KARL: Knowledge-Aware Reasoning Memory Modeling with Reinforcement Learning of Vector Space

Anonymous Author
Transactions of the ACL
submission under open review

Abstract

Founded in Atkinson-Shiffrin Memory Model's three-stage theory, the cognitive process of answering a question with stored knowledge can be seen as a reasoning process that goes from the external sensory memory via short-term knowledge-aware reasoning towards internal knowledge storage. While in the machine, this process can be interpreted as a pipeline from encoding the question's information, through decoding the reasoning query of triples, towards the knowledgebase. How to encode and decode the varying semantic information of question into accurate triple query, and how to adjust the query generation with an evolving knowledgebase, are two inevitable problems in this cognitive process. Our model KARL provides a solution by designing the three memory spaces as, an encoder to handle the language modeling, a decoder for query generation, and a self-calibration with reinforcement learning of the knowledge representation vector space. We evaluate the model's reasoning ability in knowledge-based question answering against the Question Answering over Linked Data (QALD) benchmark, and achieve significant improvements in answers accuracy as compared to other neural models.

1 Introduction

The Atkinson and Shiffrin Model(Atkinson and Shiffrin, 1968) of memory states gives proof for modeling the reasoning process as a three-stage and multi-store pipeline that consists of the sensory memory, the short-term memory, and the long-term memory. Theoretically, the sensory memory is for extracting and holding the external information, while the short-term memory represents the knowledge-aware mental activities of reasoning, and the long-term memory is the storage of memorized knowledge. In our modeling, we compare the machine reasoning as a knowledge-based question answering (KBQA)(Yahya et al., 2012; Cui et al., 2019) a three-stage problem where a machine perceives a natural language question and interprets it into a query of entity-relation triples then querying towards the answer entity in the knowledgebase. And herein lies three problems: first, given the flexibility of language, how to perceive the varying semantic information from a question is inevitable for a machine; second, it is about the appropriate representation for the query of triples(Arias et al., 2011); third, a strategy for the model's learning to adjust with the modification of knowledgebase should be self-evolutionary(Lehmann et al., 2015). However, to the best of our knowledge, the recent researches of machine knowledge-aware reasoning(Ding et al., 2019; Das et al., 2017; Soru et al., 2018) always show less focus either on the varying of semantics (e.g. the polysemy and the synonymy), or on the studies about the self-calibration to an evolving knowledge, which calls for more attention to the comprehensive modeling that can multilaterally combine all into a complete cognitive process.

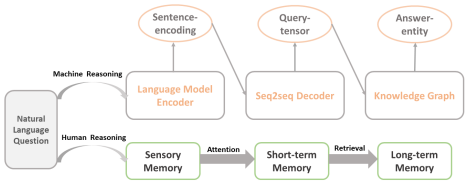

Figure 1: Machine Memory Model and Human Memory Model

The semantic meanings in natural language are abstract and varying, while the entities and relations in a knowledge graph (e.g. DBpedia, Wikidata)(Lehmann et al., 2015; Vrandecic and Krotzsch, 2014) are extracted and refined in a format of concrete nodes with connecting edges.

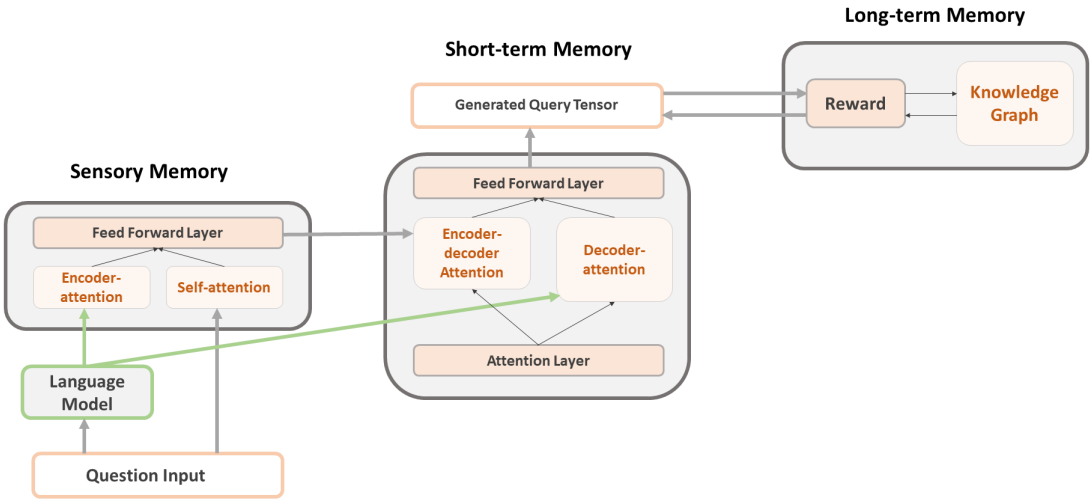

Figure 2: The model mainly has three components: **Sensory Memory**, the encoder part incorporated with pre-trained language model; **Short-term Memory**, the decoder part for query tensor generation; **Long-term Memory**, the interaction of the query tensor generation and the reward function in the reinforcement learning of knowledge-graph vector space.

