Question Entity and Relation Linking to Knowledge Bases via CLOCQ

Philipp Christmann, Rishiraj Saha Roy and Gerhard Weikum

Max Planck Institute for Informatics and Saarland University, Germany

Abstract

Curated knowledge bases (KBs) contain billions of facts with millions of entities and thousands of predicates. Question answering (QA) systems are supposed to access this knowledge to answer users' factoid questions. Entity linking and relation linking are integral ingredients of many such QA systems, and aim to link mentions in the question to concepts in the KB. The quality of these linking modules is of high importance: a single error in linking can result in a failure for the whole QA system. The SMART 2022 Task poses challenges for entity and relation linking to evaluate the performance of different approaches. In this work, we adapt and extend our prior work CLOCQ. CLOCQ computes top-k linkings for each mention to make up for potential errors, with k set automatically based on an ambiguity score. As an extension, we design a module that prunes linkings for irrelevant mentions which helps to improve precision. We found that there is a trade-off between recall and precision: higher k boosts recall (up to 0.87 for entity linking), while lower k leads to high precision performance. The best choice for the linking modules may highly depend on the specific QA system, and whether it can make use of higher recall in the presence of noise.

Keywords

Question Answering, Knowledge Bases, Entity Linking, Relation Linking

CEUR-WS.org/Vol-3337/smart-paper1.pdf

1. Introduction

Question answering (QA) *systems* provide natural interfaces for accessing human knowledge. Such human knowledge can be stored in large-scale *knowledge bases* (KBs) like Wikidata [1], DBpedia [2], YAGO [3], Freebase [4], and industrial counterparts (at Amazon, Apple, Google, Microsoft, etc.). KBs contain facts consisting of *entities*, *relations*, *types*, and *literals*. The standard way of storing *KB facts* are triples consisting of a subject, a predicate and an object.

Motivation and problem. QA systems operating over KBs mostly follow one of the following two themes [5]: (i) approaches **with an explicit query** create a logical form, for example, a SPARQL query, and fill the query slots with entities and relations linked with *mentions* in the question [6, 7], or (ii) approaches **without an explicit query** first link entities and relations to retrieve a search space consisting of KB facts, which is then searched for identifying the answer [8, 9]. For either of the two approaches, being able to identify mentions of entities and relations, and linking these mentions to *KB items*, is a key obstacle in the QA pipeline (linking may also be referred to as *disambiguating*). Even single errors in these *entity linking* or *relation*

SMART Task 2022, October 27, 2022, Hangzhou, China

pchristm@mpi-inf.mpg.de (P. Christmann); rsaharo@mpi-inf.mpg.de (R. Saha Roy); weikum@mpi-inf.mpg.de (G. Weikum)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

linking modules can lead to a complete failure of the QA pipeline, which is why their quality is integral to the performance of the entire QA system. Note that without loss of generality, mentions of types or general concepts may be linked as well, and can be used in the remainder of the QA pipeline.

Consider the following running example on the TV series House of the Dragon following the narratives of George R.R. Martin:

"Who plays Viserys in GRRM's latest HBO series?"

Linking the mentions in the question to the KB (Wikidata for this example) is a non-trivial task that requires an understanding of the question as a whole. The entity mention *"HBO"* may refer to the `HBO company`, the `HBO network`, or to the `Hollywood Bowl Orchestra`. Understanding that the question is on a TV series helps to identify `HBO network` as the correct entity. Similarly, *"plays"* semantically or lexically matches with many relations in the KB, like `plays for team`, `instrument`, `number of plays`, `character role`, or `time played`. The intended relation `character role` only becomes clear from the question context.

"Viserys" is even harder to link, since there are different characters named Viserys in the Game of Thrones universe: `Viserys III Targaryen`, the more well-known character from Game of Thrones, and `Viserys I Targaryen` from the more recent House of the Dragon series. Thus, the mention *"Viserys"* is quite *ambiguous*, even if the general context of the question is clear. A deep understanding of the question is required to correctly link *"Viserys"* to `Viserys I Targaryen`. Note that in case any of these disambiguations are incorrect, there is little hope to return the correct answer `Paddy Considine` to the user.

To further investigate QA modules and pinpoint failure cases, the SMART Task 3.0[1] (co-located with ISWC 2022) poses tasks for entity linking (Task 3) and relation linking (Task 2). There is also a task on answer type prediction (Task 1), which is not targeted in this work.

