ReLiK: Retrieve and LinK, Fast and Accurate Entity Linking and Relation Extraction on an Academic Budget

Anonymous ACL submission

Abstract

Entity Linking (EL) and Relation Extraction (RE) are fundamental tasks in Natural Language Processing, serving as critical components in a wide range of applications. In this paper, we propose ReLiK, a Retriever-Reader architecture for both EL and RE, where, given an input text, the Retriever module undertakes the identification of candidate entities or relations that could potentially appear within the text. Subsequently, the Reader module is tasked to discern the pertinent retrieved entities or relations and establish their alignment with the corresponding textual spans. Notably, we put forward an innovative input representation that incorporates the candidate entities or relations alongside the text, making it possible to link entities or extract relations in a single forward pass and to fully leverage pre-trained language models contextualization capabilities, in contrast with previous Retriever-Reader-based methods, which necessitate a forward pass for each candidate. Our formulation of EL and RE achieves state-of-the-art performance in both in-domain and out-of-domain benchmarks while using academic budget training and with up to 40x inference speed with respect to other competitors. Finally, we show how our architecture can be seamlessly used for Information Extraction (cIE), i.e. EL + RE, and setting a new state of the art by employing a shared Reader that simultaneously extracts entities and relations.

1 Introduction

Extracting structured information from unstructured text lies at the core of many AI problems, such as Information Retrieval (Hasibi et al., 2016; Xiong et al., 2017), Knowledge Graph Construction (Clancy et al., 2019; Li et al., 2023), Knowledge Discovery (Trisedya et al., 2019), Automatic Text Summarization (Amplayo et al., 2018; Dong et al., 2022), Language Modeling (Yamada et al., 2020; Liu et al., 2020b), Automatic Text Reasoning (Ji et al., 2022), and Semantic Parsing (Bevilacqua et al., 2021; Bai et al., 2022), inter alia. Looking at the variety of applications in which IE systems are used, we argue such systems should strive to satisfy three fundamental properties: Inference Speed, Flexibility, and Performance.

This work focuses on two of the most popular IE tasks: Entity Linking and Relation Extraction. While tremendous progress has recently been made on both EL and RE, to the best of our knowledge, recent approaches only focus on at most two out of the aforementioned three properties simultaneously (usually either Performance and Inference Speed (De Cao et al., 2021a), or Performance and Flexibility (Zhang et al., 2022)), hindering their applicability in multiple scenarios. Here, we show that by harnessing the Retriever-Reader paradigm (Chen et al., 2017), it is possible to use the same underlying architecture to tackle both tasks, improving the current state-of-the-art while satisfying all fundamental properties. Most importantly, our models are trainable on an academic budget with a short experiment lifecycle, leveling the current playing field and making research on these tasks accessible for academic groups.

Our system ReLiK frames EL and RE similarly to recent Open Domain Question Answering (ODQA) systems (Zhang et al., 2023) where, given an input question, a bi-encoder architecture (Retriever) encodes the input text and retrieves the most relevant text passages from an external index containing their encodings. Then, a second encoder (Reader) takes in input the question and each retrieved passage separately and extracts the answer from a specific passage if present. For our tasks, EL and RE, the input query corresponds to the sentence in which we have to link entities and/or extract relations; the retrieved passages are the entities or relations definitions; and predicting an answer translates into linking the entities and/or extracting the relations. However, our framing differs from most famous ODQA ones in two main respects: i)

for both EL and RE, the input text contains multiple questions simultaneously since there might be multiple entities to link, and/or multiple relations to extract; ii) we encode the input text with all its retrieved passages (i.e. the textual representations of the candidate entities or relations), linking all the entities or extracting all the relational triplets in a single forward pass. Our architecture can thus be conceptually divided into two main components:

- The Retriever that is tasked to retrieve the possible Entities/Relations that can be extracted from a given input text.
- The Reader, that, given the original input text and all the retrieved Entities/Relations (output of the Retriever), is tasked to connect them to the relevant spans in the text.

ReLiK innovates and integrates various unique properties and benefits: first, leveraging the non-parametric memory, i.e. the knowledge base accessed by the Retriever component, considerably lowers the number of parameters required by the final model to achieve state-of-the-art performances (**Inference Speed**). Second, using textual representations for entities/relations combined with the Retriever component makes it easier for the final model to zero-shot on unseen entities/relations (**Flexibility**). Finally, using our novel input formulation we exploit to the fullest the contextualization capabilities of novel large language models such as He et al. (2023). Indeed, with an extensive array of experiments, we show that encoding the input text and the textual representation of entities/relations and linking/extracting them in the same forward pass improves both model's final performances and processing speed (**Performance** and **Inference Speed**).

To foster research and usage of ReLiK, we release the code and models' weights at `http://www.omitted.link`.

2 Background

Entity Linking (EL) is the task of identifying all the entity mentions in a given input text and linking them to an entry in a reference knowledge base. Formally, we can define an EL system as a function that, given an input text q and a reference knowledge base $\mathcal{E}$, identifies all the mentions in q along their corresponding entities $\{(m, e) : m \in \mathcal{M}(q), e \in \mathcal{E}\}$ where $m := (s, t) \in \mathcal{M}(q)$ represents a span among all the possible spans $\mathcal{M}(q)$ in the input text q starting in s and ending in t with $1 \leq s \leq t \leq |q|$.

Relation Extraction (RE) is the task of extracting semantic relations between entities found within a given text from a closed set of relation types coming from a reference knowledge base. Formally, for an input text q and a closed set of relation types $\mathcal{R}$, RE consists of identifying all triplets $\{(m, m', r) : (m, m') \in \mathcal{M}(q) \times \mathcal{M}(q), r \in \mathcal{R}\}$ where m and m' are respectively the subject and object spans and r a relation between them. The combination of both EL and RE as a unified task is known as closed Information Extraction (cIE).

3 The Reader-Retriever (RR) paradigm

In this section, we introduce ReLiK, our Retriever-Reader architecture for EL, RE, and cIE. While the Retriever is shared by the three tasks (Section 3.1), the Reader has a common formulation for span identification but slightly differs in the last linking and extraction steps (Section 3.2). Figure 1 shows a high-level overview of ReLiK as a unified framework for EL, RE and cIE.

3.1 Retriever

For the Retriever component, we follow a retrieval paradigm similar to Dense Passage Retrieval (DPR) (Karpukhin et al., 2020) based on an encoder that produces a dense representation of our queries and passages. In our setup, given an input text q as our query and a passage $p \in \mathcal{D}_p$ in a collection of passages $\mathcal{D}_p$ that corresponds to the textual representations[1] of either entities or relations, the Retriever model computes:

$$E_Q(q) = \text{Retriever}(q),\ E_P(p) = \text{Retriever}(p)$$

and ranks the most relevant entities or relations with respect to q using the similarity function $\text{sim}(q, p) = E_Q(q)^\top E_P(p)$, where the contextualized hidden representation of a query q and a passage p are computed by the same Retriever Transformer encoder.[2]

We train the Retriever employing a multi-label noise contrastive estimation (NCE) as a training

[1] A textual representation of an entity or a relation is any text that univocally identifies them. Using Wikipedia as reference knowledge base for entity linking, a textual representation for an entity might be its Wikipedia title.

[2] The representations consist of the average of the encodings for the tokens in each of the two sequences.

