Encoding Knowledge Graph Entity Aliases in Attentive Neural Network for Wikidata Entity Linking

Isaiah Onando Mulang'[1(✉)], Kuldeep Singh[2], Akhilesh Vyas[3], Saeedeh Shekarpour[4], Maria-Esther Vidal[3], Jens Lehmann[5], and Soren Auer[3]

[1] Fraunhofer IAIS, University of Bonn, and Zerotha Research, Bonn, Germany
isaiah.mulang.onando@iais.fraunhofer.de
[2] Cerence GmbH, and Zerotha Research, Aachen, Germany
kuldeep.singh1@cerence.com
[3] TIB, Hannover, Germany
akhilesh.vyas@tib.eu, maria.vidal@tib.eu
[4] University of Dayton, Dayton, USA
saeedeh@knoesis.org
[5] Fraunhofer IAIS, Sankt Augustin, Germany

Abstract. The collaborative knowledge graphs such as Wikidata excessively rely on the crowd to author the information. Since the crowd is not bound to a standard protocol for assigning entity titles, the knowledge graph is populated by non-standard, noisy, long or even sometimes awkward titles. The issue of long, implicit, and nonstandard entity representations is a challenge in Entity Linking (EL) approaches for gaining high precision and recall. Underlying KG in general is the source of target entities for EL approaches, however, it often contains other relevant information, such as aliases of entities (e.g., Obama and Barack Hussein Obama are aliases for the entity Barack Obama). EL models usually ignore such readily available entity attributes. In this paper, we examine the role of knowledge graph context on an attentive neural network approach for entity linking on Wikidata. Our approach contributes by exploiting the sufficient context from a KG as a source of background knowledge, which is then fed into the neural network. This approach demonstrates merit to address challenges associated with entity titles (multi-word, long, implicit, case-sensitive). Our experimental study shows ≈8% improvements over the baseline approach, and significantly outperform an end to end approach for Wikidata entity linking.

Keywords: Knowledge graph context · Wikidata · Entity linking

1 Introduction

Entity linking (EL) over Web of data often referred as Named Entity Disambiguation (NED) or Entity Disambiguation is a long-standing field of research

The original version of this chapter was revised: an inadvertently forgotten co-author was added. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-62005-9_41

Z. Huang et al. (Eds.): WISE 2020, LNCS 12342, pp. 328–342, 2020.
https://doi.org/10.1007/978-3-030-62005-9_24

in various research communities such as information retrieval, natural language processing, semantic web, and databases since early approaches in 2003 [2]. EL generally comprises two subtasks: *entity recognition* that is concerned with the identification of entity surface forms in the text, and *entity disambiguation* that aims at linking the surface forms with structures and semi-structured knowledge bases (e.g. Wikipedia), or structured knowledge graphs (e.g. DBpedia [1], Freebase [4] or Wikidata [24]).

Research Objectives, Approach, and Contribution. Uniqueness of Wikidata is that the contents are collaboratively edited. As at April 2020; Wikidata contains 83,151,903 items and a total of over 1.2B edits since the project launch[1]. User-created entities add additional noise and vandalism in Wikidata [13] since users do not follow a strict naming convention nor a standardised approach; for instance, there are 1788134 labels in which each label matches with at least two different URIs. The previous approaches for EL [18,19] on the textual content consider the well-established knowledge bases such as Wikipedia, Freebase, YAGO [22], and particularly DBpedia. Thereby, Wikidata as the core background KG along with its inherent challenges, has not been studied particularly for the task of EL.

Besides the vandalism and noise in underlying data of Wikidata, collaborative editing of its content adds several aliases of the entities and its description as entity properties (attributes). This enables Wikidata as a rich source of additional information which may be useful for EL challenges. Thus, in this work, we analyse the impact of additional context from Wikidata on Attentive Neural Networks (NN) for solving its entity linking challenges. We develop a novel approach called Arjun, first of its kind to recognise entities from the textual content and link them to equivalences from Wikidata KG. An important strength of Arjun is an ability to link non-Wikipedia entities of Wikidata by exploiting unique characteristic of the Wikidata itself (i.e. availability of entity aliases as explained in Sect. 2). Please note, focus of this paper is not to propose a black-box deep learning approach for entity linking using latest deep learning models such as transformers or graph neural networks. In this paper we hypothesise that even though Wikidata is noisy and challenging, but its special property to provide aliases of entities can help an NN better understand the context of the potential entities. Since the concept of informing a neural network using contextual data from a KG is our proposed-solution in this work, we believe that traditional neural networks make it more transparent to understand the impact of KG context. Hence, our approach contributes to model attentive neural networks respecting the contextual content and trained on a sizable dataset. In particular, Arjun is a pipeline of two attentive neural networks coupled as follows:

1. In the first step, Arjun utilises a deep attentive neural network to identify the surface forms of entities within the text.
2. In the second step, Arjun uses a local KG to expand each surface form from previous step to a list of potential Wikidata entity candidates. Unlike [15],

[1] https://www.wikidata.org/wiki/Wikidata:Statistics.