The same question can be asked using different sentences with different vocabularies, and that the same vocabulary has biased meanings in different sentences. What's more, the knowledge graph is dynamic, for instance, the entity node for a famous star named *X* used to have an relation edge named *spouse* with another entity *Y*, but after they get divorced and married again to others, the edge *spouse* of *X* would have been reconnected to another entity may be named *Z*. Although in these cases the answer's explicit node content changed, the natural language sentences, no matter "Who is the spouse of X ?", "Who is the husband of X ?" or "Which one is married to X?", the logic structure like triple $\langle entity(X), relation(spouse), entity(?variable)\rangle$ is unchanged during reasoning, with its semantic similarity remains in vector space, which also keeps stable the SPARQL query(Arias et al., 2011; Pérez et al., 2006) for answering this question in the knowledge graph i.e.$SELECT\ ?var\ WHERE\{dbr{:}X\ dbo{:}spouse\ ?var.\}$. This opens the opportunity to build the reward function with reinforcement learning algorithms of vector space to self-calibrate the query generation. In our model, we base a sequence-to-sequence structure integrated with the pre-trained language model BERT on the reinforcement learning of knowledge embedding vector space, to imitate a three-component cognitive process for knowledge-aware reasoning.

The Sensory Memory is the first component, to begin with the modeling. Due to the variety of semantic biases in contexts, the phenomenon of polysemy and synonymy is frequent in asking the questions, especially when the questions are only short sentences. To disambiguate these semantic biases, the support of a pre-trained language model that has learned a vast content of language materials is necessary for universal language representation. Here we leverage the pre-trained BERT in our modeling. We incorporate the BERT into the attention-based encoder part in the sequence-to-sequence structure for the question encoding.

In the second component, the Short-term Memory, the major architecture is a decoder also based on attention mechanism. The role for this component is to decode the question encoding from the first component into a SPARQL query(Arias et al., 2011; El-Roby and Ammar, 2018) to fetch the entity from a knowledge graph, such as DBpedia.

Finally, in the third component, after vectorizing the generated SPARQL queries in the second component with global vectors for word representation(GloVe)(Pennington et al., 2014), we tune the model's training loss function to build the reinforcement learning algorithm that optimizes the semantic distance between the targeted entity of the generated query and the plausible answer entity of the golden query.

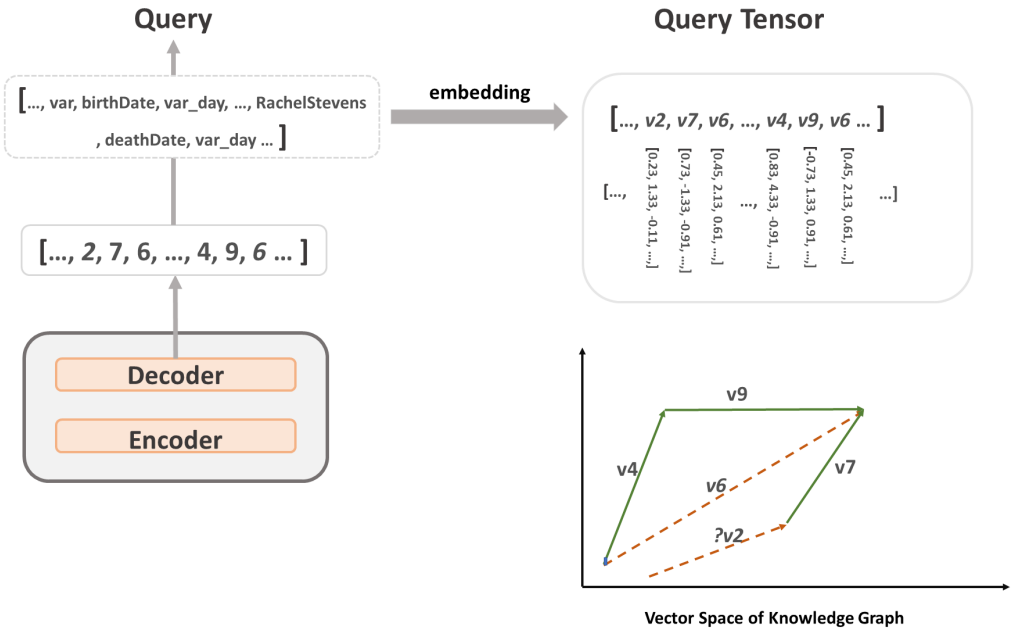

Figure 3: The visualization of the major workflow: 1) the sequence-to-sequence structure encodes the question and decode into a query tensor; 2)in the generated query tensor, each element is an integer index for the entity or relation embedding; 3)then, the embedded query tensor from step 2 is parsed into triples, and all the embeddings in the triples of the query are reduced into a vector representing what the query is asking.

2 Background and Related Works

We briefly introduce the background of reasoning models for knowledge-based question answering and review current techniques.

2.1 Memory Models

In cognitive science, the researches of memory models(Atkinson and Shiffrin, 1968; Baddeley, 2018) theorize the conscious awareness process of human perceiving and retrieving memories as a continuation of multiple stages. The Atkinson-Shiffrin model advocates that human memories consist of three separate components, namely, the sensory register, where the perceived information enters memory, the short-term holding memory also called working memory or short-term memory, where is considered as an interface that handles both the inputs from the sensory register and the retrieval of stored long-term memory, and the long-term store, where the learned knowledge is stored for a long period. Expanding Atkinson and Shiffrin's work, the Baddeley's Model of Working Memory(Baddeley, 2018) presents more detailed modeling of short-term memory, where the working memory during human's simultaneous pondering and reasoning on the perceived information is considered as an independent processing unit of instant memories instead of just a sub-component of short-term memory. These two studies provide a supportive theorem to our model KARL to compute the interaction between the short-term extracted information from natural language texts and the long-term information stored in the knowledge graph. In computer science, computational models inspired by Memory Model theories proved the feasibility of implementation in the domain of dialog question answering generation(Chen et al., 2019) and questionanswering(Wang and Nyberg, 2015). But, unlike the prior design(Chen et al., 2019) which incorporated the long-term memories into both the encoder's and the decoder's structures, our model takes the knowledge graph in the component for Long-term Memory, as the role of long-term memory and considers the generated query vector in Component II as the representation of short-term memory.