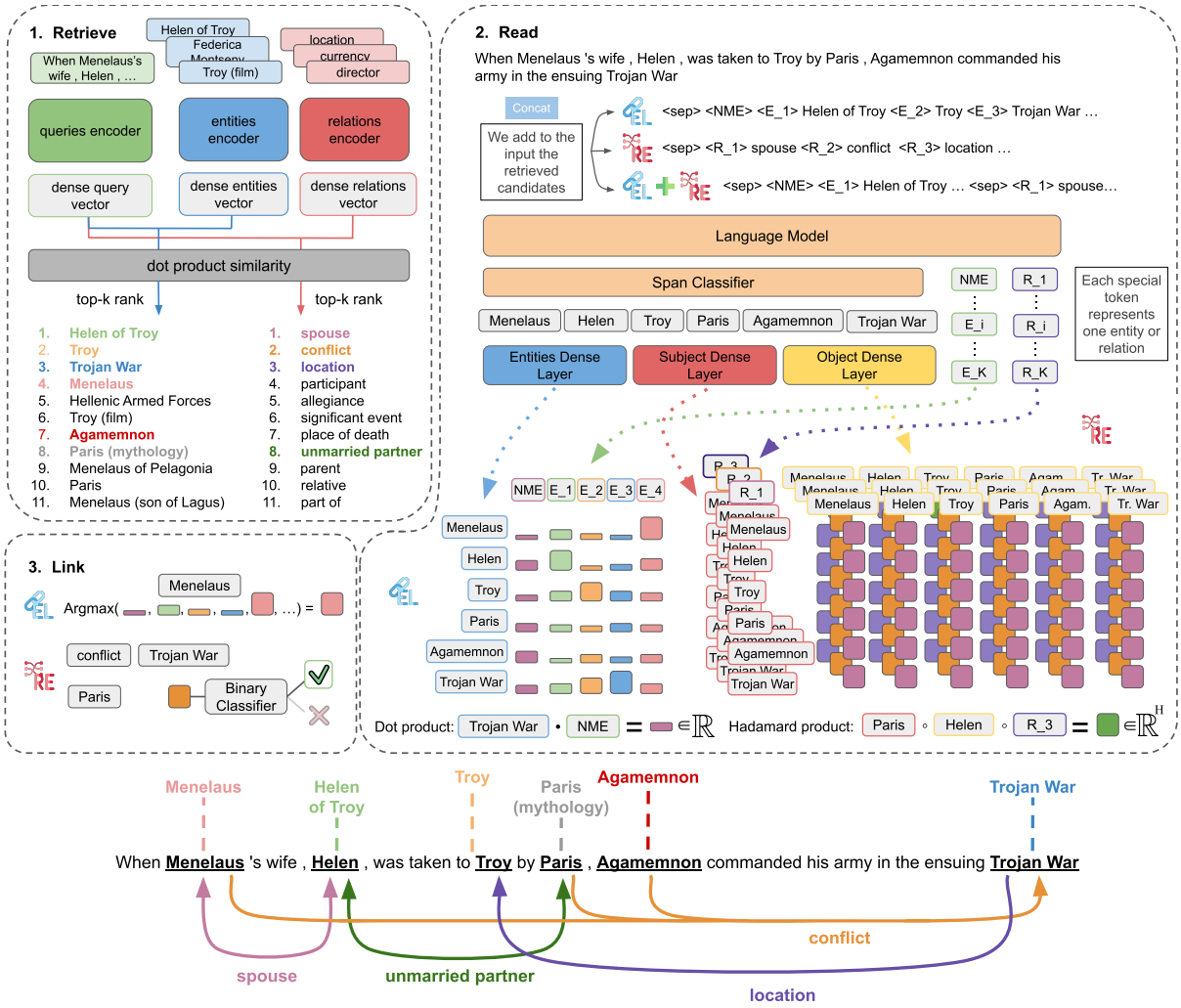

Figure 1: Description of ReLiK. Based on the RR-paradigm, we (1) Retrieve candidate entities and relations (2) Read and contextualize the text and candidates (3) Link and extract entities and triplets.

objective. The $\mathcal{L}_{Retriever}$ loss for q is defined as:

$$-\log \sum_{p^+ \in \overline{\mathcal{D}_p}(q)} \frac{e^{\text{sim}(q,p^+)}}{e^{\text{sim}(q,p^+)} + \sum_{p^- \in P_q^-} e^{\text{sim}(q,p^-)}} \quad (1)$$

where $\overline{\mathcal{D}_p}(q)$ are the gold passages of the entities or relations present in q, and P_q^- is the set of negative examples for q, constructed using in-batch negatives from gold passages of other queries and by hard negative mining using highest-scoring incorrect passages retrieved by the model.

3.2 Reader

Differently from other ODQA approaches, our Reader performs a single forward pass for each input query. We append the top-k retrieved passages, $p_{1:K} = (p_1, \ldots, p_K), p_i \in \mathcal{D}_p$,[3] to the input query q, and obtain the sequence $q\ [SEP]\ \langle ST_0 \rangle \langle ST_1 \rangle p_1 \ldots \langle ST_K \rangle p_K$, with $[SEP]$ being a special token used to separate the query from the retrieved passages, and $\langle ST_i \rangle$ being special tokens used to mark the start of the i-th retrieved passage. We obtain the hidden representations X of the sequence using a Transformer encoder:

$$X = \text{Tr}\left(q\ [SEP]\ \langle ST_0 \rangle \ldots p_K\right) \in \mathbb{R}^{l \times H} \quad (2)$$

where $l = |q| + 1 + (1 + K) + \sum_k |p_k|$ is the total length in tokens. Now, we predict all mentions within q, $\widetilde{\mathcal{M}}(q)$. We first compute the probability of each token s to be the start of a mention as:

$$p_S(s|X) = \sigma_0(W_S^T X_s + b_S) \quad \forall s \in \{1, \ldots, |q|\}$$

with $W_S \in \mathbb{R}^{H \times 2}, b_S \in \mathbb{R}^2$ being learnable parameters, and σ_i the softmax function value at position i. Then we compute the probability of a token t to be the end of a mention with starting token s is:

$$p_E(t|X, s) = \sigma_0(W_E^T X_m + b_E) \; \forall t \in \{s, \ldots, |q|\}$$

[3] The k highest scoring passages according to the sim function introduced in Section 3.1.

with $W_E \in \mathbb{R}^{2H\times 2}, b_E \in \mathbb{R}^2$ being learnable parameters and $X_m \in \mathbb{R}^{2H}$ the concatenation of X_s and X_t. We note that with this formulation we support the prediction of overlapping mentions. The loss for identifying spans in a single query is:

$$\mathcal{L}_S = -\sum_{s=0}^{|q|} \mathbb{1}_{\overline{\mathcal{M}_S}(q)}(s) log(p_S(s|X)) - \mathbb{1}_{\overline{\mathcal{M}_S}(q)^\complement}(s) log(1 - p_S(s|X))$$

$$\mathcal{L}_E = -\sum_{s\in\overline{\mathcal{M}_S}(q)} \sum_{t=s}^{|q|} \mathbb{1}_{\overline{\mathcal{M}}(q,s)}(t) log(p_E(t|X,s)) - \mathbb{1}_{\overline{\mathcal{M}}(q,s)^\complement}(t) log(1 - p_E(t|X,s))$$

where $\overline{\mathcal{M}_S}(q)$ are the gold start tokens for the mentions in q and $\overline{\mathcal{M}}(q,s)$ are the gold end tokens for mentions that start at s, $^\complement$ indicates complementary set and $\mathbb{1}$ is the indicator function. At inference time, we first compute all s with $p_S(s|X) > 0.5$ and then all ends $p_E(t|X,s) > 0.5$ for each start s to predict mentions $\widetilde{\mathcal{M}}(q)$.