Arjun does not use a pre-computed entity candidate list and search entity candidates among all the Wikidata entities.
3. Finally, the surface forms, coupled with potential Wikidata candidates, are fed into the second attentive neural network to disambiguate the Wikidata entities further.

Although simple, our approach is empirically powerful and shows ≈8% improvement over the baseline. We also release the source code and all utilised data for reproducibility and reusability on Github[2]. The remainder of the article is structured as follows: Sect. 2 motivates our work by discussing Wikidata specific entity linking challenges. Section 3 discusses related work. This is followed by the formulation of the problem in Sect. 4. Section 5 describes the approach. In Sect. 6 we discuss the experimental setup and the results of the evaluation. We conclude in Sect. 7.

2 Motivating Examples

We motivate our work by highlighting some challenges associated with linking entities in the text to Wikidata. Wikidata is a community effort to collect and provide an open structured encyclopedic data. The total number of entities described in Wikidata is over 54.1 million [24]. Wikidata entities are represented by unique IDs known as `QID` and `QIDs` are associated with entity labels. Figure 1 shows three sentences extracted from the dataset released by ElSahar et al. [9] which aligns 6.2 million Wikipedia sentences to associated Wikidata triples (<subject, predicate, object>).

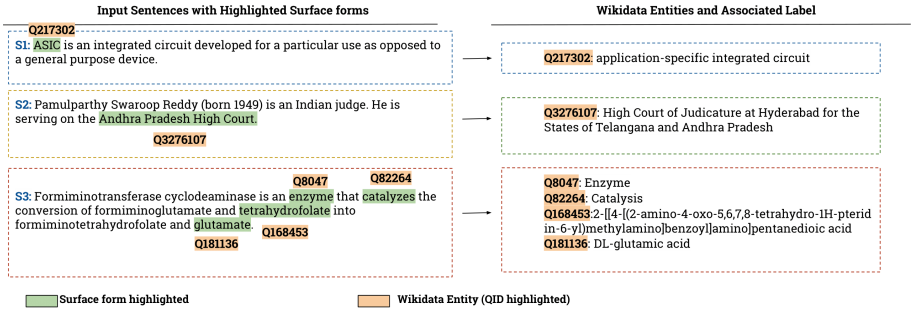

Fig. 1. Wikidata Entity linking Challenges: Besides the challenge of capitalisation of surface forms and implicit nature of entities, Wikidata has several specific challenges, such as very long entity labels and user created entities.

In the first sentence S1, the surface form `ASIC` is linked to a Wikidata entity `wiki:Q217302` and the entity is implicit (i.e. no exact string match between surface form and entity label). However, ASIC is also known as 'Application Specific

[2] https://github.com/mulangonando/Arjun.

Integrated Circuit' or Custom Chip. Therefore to disambiguate this entity, background information about the surface form will be useful. Please note, we will use this sentence as a running example "Sentence S1". In the second sentence S2 the surface form `Andhra Pradesh High Court` is linked to `wiki:Q3276107` which has 14 words in the full entity label[3]. It is also important to note here that the surface form `Andhra Pradesh High Court` also contains two sub-surface forms `Andhra Pradesh` and `High Court` which are the entity labels of the two Wikidata entities `wiki:Q1159` and `wiki:Q671721`. An ideal entity linking tool first has to identify `Andhra Pradesh High Court` as a single surface form then disambiguate the surface form to a long entity label. In Wikidata, entity labels and associated aliases can be long (e.g. `wiki:Q1156234`, `wiki:Q15885502`). In addition there are long erroneous entity labels and aliases, such as entity `wiki:Q44169790`[4] with 62 words in the label and entity `wiki:Q12766033` with 129 words in one alias. Presence of long multi-word entity labels is also specific to Wikidata and poses another challenge for entity linking. Furthermore, in sentence S3 illustrated in the Fig. 1, the surface form `tetrahydrofolate` is linked to `wiki:Q168453` which not only has a multi-word entity label and lowercase surface forms but also contains several numeric and special, non-alphanumeric ASCII characters. Such entities are not present in other public KGs. This is because unlike Wikidata other KGs do not allow users to create new entities and the entity extraction process depends on unique IRIs of Wikipedia pages, WordNet taxonomy, and GeoNames. A large number of user-created entities poses specific challenges for entity linking. Therefore, it is evident that in addition to generic entity linking challenges such as the impact of capitalisation of surface forms and the implicit nature of entities which are tackled up to a certain extent by approaches for entity linking over Wikipedia and DBpedia [19], Wikidata adds some specific challenges to the entity linking problem.

