Entity Linking and Filling for Question Answering over Knowledge Graphs

Daniel Diomedi and Aidan Hogan

DCC, Universidad de Chile; IMFD; Santiago, Chile
daniel.diomedi@ug.uchile.cl, ahogan@dcc.uchile.cl

Abstract. Question Answering over Knowledge Graphs (KGQA) aims to compute answers for natural language questions over a knowledge graph. Recent KGQA approaches adopt a neural machine translation (NMT) approach, where the natural language question is translated into a structured query language. However, NMT suffers from the out-of-vocabulary problem, where terms in a question may not have been seen during training, impeding their translation. This issue is particularly problematic for the millions of entities that large knowledge graphs describe. We rather propose a KGQA approach that delegates the processing of entities to entity linking (EL) systems. NMT is then used to create a query template with placeholders that are filled by entities identified from the text in an EL phase. This approach gives rise to what we call the "entity filling" problem, where we must decide which placeholders to replace with which entities. To address this problem, we propose a solution based on sequence labelling and constraints. Experiments for QA with complex questions over Wikidata show that our approach outperforms pure NMT approaches: while the task remains challenging, errors relating to entities in the translated queries are greatly reduced.

1 Introduction

Knowledge graphs (KGs) adopt a graph-based abstraction of knowledge, where nodes represent entities and edges represent relations [13]. Many open knowledge graphs provide public query services that can receive upwards of millions of SPARQL queries per day over the Web [18]. In order to query such services, clients must be able to formulate their queries in SPARQL, which may be unfamiliar to many users. Ideally a user could instead receive answers to questions written in a natural language familiar to them. Addressing this task automatically is known as Question Answering over Knowledge Graphs (KGQA).

Recent approaches for KGQA have moved towards neural networks [4]. Using neural machine translation (NMT) to translate the natural language question of the user into a structured query (e.g., [16,20,24,26]) has recently shown promising results, particularly for simple "factoid" questions [3] (such as "*Where was Marie*

Curie born?"), that can be answered by a single edge in the knowledge graph (`Marie Curie` $\xrightarrow{\texttt{born in}}$ `Poland`). Handling complex questions that require joining multiple edges (such as "*What was the profession of Marie Curie's father?*"), and/or involve query features such as counting, ordering, etc. (such as "*How many daughters did Marie Curie have?*") remains very challenging.

In this paper, we focus on advancing the state-of-the-art for NMT-based KGQA. After discussing related works, we analyse the performance of state-of-the-art "pure NMT" approaches, finding that their performance leaves considerable room for improvement. We identify that the most problematic issue relates to the out-of-vocabulary problem, particularly for entities. Based on this observation, we investigate the specific claim that *an approach combining entity linking (EL) with NMT can improve performance versus a pure NMT approach (without EL)* by designing, implementing and performing experiments to compare such an approach with analogous pure NMT baselines.

Specifically, in our approach, NMT is used to generate an abstract query template with generic placeholders for entities, while a specialised ensemble of entity linking systems (EL) is used to extract entities from the question and match them with the knowledge graph. This approach still leaves us with a key problem that we call *entity filling* (EF) – which can be seen as a particular form of *slot filling* – where we need to decide which placeholders in the query template should be replaced with which entities returned from the EL process. To address this problem, we propose to (1) first use a supervised sequence labelling approach (SL) to tag elements of the text with roles corresponding to the query template, and (2) thereafter apply a heuristic-based *entity assignment* (EA) that decides which placeholders in the query template to replace with which entities from EL based on the labels produced by SL. This approach then generates the final query predicted to represent the user's question.

Our results show that our approach greatly reduces the number of errors due to entities when compared with state-of-the-art pure NMT baselines, confirming our claim. However, even with these advances, the results for complex questions remain relatively poor, where we identify some open challenges for state-of-the-art methods when faced with such questions, as well as potential solutions.

2 Related work

Related tasks A number of tasks are closely related to KGQA. *Question Answering over Linked Data* (*QALD*) [22] can be seen as a type of KGQA targeting KGs specifically published as Linked Data. *Text-to-SQL* [10] relates to KGQA approaches that construct a query from the natural language question, such as NMT-based approaches; however, queries are answered over relational databases that have fixed schemas and do not exhibit the level of diversity present in large knowledge graphs. We thus focus on works for QALD and KGQA.

Non-Neural Approaches Early works on QALD avoided having to deal with the full complexity of natural language by defining control languages (e.g, GiN-

SENG [2]) or by using fixed templates (e.g., TBSL [21]). More flexible approaches involve using the question to guide a graph exploration (e.g., Treo [11], NLQ-KBQA [14]), where given a question such as "*What was the profession of Marie Curie's father?*", such an approach may first find candidate nodes in the knowledge graph referring to Marie Curie (e.g., `db:Marie_Curie` in DBpedia, `wd:Q7186` in Wikidata), thereafter navigating through relations matching "*father*" (`dbo:father`, `wdt:P22`) and "*profession*" (`dbo:occupation`, `wdt:P106`) towards a candidate answer. Further approaches (e.g., FREyA [5]) propose human-in-the-loop interaction, incorporating multiple sources on the Web (e.g., PowerAqua [15]), etc. For details on earlier systems, we refer to Unger et al. [22].

Neural Approaches A number of recent KGQA approaches leverage modern neural architectures. While neural networks have been used to classify questions, or to rank candidate queries, here we focus on approaches using neural machine translation (NMT), referring to Chakraborty et al. [4] for other uses. These NMT approaches rephrase the KGQA task into a translation task from natural language into a structured query language such as SPARQL. NMT approaches can thus be equivalently viewed as performing neural semantic parsing (NSP), mapping a natural language question into its structured query form.