While the formulation for extracting mentions from the input text is shared between EL and RE, the final steps to link them to entities and extract relational triplets are different. In what follows, we describe the two different procedures.

Entity Linking As we now describe the EL step, in this paragraph the retrieved passages will identify the textual representations of the entities we have to link to the previously identified mentions, and thus we will change the notation of $p_{1:K} = (p_1, \ldots, p_K)$ to $e_{0:K} = (e_0, \ldots, e_K), e_{i\neq 0} \in \mathcal{E}$.[4] Specifically, for each $m \in \mathcal{M}(q)$, we need to find $\mathcal{E}(q,m)$, the entity linked to mention m. To do so, we use the hidden representations X from Equation 2, and project each mention and special token in a shared dense space using a feed-forward layer:

$$M = \text{GeLU}\left(W_M^T X_m + b_M\right)$$

$$E_{0:K} = \text{GeLU}\left(W_M^T [X_{\langle ST_{0:K}\rangle}, X_{\langle ST_{0:K}\rangle}] + b_M\right)$$

where $W_M \in \mathbb{R}^{2H\times H}, b_M \in \mathbb{R}^H$ are learnable parameters, and $[X_{\langle ST_{0:K}\rangle}, X_{\langle ST_{0:K}\rangle}] \in \mathbb{R}^{K\times 2H}$ represent the repetition along the hidden representation axis of the special tokens vectors $X_{\langle ST_{0:K}\rangle} \in \mathbb{R}^{K\times H}$ in order to match the shape of X_m. The probability of mention m being linked to entity e_k is computed as:

$$\tilde{p}_{ent} = p_{ent}(\mathcal{E}(q,m) = e_k|M, E_{0:K}) = \sigma_k(E_{0:K}^T M) \quad \forall m \in \mathcal{M}(q),\ k \in \{0 \ldots K\}$$

[4]Here e_0 symbolizes NME, i.e. mentions for which the gold entity is not in $\mathcal{E}$, represented by $\langle ST_0\rangle$.

Therefore, if $\overline{\mathcal{E}}(q,m)$ is the gold entity linked to m in q, the loss for EL is:

$$\mathcal{L}_{EL} = -\sum_{m\in\overline{\mathcal{M}}(q)} \sum_{k=0}^{K} \mathbb{1}_{\overline{\mathcal{E}}(q,m)}(e_k) \log(\tilde{p}_{ent})$$

To train ReLiK for EL, we optimize $\mathcal{L}_{EL}$ and the mention detection losses from Section 3.2: $\mathcal{L} = \mathcal{L}_S + \mathcal{L}_E + \mathcal{L}_{EL}$. At inference time we will have the predicted spans $\widetilde{\mathcal{M}}(q)$ as input to the EL module and we will take $\text{argmax}_k\, p_{ent}(\mathcal{E}(q,m) = e_k|M, E_{0:K})$ for each $m \in \widetilde{\mathcal{M}}(q)$ as its linked entity.

Relation Extraction In RE, the retrieved passages for an input text q will instead identify the textual representations of relations $r_{1:K} = (r_1, \ldots, r_K), r_i \in \mathcal{R}$. Specifically for each pair of mentions $(m, m') \in \mathcal{M}(q) \times \mathcal{M}(q)$ we need to find $\mathcal{R}(q, m, m')$, i.e. the relation types between m and m' expressed in q. To do so, we use the hidden representations X from Equation 2, and project each mention and special token using three feed-forward layers:

$$S_m = \text{GeLU}\left(W_{subject}^T X_m + b_{subject}\right)$$

$$O_{m'} = \text{GeLU}\left(W_{object}^T X_{m'} + b_{object}\right)$$

$$R_k = \text{GeLU}\left(W_r^T X_{\langle ST_k\rangle} + b_r\right)$$

where $W_{subject}, W_{object} \in \mathbb{R}^{2H\times H}, W_r \in \mathbb{R}^{H\times H}$, $b_{subject}$, b_{object} and $b_r \in \mathbb{R}^H$ are learnable parameters. We obtain a hidden representation for each possible triplet with the Hadamard product:

$$T_{m,m',k} = S_m \odot O_{m'} \odot R_k \in \mathbb{R}^H$$

which is a dense representation of relation (k) between subject (m) and object (m'). Then, the probability that m and m' are in a relation r_k in q is:

$$\tilde{p}_{rel} = p_{rel}(r_k \in \mathcal{R}(q,m,m')|T_{m,m',k}) = \sigma_0(W_{rel} T_{m,m',k} + b_{rel})$$

$$\forall\ (m,m') \in \mathcal{M}(q) \times \mathcal{M}(q), k \in \{1, \ldots, K\}$$

with $W_{rel} \in \mathbb{R}^{H\times 2}, b_{rel} \in \mathbb{R}^2$ being learnable parameters. If we take $\overline{\mathcal{R}}(q,m,m')$ as the gold relations between m and m' in q, the loss for RE is defined as follows:

$$\mathcal{L}_{rel} = -\sum_{\substack{(m,m')\in \\ \mathcal{M}(q)\times\mathcal{M}(q)}} \left(\sum_{k=1}^{K} \mathbb{1}_{\overline{\mathcal{R}}(q,m,m')}(r_k) log(\tilde{p}_{rel}) - \mathbb{1}_{\overline{\mathcal{R}}(q,m,m')^\complement}(r_k) log(1 - \tilde{p}_{rel}) \right)$$

To train ReLiK for RE we optimize $\mathcal{L}_{rel}$ and the losses from Section 3.2: $\mathcal{L} = \mathcal{L}_S + \mathcal{L}_E + \mathcal{L}_{rel}$. At inference time we compute all mentions $\widetilde{\mathcal{M}}(q)$ and

then predict all triplets (m, m', r_k) where $p_{rel}(r_k \in \mathcal{R}(q, m, m')|T_{m,m',k}) > 0.5 \; \forall \; (m, m') \in \widetilde{\mathcal{M}}(q) \times \widetilde{\mathcal{M}}(q)$.

closed Information Extraction In the previous paragraphs, we described how to perform EL and RE separately with ReLiK. However, since both tasks share the same mention detection approach, ReLiK allows for closed IE with a single Reader. In this setup, we use the Retriever trained on each task separately to retrieve $e_{1:K} \in \mathcal{E}^K$ and $r_{1:K'} \in \mathcal{R}^{K'}$. Then, the Reader performs both tasks at the same time. The only difference is the input for the hidden representations in Equation 2 as $(q \; [SEP] \; \langle ST_0 \rangle \, \langle ST_1 \rangle \, e_1 \ldots \langle ST_K \rangle \, e_K \; [SEP] \, \langle ST_{K+1} \rangle \, r_1 \ldots \langle ST_{K+K'} \rangle \, r_{K'})$. Additionally, we leverage the predictions of the EL module to condition RE by taking

$$X_m = [X_s, X_t, \sigma(E^T_{0:K} M_m) X_{\langle ST_{0:K} \rangle}]$$

as the input to the RE module after EL predictions are computed. Notice that now $W_{subject}, W_{object} \in \mathbb{R}^{3H \times H}$. Finally, at training time the loss becomes $\mathcal{L} = \mathcal{L}_S + \mathcal{L}_E + \mathcal{L}_{el} + \mathcal{L}_{rel}$ for a dataset annotated with both tasks.

4 Entity Linking

We now describe the experimental setup (Section 4.1) and compare our system to current state-of-the-art solutions (Section 4.2) for EL.

