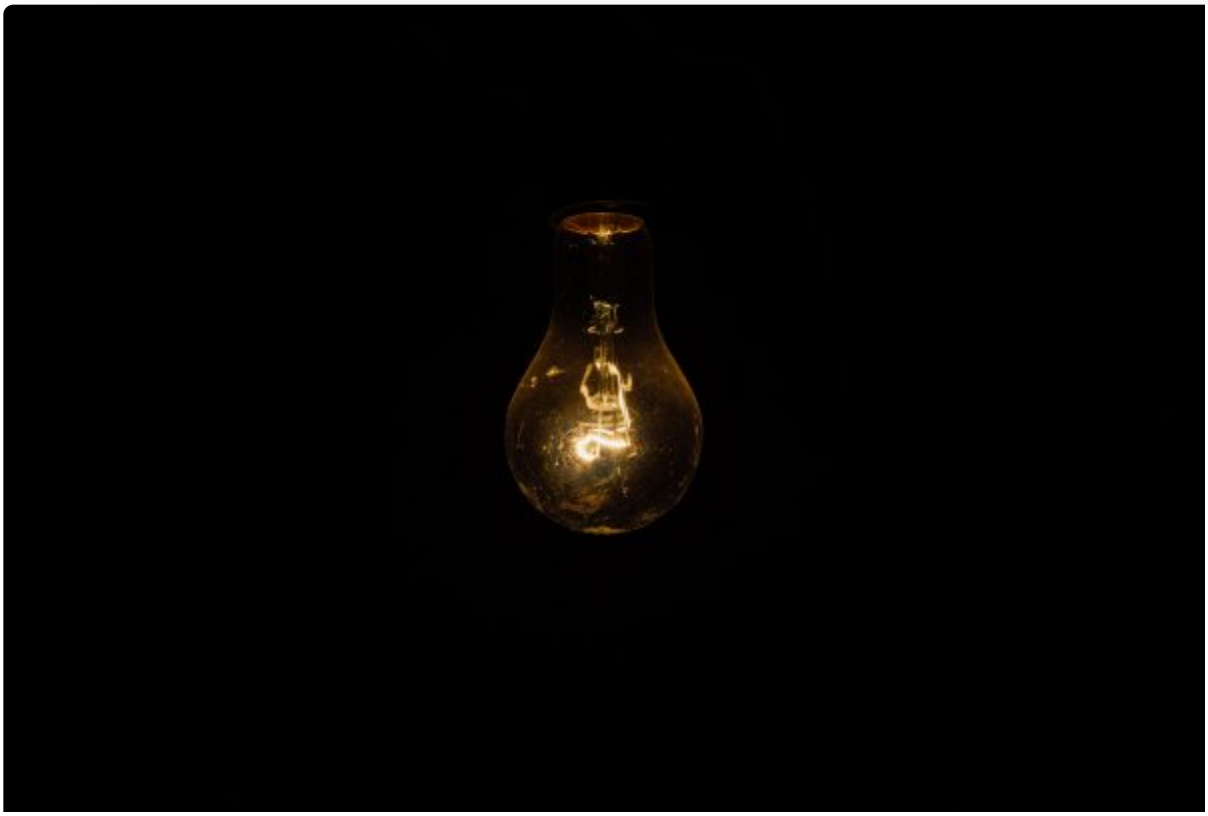
Data Scientist

Previous Post

Day 128: NLP Papers Summary - Neural Approaches to Conversational AI - KB-QA (Symbolic Methods)

Next Post

Day 130: NLP Papers Summary - Neural Approaches to Conversational AI - Text-QA (MRC)



Data Science Natural Language Processing NLP Papers Summary

Day 365: NLP Papers Summary – A Survey on Knowledge Graph Embedding

Ryan
30th December 2020

Loading [MathJax]/extensions/MathZoom.js

Data Science Implementation Natural Language Processing

Day 364: Ryan's PhD Journey – OpenKE-PyTorch Library Analysis + code snippets for 11 KE models

Ryan
29th December 2020

Loading [MathJax]/extensions/MathZoom.js