3 Related Work

Several comprehensive surveys exist that detail the techniques employed in entity linking (EL) research; see, for example, [2]. An elaborate discussion on NER has been provided by Yadav Bethard [25]. However, the use of Knowledge Graph as background knowledge for EL task is a relatively recent approach. Here, a knowledge graph is not only used for the reference entities but also offers additional signals to enrich both the recognition and the disambiguation processes. For entity linking, FALCON [19] introduces the concept of using knowledge graph context for improving entity linking performance over DBpedia. Falcon creates a local KG fusing information from DBpedia and Wikidata to support entity and predicate linking of questions. We reused the Falcon Background knowledge base and then expand it with all the entities present in the Wikidata (specially non standard entities).

[3] High Court of Judicature at Hyderabad for the States of Telangana and Andhra Pradesh.

[4] https://www.wikidata.org/wiki/Q44169790.

The developments in deep learning has introduced a range of models that carry out both NER and NED as a single end to end step using various neural network based models [15]. Kolitsas et al. [15] enforces during testing that gold entity is present in the potential list of candidates, however, Arjun doesn't have such assumption and generates entity candidates on the fly. This is one reason Arjun is not compared with Kolitsas's work in the evaluation section. Please note, irrespective of the model opted for entity linking, the existing EL approaches and their implementations are commonly evaluated over standard datasets (e.g. CoNLL (YAGO) [14]). These datasets contain standard formats of the entities commonly derive from Wikipedia URI label. Recently, researchers have explicitly targeted EL over Wikidata by proposing new neural network-based approach [5]. Contrary to our work, authors assume entities are recognised (i.e. step 1 of Arjun is already done), inputs to their model is a "sentence, one wrong Wikidata Qid, one correct Qid" and using an attention-based model they predict correct Qid in the sentence- more of a classification problem. Hence, 91.6F-score in Cetoli et al.'s work [5] is for linking correct QID to Wikidata, given the particular inputs. Their model is not adaptable for an end to end EL due to input restriction. OpenTapioca [6]) is an end to end EL approach to Wikidata that relies on topic similarities and local entity context, but ignores the Wikidata specific challenges (Sect. 2). Works in [10,20] are other attempts for Wikidata entity linking.

4 Problem Statement

Wikidata is an RDF[5] knowledge graph that contains a set of triples $(s, p, o) \in \mathcal{R} \times \mathcal{P} \times (\mathcal{R} \cup \mathcal{L})$, where $\mathcal{R} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{P}$ is the union of all RDF resources. $(\mathcal{C}, \mathcal{P}, \mathcal{E}$ are respectively a set of classes, properties, and entities), and L is the set of literals $(L \cap R = \emptyset)$. An RDF knowledge graph represents a directed graph structure which is formally defined as:

Definition 1 (Knowledge Graph). *A knowledge graph KG is a directed labelled graph $G(V, E)$, where $V = \mathcal{E} \uplus \mathcal{C} \uplus L$ is a disjoint union of entities $\mathcal{E}$, classes $\mathcal{C}$, and literal values L. The set of directed edges is denoted by $E = \mathcal{P}$, where $\mathcal{P}$ are properties connecting vertices. Please note that there is no outgoing edge from literal vertices.*

In this paper, we target **end to end EL task**. The EL for us is defined as recognising the surface forms of entities in the text and then map them to the entities in the background KG. The EL task can be defined as follows:

Definition 2 (Entity Linking). *Assume a given text is represented as a sequence of words $w = (w_1, w_2, ..., w_N)$ and the set of entities of a KG is represented by set $\mathcal{E}$. The EL task maps the text into a subset of entities denoted as $\Theta : w \rightarrow \mathcal{E}'$ where $\mathcal{E}' \subset \mathcal{E}$. Herein, the notion of Wididata entity refers to the representation of an entity based on the corresponding label because Wikidata might consider a variety of identifiers (called Q values) for the same label.*

[5] https://www.w3.org/RDF/.

The EL task can be divided into two individual sub-tasks. The first sub-task Surface Form Extraction is recognising the surface forms of the entities in the text. This task is similar to Named Entities Recognition (NER). However, it disregards identifying the type of entities (e.g. person, place, date, etc.).

Definition 3 (Surface form Extraction). *Let $w = (w_1, w_2, ..., w_N)$ be a text represented as a sequence of words. The surface form extraction is then a function $\theta_1 : w \rightarrow \mathcal{S}$, where the set of surface forms is denoted by $\mathcal{S} = (s_1, s_2, ..., s_K)$ $(K \leq N)$ and each surface form s_x is a sequence of words from start position i to end position j: $s_x^{(i,j)} = (w_i, w_{i+1}, ..., w_j)$.*

The second sub-task Entity Disambiguation (ED) is mapping each surface form into a set of the most probable entities from the background KG.