4.1 Experimental Setup

4.1.1 Data

To evaluate ReLiK on Entity Linking, we reproduce the setting used by Zhang et al. (2022). We use the AIDA-CoNLL dataset (Hoffart et al., 2011, AIDA) for the *in-domain* training (AIDA train) and evaluation (AIDA testa for model selection and AIDA testb for test). The *out-of-domain* evaluation is carried on: MSNBC, Derczynski (Derczynski et al., 2015), KORE 50 (Hoffart et al., 2012), N3-Reuters-128, N3-RSS-500 (R500) (Röder et al., 2014), and OKE challenges 2015 and 2016 (Nuzzolese et al., 2015). As our reference knowledge base, we follow Zhang et al. (2022) and use the 2019 Wikipedia dump provided in the KILT benchmark (Petroni et al., 2021). We do not use any *mention-entities* dictionary to retrieve the list of possible entities to associate with a given mention.

4.1.2 Comparison Systems

We compare ReLiK with two autoregressive approaches, namely De Cao et al. (2021b), in which the authors train a sequence-to-sequence model to produce, given a text sequence in input, a formatted string containing the entities spans along with the reference Wikipedia title; and De Cao et al. (2021a) which build on top of this approach by previously identifying the spans of text that may represent entities and then generate in parallel the Wikipedia title of each span, greatly enhancing the speed of the system.

The most similar approach to our system is arguably Zhang et al. (2022), which was the first to invert the standard Mention Detection $\rightarrow$ Entity Disambiguation pipeline for EL. They first used a bi-encoder architecture to retrieve the entities that could appear in a text sequence and then an encoder architecture to reconduct each retrieved entity to a span in the text. We want to highlight that while the Retriever part of ReLiK for EL and Zhang et al. (2022) are conceptually the same, the Reader component strongly differs. Indeed, our Reader is capable of linking all the retrieved entities in a single forward pass, while theirs must perform a forward pass for each retrieved entity, being roughly 40 times slower to achieve the same performance. Finally, we note that, with the exception of Zhang et al. (2022), the other approaches use a *mention-entities* dictionary, i.e. a dictionary that for each mention contains a list of possible entities in the reference knowledge base with which the mention can be associated. In order to build such a dictionary for Wikipedia entities, the hyperlinks in Wikipedia pages are usually utilized (Pershina et al., 2015). This means that given the input sentence "Jordan is an NBA player" in order to link the span "Jordan" to the Wikipedia page of Michael Jordan, there must be at least one page in Wikipedia in which a user manually linked that specific span (Jordan) to the Michael Jordan page. While for frequent entities, this might not represent a problem, for rare entities, it could mean the impossibility of linking them.

4.1.3 Evaluation

We evaluate ReLiK on the GERBIL platform (Röder et al., 2018), using the implementation of Zhang et al. (2022) from the paper repository `https://github.com/WenzhengZhang/EntQA`. We report the results of evaluating against the datasets described in Section 4.1.1 using the *InKB* F1 score with strong matching (prediction boundaries must match exactly gold ones).

	In-domain	Out-of-domain							Avgs		
Model	AIDA	MSNBC	Der	K50	R128	R500	O15	O16	Tot	OOD	AIT (m:s)
De Cao et al. (2021b)†	83.7	73.7	54.1	60.7	46.7	40.3	56.1	50.0	58.2	55.0	38:00
De Cao et al. (2021a)†*	85.5	19.8	10.2	8.2	22.7	8.3	14.4	15.2	—	—	00:52
Zhang et al. (2022)	85.8	72.1	52.9	64.5	**54.1**	41.9	61.1	51.3	60.5	57.3	20:00
ReLiK$_B$	85.9	71.9	55.5	67.2	49.2	41.5	62.6	53.9	61.0	57.9	00:29
ReLiK$_L$	**86.5**	**74.2**	**56.6**	**73.9**	51.4	**43.0**	**66.1**	**55.4**	**63.4**	**60.5**	01:46

Table 1: Comparison systems' evaluation (*inKB Micro* F_1) on the *in-domain* AIDA test set and *out-of-domain* MSNBC (MSN), Derczynski (Der), KORE50 (K50), N3-Reuters-128 (R128), N3-RSS-500 (R500), OKE-15 (O15), and OKE-16 (O16) test sets. **Bold** indicates the best model and underline indicates the second best competitor. † marks systems that use mention dictionaries. * For De Cao et al. (2021a), we report the results on the Out-of-domain benchmark running the model from the official repository, but without using any *mention-entity* dictionary since no implementation of it is provided. AIT column shows the time in minutes and seconds (m:s) that the systems need to process the whole AIDA test set using an NVIDIA RTX 4090, except for Zhang et al. (2022) that does not fit in 24GB of RAM and for which an A100 is used.

4.1.4 ReLiK Setup

Retriever We initialize the query encoder and passage encoder with E5$_{\text{base}}$ (Wang et al., 2022) trained on BLINK[5]. We train the encoder on the AIDA dataset for a maximum of 5000 steps using RAdam (Liu et al., 2020a) with a learning rate of 1e-5 and a linear learning rate decay schedule. We split each document into overlapping chunks of length $W = 32$ words with a stride $S = 16$, resulting in 12,995 windows in the training set, 3292 in the validation set, and 2950 in the test set. We concatenate to each window the first word of the document as in Zhang et al. (2022). We employ KILT (Petroni et al., 2021) to construct the entities index, which contains $|\mathcal{E}| = 5.9$M entities. The textual representation of each entity is a combination of the Wikipedia title and opening text for the corresponding entity contained within KILT. We optimize the NCE loss (Equation 1) with 400 negatives per batch. At the end of each epoch, we mine at most 15 hard negatives per sample in the batch among the highest-scoring incorrect entities retrieved by the model. Appendix A.2.1 shows all the parameters used during the training process.

Reader We train the Reader model with the windows produced by the Retriever on the AIDA dataset. While in the Retriever we use the Wikipedia openings as the entities' textual representations, in the Reader, due to computational constraints, and as in other works (De Cao et al., 2021b,a), we only use Wikipedia titles, which demonstrated to be informative and discriminative in most situations (Procopio et al., 2023). In order to handle the long sequences created by the concatenation of the top-100 retrieved candidates to the windows, we use DeBERTa-v3 (He et al., 2023) as our underlying encoder. We train two versions of it using DeBERTa-v3 base (183M parameters, ReLiK$_B$) and DeBERTa-v3 large (434M parameters, ReLiK$_L$). We optimize both ReLiK$_B$ and ReLiK$_L$ using AdamW and apply a learning rate decay on each layer as in Clark et al. (2020) for 50,000 optimization steps. A table with all the training hyperparamenters can be found in Appendix A.2.1.

4.2 Results

Performance We show in Table 1 the *InKB F1* score ReLiK and its alternatives attain on the evaluation datasets.[6] Arguably, the most interesting finding we report is the improvement in performance we achieve with respect to Zhang et al. (2022). Indeed, not only even ReLiK$_B$ outperforms Zhang et al. (2022) both in- and out-of-domain (85.9 vs 85.8 in-domain and 57.9 vs 57.3 average out-of-domain) with fewer parameters (289M parameters vs 650M parameters), but it does so using a single forward pass to link all the entities in a window of text, greatly enhancing the final speed of the system. A broader look at the table shows that ReLiK$_L$ surpasses all its competitors on all evaluation datasets except R128, thus setting a new state-of-the-art. Finally, another interesting finding is ReLiK$_L$ outperforming its best competitor by 9.4 points on K50. While the other datasets contain news and encyclopedic corpora annotations, K50 is specifically designed to capture hard-to-disambiguate mentions that involve a deep understanding of the context in

[5]Appendix A.1 provides details on the training process.