Ferreira Luz and Finger [17] propose to compute a vector representation for questions and queries that are then fed into an LSTM encoder–decoder model with an attention layer in order to perform the translation/parsing; the out-of-vocabulary problem is explicitly left open. Soru et al. [20] propose "Neural Semantic Machines", which accept as input a set of templates, composed of pairs of questions and queries with entities replaced by placeholders (e.g., "*What was the profession of* `<A>` *'s father?*", with the corresponding query), and a knowledge graph, from which valid entity replacements of placeholders are computed, generating concrete instances (e.g., "*What was the profession of Marie Curie's father?*" with the corresponding query) that are used to train an LSTM encoder-decoder model. While this approach partially addresses the out-of-vocabulary problem, tens of millions of (highly repetitive) instances would need to be used for training to cover all entities in a knowledge graph such as Wikidata. Yin et al. [26] compare a variety of NMT models in the context of KGQA, including RNN-, CNN- and transformer-based models, finding that the best model was the Convolutional Sequence to Sequence (ConvS2S) model. However, Yin et al. [26] acknowledge that the models encounter issues with larger vocabularies.

Tackling the out-of-vocabulary issue, Lukovnikov et al [16] merge word-level and character-level representations for both the questions and the entity labels in the knowledge graph; however, the approach is specifically designed to deal with simple questions (e.g., "*Who is Marie Curie's father*"), and would not work with knowledge graph identifiers not based on a natural language (as is the case for Wikidata). Wang et al. [24] propose gAnswer: a system that extends NMT with entity linking (EL) and relation linking (RL) in order to better handle the out-of-vocabulary problem, finding that entity linking in particular improves performance over NMT. However, only one EL system is explored, and there is no discussion of how entities are matched with their placeholders.

Table 1. Hyperparameter settings used

Model	Layers	H. Units	Attention	Optimiser	L. Rate	Dropout	Size
ConvS2S	15	512–2,048	yes	SGD	0.5	0.2	500
LSTM	4	1,000	yes	Adam	0.001	0.3	500
Transformer	6	1,024	yes	Adam	0.0005	0.3	500

QALD/KGQA Datasets Various datasets have been proposed for QALD/KGQA, including a series of QALD challenges [23], which provide a variety of small sets (in the order of hundreds) of training and test examples over knowledge graphs such as DBpedia and Wikidata. For NMT-based approaches, larger datasets are needed for training. SimpleQuestions [3] is a larger alternative, containing 108,442 question–answer pairs, but focuses on simple factoid questions. The Monument dataset [20] is based on 38 unique query templates about monuments in DBpedia, where 300–600 question–query pairs are created for each template; however, these instances focus on the data for one class in DBpedia. The DBNQA dataset [12] includes 894 thousand question–query pairs produced by taking 215 queries from a QALD training dataset, replacing the entities with placeholders, and generating other valid replacements from the DBpedia knowledge graph; there is thus a high level of repetition as the 894 thousand examples are based on 215 unique queries. Finally, LC-QuAD 2.0 [7] consists of 30 thousand question–query pairs over DBpedia and Wikidata generated from 22 templates, where the questions are paraphrased through crowdsourcing.

3 The Problem with Entities

Our goal is to advance NMT-based approaches for KGQA. We first establish a baseline for a state-of-the-art "pure NMT" approach by selecting three of the best performing models for the KGQA in the comparison of Yin et al. [26]. These three models constitute the state-of-the-art for NMT in the context of natural language, involving three diverse architectures based on Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs) and attention without recurrence or convolutions. More specifically, the three models are as follows. *ConvS2S* (Convolutional Sequence-to-Sequence): A CNN-based architecture, featuring gated linear units, residual connections, and attention. *LSTM* (Long Short Term Memory): An RNN-based architecture that uses stacking LSTM models of four layers, with attention. *Transformer*: a lightweight architecture that interleaves multi-head attention mechanisms with fully-connected layers. We use Fairseq, Google Colab, and the hyperparameters listed in Table 1 (per Yin et al. [26]). For ConvS2S, the first 9 layers had 512 hidden units and kernel width 3; the next 4 layers had 1,024 hidden units and kernel width 3; the final 2 layers had 2,048 hidden units with kernel width 1.

With respect to the datasets, we first aim to replicate the experiments of Yin et al. [26] over the LC-QuAD 2.0 dataset, which was the most complex, diverse

and challenging KGQA dataset they tested. We use the original training/test splits of LC-QuAD 2.0. However, we encountered some quality issues with the LC-QuAD 2.0 dataset, which may relate to its use of crowdsourcing platforms to generate and diversify the question texts. Specifically, participants were asked to rephrase synthetic questions such as "*What is [occupation] of [father] of [Marie Curie]?*" into a more natural form, such as "*What was the occupation of Marie Curie's father?*", or to paraphrase the question, such as "*What did Marie Curie's father do for a living?*". From revising the benchmark, though in many cases the process yielded meaningful question–query pairs, not all cases were like this. In particular, we identified issues including: (1) the question text being null, empty or marked not applicable ("*NA*"); (2) questions containing syntactic features from the synthetic question, or copying the synthetic question in a verbatim manner; (3) the question specifying the answer rather than the question; (4) the length of the question being too long or too short to be a reasonable rephrasing of the synthetic question; (5) the query containing invalid tokens. Given the number of instances in the dataset, we applied automatic heuristics to detect such cases, where we discarded 2,502 (8.2%) of the 30,226 instances due to such quality issues. We call the resulting cleaned dataset LC-QuAD 2.0*.

Given the quality issues with LC-QuAD 2.0 dataset, and the fact that its training and test datasets feature question–query pairs based on common templates, we decided to create a novel, independent test dataset with 100 question–query pairs answerable over Wikidata. This dataset does not follow standard templates, and contains queries with a range of complexities and algebraic features, ranging from simple queries such as "*Who is the president of Poland*", to complex questions such as "*Which country that does not border Germany has German as an official language?*".[1] We call this novel dataset WikidataQA; it contains 132 entities, 86 properties, and 208 triple/path patterns.