Definition 4 (Entity Disambiguation). *Let $\mathcal{S}$ be the set of surface forms and $\mathcal{E}$ the set of entities of the background KG. Entity Disambiguation is a function $\theta_2 : \mathcal{S} \rightarrow \mathbb{P}(\mathcal{E})$, which assigns a set of entities to each surface form.*

Please note that a single surface form potentially might be mapped into multiple potentially suitable entities.

5 Arjun: A Context-Aware Entity Linking Approach

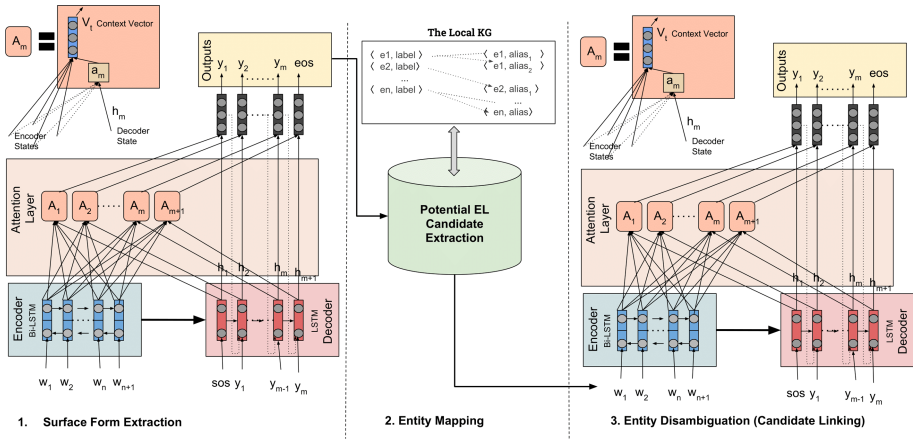

Fig. 2. Proposed Approach Arjun: Arjun consists of three tasks. First task identifies the surface forms using an attentive neural network. Second task induces background knowledge from the Local KG and associate each surface form with potential entity candidates. Third task links the potential entity candidates to the correct entity labels.

Arjun is illustrated in Fig. 2. Arjun performs three sub-tasks:

1. surface form extraction which identifies the surface forms of the entities,
2. entity mapping (or candidate generation) which maps the surface forms to a list of candidate entities from the Local KG,
3. entity disambiguation which selects the most appropriate candidate entity for each surface form.

We devise a context-aware approach based on attentive neural networks for tasks (1) and (3). We initially introduce our derived Local KG. Then we present the details of our approach for the tasks (1), (2) and (3).

Local KG and Refinement Strategies. Arjun relies on Wikidata as the background knowledge graph. Wikidata consists of over 100 million triples in RDF format. Wikidata provides dumps of all the entities and associated aliases[6]. Although Wikidata has specific challenges for EL, its unique characteristic to provide entity aliases can be utilised in developing an approach for entity linking. Since the training dataset is in English, we extracted all 38.6 million Wikidata entities with English labels and 4.8 million associated aliases from the dumps. We use entity labels and aliases as indexed documents in the Local KG and large portion of it is reused from Local KG built by Sakor et al. [19]. For example, the entity described in exemplary "Sentence S1" (cf. Fig. 1), entity `wiki:Q217302` with label *application-specific integrated circuit* is enriched in the Local KG with its aliases: ASIC, Custom Chip, and Custom-Chip.

Model Architecture. For task (1) and (3), our attentive neural model is inspired by the work of Luong et al. [16] and consists of an encoder, a decoder, and an attention layer. We don't claim that an extension of Luong's NN architecture used in this work as novelty, indeed we experiment with already established concepts of LSTM and attentive Neural Networks. We view our attempt of combining these NNs with *background contextual knowledge from a KG* as an interesting perspective for researchers within the community to solve Wikidata KG challenges and is our main novelty. The model in the task (1) is used to identify the surface forms of the entities in the input text. The similar attentive neural model used in the task (3) that selects the most appropriate candidate entity for each surface form (cf. Fig. 2).

We extended Luong's model by using a Bidirectional Long Short-Term Memory (Bi-LSTM) model for the encoder and one-directional LSTM model for the decoder. The input of the encoder is the source text sequence $w = (w_1, w_2, ..., w_n, .., w_N)$ where w_n is the n-th word at time step n and N is the length of the text. The encoder encodes the complete sequence and the decoder unfolds this sequence into a target sequence $y = (y_1, y_2, .., y_m, ..., y_M)$ where y_m is the m-th word at time step m and M is the length of the target sequence. In our assumption each target sequence ends with EOS (end of sequence) token. The N and M values can be considered as the last time steps of the source sequence w and the target sequence y respectively.

[6] https://dumps.wikimedia.org/wikidatawiki/entities/.

Each word of the source sequence is projected to its vector representation acquired from an embedding model $\mathcal{R}^d$ with dimensionality d. The transformation of the input is represented in the matrix X as:

$$X = [x_1, x_2, ., x_n, ..., x_N] \tag{1}$$

where x_n is a vector with the size d and represents the low dimensional embedding of the word w_n.