[6]Additional comparison systems can be found in Table 5.

which they appear. A qualitative error analysis of the predictions can be found in Appendix A.6.

Speed and Flexibility As we can see from Table 1 last column, ReLiK$_B$ is the fastest system among the competitors. Not only that, the second fastest system, i.e. (De Cao et al., 2021a), requires a *mention-entities* dictionary that contains the possible entities to which a mention can be linked. When not using such a dictionary, the results on the AIDA test set drop by 43% (De Cao et al., 2021a) and, as reported in Table 1, it becomes unusable in out-of-domain settings. We want to stress that systems that leverage such dictionaries are less flexible in predicting unseen entities during training and, most importantly, cannot link at all entities to mentions to which they are not specifically paired in the reference dictionary. Finally, our formulation allows the use of relatively large language models such as DeBERTa-v3 large and achieves unprecedented performance while keeping competitive inference speed. Report and ablations on ReLiK efficiency can be found in Appendices A.4 and A.5.

5 Relation Extraction and closed Information Extraction

In this section, we present the experimental setup (Section 5.1) for RE and cIE, and compare the results of our systems to the current state of the art (Section 5.2).

5.1 Experimental Setup

5.1.1 Data

RE We choose two of the most popular datasets available. NYT (Riedel et al., 2010), which has 24 relation types, 60K training sentences and 5K for validation and test; and CONLL04 (Roth and Yih, 2004) with 5 relation types, 922 training sentences, 231 for validation and 288 for testing.

cIE We follow previous work and report on the REBEL dataset (Huguet Cabot and Navigli, 2021), which leverages entity labels from Wikipedia and relation types (10,936) from Wikidata. We subsample 3M sentences for training, 10K for validation and and keep the same test set as Josifoski et al. (2022) containing 175K sentences.

5.1.2 Comparison Systems

RE We compare ReLiK with recent state-of-the-art systems for RE. As with EL, we compare to a recent trend in RE systems using seq2seq approaches. Huguet Cabot and Navigli (2021) reframed the task as a triplet sequence generation, in which the model learns to *translate* the input text into a sequence of triplets. Lu et al. (2022) followed a similar approach to tackle several IE tasks, including RE. They were the first to include labels as part of the input to aid generation. However, while these approaches are flexible and end-to-end, they suffer from efficiency, as they are autoregressive. Lou et al. (2023) built upon Lu et al. (2022), dropping the need for a decoder by having labels as part of the input and reframing the task as linking mention spans and labels between each other, pairwise. This approach is somewhat similar to our EL Reader component. However, it does not include a Retriever, limiting the number of relation types that can be predicted, and their linking pairwise strategy leads to ambiguous decoding for triplets (See A.7 for more details).

cIE The task of cIE has been traditionally tackled using pipelines with systems trained separately for EL and RE. We compare ReLiK to two recent autoregressive approaches. Josifoski et al. (2022), inspired by Huguet Cabot and Navigli (2021), generate the triplets with the unique Wikipedia title of each entity instead of their surface form with the aid of constraint decoding from De Cao et al. (2021b). Rossiello et al. (2023) extended their approach by outputting both surface forms and titles. As with RE, autoregressive approaches did lift the ceiling for cIE, however, they are still slow and computationally heavy at inference time.

5.1.3 Evaluation

We report on micro-F1, using boundaries evaluation, i.e. a triplet is considered correct when entity boundaries are properly identified with the relation type. For cIE, we consider a triplet correct only when both entity spans, their disambiguation, and the relation type between the two entities are correct. To ensure a fair comparison with previous autoregressive systems, we only consider entities present in triplets for EL, albeit ReLiK is able to disambiguate all of them.

5.1.4 ReLiK Setup

Retriever As in the EL setting (Section 4.1.4), we initialize the query and passage encoders with E5 (Wang et al., 2022). In this context, we utilize the `small` version of E5. This choice is driven by the limited search space in contrast to the Entity

Model	Params.	NYT		CONLL04		REBEL	
			Pretr.		Pretr.	EL	RE
Huguet Cabot and Navigli (2021)	460M	93.1	93.4	71.2	75.4	—	—
Lu et al. (2022)	770M	93.5	—	71.4	72.6	—	—
Lou et al. (2023)	355M	94.0	94.1	75.9	**78.8**	—	—
Josifoski et al. (2022)	460M	—	—	—	—	79.7	68.9
Rossiello et al. (2023)	460M	—	—	—	—	82.7	70.7
ReLiK$_S$	33M + 141M	94.4	94.4	71.7	75.8	83.7	73.8
ReLiK$_B$	33M + 183M	94.8	94.7	72.9	77.2	84.1	74.3
ReLiK$_L$	33M + 434M	**95.0**	94.9	75.0	78.1	**85.1**	**75.6**

Table 2: Micro-F1 results for systems trained on NYT, CONLL04 and REBEL datasets. Params. column shows the number of parameters for each system. EL reports only on entities belonging to a triplet. Pretr. indicates the model underwent pretraining on additional task-specific data.

Linking setting. Consequently, this enables us to significantly lower the computational demands for both training and inference. We train the encoder for a maximum of 40,000 steps using RAdam (Liu et al., 2020a) with a learning rate of 1e-5 and a linear learning rate decay schedule. For NYT we have $|\mathcal{R}| = 24$ while for REBEL we use all Wikidata properties with their definitions, $|\mathcal{R}| = 10,936$. For EL we use the same settings explained in Section 4.1 with KILT as KB, $|\mathcal{E}| = 5.9$M. We optimize the NCE loss (1) using 24 negatives per batch for NYT and 400 for REBEL. More details are given in Appendix A.2.1.

Reader The Reader setup mirrors that of EL. We use DeBERTa-v3 in all three sizes with AdamW as the optimizer and a linear decay schedule. For NYT we set $K = 24$, effectively utilizing the Retriever as a ranker. For the CONLL04 dataset, we use the NYT's Retriever. We explore a setup where ReLiK is pretrained using data from REBEL and NYT[7]. In the context of closed Information Extraction (cIE) we set $K = 25$ and $K' = 20$ as the number of passages for EL and RE respectively. In all cases, we select the best-performing validation step for evaluation. A table with all the parameters utilized during training can be found in Appendix A.2.1.

5.2 Results

RE In Table 2, we present the performance of ReLiK in comparison to other systems. Notably, on NYT ReLiK$_S$ achieves remarkable results, outperforming all previous systems while utilizing fewer parameters and with remarkable speed, around 10 seconds to predict the entire NYT test set (see Appendix A.4 for more details). The only exception is the CONLL04 dataset, where ReLiK is outperformed by Lou et al. (2023). However, it is important to note that CONLL04 is an extremely small dataset, where a few instances can lead to a big gap in performance.

cIE The right side of Table 2 reports on closed Information Extraction. Here, ReLiK truly shines as the first efficient end-to-end system for jointly performing EL and RE with exceptional performance. It outperforms previous approaches in all its model sizes by a significant margin and is up to 35x times faster (see Appendix A.4 for more details). ReLiK enables dowsntream cIE use in a previously unattainable capacity.

A qualitative Error Analysis of the predictions can be found in Appendix A.6.

6 Conclusion

In this work, we presented ReLiK, a novel and unified Retriever-Reader architecture that seamlessly attains state-of-the-art performance for both Entity Linking and Relation Extraction. Furthermore, taking advantage of the common architecture and using a shared Reader, our system is capable of achieving unprecedented performance and efficiency even on the closed Information Extraction task (i.e. Entity Linking + Relation Extraction). Our models are considerably lighter, an order of magnitude faster, and trained on an academic budget. We believe that ReLiK can advance the field of Information Extraction in two directions: first, by providing a novel framework for unifying other IE tasks beyond EL and RE, and, second, by providing accurate information for downstream applications in an efficient way.