In Table 2, we present the results for the three models trained on the LC-QuAD 2.0* dataset, applied to the LC-QuAD 2.0* test set and the full WikidataQA set. These results consider a number of different metrics. First we present the *BLEU score*, which is a modified version of precision used to compare a given translation against a set of reference translations. This measure was reported by Yin et al. [26]. However, a translated query may have an excellent BLEU score but may be invalid or distinct from the reference query. We thus also present *accuracy*, which indicates the percentage of questions for which the exact reference query was generated. This measure offers a lower-bound for performance since two queries that are syntactically different may be semantically equivalent, returning the same results on any knowledge graph. Unfortunately, determining the equivalence of SPARQL queries (or queries for any language that encapsulate first-order logic, including SQL) is undecidable. A complementary measure is thus to compare the answers generated by the reference query and the computed query in terms of *precision* and *recall*; we then finally present the macro precision, recall and F_1-score (averaged over each question, which is given equal

[1] While QALD datasets could have been used, LC-QuAD 2.0 may use templates inspired by QALD, creating a possible leak between training and test datasets.

Table 2. KGQA performance of baseline models in terms of BLEU (B), Accuracy (A), Precision (P), Recall (R), and F_1 score (F_1) for three models and two datasets

Model	**Dataset**	B	A	P	R	F_1
ConvS2S	LC-QuAD 2.0*	**51.5%**	**3.3%**	**16.4%**	**16.6%**	**16.4%**
LSTM	LC-QuAD 2.0*	45.6%	0.2%	12.9%	13.0%	12.9%
Transformer	LC-QuAD 2.0*	48.2%	2.1%	15.0%	15.2%	14.9%
ConvS2S	WikidataQA	**18.8%**	0%	5.5%	6.0%	5.3%
LSTM	WikidataQA	18.3%	0%	**7.9%**	7.8%	**7.9%**
Transformer	WikidataQA	18.3%	0%	7.2%	**8.9%**	7.3%

weight). Again however, this measure is somewhat flawed in that – in particular for `ASK` queries that return a boolean result – a query that is completely unrelated may return a correct result. While no measure perfectly captures KGQA performance, each provides a different insight.

The presented results indicate that while the BLEU score for LC-QuAD 2.0* is mediocre, accuracy is very poor, while precision and recall of answers is somewhat better than accuracy. We highlight that over the LC-QuAD 2.0 dataset, Yin et al. [26] report BLEU scores of 59.5%, 51.1% and 57.4% for ConvS2S, LSTM and Transformer, respectively; although BLEU scores drop over our cleaned dataset for the corresponding models – perhaps due to the removal of trivial cases – the relative ordering of models remains the same. It is important to note that Yin et al. [26] base their F_1 scores on correct terms in the output query, rather than solutions; thus their scores are higher than seen here. We can also compare these results with those reported for gAnswer [24] over the QALD-9 and Monument datasets, where Transformer and ConvS2S outperformed LSTM.

From these baseline results, we can conclude that while NMT models do sometimes find correct answers, overall the performance leaves much room for improvement. Furthermore, the results for WikidataQA are consistently worse than those for LC-QuAD 2.0*, indicating that when presented with questions not following similar patterns seen in training, the models struggle to generalise.

Comparing the models, we see that ConvS2S provides the best results for all metrics in the case of LC-QuAD 2.0*. However, while ConvS2S provides the best BLEU score in the case of WikidataQA, it is outperformed by LSTM and Transformer in terms of precision, recall and F_1.

Although the NMT-based gAnswer system won the recent QALD-9 challenge by a clear margin, these results highlight the difficulty of KGQA for complex questions. Analysing the errors, we found that *in 91.1% of the cases, the set of entities in the output query did not correspond with that of the reference query, thus constituting the primary source of error found.* This is not surprising as the specific entities of a question–query pair may not have been seen during training, causing more frequent out-of-vocabulary errors than in the case of properties, which are fewer in number and easier to cover in the training set. In the following, we propose to combine Entity Linking with NMT to address this issue.

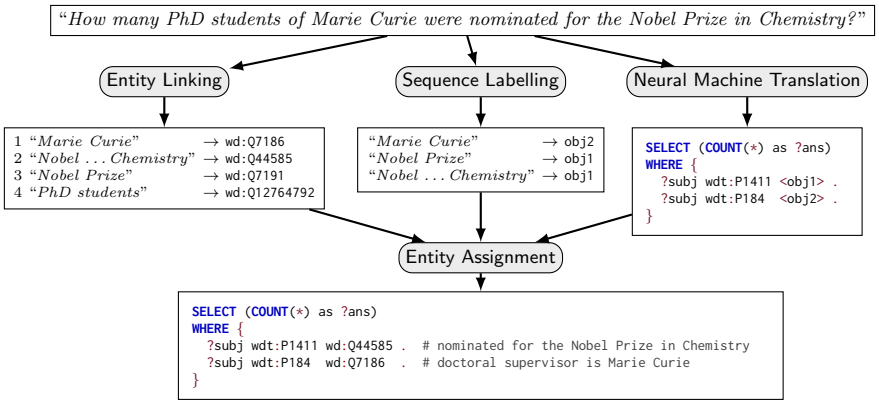

Fig. 1. Overview of ElNeuKGQA approach for an example input question

4 Proposal: ElNeuKGQA

In order to reduce entity-related errors, we propose to combine NMT with Entity Linking (EL). EL identifies mentions of entities in a text and links them to knowledge graph identifiers [25]. Given a text "*What did Marie Curie's father do for a living?*", an EL tool selecting DBpedia or Wikidata as a target knowledge graph would be expected to link "*Marie Curie*" to `db:Marie_Curie` or `wd:Q7186`, respectively. Some EL systems may further target common entities such as "*father*", linking them to `dbr:Father` and `wd:Q7565`, respectively.