2.2 Knowledge-Based Question Answering

For the recently published knowledge-based question answering models(Cui et al., 2019; Ding et al., 2019; Das et al., 2017; Huang et al., 2019) which also attempt to answer a question with explainable reasoning, they store their knowledge in three styles of methods. The models of the first style(Cui et al.,

2019) is to store knowledge in the traditional type of knowledge graph, such as Wikidata and DBpedia, which are already prepared and fixed external knowledge bases. On the contrary, the second type(Ding et al., 2019) tried to build the graph from texts as the model training goes on, whose knowledge base, in short, can be generated by the instant segmentation or extraction when building the model. The third type, it shows an ambition to go beyond the restricted structure of nodes connected with edges, instead they transform the graph into homogeneous or heterogeneous embedding spaces. But for this type of method(Huang et al., 2019), on one hand it alleviates the limitations brought by a fixed graph structure on the other hand it requires more time and resource to train a set of embedding(Yang et al., 2014) from a knowledge graph at the cost of losing much explicit information in the concrete nodes and edges. What's more, once the knowledge graph gets re-edited, i.e. several nodes modified or a few edges redirected even deleted which causes an impact over the whole graph, then the embedding needs to be trained again.

The strategies used by current knowledge-based question answering models can be typically classified into two branches. The first is to search for a path and arrive at the answer's node in the knowledge graph. The path-searching, it can be done by pattern matching of the text (i.e. matching some phrases or vocabulary) in a graph that's constructed with the segmented parts from texts, or it can be helped by algorithms of reinforcement learning in wandering the path towards the right answer node with a simplified reward function that if it is right path then return a positive reward value else return a negative value to update its learning state.

3 Architecture

The model architecture can be seen as three components to bridge the gaps among the three states. The cognitive process is perpetuated with the three-state pipeline.

3.1 Sensory Memory

The Sensory Memory component of our model is a question encoder incorporated with a pre-trained language model.

3.1.1 Language Modeling

The pretrained language model(Dai et al., 2019; Radford, 2018; Devlin et al., 2018) helps the model in better extracting the semantic information from a rather short question sentence that lacks of abundant contextual background information. The pre-trained language model works in two layers, in the architecture's sequence-to-sequence encoder, and in decoder (Sutskever et al., 2014). This design is grounded in the previous efforts from neuroscience and psychological experiments(Cao and Perfetti, 2016; Kuhl, 2010; May L, 2011) on the observation and analyses of how human brain neural structure differently responses to reading familiar and unlearned languages(Cao and Perfetti, 2016; Kuhl, 2010) and how the psychological activities develop during these readings(May L, 2011). These researches assert that a pre-acquired language system(May L, 2011) for a learned language is necessary for the process of perceiving external information into internal sensory memory. Without the pre-acquisition of the language, it is not only impossible to interpret the language into our memory system but also tended to cause the psychological anxiety for unlearned language with a vibration effect in the neural aspect. Thus, in our computational cognitive modeling of memory states for reasoning in knowledge-based question answering, it is reasonable and essential to introduce the pre-trained language model to serve the role as the pre-acquired language system works in interpreting the information from natural language question sentences.

In regards to the length limited and the lack of context of a question sentence, the semantic bias can be a problem in the process of comprehensively and multilaterally interpreting the sentence into sensory memory. For example, the question "where is the bank of paris" can be both the river beach of Paris city and a financial institution of Paris. Without the pre-trained language model, the encoding for such sentences will be unilateral and limited in information.

First, in the encoder, the BERT(Devlin et al., 2018) outputs the embedded sentence tensor as the input of the Encoder-attention layer. Through this process, the sentence gets weighted by the trained parameters in the pre-trained model which is proven to be efficient in universal language representation(Edunov et al., 2019). Our model incorporates the BERT into encoder-decoder gets inspiration from the BERT-fused-NMT(Zhu et al., 2020). Then, in the decoder, there's also an attention layer that accepts the sentence embedding from BERT. This strongly helps in better maintaining the se-

mantic information of question sentence all along through the encoder to the decoder.

3.1.2 Encoder

We construct the encoder with an attention layer and a feed-forward layer. The attention layer(Vaswani et al., 2017) consists of two attention mechanisms to compute the output from the pre-trained language model and the tokens index tensor from the embeddings of the segmentation of sentence. We use the general design of attention layer from the paper(Vaswani et al., 2017), denoting the Key, $Value$, and$Query$ in the original algorithm as K ,V, and$Q$, then the attention layer is defined as

$$Attn(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

$$head_i = Attn(QW_i^Q, KW_i^K, VW_i^V) \quad (2)$$

Then we build the multi-head attention function as follows, the $concat$ is the function to concatenate the tensors,

$$MultH(Q, K, V) = concat(head_1, ..., head_i)W^O \quad (3)$$

which's used in the *Encoder-attention*, *Self-attention* of encoder, the *Encoder-decoder Attention*, *Decoder-attention* of decoder depicted in the Figure22. The structure of feed-forward layers in our model is position-wise feed-forward network(Vaswani et al., 2017; Ott et al., 2019) that each time adds up the two outputs from the previous two attention mechanisms through layer normalization(Vaswani et al., 2017). This structure is also used in both the encoder and the decoder.