Approach and contribution. In this work we adapt our recently proposed CLOCQ framework [8] to these tasks. CLOCQ is an unsupervised framework that provides many functionalities related to QA, and is made available as open-source code and as a public API[2]. These functionalities include basic KB methods like retrieving the aliases, frequency, or 1-hop neighborhood of a KB item, or computing the KB connectivity or the shortest path between two KB items. The core algorithm presented in the paper aims to retrieve a search space for a user question, faciliating QA methods *without an explicit query*. As an intermediate step and result, mentions in the question are linked to KB items when retrieving the search space. For linking to KB items CLOCQ implements two key ideas. First, all mentions should be linked jointly, considering the coherence of the disambiguated KB items. This follows the intuition that the question needs to be considered as a whole. CLOCQ links not only entities and relations, but also types and general concepts, providing disambiguations for each mention in the question. Second, the linking modules should make up for potential errors. When disambiguating highly ambiguous mentions, like *"Viserys"* in the running example, the linking modules should take this ambiguity into account and provide the QA system with several possible linkings. CLOCQ provides a mechanism to detect the ambiguity of a mention, based on an entropy measure of

[1] https://smart-task.github.io/2022/
[2] https://clocq.mpi-inf.mpg.de

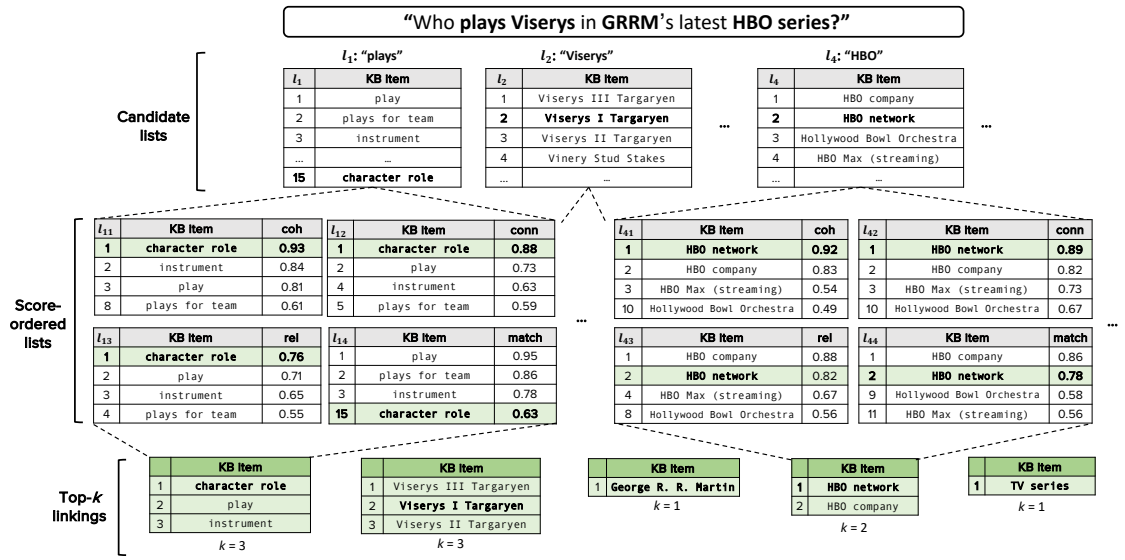

Figure 1: Overview of the CLOCQ linking process. k is chosen automatically for each individual mention in the question.

the candidate distribution. The method can then return top-k linkings per mention, where k can be automatically set based on the ambiguity measure.

In this work, we investigate how CLOCQ can be adapted and extended for QA methods *with an explicit query*, which leverage entity and relation linkings for filling slots in the query. One obstacle with adapting CLOCQ on entity or relation linking tasks, is that it, by design, *disambiguates all mentions* in the question. It does not differentiate between entities, relations, types or other concepts. This helps when retrieving a search space, but can hurt precision of linking results. For example, CLOCQ might link *"latest"* and *"series"* to the KB, even if these mentions are irrelevant. We therefore propose a simple *pruning module*, that identifies which mentions should be linked, and prunes linkings for other mentions. The module is implemented with a fine-tuned sequence generation model that is trained using distant supervision.

By evaluating CLOCQ on the entity and relation linking tasks of SMART 3.0 challenge, we essentially investigate its applicability to QA approaches generating *an explicit query*. We show that top-k disambiguations can help boosting recall, at the cost of decreasing scores for precision and F1 score. Further, we find that the mention-pruning module helps to improve the precision and F1 score substantially on the entity linking task.

2. The CLOCQ linking process

We first introduce the complete workflow of the CLOCQ algorithm. For further discussion and details (e.g. on the fact-centric KB index underlying the CLOCQ framework), please refer to the original paper [8]. Fig. 1 shows an overview of the linking process.

2.1. Retrieving disambiguation candidates

Consider our running example *"Who plays Viserys in GRRM's latest HBO series?"*. Our goal is to link mentions in the question (*"plays"*, *"Viserys"*, *"GRRM"*, *"HBO"*, *"series"*) to items in the KB. Mentions in the question can be single question words or phrases. Named entity phrases can for example be detected using named entity recognition (NER).