Numerous EL techniques have been proposed down through the years, where a common theme is to use the context of the text to disambiguate an entity. For example, in a question "*Who is the lead singer of Boston?*", while the mention "*Boston*" has a higher prior probability of referring to the city, in this case it clearly refers to the rock band Boston, based on contextual clues such as "*singer*". A natural approach is then to delegate the translation of entities in KGQA to external EL systems. This would avoid the need to train the NMT model with examples for every entity, and by taking into account the context, it should improve results versus the approach of applying disambiguation using only prior probabilities, as was proposed for Neural Semantic Machines [20].

The idea of using EL in the context of KGQA is far from new [11,8,4,24], but to the best of our knowledge only Wang et al. [24] have recently looked at combining EL with NMT. Their system, called gAnswer, won the QALD-9 challenge, and is thus clearly a promising approach. However, their approach is based on an individual EL system, and details are missing in terms of how the entity filling task is addressed, i.e., how cases where the EL system generates multiple entities, or where the query involves multiple entities, are handled.

ElNeuKGQA Overview We propose a novel approach, called *ElNeuKGQA*, that combines EL with NMT. Specifically, an EL system is used to identify entity mentions in the question and link them to the knowledge graph. We combine

this with an NMT model that is trained and used to generate template queries with placeholders for entities. The results of the EL system can then be used to fill in these placeholders in the template query. In initial experiments, we noted some key design factors underlying such an approach: (1) the EL system should not produce fewer entity links than there are placeholders in the template query, as this will lead to an invalid query; (2) the EL system may sometimes identify more entity links than placeholders, where the correct entities must be chosen; (3) for questions/queries with multiple entities, we must choose which placeholder to replace with which entity, which is a non-trivial problem.

We thus propose a custom ensemble EL approach, which incorporates multiple individual EL systems in order to boost recall (addressing (1)), and is combined with a voting method to rank individual entities (addressing (2)). Though various ensemble EL systems have already been proposed in the literature, we opted to build a task-specific ensemble system for two main reasons: (i) the nature of the scoring function changes per the aforementioned requirements where, rather than voting for entities to link a given mention in the text as in a typical EL scenario, we rather need to score entities to fill "placeholders" in the query; (ii) existing ensemble systems do not target Wikidata (as targeted by the LQ-QuAD 2.0 dataset), where although Wikidata contains a large intersection of interlinked entities with knowledge bases such as DBpedia, Wikipedia, and YAGO that are targeted by the existing systems, it also contains a much broader set of unique entities, where our ensemble includes an EL system for Wikidata.

We further propose to combine EL with an *entity filling* (EF) technique that chooses which placeholders in the query to replace with which entities (addressing (3)). This entity filling component consists of two phases: (SL) a sequence labeller is trained on the question–query pairs in order to identify the role of entities in the question, matching them with their role in the query; and (EA) an entity assigner uses these roles and specific heuristics to map each query placeholder to a specific entity produced by EL.

We provide an example of our proposed ElNeuKGQA system in Figure 1. Entity linking (EL) provides a ranked list of entities mentioned in the question text. Neural machine translation (NMT) is used to generate a query template with placeholders, such as `<obj1>` and `<obj2>`, to be filled later. Entity filling (EF) requires two steps: sequence labelling (SL) annotates phrases in the question with roles corresponding to placeholder labels, while entity assignment (EA) decides which entity to use to replace which placeholder in the query template.

There exist a number of ways in which this process can give an incorrect query: EL may fail to identify all of the relevant entities; sequence labelling may produce labels that do not match the template; NMT may produce an incorrect template; etc. Later we will look at the most common forms of errors in practice.

In order to train the system, the mentions of entities found in the output query must be annotated in the question with their knowledge graph identifiers, which is the case for LC-QuAD 2.0* and our novel WikidataQA dataset.

We now describe the individual components in more detail.

Neural Machine Translation The NMT component follows the baseline approach, with the sole exception that the model is trained to produce query templates with generic placeholders replacing the specific knowledge graph nodes referring to a particular entity mention (where specific nodes are rather identified by the EL component). In particular, all constant subjects and objects in the query are replaced, in a one-to-one manner, with placeholders P of the form `sbj`k/`obj`k for subject/object IRIs; and `num`k/`str`k for numbers/strings, where k indicates the k^{th} placeholder of that form in the query.Formally, let $\mathbf{Q}$ denote the set of all finite-length sequences of (Unicode) characters of the form $\langle c_1, ..., c_n \rangle$. Let $\mathbf{S}$ denote the set of all finite-length sequences of the form $\langle t_1, ..., t_n \rangle$, where each t_i $(1 \leq i \leq n)$ is a query token that may be a syntactic terminal (e.g., "{"), an operator (e.g., `COUNT`), a query variable, an RDF term, or a placeholder. Then $NMT : \mathbf{Q} \rightarrow \mathbf{S}$ is a (learnt) function that maps an input question $q \in \mathbf{Q}$ to an output sequence of query tokens $NMT(q) \in \mathbf{S}$. Though most elements of $\mathbf{S}$ are not valid queries, the goal is to learn a function NMT that generates a valid query representing the intent of the input question.