3.2 Short-term Memory

The Short-term Memory is the decoder part for the query tensor generation, where reflects the explainable logic reasoning of the triples in the question sentence.

3.2.1 Decoder

The decoder component consists of three layers. The first is an attention layer, following with the second attention layer, and the last is a feedforward layer. The second layer is stacked with two parallel attention mechanism that respectively accepts the output tensor from the last layer of the encoder and the output from the pre-trained language model, and the outputs of these two attention mechanisms will be added and normalized at the end of the second layer as the input for the third layer. At last, as the output of the third layer, a query tensor is generated, which represent a SPARQL query sentence that can fetch back a correspondent entity in the knowledge graph. The generated query tensor is a one-dimension integer tensor, in which each integer element is a numeric index of one token in SPARQL language. To better understand how the SPARQL query works in interpreting the basic logical structure of the entities and relations in a question sentence, Figure 4 is two examples that visualized the comparison and relation of the natural language question with its tokenized SPARQL representation and the matching answer entity in the knowledge graph.

3.2.2 Generation of Query Tensor

This Figure 4 displays the two types of queries in SPARQL language. In a query, the core is the clause starting with ***WHERE*** surrounded by two braces. The first type starting with ***SELECT*** is to query a certain entity node. And, the second type starting with ***ASK***, different from the first type, is to assert whether the RDF relations described in the **WHERE** exist, if it exists return ***True*** else ***False***. In the **WHERE** clause, there are single or multiple triples in the form of $\langle head\_entity, relation, tail\_entity \rangle$, e.g.in the examle for Type.1 $\langle var, dbo : restingPlace, dbr : Great\_Pyramid\_of\_Giza \rangle$, and the triples are separated by a dot . In these triple, those tokens starts with string ***var*** are the representative symbols of those undetermined entities. So, these generated query tensors can be mapped to a list of SPARQL tokens and join into a string of query which's able to fetch back certain entities nodes through the server endpoint of a knowledge. Even though the graph construction of a knowledge base is periodically modified, the verbal form of query for th certain answer can remain stable. In some rare cases, although the name of certain entity or relation will get renamed, e.g. the $Obama, dateOfBirth, var$ might turn to $Obama, birthday, var$, the semantic similarity based on vector cosine distance between the adding-up of embeddings of ***Obama*** and ***dateOfBirth*** the embedding of ***Obama*** and ***birthday***. As a result, according to the algorithm of TransE(Bordes et al., 2013), in the embedding vector space of knowledge representation, the *entity_head* plus the *relation* equals to the *entity_tail*, and the *entity_tail* minus the *relation* equals to the

entity_head .

3.2.3 Reasoning with the Query Tensor

SPARQL Query SPARQL query can be considered as a path that can wind through the entity nodes and relation edges and finally arrive at the answer node. The interesting part of SPARQL is, its nodes and edges are determined by their verbal form of names, i.e. the composition of the vocabularies used in the names of the entity or relation. In short, as an analogy that maybe not very descriptive, our sequence-to-sequence model is an laser-gun sharpshooter who picks up the natural language questions as the bullets and shot different bullets as laser vectors in different directions into the vector space, as the vector space develops our model learns to adjust the shooting and improves its hit rate at the target entities.

Reasoning The generated query tensor, after embedding, will be parsed into a graph of triples for non-monotonic reasoning(Hunter, 2018). That's to say, preasuming that in each triple we have $head$+$relation$=$tail$ in the knowledge-graph vector space following closed-world assumption, if a triple is defaulted (i.e., the triple contains the uncertain variable entity), the defaulted variable entity can be represented by: $vector(entity_{head}) = vector(entity_{tail}) - vector(relation)$, if $entity_{head}$ is var; $vector(entity_{tail}) = vector(entity_{head}) + vector(relation)$,if $entity_{tail}$ is variable. Similar to TransE(Bordes et al., 2013), we use GloVe(Pennington et al., 2014) to embed the tokens in triple to calculate the representation vector for the query in vector space. For instance, the token for the entity of *City of Westminster* in SPARQL language is *dbr:City_of_Westminster*, then the embedding for this entity is the add-up of the GloVe embedding of the *city* and the *Westminster* after removing the stop-word *the*. The algorithm for calculating the query representation vector is a loop: if the triple contains only one variable, the variable can be easily calculated by the other two certain elements in triple; else if the triple contains more than one variable, we search other triples that contain one of these variables. If the searched triple also contains variables in addition to the previous one, we trace the new variable back to search other triples. This will be considered as build the triples of the query into a graph with a tree structure in Figure 4.

3.3 Long-term Memory

In Long-term Memory, there are two major parts: the knowledge graph, and the reward function. The reward function is the core of the reinforcement learning of vector space, where the SPARQL query towards the knowledgebase is turned into a vector and this vector's distance is the objective to be optimized.