The LSTM Layer: In our model, the encoder and the decoder consist of single layer of Bi-LSTM and LSTM respectively. Now we explain the LSTM layer.

We model the first layer of our network using a LSTM layer since it has been successfully applied to various NLP tasks. Each LSTM unit contains three gates (i.e., input i, forget f and output o), a hidden state h and a cell memory vector c. The forget gate is a sigmoid layer applied on the previous state h_{t-1} at time step t-1 and the input x_t at time step t to remember or forget its previous state (Eq. 2).

$$f_t = \sigma(W^f[x_t, h_{t-1}] + b^f) \tag{2}$$

Please note that W is the weight matrix and b is the bias vector. The next step determines the update on the cell state. The input gate which is a sigmoid layer updates the internal value (Eq. 3), and the output gates alter the cell states (Eq. 4).

$$i_t = \sigma(W^i[x_t, h_{t-1}] + b^i) \tag{3}$$

$$o_t = \sigma(W^o[x_t, h_{t-1}] + b^o) \tag{4}$$

The next tanh layer computes the vector of a new candidate for the cell state $\tilde{C}_t$ (Eq. 5). Then the old state C_{t-1} is updated by the new cell state C_t via multiplying the old state with the forget gate and adding the candidate state to the input gate (Eq. 6). The final output is a filtering on the input parts (Eq. 4) and the cell state (Eq. 7).

$$\tilde{C}_t = \tanh(W^C[x_t, h_{t-1}] + b^C) \tag{5}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{6}$$

$$h_t = o_t \odot \tanh(C_t) \tag{7}$$

where the model learning parameters are weight matrices W^f, W^i, W^o, W^C and bias vectors b^f, b^i, b^o, b^C. The σ denotes the element-wise application of the sigmoid function and $\odot$ denotes the element-wise multiplication of two vectors.

The Bi-LSTM of the encoder consists of two LSTM layers. The first layer takes an input sequence in forward direction (1 to N) and the second layer takes the input in backward direction (N to 1). We employ same Eqs. 2, 3, 4, 5, 6, 7 for each LSTM Layer. The final encoder hidden state is produced by the sum of hidden states from both LSTM layers ($h_n = \overrightarrow{h_n} + \overleftarrow{h_n}$) at timestep n.

Attention and Decoder Layer. The decoder layer takes SOS token (start of the sequence) vector and the encoder final states (h_N and C_N) as the initial

inputs to start decoding source text sequence into a target text sequence. Here we differentiate between the encoder hidden state and decoder hidden state by using the notations h_n at time step n and h_m time step m respectively. Below we explain how the decoder generates target text sequence words y_m one by one.

In the attention layer, we define attention weights as $a_m = [a_{m1}, a_{m2}, ..., a_{mN}]$ for a decoder state at time step m which has the size equals to the number of total time steps in the encoder side. The attention weights contain only scalar values which are calculated by comparing all encoder states h_n and decoder state h_m. To calculate the attention weight (a_{mn}) of an encoder state at time step n wrt. A decoder state at time step m, we use the following Eq. (8) [16].

$$a_{mn} = \frac{\exp(h_m \cdot h_n)}{\sum_{n'=1}^{N} \exp(h_m \cdot h_{n'})} \tag{8}$$

Where $(h_m \cdot h_n)$ denotes the dot product. The Eq. 9 computes the context vector V_m as weighted average over all the encoder hidden states (h_n) that captures the relevant encoder side information to help in predicting the current target sequence word y_m at time step m and can be defined as:

$$V_m = \sum_{n=1}^{N} a_{mn} h_n \tag{9}$$

We calculate Attention Vector ($\tilde{h}_m$) using the concatenation layer on the context vector V_m and decoder hidden state h_m for combining information from both the vectors. The Eq. 10 represent it mathematically (where tanh is an activation function same as describe in [16]).

$$\tilde{h}_m = \tanh\left(W_v[v_m; h_m]\right) \tag{10}$$

Finally, we apply softmax layer on the attention vector $\tilde{h}_m$ for predicting a word of a target text sequence from the predefined vocabulary of the complete target text sequences.

$$p(y_m|y_{<m}, x) = \text{softmax}(W_s \tilde{h}_m) \tag{11}$$

Where W_s is weight matrix of softmax layer and p is probability. Please note that the decoder stops producing words once it encounters EOS (end of sequence) token or m is equal to M.

5.1 Entity Mapping Process

The local KG acts as a source of background knowledge. It is an indexed graph (created using the same methodology proposed by Sakor et al. [19] and reusing a large portion of the indexed graph built by the authors) where each entity label is extended with its aliases from Wikidata. Once Task 1 identifies surface forms

in the input sentence, the entity mapping step (Task 2) takes each surface form and retrieves all the entities for which entity label(s) in the local KG matches with the surface form. Next, the full list of the entity candidates is then passed into the Step 3 of Arjun as input to predict (disambiguate) the best Wikidata entity labels.