We first collect candidates from the KB using a standard lexical matching score (like TF-IDF or BM25) for each mention $q_1 \dots q_m$. m would be 5 in our example, and stopwords are dropped. Here q_i is analogous to a search query, while each item x in the KB resembles a document in a corpus. This "document" is created by concatenating the item label with textual aliases and descriptions available in most KBs [1, 4]. This results in m ranked lists $\{l_1 = \langle x_{11}, x_{12}, \dots\rangle; l_2 = \langle x_{21}, x_{22}, \dots\rangle; \dots l_m = \langle x_{m1}, x_{m2}, \dots\rangle\}$ of KB items x_{ij}, one list l_i for each q_i, scored by degree of match between the mentions and KB items.

A ranked lexical match list for *"plays"* could look like:

$l_1 = \langle$`1: play, 2: plays for team, 3: instrument, 4: number of plays, 5: time played, 6: playwright, 7: guitarist, 8: Plays collection,...,` **`15: character role`**`,...`$\rangle$

with the ideal disambiguation being shown in **bold**. The list for *"HBO"* could be:

$l_4 = \langle$`1: HBO company,` **`2: HBO network`**`, 3: Hollywood Bowl Orchestra,...`$\rangle$

Note that the correct KB item for q_i can sometimes be *very deep in individual lists* l_i. For example, `character role` is at rank 15 in l_1.

Next, each list l_i is traversed up to a depth d to fetch the top-d items per mention. The goal is to find *combinations* of KB items $\langle x_i \rangle_{i=1}^{m}$ that best match the question. For instance, an ideal combination for us would be:

`{character role, Viserys I Targaryen, George R.R. Martin, HBO network, TV series}`

These combinations come from the Cartesian product of items in the m lists, and would have d^m possibilities if each combination is explicitly enumerated and scored. This is cost-prohibitive: since we are only interested in some top-k combinations, as opposed to a full or even extended partial ordering, a more efficient way of doing this would be to apply top-k algorithms [10, 11]. These prevent complete scans and return the top-k best combinations efficiently.

2.2. Scoring candidates

Thus, we propose an approach using top-k algorithms to overcome this challenge. To go beyond shallow lexical matching, our proposal is to construct *multiple lists per question token, each reflecting a different relevance signal.* Specifically, we obtain one list l_{is} for each mention q_i and score s. Then, we apply top-k algorithms *on these lists* to obtain the disambiguation of each question token individually.

Note that considering the question as a whole is a key criterion for our scoring mechanism. Therefore, we integrate two global relevance signals. Specifically, a candidate KB item combination that fits well with the intent in the question is expected to have high semantic coherence and high graph connectivity within its constituents. These can be viewed as proximity in latent and symbolic spaces. Further, candidates should match well on question and mention levels.

These motivate our four relevance signals for each item x_{ij} in list l_i below.

Coherence. We consider global signals for semantic coherence and graph connectivity, which are inherently defined for KB item pairs, on a global level, instead of single items. Therefore, we need a technique to convert these signals into item-level scores. The idea is to use a max operator over all candidate KB item pairs involving a candidate at hand. More precisely, the coherence score of an item x_{ij} is defined as the maximum item-item similarity (averaged over pairs of lists) this item can contribute to the combination. The pairwise similarity is obtained by the cosine value between the embedding vectors of the two KB items, min-max normalized from $[-1, +1]$ to $[0, 1]$:

$$coh(x_{ij}) = \frac{1}{m-1} \sum_{k \neq i} \max_{l} cos(\vec{x_{ij}}, \vec{x_{kl}}) \tag{1}$$

Connectivity. This is the second global signal considered, and captures a different form of proximity. Every KB can be viewed as an equivalent knowledge graph (KG), where entities, predicates and other KB items are nodes, and edges run between components of the same fact [12, 13]. We define KB items that are *part of the same fact* to be in the 1-hop neighborhood of each other, those that are connected *via members of another fact* as in the 2-hop, and so on [8]. We assign items in one hop of each other to have a distance of 1, those in two hops to have a distance of 2, and ∞ otherwise. Almost all KB items are within three or four hops of each other, and thus distances beyond two hops cease to be a discriminating factor. We define connectivity scores as the inverse of this KB distance. So we obtain $1, 0.5$, and 0, respectively for 1-, 2-, and >2-hop neighbors.

The global connectivity score is then converted to an item-level score analogously to the coherence, using max aggregation over pairs. Formally, we define the connectivity of x_{ij} as:

$$conn(x_{ij}) = \frac{1}{m-1} \sum_{k \neq i} \max_{l} conn(x_{ij}, x_{kl}) \tag{2}$$

Note that $conn(x_{ij}) \in [0, 1]$ for all x_{ij}.