3.3.1 Knowledge Graph As Vector Space

The long-term knowledge storage structure of the Sensory Memory component in our model KARL is the knowledge graph of DBpedia. A knowledge graph can be represented in an embedding vector space, where the entities and relations are mapped as vectors in the vector space. This is highly efficient for more direct computation of the triples, in the form of calculable embedding tensors. In the embedding representation of the knowledge graph, which is elaborated in the methodology of TransE(Bordes et al., 2013) and DistMult(Yang et al., 2014), the problem of pointing the query toward the answer entity can be achieved with the arithmetic operations of the embedding tensors. Thus, we transform the embedding vector space of a knowledge graph into a vectorized environment for reinforcement learning.

3.3.2 Reward Function And Reinforcement Learning

Since the query can be embedded into a vector that arrows at a certain point in the space, the metric for how to measure the loss between the generated query tensor and the correct query tensor is defined as a function that calculates the similarity based on cosine distance. The objective function is to maximize the similarity by minimizing their cosine distance.

To design our reward function, we first set a label-smoothed cross-entropy function to get the loss of distribution of the generated query as the basis value of the reward. Then, on this basis, we calculate the cosine similarity between the generated query and its standard correct query to get a value as an additional weight for the basis cross-entropy loss value. Finally, the model multiplies the basis cross-entropy loss with the weight of cosine similarity. After the training of each epoch, by comparing the current loss of the latest epoch with the best loss among the previous epochs, the model automatically selects the training state with

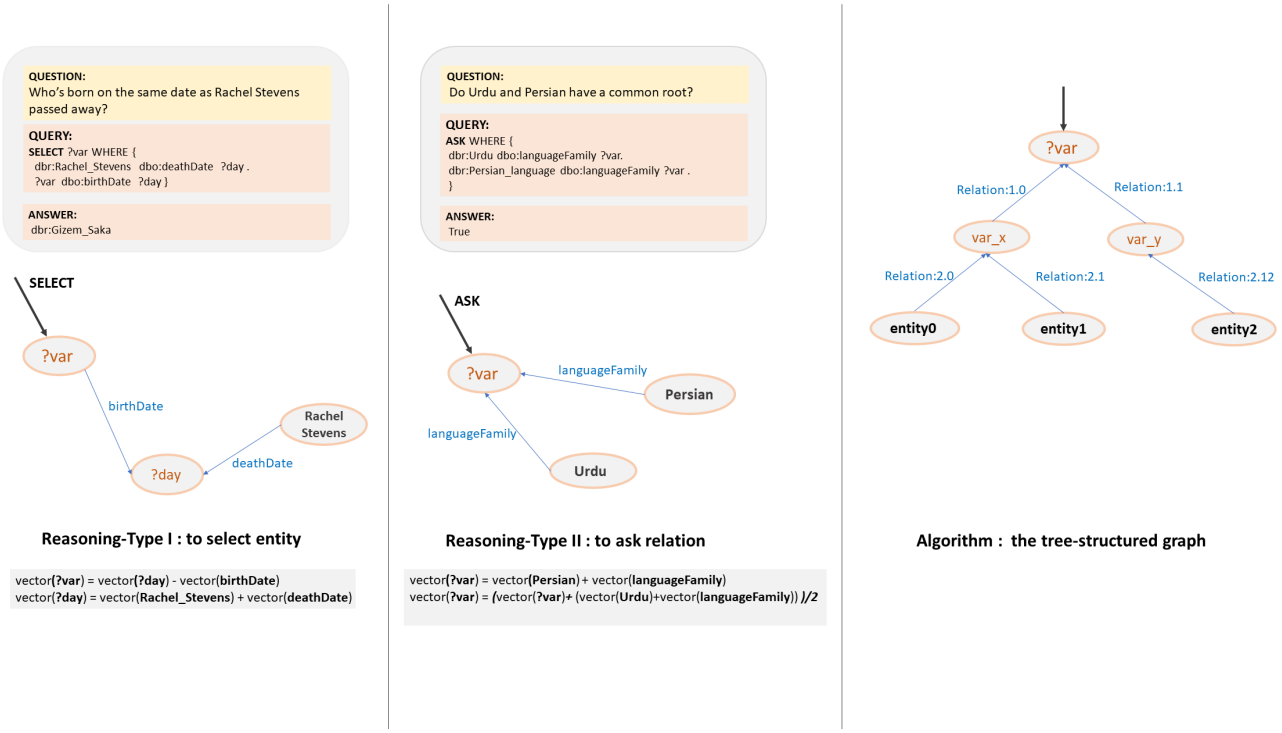

Figure 4: For vivid case studies, here is the figure displaying three columns:1)Reasoning Type-I, is the logic structure for the first type of query which aims to select an entity that conforms with all the premises(i.e. all the triples in the query); 2)Reasoning Type-II, is the logic structure for the first type of query which aims to ask whether all the containing triples conform to the logic relations in the query, the answer of which is true or false in closed-world assumption of a knowledge graph; 3)it is the method in section3.2.3 for calculating the variables from the given entities and relations.

the best performance to update for optimization in the training.

States: Defining the whole embedding vector space as the state space $\mathcal{S}$ with all its valid triples $T = \{(e_{head}, r, e_{tail}) | e_{head} \in E, e_{tail} \in E, r \in R\}, T \in S$ where $\mathcal{R}$ and $\mathcal{E}$ denote all entities and relations in $\mathcal{S}$. the model KARL is the agent that's expected to transform the natural language question through a sequence-to-sequence structure into the query tensor that represents the basic logic triples in the form of SPARQL query language, in order to point the query as accurately as possible to the answer entity. Thus, a state is represented by the generated query tensors $q \in Q$, $Q = \{t(e_{head}, r, e_{tail}) | e_{head} \in E, e_{tail} \in E, r \in R, t \in T\}$.