Let us trace our approach for the sentence S1 of Fig. 1 to understand the steps better. The sentence S1 `ASIC is an integrated circuit developed for particular use as opposed to a general-purpose device` is fed to the attentive neural model comprises of an encoder (Bi-LSTM), decoder (LSTM), and an attention layer as an input for the surface form extraction task. Thereby, the term `ASIC` is recognised as a surface form. Then, for the entity mapping task, we populate a Local KG to generate candidate entities associated with this surface form. We employ semantic search (reused from Falcon [19]) to identify entity candidate labels for `ASIC` which returns `Application Specific Integrated Circuit`. The last step of Arjun is entity disambiguation. In this step, the surface form `ASIC` along with `Application Specific Integrated Circuit` is fed into the encoder as the input sequence. Here, we utilise identical attentive neural network used for surface form extraction task. This attentive neural network decides the context of `ASIC` using extra information in the form of associated alias to correctly link to the Wikidata entity `application-specific integrated circuit` (Q217302).

6 Experimental Setup

6.1 Dataset

We rely on the recently released T-REx [9] dataset that contains 4.65 million Wikipedia extracts (documents) with 6.2 million sentences. These sentences are annotated by 11 million Wikidata triples. In total, over 4.6 million surface forms are linked in the text to 938,642 unique entities. T-REx is the only available dataset for Wikidata with such a large number of triple alignment. We are not aware of any other dataset explicitly released for Wikidata entity linking challenges. Please note that the popular entity linking datasets (e.g. CoNLL (YAGO) [14]) have linked entities either to Wikipedia, YAGO, Freebase or DBpedia. Work in [5,6] attempt to develop approaches for EL over Wikidata and simply align (map using owl:sameAs) existing Wikipedia based dataset to Wikidata. However, our focus in this paper is to solve Wikidata specific challenges and these datasets do no embrace Wikidata specific challenges for entity linking. We divide the T-REx dataset into an 80:20 ration for training and testing.

6.2 Baseline

In this work, we pursue the following research question: "How well does the attentive neural network perform for entity linking task leveraging background knowledge particularly for a challenging KG such as Wikidata?" To the best

of our knowledge, it is a pioneering work for the task of entity linking on the Wikidata knowledge graph where it considers the inherent challenges (noisy nature, long entity labels, implicit entities). Therefore, we do not compare our approach to generic entity linking approaches which typically either do not use any background knowledge or employ the well-established knowledge graphs such as DBpedia, YAGO, Freebase. Our approach Arjun comprises all three tasks illustrated in Fig. 2. To elaborate the advantage of inducing additional context post NER step, we built a "baseline" which is an end to end neural model. The "baseline" in our case is the attentive neural network employed in Task 1 without any background knowledge (or can be seen as end to end EL using attentive neural network). In fact, in the task (1) (cf. Fig. 2), the baseline directly maps the text to a sequence of Wikidata entities without identifying surface form candidates. Hence, the baseline approach is the modified version of Arjun. With a given input sentence, the baseline implicitly identifies the surface forms and directly links them to Wikidata entities. Unlike Arjun, the baseline does not use any KG context for the expansion of the surface forms. We also compare Arjun with recently released SOTA for Wikidata entity linking- OpenTapioca [6] which is an end to end EL approach. We are not aware of any other end to end EL tool/approach released for Wikidata.

6.3 Training Details

Implementation Details. We implemented all the models using the PyTorch framework. The local KG and the semantic search is implemented using Apache Lucene Core[7] and Elastic search [12]. The semantic search returns entity candidates with a score (higher is better). We reuse the implementation of Falcon local KG [19] for the same. After empirically observing the performance, we set the threshold score to 0.85 for selecting the potential entity candidates per surface form (i.e. the parameter is optimised on the test set). We reused pre-trained word embeddings from Glove [17] for the attention based neural network. These embeddings have been pre-trained on Wikipedia 2014 and Gigaword 5[8]. We employ 300-dimensional Glove word vectors for the training and testing of Arjun. The models are trained and tested on two Nvidia GeForce GTX1080 Ti GPUs with 11 GB size. Due to brevity, detailed description of training details can be found in our public Github.

Dataset Preparation. We experimented initially with higher text sequence lengths but resorted to 25 words due to GPU memory limitation. In total, we processed 983,257 sentences containing 3,133,778 instances of surface forms (not necessarily unique entities) which are linked to 85,628 individual Wikidata entities. From these 3,133,778 surface forms occurrences, approximately 62% do not have exact match with a Wikidata entity label.

[7] https://lucene.apache.org/core/.
[8] https://nlp.stanford.edu/projects/glove/.