Entity Linking Entity linking (EL) identifies entity mentions in text, and links them to entity identifiers in a knowledge base. In our scenario, given an input question $q \in \mathbf{Q}$, and an RDF knowledge graph G referring to its entities with a set of IRIs E, the function $EL(q, G) = M$ provides us with a set of entity mentions M, which are quads of the form (e, k_1, k_2, z), where $k_1 < k_2$ indicate the start and end offsets of a mention of entity e in the text q and z provides a score. ElNeuKGQA is instantiated with an ensemble of four EL systems: AIDA [27] targeting YAGO, DBpedia Spotlight [19] targeting DBpedia, OpenTapioca [6] targeting Wikidata, and TagME [9] targeting Wikipedia. Though AIDA, DBpedia Spotlight and TagME target other knowledge bases, we can use existing mappings to convert DBpedia, Wikipedia and YAGO links to Wikidata links; however, Wikidata has entities not appearing in DBpedia, Wikipedia nor YAGO, where OpenTapioca is important to be able to target these Wikidata-specific entities. We primarily choose these four EL systems as they provide online APIs, where other systems could be added to the ensemble system in future. It is important for later entity assignment that the EL component rank the entity links that are found. Along these lines, we establish a voting system for z where, for a given mention, the output of each EL system gives a vote to the top-scored entity for that mention based on the order (not the score) in which the results are output. In the case of a tie, we boost z for individual EL systems that provide more precise results for the training dataset; if still tied, we use the scores produced by individual systems to boost z and break the tie.

Sequence Labelling We are then left to perform entity filling, which consists of two phases, starting with sequence labelling (SL). The SL phase takes the question q as input, and produces a set of roles $SL(q) = R$, i.e., a set of triples of the form (p, k_1, k_2), where $p \in P$ is a placeholder (indicating a role, e.g., `obj2`), and $k_1 < k_2$ indicate the start and end position for the placeholder tag. We assume that the dataset provides annotations to enable training of SL (as

provided, e.g., by LC-QuAD 2.0). In order to implement SL in ElNeuKGQA, an embedding layer is used, composed of a stacked embedding that includes GloVe embeddings and Flair contextual embeddings. We use the model proposed by Akbik et al. [1]: GloVe and Flair embeddings are concatenated and passed to BiLSTM with 256 layers, with a final conditional random field (CRF) layer.

Entity Assignment The second phase of entity filling, and the last step overall, involves entity assignment (EA). The EA component accepts the output of the previous three components – (s, M, R) where $s = NMT(q), M = EL(q, G), R = SL(q)$ – and produces the final query $EA(s, M, R) \in \mathbf{S}$. Specifically, EA computes an injective mapping $\gamma : P_s \to E_M \cup D$ from the placeholders P_s mentioned in s to the entities E_M mentioned in M and a set of datatype literals D, returning the image of s under γ (i.e., replacing each placeholder p in s with $\gamma(p)$).

In Algorithm 1, function MAPP2E describes how γ is computed. For brevity, we focus on how entities are processed; datatype literals are generated directly from R for sub-strings labelled with `str`k or `num`k. We use domain(γ) to denote the set of placeholders for which γ is defined, range(γ) to denote the set of entities returned for some placeholder by γ, and role(p) to indicate the role of a placeholder p (e.g., role(`obj3`) = `obj`). The algorithm throws an error if there are more unique entity placeholders appearing in s than unique entities appearing in M ($|P_s| > |E_m|$).[2] We apply three phases of assignment. In the first phase, each placeholder is mapped to the highest ranking entity (from EL) whose mention in the question text is labelled with the same placeholder (by SL), if any; if no exactly matching positions in the question text are found, overlapping positions are considered. In the second phase, each unmapped placeholder is mapped to the highest ranking entity whose mention is labelled with a placeholder of the same role (e.g., any `obj*` for `obj2`); if no exactly matching positions are found, overlapping positions are considered. Finally, each unmapped placeholder is paired with an entity based on the order in which they appear in the query template and question text, respectively. Throughout, the algorithm enforces that γ maps each placeholder to a distinct entity. The algorithmic complexity is $O(pe + p \log p + e \log e)$ for $p = |P_s|$ and $e = |E_m|$.

5 Experiments

Our hypothesis is that *combining entity linking (and entity filling) with neural machine translation can improve the quality of results for the KGQA task versus pure neural machine translation approaches.* The following experiments explore this hypothesis by extending the best three NMT models in the recent results by Yin et al. [26] and the three models used by gAnswer [24] – namely ConvS2S, LSTM, and Transformer – with the proposed EL and EF methods. Experiments comparing NMT and non-NMT approaches are not relevant for our hypothesis, where relevant results can rather be found in QALD-9 [24].

[2] This could optionally be relaxed to replace extra placeholders with variables, or to map multiple placeholders to a given entity.