Observations: The agent KARL observes the natural language questions, interpreting them to query tensors that are as close as possible to the expected target entity in the graph.

Actions: The actions are the query tensor generation3.2.2. The generated query tensor from interpreting its natural language question is required to contain all the triples with the potential entities and relation in the question to form a sequence of SPARQL queries to reach the answer entity.

Transition: The model's training evolution for the environment is driven by the semantic similarity distance(Wieting et al., 2019) towards the target. The transition is achieved by updating the loss of the generation distribution of the query tensor, utilizing minimizing the label-smoothed cross-entropy(Müller et al., 2019) loss compared with the distribution of the optimal query. The label-smoothing turns the label y_k into y_k^{LS}, with denoting p_t as the probability of correct generation, to get $y_k^{LS} = y_k(1-\alpha) + \alpha, y_k \in \{0, 1\}$, then the label-smoothed cross-entropy is defined as

$$H(y, p) = \sum_{k=1}^{K} -y_k log(p_k) = -log(p_t) \quad (4)$$

Reward: The rewards in the vector space of the knowledge graph are continuous, and this reward is considered as a weighted value based on the label-smoothed cross-entropy loss. Denote by R, $vector_{gen}$, and $vector_{gold}$ respectively the reward, the generated query's embedding vector, and the

correspondent golden-standard query's embedding vector, with setting $R \in [1, 2]$, the reward is defined as

$$R = 2.0 - sim(vector_{gen}, vector_{gold}) \quad (5)$$

where the sim is the similarity function from the equation() mentoned above.

4 Experiment

4.1 Main Experiment

The encoder-decoder structure in our model is implemented with the help of the package Fairseq(Ott et al., 2019).

4.1.1 Dataset

The datasets for the experiments contain the source data which are natural language questions and the target data which the corresponding SPARQL queries that provide the basic entity-relation tokens(Ricardo Usbeck and Ngomo, 2018; Soru et al., 2018) and syntactic structure(El-Roby and Ammar, 2018) for the model to learn. Our training data is from the DBNQA(Marx, 2018), and we evaluate the model's reasoning performance in knowledge-based question answering with the tasks in QALD datasets.

Training Data The DBNQA is the dataset for training our model. To the best of our knowledge, it is the largest natural language to SPARQL dataset with different samples of SPARQL queries from different development versions of the DBpedia. As a result, the queries in DBNQA contains not triples from multiple previous modifications. Thus, it is one sufficient resource dataset to learning about the changes in the verbal forms and the triple relations.

Preprocessing To enlarge the vocabulary volume for interpretation and generation also handling the out-of-vocabulary phenomenon, the BPE(byte pair encoding)(Sennrich et al., 2015) method from neural machine translation is introduced into the preprocessing works of both the source and target data. We use BPE to get te subtokens of all the vocabulary, and these subtokens are made into a vocabulary dictionary with unique integer indexes for each subtokens. During the sequence-to-sequence structure that turns natural language questions into query tensors, the vocabulary dictionary helps in handling those vocabularies that are unprecedented in the training dataset, which on one hand fortifies the model's flexibility and capacity in dealing with the modification of entities and relation in knowledge graph, on the other hand also improves its ability in interpreting the flexible composition of words in the natural language questions.

4.1.2 Evaluation

The QALD is an annual challenge for knowledge-based question answering which provides updated datasets that match each year's modification in the knowledgebase of DBpedia. QALD provides the task where multilingual question answering over DBpedia is available in eleven different languages (e.g., English, Spanish, German, Italian, French, Dutch, Romanian or Farsi). For our evaluation, we currently choose the datasets for knowledge-based question-answering in English. Each data item contains a SPARQL query with its questions written in multiple languages4. In our experiments, we denote the given two official datasets, the *qald-9-train-multilingual.json* and the *qald-9-test-multilingual.json*, as *Set-I* and *Set-II*.

4.1.3 Results

For fair experiments, the evaluations were all run on the online server of GERBIL[1] which is the official open platform to provide QALD bench. To the best of our knowledge, the NSpM(Soru et al., 2018) is the precedent state-of-the-art in neural models which depends on translating natural language questions into SPARQL queries with neural machine translation methods, we compare our model's performance with NSpM's, the results were calculated and recorded by GERBIL.

Up to QALD9, the non-neural models have been dominating the leaderboard, and the previous neural models are not yet comparable with the non-neural ones. In Table1, we show significant improvement compared to the neural model NSpM,. In **Set-I**, as the highest score in leaderboard2, our model is 52.78 percent higher than the precedent neural model in F-1 score1; in **Set-II**, like the one neural model that can be comparable with the rule-based models, our model is 27.18 percent higher than the precedent neural model in F-1 score1. It is of our contribution to largely improve the neural model's performance in knowledge-based question answering benchmarks.