Question relatedness. We estimate semantic relatedness of the KB item x_{ij} to the whole input question q by averaging pairwise cosine similarities between the embeddings of the item and each term q_i. The same min-max normalization as for coherence is applied. To avoid confounding this estimate with the question term for which x_{ij} was retrieved, we exclude this from the average. We define semantic relatedness as:

$$rel(x_{ij}) = \text{avg}_{q_i \neq q_k} cos(\vec{x_{ij}}, \vec{q_k}) \tag{3}$$

Term match. This score is intended to take into account the degree of lexical term match (via TF-IDF, BM25, or similar) for which x_{ij} was admitted into l_i. However, such TF-IDF-like weights are often unbounded and may have a disproportionate influence when aggregated with the other signals, that are all in the closed interval $[0, 1]$. Thus, we simply take the reciprocal rank of x_{ij} in l_i as a representative match score to have it in the same $[0, 1]$ interval:

$$match(x_{ij}) = 1/rank(x_{ij}, l_i) \tag{4}$$

2.3. Finding top-k across sorted lists

We then sort each of these $4 \cdot m$ lists in descending score-order. Note that for each q_i and each score s, all lists l_{is} hold the same items (those in the original l_i). Top-k algorithms operating over such multiple score-ordered lists, where each list holds the same set of items, require a monotonic aggregation function over the item scores in each list [10, 11, 14, 15]. Here, we use a linear combination of the four relevance scores as this aggregate: $aggScore(x_{ij}) = h_{coh} \cdot coh(x_{ij}) + h_{conn} \cdot conn(x_{ij}) + h_{rel} \cdot rel(x_{ij}) + h_{match} \cdot match(x_{ij})$, where hyperparameters are tuned on a dev set, and $h_{coh} + h_{conn} + h_{rel} + h_{match} = 1$. Since each score lies in $[0, 1]$, we also have $aggScore(\cdot) \in [0, 1]$. We use the threshold algorithm (TA) with early pruning [11] on these score-ordered lists. TA is run over each set of 4 sorted lists $\langle l_{i1}, l_{i2}, l_{i3}, l_{i4} \rangle$, corresponding to one mention q_i, to obtain the top-k best KB items $\{x_i^*\}_k$ per q_i. These KB items are then the top-k linkings for a specific mention as predicted by our system.

2.4. Automatically setting k

Choosing an appropriate k is non-trivial, a process that is often mention-specific. Intuitively, one would like to increase k for *ambiguous* mentions in the question. For example, *"plays"* can refer to many KB items. By increasing k one can account for potential disambiguation errors. On the other hand, *"GRRM"* is not as ambiguous, which is why setting k=1 should suffice. The ambiguity of a mention is closely connected to that of uncertainty or randomness: the more uncertainty there is in predicting what a mention refers to, the more ambiguous it is. This makes *entropy* a suitable measure of ambiguity. More specifically, for each mention, d KB items are retrieved initially. These items form the sample space of size d for the probability distribution. The numbers of KB facts with these items form a frequency distribution that can be normalized to obtain the required probability distribution. We compute the entropy of this probability distribution as the ambiguity score of a mention, and denote it as $ent(q_i)$. By definition, $0 \leq ent(q_i) \leq \log_2 d$. Practical choices of k and d does not exceed 5 and 50 respectively, and hence k and $\log_2 d$ are in the same ballpark ($\log_2 50$=5.6). This motivates us to make the simple choice of directly setting k as $ent(q_i)$. Specifically, we use $k = \lfloor ent(q_i) \rfloor + 1$ to avoid the situation of k=0. Fig. 1 shows a possible "auto-k" (automatic choice of k) setting for our running example, and the corresponding top-k linkings.

"plays" is highly ambiguous, and thus k is set to a relatively high value. *"Viserys"* and *"HBO"* can also refer to different concepts. The word *"GRRM"* is relatively unambiguous.

3. Adapting CLOCQ to linking tasks

The native CLOCQ method was primarily designed for retrieving a search space of relevant KB facts for a given user question. Therefore, the linking of entities and relations is more of a means to an end here, and is not optimized for the specific tasks. We identified two key obstacles when using the plain CLOCQ method for entity and relation linking.

The first obstacle is that CLOCQ links *all mentions* in the question. Not only entities and relations are linked, but also types (like *"series"*) and other mentions (e.g. *"latest"* in the running example). While linking such mentions is beneficial for coherence among linkings, and can

improve initiating a search space, it often adds undesired noise to the outputs when evaluating entity or relation linking capabilities.

For example, CLOCQ would link *"series"* to the entities `TV series` and the relation *"latest"* to `latest start date` for the running example, which would both decrease precision.

The second obstacle is that CLOCQ does not differentiate between entity and relation mentions. Any mention is disambiguated to the KB items that score best w.r.t. the specified scoring mechanism. For example, the relation mention *"plays"* could also be linked to the entities `play` or `playwright`, similarly as *"director"* could be linked to the relation `director` or to the type `director`. Again, this does not hurt when initiating the search space, but definitely restricts the relation linking capabilities of CLOCQ.