Algorithm 1 Mapping placeholders to entities

function MAPP2E(s, M, R) ▷ i.e., MAPP2E($NMT(q), EL(q, G), SL(q)$)
  $P_s \leftarrow \{p \in P \mid p$ appears in s and role$(p) \in \{\texttt{sbj}, \texttt{obj}\}\}$
  $E_M \leftarrow \{e \mid$ there exists $k_1, k_2, z : (e, k_1, k_2, z) \in M\}$
  $P_R \leftarrow \{p \in P \mid$ there exists $k_1, k_2 : (p, k_1, k_2) \in R\}$
  if $|P_s| > |E_M|$ **then**
    throw error "more placeholders than entities"
  initialise γ ▷ an empty mapping
  $\mathsf{P}_s \leftarrow$ sortRole(P_s) ▷ by sbj1, ..., sbj$a$, obj1, ..., obj$b$
  for $i \in 1 : |\mathsf{P}_s|$ **do** ▷ iterate in order
    ASSIGNBYROLE($e, M, R, \{\mathsf{P}_s[i]\}, \gamma$)
  for $i \in 1 : |\mathsf{P}_s|$ such that $\mathsf{P}_s[i] \notin$ domain(γ) **do** ▷ skip assigned placeholders
    $P' \leftarrow \{p' \in P_s \mid$ role$(\mathsf{P}_s[i]) =$ role(p') and $\mathsf{P}_s[i] \neq p'\}$
    ASSIGNBYROLE(e, M, R, P', γ)
  $\mathsf{P}'_s \leftarrow$ sortPos($P_s \setminus$ domain$(\gamma), s$) ▷ asc. by start position of first appearance in s
  $\mathsf{E}'_M \leftarrow$ sortPos($E_M \setminus$ range$(\gamma), M$) ▷ asc. by start position of first mention in q
  for $i \in 1 : |\mathsf{P}'_s|$ **do**
    $\gamma \leftarrow \gamma \cup \{\mathsf{P}'_s[i] \mapsto \mathsf{E}'_M[i]\}$
  return γ
function ASSIGNBYROLE(e, M, R, P', γ)
  $C \leftarrow \{(e, z) \mid$ there exists $k_1, k_2, p \in P' : (e, k_1, k_2, z) \in M$ and $(p, k_1, k_2) \in R\}$
  $C' \leftarrow \{(e, z) \in C \mid e \notin$ range$(\gamma)\}$ ▷ ensure entities mapped to at most once
  if C' is not empty **then**
    $\gamma \leftarrow \gamma \cup \{p \mapsto \arg\max_e C'\}$ ▷ map p to entity e with highest score z in C'
  else
    $D \leftarrow \{(e, z) \mid \exists k_1, k_2, k_3, k_4, x, p \in P' : (e, k_1, k_2, z) \in M, (p, k_3, k_4) \in R,$
      $k_1 \leq x \leq k_2$ and $k_3 \leq x \leq k_4\}$ ▷ has overlapping text
    $D' \leftarrow \{(e, z) \in D \mid e \notin$ range$(\gamma)\}$
    if D' is not empty **then**
      $\gamma \leftarrow \gamma \cup \{p \mapsto \arg\max_e D'\}$

Setting As per the baselines of Section 3, the NMT and SL models are built with the Fairseq framework. We use the same hyperparameters (Table 1). We will again use the cleaned LC-QuAD 2.0* for training and testing (following the original splits), and the WikidataQA dataset for testing only.

Query Generation and Question Answering Table 3 presents high-level results that compare the baseline results for "pure NMT" (shown previously in Table 2) with the results for our proposed ElNeuKGQA system using the same metrics. We see that ElNeuKGQA consistently – and in most cases considerably – outperforms the baseline approaches: ElNeuKGQA variants achieve equal or better results for *all* models, metrics and datasets compared to their baseline pure NMT counterparts. These results over two KGQA datasets and three sequence-to-sequence models thus *support* our hypothesis that combining entity linking and filling with NMT (the ElNeuKGQA variants) can improve the quality of re-

Table 3. KGQA performance of baseline and ElNeuKGQA

System	**Dataset**	B	A	P	R	F_1
ConvS2S	LC-QuAD 2.0*	51.5%	3.3%	16.4%	16.6%	16.4%
ElNeuKGQA-ConvS2S	LC-QuAD 2.0*	**59.3%**	**14.0%**	**26.9%**	**27.0%**	**26.9%**
LSTM	LC-QuAD 2.0*	45.6%	0.2%	12.9%	13.0%	12.9%
ElNeuKGQA-LSTM	LC-QuAD 2.0*	52.9%	7.9%	20.2%	20.3%	20.2%
Transformer	LC-QuAD 2.0*	48.2%	2.1%	15.0%	15.2%	14.9%
ElNeuKGQA-Transformer	LC-QuAD 2.0*	51.8%	7.1%	19.9%	20.1%	19.8%
ConvS2S	WikidataQA	18.8%	0%	5.5%	6.0%	5.3%
ElNeuKGQA-ConvS2S	WikidataQA	**20.5%**	0%	**12.9%**	**12.9%**	**12.9%**
LSTM	WikidataQA	18.3%	0%	7.9%	7.8%	7.9%
ElNeuKGQA-LSTM	WikidataQA	18.8%	0%	8.9%	8.9%	8.9%
Transformer	WikidataQA	18.3%	0%	7.2%	8.9%	7.3%
ElNeuKGQA-Transformer	WikidataQA	19.8%	0%	11.9%	11.9%	11.9%

sults for the KGQA task (versus pure NMT approaches). As an auxiliary result, in terms of models, we see that ConvS2S clearly outperforms the other two.

Although we see clear *relative* gains over the baseline, even in the case of ElNeuKGQA, the results leave a *lot* of room for improvement. For example, although correct answers are generated for some WikidataQA queries, no query generated corresponded (precisely) to the gold standard query for any model or variant (see A: accuracy). KGQA for complex questions remains a challenging task, and in what follows we characterise the performance, errors and limitations of ElNeuKGQA by each component, in order to better understand the origin of these relative gains, and also ways in which the approach could be improved.

NMT Template Generation We first look at the results for generating query templates using NMT. Given that the query templates are not expected to produce answers to the question, we present only BLEU and accuracy. These results are shown in Table 4. Looking first at the results for LC-QuAD 2.0*, we observe a moderate increase in the BLEU score versus the baseline for generating the full query in Table 2 (51.5% → 65.2%, 45.6% → 58.2%, 48.2% → 56.9%. resp.). We also see a marked improvement in accuracy (3.3% → 34.3%, 0.2% → 19.3%, 2.1% → 16.4%, resp.), indicating that the models can generate query templates much more accurately than full queries mentioning specific entities (further highlighting the out-of-vocabulary issues encountered by pure NMT models when processing entities). On the other hand, the BLEU score for WikidataQA sees only a slight increase (18.8% → 20.1%, 18.3% → 19.0%, 18.3% → 20.2%) and accuracy remains at zero: not a single WikidataQA template is generated in a syntactically identical manner to the gold standard template by any model. This highlights that the models trained on the LC-QuAD 2.0* dataset do not generalise to WikidataQA. However, it is still possible for queries that are not syntactically identical to return (some) correct answers, as seen in Table 3.