Model	Dataset	C2KB	P2KB	RE2KB	Macro-F1 on QALD
NSpM	Set-I	0.049	0.049	0.049	**0.0049**
-	Set-II	0.1733	0.18	0.18	**0.0132**
Our Model	Set-I	0.4338	0.4338	0.4338	**0.5327**
-	Set-II	0.2111	0.2133	0.2133	**0.285**

Table 1: **Evaluation Results**:In the header, in addition to the first column that indicates different neural models, there are four metrics(et al., 2018): 1)**C2KB**(Concept to Knowledge Base.), aims to identify all relevant resources(i.e. the entities in a triple) for the input question, which calculates the F-1 score of the resources generated by the model; 2)**P2KB**(Properties to Knowledge Base), measure how well the model can identify the relevant predicates(i.e. the relation in a triple) in the question, and returns the F-1 score; 3)**RE2KB**(Relation to Knowledge Base), calculates the F-1 score based on the correctness of the generated triples from the question; 4)**Macro-F1 on QALD**, measures the models' answering accuracy based on the precision and recall on the over-all dataset.

System	F1-train	F1-test
Elon(dictionary-based)	0.1327	0.1001
gAnswer(graph-matching-based)	0.5307	0.4296
QASystem(rule-based)	0.3321	0.1995
TeBaQA(templates-based)	0.2678	0.2216
wdaqua-core1(rule-based)	0.4065	0.2887
our model(neural-network-based)	**0.5327**	**0.285**

Table 2: **Contribution:we make the neural model competable against the other non-neural methodologies.**This table shows the comparison between the latest models with leading performances and our model. The previous models were trained on the train-set and evaluated on both the train-set and test-set. However, to see our model's performance on adjusting with modified versions of the knowledge graph, we trained our model KARL on the DBNQA dataset that contains no-longer valid queries. And, the DBNQA dataset was earlier before the publication of QALD-9 dataset. Thus, the QALD-9 datasets are suitable for evaluating the knowledge-based question answering ability in spite of the modifications varying with time.

Encoder structure	Accu
encoder with BERT	0.4722
encoder with XLNet	0.4763
encoder without any pretrained model	0.0297

Table 3: **Encoder Ablation Study** The accuracy different in question classification by question encoding from variant decoders.

4.2 Experiments on Ablation Study

4.2.1 Question Encoding

To investigate the differences in question encoding brought by incorporating different pre-trained language models, we use customized K-means(Pelleg and Moore, 1999) with setting cosine-based similarity as the metric for the distance between question embeddings. In QALD, each answer for the question is classified into different categories in the knowledge graph, and this is the label for classification. Here, denote by x, μ the encoded tensor and the cluster centroid respectively, we define the similarity function as

$$sim(x_i, x_j) = \frac{\cos(x_i, x_j) + 1}{2} \tag{6}$$

The clustering algorithm updates by calculating $S_i^{(t)} = \{x_p : sim(x_p, \mu_i^{(t)}) \leq sim(x_p, \mu_j^{(t)}) \; \forall j, 1 \leq j \leq k\}$ with the $\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$
to optimize the objective function

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} sim(x_n - \mu_k)$$

. Denote by $Accu$, C the probability of predicted category for the cluster and the prediction accuracy by the similarity-based clustering respectively,

$$Accu = \sum_{k=1}^{K} p(x_k|C)$$

We use *Set-I* as the training set and *Set-II* as the testing set. Their results are shown in Table 33:

According to Table2, it shows that: first, the encoders with pre-trained language model performs better than the one without the pre-trained model; second, the one with XLNet achieves higher but subtle score than the one with BERT.

4.2.2 Decoder

The Table 4 compares structures using different decoders. The structure with decoder of bideirectional-RNN(Sutskever et al., 2014) and the structure with decoder of vanilla Transformer(Vaswani et al., 2017) do not achieve significant improvement compared to 1 performance on F-1 score. And their improvements are not competable with the non-neural models in Figure 2.

[1] http://gerbil-qa.aksw.org/gerbil/config

Decoder as Variant	C2KB	P2KB	RE2KB	Macro-F1 on QALD
Decoder of Bi-RNN with Attention(Sutskever et al., 2014)	0.18	0.1867	0.1867	**0.0262**
Decoder of vanilla Transformer(Vaswani et al., 2017)	0.18	0.1867	0.1867	**0.0391**

Table 4: **Decoder Ablation Study evaluated on Set-II** We remove the component of reinforcement reward function, only keeping the major encoder-decoder structure. We leave the encoder part unchanged as proposed in our model, and choose the two types of decoders that are frequently used, i.e., the "Decoder of Bi-RNN with Attention " and the "Decoder of vanilla Transformer". With the control variates method, the two experimental models both have the same encoder incorporated with BERT-base-uncased and are trained on DBNQA. The two are evaluated on Test-set.

4.2.3 Vector Space

The generated query tensor is parsed into a vector pointing towards the answer in the knowledge-graph vector space. We experiment to evaluate the answer prediction ability by ranking the cosine-based similarity between the query vector and all the vectors of answer entities in the **Set-I** and **Set-II**. The MRR(Mean Reciprocal Rank), Hits@10 and Hits@100 are to measure the accuracy that the vector can infer to the correct answer.

Dataset	MRR	Hits@10	Hits@100
Set-I	0.0065	0.0288	0.2877
Set-II	0.0163	0.0808	0.7576

Table 5: **Ablation Study on the Query Vectors' Answer Prediction**: the generated query will be parsed and embeded into a query vector, we use this same method to generate the query vectors based on the given SPARQL query set in **Set-I** and **Set-II** and evaluate the accuracy that the vector infers towards the answer.