In the following, we will discuss how we optimized CLOCQ for the entity and relation linking tasks of the SMART 2022 challenge. The same intuitions apply to other entity or relation linking problems as well.

3.1. Post-hoc pruning module for entity linking

As discussed, linking all mentions jointly is beneficial for linking results, since it considers information on the whole question in the linking stage. This follows our intuition of understanding the question in its entirety. Also, we did not want to touch the main CLOCQ algorithm itself. Instead, our idea is to *prune* the linkings returned by CLOCQ. We propose a simple approach: the decision, whether an entity should be included in the linking results or not, should be made depending on the mention the entity was disambiguated for. If the mention should be disambiguated, we add the linking, if not it is dropped from the results. For example, the mentions *"plays"*, *"latest"* and *"series"* should not be disambiguated when solving an entity linking task.

Training. We aim to learn which mentions should be linked (and which not) using *distant supervision* on the training data provided as part of the SMART task. Given a training instance, we first obtain all ⟨mention, KB entity⟩ pairs (i.e. the linkings) using the native CLOCQ method. From the training instance, we know the gold entities that should be linked. We then consider all mentions, that are linked with a gold entity by CLOCQ, as mentions that should be disambiguated. For the running example we would obtain the mentions *"Viserys"*, *"GRRM"*, and *"HBO"*, assuming the gold entity set ⟨`Viserys I Targaryen, George R.R. Martin, HBO network`⟩. With this information, we can create a training instance for learning the relevant entity mentions. The input is the question, and the output is the concatenation of the mentions linked to gold entities *"Viserys|GRRM|HBO"* separated by the special token "|". We then simply fine-tune a pre-trained sequence generation model on this data. For this purpose we used BART [16], which was found to be effective when text is copied and manipulated from the input to autoregressively generate the output.

Inference. At inference time, the pruning module is applied in a *post-hoc* manner. We first run CLOCQ and our trained pruning module on the input question. We then keep a ⟨mention, KB entity⟩ pair only if the disambiguated mention matches with any mention generated by our pruning module. Here, matching is relaxed to substring matching. For example, if the pruning module generates *"in GRRM"* or *"GRR"*, linking pairs for *"GRRM"* are still kept. In addition, for

Table 1
Basic statistics of the SMART 2022 entity linking task.

No. of total instances	63,781
No. of train instances	49,987
No. of test instances	13,794
Avg. no. of entities per question (train set)	1.13
No. of questions with $\geq$ 2 entities (train set)	5,873
No. of questions with $\geq$ 3 relations (train set)	390

Table 2
Basic statistics of the SMART 2022 relation linking task.

No. of total instances	30,141
No. of train instances	24,112
No. of test instances	6,029
Avg. no. of relations per question (train set)	1.74
No. of questions with $\geq$ 2 relations (train set)	16,588
No. of questions with $\geq$ 3 relations (train set)	1,234

the entity linking task, we remove all relations from the linkings (relation identifiers start with a "P" in Wikidata).

Note that the post-hoc pruning module is also capable of learning benchmark-specific properties. The SMART 2022 entity linking task can often (but not always) require linking types or concepts. For example, *"airline"* in *"DC-3 is operated by which airline?"* should be linked, but not *"continents"* in *"How many continents are in Antarctica?"*. Such benchmark characteristics are learned implicitly by our pruning module, which can help improve the performance.

3.2. Increasing k for relation linking

As mentioned earlier, relation mentions may also be linked to entities that are coherent with the other linkings. We found that this can often be the case for CLOCQ linkings, and that the appropriate relation can be deeper in the ranked linkings of a mention than the automatically set cut-off length k. However, relations can easily be differentiated from entities via the identifier (relation identifiers start with a "P", entity identifiers with a "Q" in Wikidata). We therefore simply set d=50 and k=40 to increase the probability of obtaining relations, and prune all entities from the linkings. Finally, we explore the effect of keeping either the *top-ranked relation per mention*, or *all relations per mention* as the final result.

4. Experiments

4.1. Experimental setup

SMART 2022 tasks. Statistics on the entity linking and relation linking tasks of the SMART Task 2022 can be found in Table 1 and 2. For both tasks, the question and the corresponding gold entities or relations are given for the train set. For the test set, only the question is given. The datasets are made publicly available[3].

Metrics. We use the standard metrics of the SMART 2022 Task for both tasks: i) **precision**, that measures what fraction of the predicted linkings are correct, ii) **recall**, that measures what fraction of the gold linkings are found, and iii) **F1 score**, the harmonic mean of precision and recall. The results on the test set were provided by the task organizers, after we submitted our system results (since the gold standard is not publicly available).

Initialization of CLOCQ. In our experiments, we use Wikidata [1] as the knowledge base. We access CLOCQ via the public API[4]. The API currently uses a cleaned Wikidata dump[5] from 31 January 2022, which has 94 million entities and 3,000 predicates.