[7]We replicate the approach from Lou et al. (2023) by sampling 300K from REBEL dataset plus NYT train set. We pretrain for 250,000 steps with the same settings as NYT.

7 Limitations

The main limitation of our work is that while it enables efficient downstream use of very relevant IE tasks, the experiments on the paper are performed on held-out benchmarks which enable comparisons across systems but do not test or demonstrate its effectiveness on a wider range of data, other than the OOD experiments for EL. While this is true for any EL or RE model evaluated in the most common benchmarks, we expect the lightweight computation requirements of ReLiK, as well as its state-of-the-art performance to make it attractive to NLP and real-world applications which should always be done cautiosly, by considering shortcomings or limitations such as an Entity index frozen in time (KILT was built from a Wikipedia dump from 2020), or AIDA as an old dataset that while manually annotated it contains biases of its own, such as conflicting labels regarding Taiwan and China. The NYT and REBEL datasets were distantly annotated which may contain wrong or missing annotations. Again, while these shortcomings are not unique of our work, they should be considered.

References

Reinald Kim Amplayo, Seonjae Lim, and Seung-won Hwang. 2018. Entity commonsense representation for neural abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 697–707, New Orleans, Louisiana. Association for Computational Linguistics.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One spring to rule them both: Symmetric amr semantic parsing and generation without a complex pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573.

Samuel Broscheit. 2019. Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 677–685, Hong Kong, China. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Ryan Clancy, Ihab F. Ilyas, and Jimmy Lin. 2019. Scalable knowledge graph construction from text collections. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 39–46, Hong Kong, China. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021a. Highly parallel autoregressive entity linking with discriminative correction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7662–7669. Association for Computational Linguistics.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021b. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2015. Analysis of named entity recognition and linking for tweets. *Inf. Process. Manag.*, 51(2):32–49.

Yue Dong, John Wieting, and Pat Verga. 2022. Faithful to the document or to the world? mitigating hallucinations via entity-linked knowledge in abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1067–1082, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning.

Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. Exploiting entity linking in queries for entity retrieval. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval, ICTIR 2016, Newark, DE, USA, September 12- 6, 2016*, pages 209–218. ACM.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. Kore: Keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 545–554, New York, NY, USA. Association for Computing Machinery.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Networks Learn. Syst.*, 33(2):494–514.

Martin Josifoski, Nicola De Cao, Maxime Peyrard, Fabio Petroni, and Robert West. 2022. GenIE: Generative information extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4626–4643, Seattle, United States. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529, Brussels, Belgium. Association for Computational Linguistics.

Yangning Li, Jiaoyan Chen, Yinghui Li, Yuejia Xiang, Xi Chen, and Hai-Tao Zheng. 2023. Vision, deduction and alignment: An empirical study on multi-modal knowledge graph alignment. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020a. On the variance of the adaptive learning rate and beyond. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020b. K-bert: Enabling language representation with knowledge graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2901–2908.

Jie Lou, Yaojie Lu, Dai Dai, Wei Jia, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2023. Universal information extraction as unified semantic matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13318–13326.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2019. Joint learning of named entity recognition and entity linking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 190–196, Florence, Italy. Association for Computational Linguistics.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Darío Garigliotti, and Roberto Navigli. 2015. Open knowledge extraction challenge. In *Semantic Web Evaluation Challenges - Second SemWebEval Challenge at ESWC 2015, Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers*, volume 548 of *Communications in Computer and Information Science*, pages 3–15. Springer.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243, Denver, Colorado. Association for Computational Linguistics.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Luigi Procopio, Simone Conia, Edoardo Barba, and Roberto Navigli. 2023. Entity disambiguation with entity definitions. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1297–1303, Dubrovnik, Croatia. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Berlin, Heidelberg. Springer Berlin Heidelberg.

Michael Röder, Ricardo Usbeck, Sebastian Hellmann, Daniel Gerber, and Andreas Both. 2014. N^3 - a collection of datasets for named entity recognition and disambiguation in the NLP interchange format. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3529–3533, Reykjavik, Iceland. European Language Resources Association (ELRA).

Michael Röder, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. 2018. GERBIL - benchmarking named entity recognition and linking consistently. *Semantic Web*, 9(5):605–625.

Gaetano Rossiello, Md. Faisal Mahbub Chowdhury, Nandana Mihindukulasooriya, Owen Cornec, and Alfio Gliozzo. 2023. Knowgl: Knowledge generation and linking from text. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Nadine Steinmetz and Harald Sack. 2013. Semantic multimedia information retrieval based on contextual descriptions. In *The Semantic Web: Semantics and Big Data*, pages 382–396, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dianbo Sui, Xiangrong Zeng, Yubo Chen, Kang Liu, and Jun Zhao. 2023. Joint entity and relation extraction with set prediction networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Bayu Distiawan Trisedya, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. 2019. Neural relation extraction for knowledge base enrichment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 229–240, Florence, Italy. Association for Computational Linguistics.

Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. 2020. Rel: An entity linker standing on the shoulders of giants. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 2197–2200, New York, NY, USA. Association for Computing Machinery.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics.

Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2017. Word-entity duet representations for document ranking. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 763–772. ACM.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Qin Zhang, Shangsi Chen, Dongkuan Xu, Qingqing Cao, Xiaojun Chen, Trevor Cohn, and Meng Fang. 2023. A survey for efficient open domain question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14447–14465, Toronto, Canada. Association for Computational Linguistics.

Wenzheng Zhang, Wenyue Hua, and Karl Stratos. 2022. EntQA: Entity linking as question answering. In *International Conference on Learning Representations*.

Hengyi Zheng, Rui Wen, Xi Chen, Yifan Yang, Yunyan Zhang, Ziheng Zhang, Ningyu Zhang, Bin Qin, Xu Ming, and Yefeng Zheng. 2021. PRGC: Potential relation and global correspondence based joint relational triple extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6225–6235, Online. Association for Computational Linguistics.

Wenxuan Zhou and Muhao Chen. 2022. An improved baseline for sentence-level relation extraction. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 161–168, Online only. Association for Computational Linguistics.

A Appendix

A.1 Retriever Pretrain For Entity Linking

We train the E5_{base} encoder Retriever on BLINK (Wu et al., 2020) before finetuning it on AIDA. We split each document d in overlapping windows q of $W = 32$ words with a stride $S = 16$. To reduce the computational requirements, we (1) random subsample 1 million windows from the entire BLINK dataset, and (2) we retrieve hard negatives at each 10% of an epoch.We employ the same strategy as in Section 4.1.4 to construct the entity index, namely, we utilize KILT (Petroni et al., 2021) as our knowledge base, and we construct the textual representation of each entity within KILT by concatenating the Wikipedia title and opening text. We optimize the NCE loss (1) with 400 negatives per batch. At each hard-negatives retrieval step we mine 15 hard negatives per sample in the batch with a probability of 0.2 among the highest-scoring incorrect entities retrieved by the model. We train the encoder for a maximum of 110,000 steps using RAdam (Liu et al., 2020a) with a learning rate of 1e-5 and a linear learning rate decay schedule.

A.2 Experimental Setup

A.2.1 Hyperparameters

Retriever We report in Table 3 the hyperparameters we used to train our Retriever for both Entity Linking and Relation Extraction.

Reader We report in Table 4 the hyperparameters we used to train our Reader for both Entity Linking and Relation Extraction.