6.4 Results

Table 1 summarises performance of Arjun compared to the baseline model and another NED approach. We observe nearly 8% improvement in the performance over baseline and Arjun significantly outperforms another end to end EL tool OpentTapioca. Arjun and OpenTapioca generate entity candidates on the fly, i.e., out of Millions of Wikidata entities, the task here is to reach to top-1 entity. This contrasts with other end to end entity linking approaches such as [15], which rely on a pre-computed list of 30 entity candidates per surface form. This translates into extra complexity due to the a large search space for generating entity candidates in the case of Arjun. Our solution demonstrates a clear advantage of using KGs as background knowledge in conjunction with a attention neural network model. We now detail some success and failure cases of Arjun.

Table 1. Performance of Arjun compared to the Baseline.

Method	Precision	Recall	F-Score
baseline	0.664	0.662	0.663
OpenTapioca [6]	0.407	0.829	0.579
Arjun	0.714	0.712	0.713

Success Cases of Arjun. Arjun achieves 0.77F-Score for the surface form extraction task. Arjun identifies the correct surface form for our exemplary sentence S1 (i.e. `ASIC`) and links it to the entity label *Application Specific Integrated Circuit* of `wiki:Q217302`. The baseline can not achieve the linking for this sentence. In the Local KG, the entity label of `wiki:Q217302` is enriched with aliases that also contain `ASIC`. This allows Arjun to provide the correct linking to the Wikidata entity containing the long label. Background knowledge induced in the attentive neural network also allows us to link several long entities correctly. For example, in the sentence "The treaty of London or London convention or similar may refer to," the gold standard links the surface form `London convention` with the label *Convention on the Prevention of Marine Pollution by Dumping of Wastes and Other Matter* (c.f. `wiki:Q1156234`). The entity label has 14 words, and Arjun provides correct linking. OpenTapioca on the other hand have high recall(it has high number of False Positives), however, the precision is relatively quite low. The limited performance of OpenTapioca was due to the fact that it finds limitation in linking non Wikipedia entities which constitute a major portion in the dataset. This demonstrate strength of Arjun in also linking non standard, noisy entities which are not part of Wikipedia.

Failure Cases of Arjun. In spite of the successful empirical demonstration of Arjun, we have a few types of failure cases. For example in the sentence: 'Two vessels have borne the name HMS Heureux, both of them captured from the French' has two gold standard entities (`Heureux` to *French ship Heureux*

(wiki:Q3134963) and French to *French* (wiki:Q150)). Arjun links Heureux to *L'Heureux* (wiki:Q56539239). This issue is caused by the semantic search over the Local KG while searching for the potential candidates per surface form. In this case, L'Heureux is also returned as one of the potential entity candidates for the surface form Heureux. A similar problem has been observed in correctly mapping the surface form Catalan to wiki:Q7026 (*Catalan Language*) where Arjun links Catalan to Catalan (wiki:Q595266). Another form of failure case is when Arjun identifies and links other entities which are not part of the gold standard. The sentence 'Tom Tailor is a German vertically integrated lifestyle clothing company headquartered in Hamburg' has two gold standard entity mappings: vertically integrated to *vertical integration* (wiki:Q1571520 and Hamburg to *Hamburg* (wiki:Q1055). Arjun identifies *Tom* (wiki:Q3354498) and *Tailor* (wiki:Q37457972) as the extra entities and can not link vertically integrated. For brevity, a detailed analysis of the failure cases per entity type (very long label, noisy nonstandard entity), performance loss due to semantic search can be found in our Github.

Limitations and Improvements for Arjun. Arjun is the first step towards improving a deep learning model with additional contextual knowledge for EL task. Arjun can be enhanced in various directions considering current limitations. We list some of the immediate future extensions:

1. *Enhancing Neural Network with Multiple layers:* Arjun currently has a Bi-LSTM and a single layer LSTM for the encoder and the decoder respectively. It has been empirically observed in sequence to sequence models for machine translations that the models show significant improvements if stacked with multiple layers [21]. Therefore, with more computing resources, the neural network model used in Arjun can be enhanced with multiple layers.
2. *Alternative Models:* In this article, our focus is to empirically demonstrate how background knowledge can be used to improve an attentive neural network for entity linking. Several recent approaches [7,8,23] enhance the performance NER and can be used in our models for task (1) and task (3).
3. *Improving NER:* there is a room of improvement regarding surface form extraction where Arjun currently achieves an F-score of 0.77. The latest context-aware word embeddings [3] can be re-used in Arjun or completely replacing NER part with latest language models such as BERT [7].
4. *Replacing Semantic Search:* Another possibility of improvement is in the second step of our approach (i.e., inducing background knowledge). Currently, we rely on very trivial semantic search (same as [19]) over the Local KG to extract Wikidata entity candidates per surface form. Ganea et al. [11] developed a novel method to embed entities and words in a common vector space to provide a context in an attention neural network model for entity linking. This approach could potentially replace semantic search. Classification is seen as one of the most reasonable and preferred ways to prevent out of scope entity labels [15]. On the contrary, Sakor et al. [19] illustrated that expanding the surface forms the way we did, works pretty well for short text.