All parameters are kept at default values (d=20, h_{coh}=0.1, h_{conn}=0.3, h_{rel}=0.2, h_{match}=0.4) [8] unless stated otherwise. For the entity linking task, we randomly sample 10,000 training instances and use it as our development set (dev set) for choosing the best pruning module. Since CLOCQ is an unsupervised method, the train set is only used for training the pruning module on the entity linking task, and for tuning the parameters d (=50) and k (=40) on the relation linking task.

Initialization of pruning module. For implementing the pruning module, we use the pre-trained BART model available on the Hugging Face library[6]. We make the code for the pruning module publicly available[7]. We choose k=1 for CLOCQ during distant supervision (Sec. 3.1). The model is fine-tuned for 5 epochs, with 500 warm-up steps, a batch size of 10, and a weight decay of 0.01. We employ cross-entropy as the loss function. After each epoch, we run the model on the withheld dev set, and finally choose the model with the lowest loss there.

CLOCQ variants. On the entity linking task, we compare the linking results of the native CLOCQ method with k=1 or k=AUTO, with the linking results after applying the post-hoc pruning module (again, k=1 or k=AUTO). On the relation linking task, we consider either the top-ranked relation per mention, or all relations per mention, as returned by CLOCQ.

4.2. Entity linking

The results on the entity linking task are shown in Table 3. When considering only the top-1 entity per mention, CLOCQ obtains a recall of 0.766. Setting k=AUTO improves the recall by $\simeq$ 0.1, indicating that potential errors can be overcome. Further, activating the pruning module

[3] https://github.com/smart-task/smart-2022-datasets
[4] https://clocq.mpi-inf.mpg.de
[5] https://github.com/PhilippChr/wikidata-core-for-QA
[6] https://huggingface.co/facebook/bart-base
[7] https://github.com/PhilippChr/CLOCQ-pruning-module

Table 3
Results of CLOCQ variants on the SMART 2022 task for entity linking.

Entity Linking				
System	**Precision**	**Recall**	**F1**	**No. of entities**
CLOCQ (*k*=1)	0.281	0.766	0.399	3.40
CLOCQ (*k*=AUTO)	0.147	**0.870**	0.240	8.37
CLOCQ (*k*=1) + pruning	**0.714**	0.732	**0.708**	1.34
CLOCQ (*k*=AUTO) + pruning	0.410	0.831	0.500	3.52

can drastically improve the precision of CLOCQ, and thus also the F1 score. When adding the pruning module for k=1, precision jumps from 0.281 to 0.714. Also, the results indicate that mostly noise is pruned, since the recall remains fairly stable. Again, recall can be substantially improved ($\simeq$ 0.1) by setting k=AUTO, with the cost of a lower precision and F1 score.

The results indicate that the pruning module can successfully reduce noise in the entity linking results. Further, we found that there is a trade-off between precision and recall, which makes it impossible to determine a best variant for all scenarios. The best choice may highly depend on the specific QA system. Some QA systems require precise linkings for each mention [17], while others can cope with some noise [18], and leverage the boosted recall.

For example, a QA system optimized for efficiency may only issue exactly one explicit query to the KB [19], and may therefore rather go with k=1 and an activated pruning module. On the other hand, if executing multiple queries is affordable for the QA system [7], using top-k linkings might help.

For example, the queries with incorrectly linked entities in top positions might not return any result, while queries with lower ranked entities are able to identify the correct answer. Re-ranking results after query execution might also be an option. When following a graph-based approach without explicit queries, setting k=AUTO was found to be beneficial [8].

An anecdotal example from the dev set for which an automatically increased k helps is: *"What was Toby Wright's profession?"*. There are different persons named *"Toby Wright"* in the KB, and the context does not help to resolve the ambiguity. With k=1, only the incorrect `Toby Wright (football player)` is returned. When setting k=AUTO, CLOCQ identifies the ambiguity of the mention and sets k=2 for this mention. The correct entity `Toby Wright (record producer)` is then fetched at second rank of the results.

Another interesting question from the dev set is *"which footballer was born in middlesbrough?"*. The mention *"footballer"* indicates that the question is on the topic of football, and therefore CLOCQ provides `Middlesbrough F.C. (football club)` as the top-ranked linking for *"middlesbrough"*. However, in this question *"middlesbrough"* refers to the corresponding town. Again, in the auto-k mode, CLOCQ chooses a higher k (=3), and includes the correct entity in these top-3 linkings ⟨`Middlesbrough F.C. (football club)`, `Middlesbrough (borough)`, `Middlesbrough (town)`⟩.

4.3. Relation linking

The results on the relation linking task are shown in Table 4.

Table 4
Results of CLOCQ variants on the SMART 2022 task for relation linking.