A.2.2 Implementation Details

We implement our work in PyTorch (Paszke et al., 2019), using PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019) as the underlying framework. We use the pretrained models for E5 and DeBERTa-v3 from HuggingFace Transformers (Wolf et al., 2020).

A.2.3 Hardware

We train every model on a single NVIDIA RTX 4090 graphic card with 24GB of VRAM.

A.3 Additional Results for Entity Linking

Similarly to Table 1, we report in Table 5 the *InKB F1* score of ReLiK compared with other systems.

A.4 Efficiency

Efficiency is a crucial factor in the practical deployment of Information Extraction systems, as real-world applications often require rapid and scalable information extraction capabilities. ReLiK excels in this regard, outperforming previous systems in performance, memory requirements, and speed. Table 6 shows the training and inference speeds of ReLiK.

EL Until now, efficiency had been a clear bottleneck for most EL systems, which rendered them useless or highly expensive on real-world applications. Therefore the efficiency gains for EL were extensively discussed in the main body of the paper at Section 4.2.

RE On the RE side, the only system on-par in terms of speed and performance would be USM. Unfortunately, it is not openly available, limiting its utility for the broader research community and hindering our ability to asses its speed. In Section A.7 we discuss other of its shortcomings. Instead, Table 6 compares the current openly available RE system with the best performance on NYT, REBEL. As an autoregressive system, inference speeds are

Hyperparameter	Values		
	BLINK	EL	RE
Optimizer	RAdam	RAdam	RAdam
Learning Rate	1e-5	1e-5	1e-5
Weight Decay	0.01	0.01	0.01
Training Steps	110,000	5000	40,000
Patience	0	5	5
Query Batch Size	64	64	64
Max Query Length	64	64	64
Passage Batch Size	400	400	[24, 400]
Max Passage Length	64	64	64
Hard-Negative Probability	0.2	1.0	1.0

Table 3: Hyperparameter we used to train the Retriever for the Entity Linking Pretrain (BLINK), Entity Linking (EL), and Relation Extraction (RE).

Hyperparameter	Values			
	AIDA	NYT	CONLL04	REBEL
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning Rate	1e-5	2e-5	8e-5	2e-5
Layer LR Decay	0.9	–	–	–
Weight Decay	0.01	0.01	0.01	0.01
Training Steps	50000	750,000	1,000	600,000
Warmup	5000	75,000	0	10,000
Token Batch Size	2048	2048	4096	4096
Max Sequence Length	1024	1024	1024	1024
EL passages	100	–	–	25
RE passages	–	24	5	20

Table 4: Hyperparameter we used to train the Reader for Entity Linking (AIDA), Relation Extraction (NYT) and cIE (REBEL).

several orders of magnitude higher. ReLiK$_L$ outperforms it by more than 2 F1 points and it is still around 3x faster, while ReLiK$_L$, which still outperforms any previous system, takes only 10s, a 10x gain in terms of speed.

cIE ReLiK continues to shine in the domain of closed Information Extraction, where it outperforms existing systems in terms of efficiency and performance. Compared with two other leading systems, ReLiK$_S$ surpasses them in F1 score while significantly outpacing them in terms of speed. These systems rely on BART-large, making them several orders of magnitude slower. In Table 6 we report on GenIE as its inference and train time are reported, but it should be noted that both GenIE and KnowGL are roughly equivalent in terms of compute. Here, again, the speed gains are multiple orders of magnitude, from 40x with ReLiK$_S$ to 15x with ReLiK$_L$.

In conclusion, ReLiK redefines the efficiency landscape in Information Extraction. Its unified framework, reduced computational requirements, and speed make it a compelling choice for a wide range of IE applications. Whether used in research or practical applications, ReLiK empowers users to extract valuable information swiftly and efficiently from textual data, setting a new standard for IE system efficiency.

A.5 Ablations

A.5.1 Entity Linking

Retriever Table 7 presents the findings of our ablation study conducted on the Retriever using the validation set from AIDA. In the baseline configura-

	In-domain	Out-of-domain							Avgs	
Model	AIDA	MSNBC	Der	K50	R128	R500	O15	O16	Tot	OOD
Hoffart et al. (2011)	72.8	65.1	32.6	55.4	46.4	42.4	63.1	0.0	47.2	43.6
Steinmetz and Sack (2013)	42.3	30.9	26.5	46.8	18.1	20.5	46.2	46.4	34.7	33.6
Moro et al. (2014)	48.5	39.7	29.8	55.9	23.0	29.1	41.9	37.7	38.2	36.7
Kolitsas et al. (2018)	82.4	72.4	34.1	35.2	50.3	38.2	61.9	52.7	53.4	49.2
Broscheit (2019)	79.3	—	—	—	—	—	—	—	—	—
Martins et al. (2019)	81.9	—	—	—	—	—	—	—	—	—
van Hulst et al. (2020)	80.5	72.4	41.1	50.7	49.9	35.0	63.1	58.3	56.4	52.9
De Cao et al. (2021b)	83.7	73.7	54.1	60.7	46.7	40.3	56.1	50.0	58.2	55.0
De Cao et al. (2021a)	85.5	19.8	10.2	8.2	22.7	8.3	14.4	15.2	—	—
Zhang et al. (2022)	85.8	72.1	52.9	64.5	**54.1**	41.9	61.1	51.3	60.5	57.3
ReLiK$_B$	85.9	71.9	55.5	67.2	49.2	41.5	62.6	53.9	61.0	57.9
ReLiK$_L$	**86.5**	**74.2**	**56.6**	**73.9**	51.4	**43.0**	**66.1**	**55.4**	**63.4**	**60.5**

Table 5: Comparison systems' evaluation (*inKB Micro* F_1) on the *in-domain* AIDA test set and *out-of-domain* MSNBC (MSN), Derczynski (Der), KORE50 (K50), N3-Reuters-128 (R128), N3-RSS-500 (R500), OKE-15 (O15), and OKE-16 (O16) test sets. **Bold** indicates the best model and underline indicates the second best competitor.

	Train					
	Retriever	ReLiK$_S$	ReLiK$_B$	ReLiK$_L$	Previous SotA	GPU
AIDA (EL)	4 h	–	11h	36h	48 h	A100
NYT (RE)	2 h	14 h	21 h	48 h	34 h	3090
REBEL (cIE)	6 h	20 h	30 h	3 d	18.5 d	V100
	Inference					
AIDA (EL)	6 s	–	23s	100s	20 m	A100
NYT (RE)	2 s	8 s	14 s	28 s	105 s	4090
REBEL (cIE)	5 m	10 m	17 m	36 m	10 h	4090

Table 6: Training and inference times for ReLiK on a single NVIDIA RTX 4090 GPU. Retriever times are reported separately, as they are shared across Reader sizes. The total time for any model size X is Retriever + ReLiK$_X$. Results for previous SotA (State-of-the-Art) in the right side are taken from the best performing openly available systems trained on each dataset and task. Zhang et al. (2022, entQA) for AIDA, Huguet Cabot and Navigli (2021, REBEL) for NYT and Josifoski et al. (2022, GenIE) for REBEL. Inference times refer to the time needed to annotate the corresponding test split for each dataset.

Model Name	Recall@100	Recall@50
Baseline	81.9	71.6
+ Hard-Negatives	98.5	97.9
+ Document-level information	98.8	98.0
+ BLINK Pretrain	99.2	98.8

Table 7: Ablation for the Retriever module. Each line represents an additional change built upon the previous one.