Table 4. Performance of template generation

Model	**Dataset**	B	A
ConvS2S	LC-QuAD 2.0*	**65.2%**	**34.3%**
LSTM	LC-QuAD 2.0*	58.2%	19.3%
Transformer	LC-QuAD 2.0*	56.9%	16.4%
ConvS2S	WikidataQA	20.1%	0%
LSTM	WikidataQA	19.0%	0%
Transformer	WikidataQA	**20.2%**	0%

Table 5. Performance of individual and ensemble EL systems for LC-QuAD 2.0*

EL System	Micro			Macro		
	P	R	F_1	P	R	F_1
AIDA	**72.5%**	31.3%	**43.7%**	**38.5%**	30.5%	33.1%
DBpedia Spotlight	20.8%	52.7%	29.9%	23.3%	52.5%	30.8%
OpenTapioca	61.8%	32.0%	42.2%	34.9%	30.2%	31.4%
TagME	25.0%	59.5%	35.2%	29.5%	59.4%	**37.4%**
Ensemble	19.5%	**68.6%**	30.3%	23.9%	**68.4%**	33.6%

EL Systems Next we look at the performance of the four individual EL systems that we use for our ensemble, and of the voting-based ensemble itself. The results for micro and macro precision, recall and F_1-score are presented in Table 5 over the LC-QuAD 2.0* dataset. These results consider the entity mentions in the question text that correspond to entities in the reference query as being "correct"; it may be the case that an EL system detects entities that would otherwise be considered valid, but are not used in the query. Individual systems find different trade-offs between precision and recall, where AIDA offers the highest precision, while TagME offers the highest recall. Our ensemble offers the best recall, but has (almost) the worst precision. This is a result of its design: we return all entities with a vote-based ranking in order to offer subsequent steps as many (ranked) options as possible in order to ensure that all slots are filled. The results of Table 5 do not consider that ensemble results are ranked.

Sequence Labelling We now look at the results for sequence labelling, where Table 6 again presents the micro and macro precision, recall and F_1-scores. Precision and recall tend to be balanced. However, we note that there is a significant drop in performance when considering the WikidataQA dataset; the model is trained on LC-QuAD 2.0*, and only partially generalises to the other dataset.

Entity Filling We specifically analyse the micro and macro precision, recall and F_1-scores for EF over the pairs (p, e) that are generated, where p is a placeholder, and e is an entity or literal. These results are presented in Table 7. We see that the results are the lowest thus far, which is to be expected as EF depends on the EL and SL components and thus accumulates their errors.

Table 6. Performance of sequence labelling

Dataset	Micro			Macro		
	P	R	F_1	P	R	F_1
LC-Quad 2.0*	62.0%	66.9%	64.4%	63.8%	66.9%	64.8%
WikidataQA	22.7%	29.8%	25.8%	30.9%	31.8%	31.0%

Table 7. Performance of entity filling

Dataset	Micro			Macro		
	P	R	F_1	P	R	F_1
LC-Quad 2.0*	47.2%	45.6%	46.4%	49.6%	47.4%	47.4%
WikidataQA	18.0%	17.0%	17.5%	18.3%	21.5%	18.8%

Sources of error We finally summarise and compare the sources of error found for ElNeuKGQA and baseline pure NMT approaches. Since the accuracy for WikidataQA was 0%, in Table 8 we present the error rates for the LC-QuAD 2.0* dataset, where T refers to a correct template, E refers to the query having the correct set of entities, and F refers to correct entity filling. We use ! to denote an error in the corresponding step and omit both T!F and !!F since an error in E implies an error in F. Of note is that all three pure NMT baselines produce a much lower percentage of queries containing correct entities versus ElNeuKGQA; for example, comparing ConvS2S with ElNeuKGQA-ConvS2S, only 10.0% (TEF + TE! + !EF + !E!) of the queries produced by the former contain correct entities, while 58.2% of the latter contain correct entities. This indicates that ElNeuKGQA primarily gains over pure NMT approaches due to its handling of entities. Focusing on ElNeuKGQA, template errors were the most common.

6 Conclusions

Question Answering over Knowledge Graphs (KGQA) has seen significant advances in recent years, but more complex questions remain a major challenge. Although recent approaches using neural machine translation (NMT) yield promising results, NMT suffers from the out-of-vocabulary problem, particularly for entities. We propose ElNeuKGQA: an approach that combines entity linking (EL) with NMT through a novel entity filling approach. Our experiments show that combining EL with NMT outperforms a pure NMT approach for all models and datasets considered. These results – along those of Wang et al. [24] – highlight the potential for NMT and EL to complement each other.

Regarding limitations, the results for more complex questions – particularly those having unseen templates – remain poor. KGQA over such questions is a challenging task, but also an important one. Regarding future work, we identified some quality issues with the LC-QuAD 2.0 dataset, where NMT-based approaches may perform better given larger, higher-quality datasets for train-

Table 8. Sources of error in the LC-QuAD 2.0* dataset

System	TEF	TE!	T!!	!EF	!E!	!!!
ConvS2S	3.3%	1.3%	25.3%	2.8%	2.6%	64.9%
ElNeuKGQA-ConvS2S	14.0%	4.0%	13.5%	15.7%	24.5%	28.5%
LSTM	0.2%	0.1%	20.3%	0.7%	0.6%	78.1%
ElNeuKGQA-LSTM	7.9%	2.5%	8.2%	21.4%	27.3%	33.7%
Transformer	2.1%	1.0%	22.1%	4.0%	4.3%	66.6%
ElNeuKGQA-Transformer	7.1%	1.7%	6.8%	17.6%	31.6%	35.1%

ing; however, creating such datasets appears costly. Another avenue to explore is to extend NMT with relation linking (RL), as proposed by Wang et al. [24]; however, sometimes a query relation is not explicitly mentioned by a query, and while large knowledge graphs typically contain millions of entities, they only contain thousands of relations (or properties), meaning that it would be more feasible to cover all relations in training. It may also be of interest to leverage the knowledge graph in order to avoid generating queries with empty results; however, user questions may sometimes yield empty results, so it would seem best to avoid always "forcing" a query with results.