Relation Linking				
System	**Precision**	**Recall**	**F1**	**No. of relations**
CLOCQ (top-ranked relation)	**0.380**	0.365	**0.352**	1.69
CLOCQ (all relations)	0.266	**0.414**	0.279	4.40

As for the entity linking task, considering more linkings per mention can help to boost recall, by 0.05 in this case. Again, precision drops substantially, leading to a decreased F1 score. Considering only the top-ranked relation per mention achieves the better F1 score. Interestingly, the average number of relations per question in the system results is quite close to the average number of gold relations (we assume that this property is similar on the train and test sets).

Overall, the results indicate that relation linking may require methods optimized specifically for this purpose. Still, being a general linking method, CLOCQ can provide the correct relation for a substantial part of the questions, often bridging the lexical gap between the relation mention and the surface form of the relation in the KB. Note that there are very few existing systems that can perform both entity and relation linking: this is one of the novelties in CLOCQ. Another such system is [20].

For example, for the question *"Which child of John Adams died on February 23, 1848?"* from the training set, CLOCQ correctly links *"child"* to `child`, and *"died on"* to `date of death`. However, for questions like *"What is the point in time that Nicolaus Cusanus was made cardinal by the Holy Roman Church?"*, CLOCQ failed to link the correct relations `start time` and `position held`.

5. Related work

5.1. Entity linking

There has been extensive research on entity linking and we discuss some prominent works here. TagMe [21], one of the early yet effective systems, makes use of Wikipedia anchors to detect entity mentions, looks up possible mappings, and scores these with regard to a collective agreement implemented by a voting scheme. In AIDA [22], a mention-entity graph is established. Then, the entity mentions are linked jointly by approximating the densest subgraph.

Coming to more recent neural systems, REL [23] is a framework for end-to-end entity linking, building on state-of-the-art neural components. ELQ [24] jointly performs mention detection and linking, leveraging a BERT-based bi-encoder. These methods are optimized for computing the top-1 entity per mention, and mostly give only the top-ranked entity in the disambiguation. Top-1 entity linking is prone to errors that can affect the whole QA pipeline [25, 26]. S-MART [27] introduces structured multiple additive regression trees, and applies the statistical model on a set of (mention, entity)-pairs and corresponding features. Unlike most other works, S-MART returns the top-k disambiguations per mention. However, since it is a proprietary entity linking system, their code is not available.

5.2. Relation linking

Relation linking is particularly useful for QA systems constructing an explicit query. Early approaches used paraphrase-based dictionaries [28] or patterns [29] to link relation mentions. Following approaches often leveraged semantic parses [30] for relation linking, which has also been shown to be effective in combination with neural models [31]. There is also a line of work that approaches relation linking as a classification task [32, 26, 33]. While these methods often achieve high accuracy, a common bottleneck is that only a fraction of all KB relations that are provided in the benchmark can be recognized. Therefore, they are mostly applied in the context of information extraction (IE), rather than QA. Finally, for previous iterations of the SMART Task in 2020 and 2021, a range of relation linking methodologies has been proposed and evaluated [34, 35].

5.3. Joint entity and relation linking

Entity and relation linking are complementary problems, where the results of one task can help solving the other. Thus, either linking is often an intrinsic part of the QA pipeline itself, in which entity and relation linking are implicitly solved in a joint manner [13, 28, 36]. EARL [20] is a dedicated linking system that aims to leverage this intuition of joint disambiguation for entity and relation linking tasks. CLOCQ generalizes this idea further, by initially linking any mention to the KB. In this work, we evaluate the applicability of CLOCQ to both tasks, entity and relation linking.

6. Conclusion

We apply CLOCQ [8] on the entity and relation linking challenges of the SMART 2022 Task. Since the original unsupervised algorithm links all mentions in the question, leading to a substantial amount of noise, we propose a post-hoc pruning module. This supervised module works on top of the linking results by CLOCQ, and prunes linkings for irrelevant mentions. The pipeline depicts a hybrid of supervised and unsupervised modules, leveraging the strengths of both worlds. The results on the SMART entity linking task indicate that the module successfully reduces noise in the linkings, and helps to achieve the overall best F1 score of the CLOCQ variants. Future work could target entity linking and relation linking in conversational settings, where linking mentions can require understanding the whole conversation [18, 37].

References

[1] D. Vrandečić, M. Krötzsch, Wikidata: A free collaborative knowledgebase, in: CACM, 2014.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A nucleus for a Web of open data, in: The Semantic Web, 2007.