Our hypothesis was that it should also work for Arjun, which is not completely true if we see our empirical results. Hence, in this paper, we do not claim that every step we took was the best, but after our empirical study, we demonstrate that the candidate expansion by Sakor et al. doesn't work well. However, it solves our purpose of inducing context in the NN which is the main focus of the paper. It leads to an interesting discussion: what is the most efficient way to induce KG context in a NN, maybe the classification one?- one need to empirically prove and we leave it for future work.

5. *Coverage restricted to Wikidata:* Effort can be made in the direction to develop common EL approach targeting multiple knowledge graphs with standard and nonstandard entity formats.

7 Conclusion

In this work, we focused on introducing the limitations of EL on Wikidata in general, presented the novel approach Arjun, and outlined deficiencies of Arjun, which in particular will guide future work on this topic. In this work, we empirically illustrate that for a challenging KG like Wikidata, if a model is fused with additional context post-NER step, it improves entity linking performance. However, this work was our first attempt towards a longer research agenda. We plan to extend our contribution particularly in the following directions (i) extending towards joint entity and predicate linking and use latest language models for NER task, (ii) enriching the background KG to several interlinked KG from Linked Open Data (DBpedia, Freebase, YAGO), (iii) extending Arjun for the learning entities across languages (currently limited to English).

Acknowledgments. This work is Co-funded by the European Union's Horizon 2020 research and innovation programme under the QualiChain Project, Grant Agreement No. 822404; and the IASIS project, Grant Agreement No. 727658.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC - 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
2. Balog, K.: Entity linking. Entity-Oriented Search. TIRS, vol. 39, pp. 147–188. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93935-3_5
3. Blythe, D., Akbik, A., Vollgraf, R.: Syntax-aware language modeling with recurrent neural networks (2018)
4. Bollacker, Kurt D., Cook, R.P., Tufts, P.: A shared database of structured general human knowledge, Freebase (2007)
5. Cetoli, A., Bragaglia, S., O'Harney, A.D., Sloan, M., Akbari, M.: A neural approach to entity linking on Wikidata. In: Azzopardi, L., Stein, B., Fuhr, N., Mayr, P., Hauff, C., Hiemstra, D. (eds.) ECIR 2019. LNCS, vol. 11438, pp. 78–86. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-15719-7_10

6. Delpeuch, A.: Opentapioca: lightweight entity linking for Wikidata. arXiv preprint arXiv:1904.09131 (2019)
7. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT, pp. 4171–4186 (2019)
8. Edunov, S., Ott, M., Auli, M., Grangier, D.: Understanding back-translation at scale. In: EMNLP, pp. 489–500 (2018)
9. ElSahar, H., et al.: T-rex: a large scale alignment of natural language with knowledge base triples. In: LREC (2018)
10. Mulang, I.O., et al.: Evaluating the impact of knowledge graph context on entity disambiguation models. In: CIKM 2020 (to appear, 2020)
11. Ganea, O.-E., Hofmann, T.: Deep joint entity disambiguation with local neural attention. In: EMNLP, pp. 2619–2629 (2017)
12. Gormley, C., Tong, Z.: Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine. O'Reilly Media, Inc. (2015)
13. Heindorf, S., Potthast, M., Stein, B., Engels, G.: Vandalism detection in Wikidata. In: CIKM (2016)
14. Hoffart, J., et al.: Robust disambiguation of named entities in text. In: EMNLP (2011)
15. Kolitsas, N., Ganea, O.-E., Hofmann, T.: End-to-end neural entity linking. In: CoNLL, pp. 519–529 (2018)
16. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: EMNLP, pp. 1412–1421 (2015)
17. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: EMNLP, pp. 1532–1543 (2014)
18. Raiman, J., Raiman, O.: Deeptype: multilingual entity linking by neural type system evolution. In: AAAI 2018 (2018)
19. Sakor, A., et al.: Old is gold: linguistic driven approach for entity and relation linking of short text. In: NAACL HLT, pp. 2336–2346 (2019)
20. Sakor, A., Singh, K., Patel, A., Vidal, M.-E.: Falcon 2.0: an entity and relation linking framework over Wikidata. arXiv preprint arXiv:1912.11270 (2019)
21. Shu, R., Miura, A.: Residual stacking of RNNs for neural machine translation. In: WAT@COLING, pp. 223–229 (2016)
22. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW, pp. 697–706 (2007)
23. Vaswani, A., et al.: Attention is all you need. In: NeurIPS 2017 (2017)
24. Vrandecic, D.: Wikidata: a new platform for collaborative data collection. In: WWW Companion, pp. 1063–1064 (2012)
25. Yadav, V., Bethard, S.: A survey on recent advances in named entity recognition from deep learning models. In: COLING (2018)