Supplementary material, including code and queries, are available from the repository `https://github.com/thesemanticwebhero/ElNeuKGQA`.

Acknowledgements Hogan was supported by Fondecyt Grant No. 1221926 and by ANID – Millennium Science Initiative Program – Code ICN17_002.

References

1. Akbik, A., Blythe, D., Vollgraf, R.: Contextual String Embeddings for Sequence Labeling. In: COLING. pp. 1638–1649. ACL (2018)
2. Bernstein, A., Kaufmann, E., Göhring, A., Kiefer, C.: Querying Ontologies: A Controlled English Interface for End-Users. In: The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, Nov. 6-10, 2005, Proc. LNCS, vol. 3729, pp. 112–126. Springer (2005)
3. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale Simple Question Answering with Memory Networks. CoRR **abs/1506.02075** (2015)
4. Chakraborty, N., Lukovnikov, D., Maheshwari, G., Trivedi, P., Lehmann, J., Fischer, A.: Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs. CoRR **abs/1907.09361** (2019)
5. Damljanovic, D., Agatonovic, M., Cunningham, H., Bontcheva, K.: Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues. J. Web Semant. **19**, 1–21 (2013)
6. Delpeuch, A.: OpenTapioca: Lightweight Entity Linking for Wikidata. In: Wikidata Workshop. CEUR, vol. 2773 (2020)
7. Dubey, M., Banerjee, D., Abdelkawi, A., Lehmann, J.: LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia. In: ISWC. LNCS, vol. 11779, pp. 69–78. Springer (2019)

8. Dubey, M., Banerjee, D., Chaudhuri, D., Lehmann, J.: EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs. In: ISWC. LNCS, vol. 11136, pp. 108–126. Springer (2018)
9. Ferragina, P., Scaiella, U.: TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In: CIKM. pp. 1625–1628. ACM (2010)
10. Finegan-Dollak, C., Kummerfeld, J.K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., Radev, D.R.: Improving Text-to-SQL Evaluation Methodology. In: ACL. pp. 351–360. ACL (2018)
11. Freitas, A., Oliveira, J.G., O'Riain, S., Curry, E., da Silva, J.C.P.: Treo: Best-Effort Natural Language Queries over Linked Data. In: NLDB. LNCS, vol. 6716, pp. 286–289. Springer (2011)
12. Hartmann, A.K., Soru, T., Marx, E.: Generating a Large Dataset for Neural Question Answering over the DBpedia Knowledge Base. In: LD Management (2018)
13. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. **54**(4), 71 (2021)
14. Jung, H., Kim, W.: Automated conversion from natural language query to SPARQL query. J. Intell. Inf. Syst. **55**(3), 501–520 (2020)
15. López, V., Fernández, M., Motta, E., Stieler, N.: Poweraqua: Supporting users in querying and exploring the semantic web. Semantic Web **3**(3), 249–265 (2012)
16. Lukovnikov, D., Fischer, A., Lehmann, J., Auer, S.: Neural network-based question answering over knowledge graphs on word and character level. In: WWW. pp. 1211–1220. ACM (2017)
17. Luz, F.F., Finger, M.: Semantic Parsing Natural Language into SPARQL: Improving Target Language Representation with Neural Attention. CoRR **abs/1803.04329** (2018)
18. Malyshev, S., Krötzsch, M., González, L., Gonsior, J., Bielefeldt, A.: Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph. In: ISWC. LNCS, vol. 11137, pp. 376–394. Springer (2018)
19. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia spotlight: shedding light on the web of documents. In: I-SEMANTICS. pp. 1–8. ACM (2011)
20. Soru, T., Marx, E., Moussallem, D., Publio, G., Valdestilhas, A., Esteves, D., Neto, C.B.: SPARQL as a foreign language. In: SEMANTiCS P&D. CEUR, vol. 2044. CEUR-WS.org (2017)
21. Unger, C., Bühmann, L., Lehmann, J., Ngomo, A.N., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: WWW. pp. 639–648 (2012)
22. Unger, C., Freitas, A., Cimiano, P.: An Introduction to Question Answering over Linked Data. In: Reasoning Web. LNCS, vol. 8714, pp. 100–140. Springer (2014)
23. Usbeck, R., Gusmita, R.H., Ngomo, A.N., Saleem, M.: 9th Challenge on Question Answering over Linked Data (QALD-9) (invited paper). In: NLIWoD. CEUR, vol. 2241, pp. 58–64. CEUR-WS.org (2018)
24. Wang, S., Jiao, J., Li, Y., Zhang, X., Feng, Z.: Answering Questions over RDF by Neural Machine Translating. In: ISWC P&D. CEUR, vol. 2721, pp. 189–194. CEUR-WS.org (2020)
25. Wu, G., He, Y., Hu, X.: Entity Linking: An Issue to Extract Corresponding Entity With Knowledge Base. IEEE Access **6**, 6220–6231 (2018)
26. Yin, X., Gromann, D., Rudolph, S.: Neural machine translating from natural language to SPARQL. Future Gener. Comput. Syst. **117**, 510–519 (2021)
27. Yosef, M.A., Hoffart, J., Bordino, I., Spaniol, M., Weikum, G.: AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables. Proc. VLDB Endow. **4**(12), 1450–1453 (2011)