[3] F. M. Suchanek, G. Kasneci, G. Weikum, YAGO: A core of semantic knowledge, in: WWW, 2007.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in: SIGMOD, 2008.
[5] R. Saha Roy, A. Anand, Question Answering for the Curated Web: Tasks and Methods in QA over Knowledge Bases and Text Collections, Springer, 2022.
[6] D. Guo, D. Tang, N. Duan, M. Zhou, J. Yin, Dialog-to-action: conversational question answering over a large-scale knowledge base, in: NeurIPS, 2018.
[7] H. Bast, E. Haussmann, More accurate question answering on freebase, in: CIKM, 2015.
[8] P. Christmann, R. Saha Roy, G. Weikum, Beyond NED: Fast and effective search space reduction for complex question answering over knowledge bases, in: WSDM, 2022.
[9] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, W. Cohen, Open domain question answering using early fusion of knowledge bases and text, in: EMNLP, 2018.
[10] V. N. Anh, A. Moffat, Pruned query evaluation using pre-computed impacts, in: SIGIR, 2006.
[11] R. Fagin, A. Lotem, M. Naor, Optimal aggregation algorithms for middleware, Journal of computer and system sciences 66 (2003).
[12] X. Lu, S. Pramanik, R. Saha Roy, A. Abujabal, Y. Wang, G. Weikum, Answering complex questions by joining multi-document evidence with quasi knowledge graphs, in: SIGIR, 2019.
[13] P. Christmann, R. Saha Roy, A. Abujabal, J. Singh, G. Weikum, Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion, in: CIKM, 2019.
[14] H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum, IO-Top-k: Index-access Optimized Top-k Query Processing, in: VLDB Conference, 2006.
[15] C. Buckley, A. F. Lewit, Optimization of inverted vector searches, in: SIGIR, 1985.
[16] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, in: ACL, 2020.
[17] A. Abujabal, R. Saha Roy, M. Yahya, G. Weikum, Never-ending learning for open-domain question answering over knowledge bases, in: WWW, 2018.
[18] P. Christmann, R. Saha Roy, G. Weikum, Conversational question answering on heterogeneous sources, in: SIGIR, 2022.
[19] D. Ziegler, A. Abujabal, R. S. Roy, G. Weikum, Efficiency-aware answering of compositional questions using answer type prediction, in: IJCNLP, 2017.
[20] M. Dubey, D. Banerjee, D. Chaudhuri, J. Lehmann, EARL: joint entity and relation linking for question answering over knowledge graphs, in: ISWC, 2018.
[21] P. Ferragina, U. Scaiella, TAGME: On-the-fly annotation of short text fragments (by Wikipedia entities), in: CIKM, 2010.
[22] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, G. Weikum, Robust disambiguation of named entities in text, in: EMNLP, 2011.
[23] J. M. van Hulst, F. Hasibi, K. Dercksen, K. Balog, A. P. de Vries, Rel: An entity linker standing on the shoulders of giants, in: SIGIR, 2020.
[24] B. Z. Li, S. Min, S. Iyer, Y. Mehdad, W.-t. Yih, Efficient one-pass end-to-end entity linking for questions, in: EMNLP, 2020.
[25] T. Shen, X. Geng, Q. Tao, D. Guo, D. Tang, N. Duan, G. Long, D. Jiang, Multi-task learning for

conversational question answering over a large-scale knowledge base, in: EMNLP-IJCNLP, 2019.
[26] W.-t. Yih, M.-W. Chang, X. He, J. Gao, Semantic parsing via staged query graph generation: Question answering with knowledge base, in: ACL-IJCNLP, 2015.
[27] Y. Yang, M.-W. Chang, S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking, in: ACL-IJCNLP, 2015.
[28] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, G. Weikum, Natural language questions for the web of data, in: EMNLP, 2012.
[29] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, P. Cimiano, Template-based question answering over RDF data, in: WWW, 2012.
[30] W.-t. Yih, X. He, C. Meek, Semantic parsing for single-relation question answering, in: ACL), 2014.
[31] T. Naseem, S. Ravishankar, N. Mihindukulasooriya, I. Abdelaziz, Y.-S. Lee, P. Kapanipathi, S. Roukos, A. Gliozzo, A. Gray, A semantics-aware transformer model of relation linking for knowledge base question answering, in: ACL-IJCNLP, 2021.
[32] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, Relation classification via convolutional deep neural network, in: COLING, 2014.
[33] J. Feng, M. Huang, L. Zhao, Y. Yang, X. Zhu, Reinforcement learning for relation classification from noisy data, in: AAAI, 2018.
[34] N. Mihindukulasooriya, M. Dubey, A. Gliozzo, J. Lehmann, A.-C. N. Ngomo, R. Usbeck, Semantic answer type prediction task (smart) at iswc 2020 semantic web challenge, arXiv (2020).
[35] N. Mihindukulasooriya, M. Dubey, A. Gliozzo, J. Lehmann, A.-C. N. Ngomo, R. Usbeck, G. Rossiello, U. Kumar, Semantic answer type and relation prediction task (smart 2021), arXiv (2021).
[36] A. Abujabal, M. Yahya, M. Riedewald, G. Weikum, Automated template generation for question answering over knowledge graphs, in: WWW, 2017.
[37] H. Joko, F. Hasibi, K. Balog, A. P. de Vries, Conversational entity linking: Problem definition and datasets, 2021.