5 Discussion and Conclusion

In the model KARL, we propose computational cognitive modeling of the memory model for reasoning on knowledge-based question answering. The research of how the external information is transformed then connected towards the stored information via an explainable logic process is still of much empty space to be explored. Besides, by challenging the previous leaderboard dominated by rule-based and matching-based systems, our methodology KARL demonstrates the feasibility of neural reasoning modeling in knowledge-based question answering, showing significant improvements in benchmarks.

Why does the model build vectors for entities with the GloVe instead of the knowledge embeddings? As we have discussed in the previous section2.1, we focus on the reasoning modeling that can maintain stable in spite of the periodical modifications of the knowledge base, which in human cognition is the belief revision(Chi, 2008; van Benthem, 2007), i.e., the re-edition of stored knowledge. And, the modification of a knowledge graph will inevitably cause biases upon its previous embeddings. We would hope to find a method that can be of a certain level of independence to the knowledgebase. The SPARQL query can be considered as a path to arrive at the expected answer entity. However, if a query is only represented by a list of tokens, the query easily gets expired when it loses its semantic meanings during the knowledge modification and tokens change. Thus, we evaluated how well this vectoring method can achieve in the experiment5 of ranking evaluations on answers prediction.

In fact, is the SPARQL query a restriction of the development of knowledge-based reasoning and question-answering? Surely yes, not only the query but also the knowledge graph itself is a barrier, as one's self is just its own biggest enemy. The query, entity(node) and relation(edge) in a knowledge graph are categorized into the ontologies, however, the relations and ontologies for the same item can be varying according to different contexts. Thus, a computable probabilistic graph model for knowledge storing will be better in handling the likelihood of choosing the right path when standing at one entity and facing multiple existing relation edges, for instance, if an entity both has the relation edges *husband* and *wife*, in complex context, which relation is more pertinent for the context *who is her spouse*.

For the future works: As the two questions discussed above, we will continue to: (1) Auto-simultaneous vectorization of the knowledge base, which gets rid of the training the embeddings again when the knowledge graph modifies. (2) Work on more profound knowledge-graph reasoning. We are currently staying in an initial stage for exploring the logic of knowledge, and the computational non-monotonic reasoning modeling is worth more attention.

References

Ricardo Usbeck et al. 2018. Benchmarking question answering systems. *Semantic Web Journal*, 1838-3051.

Mario Arias, Javier D. Fernandez, Miguel A. Martinez Prieto, and Pablo de la Fuente. 2011. An empirical study of real-world SPARQL queries. *CoRR*, abs/1103.5043.

R.C. Atkinson and R.M. Shiffrin. 1968. *The psychology of learning and motivation*. New York: Academic Press.

A. Baddeley. 2018. *Exploring Working Memory*. London: Routledge.

Johan van Benthem. 2007. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics*, 17(2):129–155.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Fan Cao and Charles A. Perfetti. 2016. Neural signatures of the reading-writing connection: Greater involvement of writing in chinese reading than english reading. *PLoS ONE*, 11(12).

Xiuyi Chen, Jiaming Xu, and Bo Xu. 2019. A working memory model for task-oriented dialog response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2687–2693, Florence, Italy. Association for Computational Linguistics.

Michelene Chi. 2008. *International Handbook of Research on Conceptual Change*. Stella Vosniadou.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2019. KBQA: learning question answering over QA corpora and knowledge bases. *CoRR*, abs/1903.02419.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alexander J. Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *CoRR*, abs/1711.05851.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703, Florence, Italy. Association for Computational Linguistics.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. *CoRR*, abs/1903.09722.

Ahmed El-Roby and Khaled Ammar. 2018. Sparql endpoint sparql endpoint sparql endpoint query completion cached predicates and literals query suggestion federated query processor query terms term suggestions user query answers query query suggestions client sapphire server web predictive user model.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 105–113, New York, NY, USA. ACM.

Anthony Hunter. 2018. Non-monotonic reasoning in deductive argumentation. *CoRR*, abs/1809.00858.

Patricia K. Kuhl. 2010. Brain mechanisms in early language acquisition. *Neuron*, 67:713–727.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sren Auer, and Christian Bizer. 2015. Dbpedia a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

Edgard Marx. 2018. DBpedia Neural Question Answering (DBNQA) dataset.

Gervain J Werker JF May L, Byers-Heinlein K. 2011. Language and the newborn brain: does prenatal language experience shape the neonate neural response to speech? *Front Psychol*.

Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. 2019. When does label smoothing help? *CoRR*, abs/1906.02629.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Dan Pelleg and Andrew Moore. 1999. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 277–281, New York, NY, USA. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference*

on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Jorge Pérez, Marcelo Arenas, and Claudio Gutiérrez. 2006. Semantics and complexity of sparql. In *International Semantic Web Conference*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Muhammad Saleem Ricardo Usbeck, Ria Hari Gusmita and Axel-Cyrille Ngonga Ngomo. 2018. 9th challenge on question answering over linked data (qald-9). In *CEUR Workshop Proceedings*, volume 2241.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Tommaso Soru, Edgard Marx, André Valdestilhas, Diego Esteves, Diego Moussallem, and Gustavo Publio. 2018. Neural machine translation for query construction and composition.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Denny Vrandecic and Markus Krotzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 707–712, Beijing, China. Association for Computational Linguistics.

John Wieting, Taylor Berg-Kirkpatrick, Kevin Gimpel, and Graham Neubig. 2019. Beyond BLEU:training neural machine translation with semantic similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4344–4355, Florence, Italy. Association for Computational Linguistics.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *EMNLP-CoNLL*.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*.