	EL			RE				
K	100	50	20	24	16	12	8	4
ReLiK$_S$	—	—	—	94.4	94.5	94.5	94.5	94.2
Time	—	—	—	10 s	10 s	10 s	8 s	6 s
ReLiK$_B$	85.9	**86.1**	85.8	94.8	94.8	94.8	94.8	94.5
Time	23 s	14 s	6 s	14 s	14 s	12 s	10 s	9 s
ReLiK$_L$	**86.5**	**86.5**	86.1	95.0	**95.1**	95.0	95.0	94.8
Time	100 s	47 s	22 s	28 s	24 s	22 s	20 s	18 s

Table 8: Micro-F1 results and inference time on AIDA for EL and NYT for RE when we reduce the number of retrieved passages as input to the Reader. Times reported are just for the Reader, without the retrieval step. Notice that for $K = 24$, all relation types in NYT are part of the input.

tion, we initialize the model with E5$_{\texttt{base}}$ and train it by optimizing the loss (1) with a focus solely on in-batch negatives. The introduction of hard-negatives substantially improve recall rates. Additionally, document-level information proves beneficial to the Retriever, albeit particularly benefiting AIDA, where relevant information is concentrated in the first token. Furthermore, the pretraining on BLINK demonstrated significant impact, especially on Recall@50, suggesting that pretraining enhances the Retriever ability to efficiently rank the candidate entities.

Passages Trimming The Retriever serves as a way to limit the number of passages that we consider as input to the Reader. At train time, we set $K = 100$, which, as Table 7 just showed, has a high Recall@K. However, as the computational cost of the Transformer Encoder that serves as the Reader grows quadratically on the input length, the choice of K affects efficiency. Table 8 shows what happens when we reduce the number of passages at inference time. Surprisingly, performance is not affected; in some cases, it even improves while time is halved. This showcases the usefulness of the Retriever which while fast is still able to rank passages effectively.

A.5.2 Relation Extraction

No Retriever Our benchmarks for RE contain a small number of relation types (5 and 24). Therefore the Retriever component is not extrictly necessary when all types fit as part of the input. Still, we believe it is an important part of the RE pipeline, as it is more flexible and robust to cases outside of the benchmarks. For instance, in long-text RE where the input text is longer, there is a need to reduce the number of passages as input to the Reader. Or as the case with cIE with REBEL, when the relation type set is larger, the Retriever enables an unrestricted amount of relation types. Nevertheless, we assess the influence of the Retriever as a reranker for NYT and explore a version of ReLiK without a Retriever. To do so we train a version of our Reader where the relation types are shuffled (ie. without a Retriever step). We obtained a micro-F1 of 94.2 for ReLiK$_S$, which is just slightly worse. Given how fast the Retriever component is at inference time, this result showcases how even when not strictly needed, it does not hurt performance.

Passages Trimming The previous section seemed to indicate that for datasets with a small set of relation types there is no need of a Retrieval step and a standalone Reader would be enough. While this is certantly an option, the Retrieve step is still very fast and doesn't add much overhead computation. On the other hand, the Reader is considerably slower, as the input is larger with additional computation that adds to the overall computational time. For RE the Hadamard product step grows quadratically with the number of passages. Therefore we explore how it affects downstream performance to reduce the number of passages once the system is already trained. We want to find out 1) is performance affected 2) is it considerably faster to reduce the number of passages. As Table 8 shows, reducing the number of passages up to just 8 doesn't impact performance. In fact, we even obtained better results with just 16 passages instead of 24.

Entity Linking as an aid to Relation Extraction On the cIE setup where Entity Linking and Relation Extraction are performed by the same Reader, each task is performed sequentially and then RE predictions are conditioned on EL. But does EL aid RE? Or does having a shared Reader between both tasks impact RE negatively? Entity types were often included in Relation Classification to improve

System using BERT-base	P	R	F
(Sui et al., 2023)	92.5	92.2	92.3
(Zheng et al., 2021)	93.5	91.9	92.7
(Lou et al., 2023, USM$_{BERT-base}$)	**93.7**	91.9	92.8
ReLiK$_{BERT-base}$	93.2	**92.9**	**93.1**

Table 9: Results for systems using BERT-base on the NYT dataset.

the overall performance (Zhou and Chen, 2022). In our case, RE is conditioned on EL implicitly, without explicit ad-hoc information, i.e. just by leveraging the predictions of the EL component. We train ReLiK$_S$ on REBEL without EL, which performs solely RE under the same conditions and hyperparameters as the cIE counterpart. The system without EL obtained a micro-F1 of 75.4 with boundaries evaluation. On the other hand, the cIE approach that combines both EL and RE, we obtain 76.0 micro-F1[8], which considering the size of the test set (175K sentences) is a considerable difference. This is an exciting result as it validates end-to-end approaches for cIE where both tasks are combined.

BERT-base Our Reader is based on DeBERTa-v3, while previous RE systems may be based on older models. To enable a fair comparison and assess the flexibility of our RR approach, we train our Reader on NYT using BERT-base and compare with other systems. Table 9 shows how ReLiK$_{BERT-base}$ outperforms previous approaches, including USM.

A.6 Error Analysis

Entity Linking Figure 2 shows an example of the predictions generated by our system when trained on EL. This particular example showcases a common error when evaluating the AIDA dataset. AIDA was manually annotated in 2011 on top of a Named Entity Recognition 2003 dataset (Tjong Kim Sang and De Meulder, 2003). While widely used as the de-facto EL dataset, it contains errors and inconsistencies. A common one is the original entity spans not being linked to any entity in the KB. This could either be because at the time such an entity was not present in the KB, or an annotation error due to the complexity of the task. This leads to NME annotations which at evaluation time are considered false positives, as our system links to the correct entity, such as *Bill Brett* in the example. Another source of errors is document slicing in windows. While necessary to overcome the length constraints of our Encoder, it can lead to inconsistent or incomplete predictions. For instance, *ILO* was linked to an entity in a window that did not see further context (*Workers Group*), while the next window correctly identified *ILO Workers Group* as an NME.

Relation Extraction The example shown in Figure 2 is a common error found in predictions on NYT by ReLiK. Due to the semiautomatic nature of NYT annotations, some relations, such as the ones shown in the example, lack the proper context to ensure consistency at inference time. In this case, the system predicts a relation (*place_lived*) which cannot really be inferred from the text or is ambiguous at best. We believe this is due to certain biases introduced at training time. This can be exemplified by the false negative, annotated as correct (*place_of_birth*), which is impossible to infer from the sentence.

closed Information Extraction Finally, the last example in Figure 2 shows a prediction by our model when trained on both tasks simultaneously with the REBEL dataset. Notice the missing prediction (*participant*), and the false positives. While the passages retrieved contained all the necessary relation types, the system still failed to recover one of the gold triplets, even if all the spans were correctly identified. Then, for the two false positives, while they were not annotated in the dataset, probably due to its automatic annotation, they are correct, and ReLiK predicted them even if, at evaluation time, this will decrease the reported performances.

A.7 USM

In this section, we want to discuss in detail how ReLiK compares with USM. USM is the current state-of-the-art for RE and was the first modern RE system that jointly encoded the input text with the relation types, breaking from ad-hoc classifiers with weak transfer capabilities or autoregressive approaches that leverage its large language head but are inefficient. Therefore, it shares a similar strategy to our RE component, in that both rely on the relation types being part of the input, and the core idea is to link mention spans to their corresponding triplet. However, this is where the similarities end. In USM, the probabilities of a mention span being linked to a triplet (i.e. to another entity

[8]This value differs from the one reported in Table 2 since it is evaluated without entity disambiguation

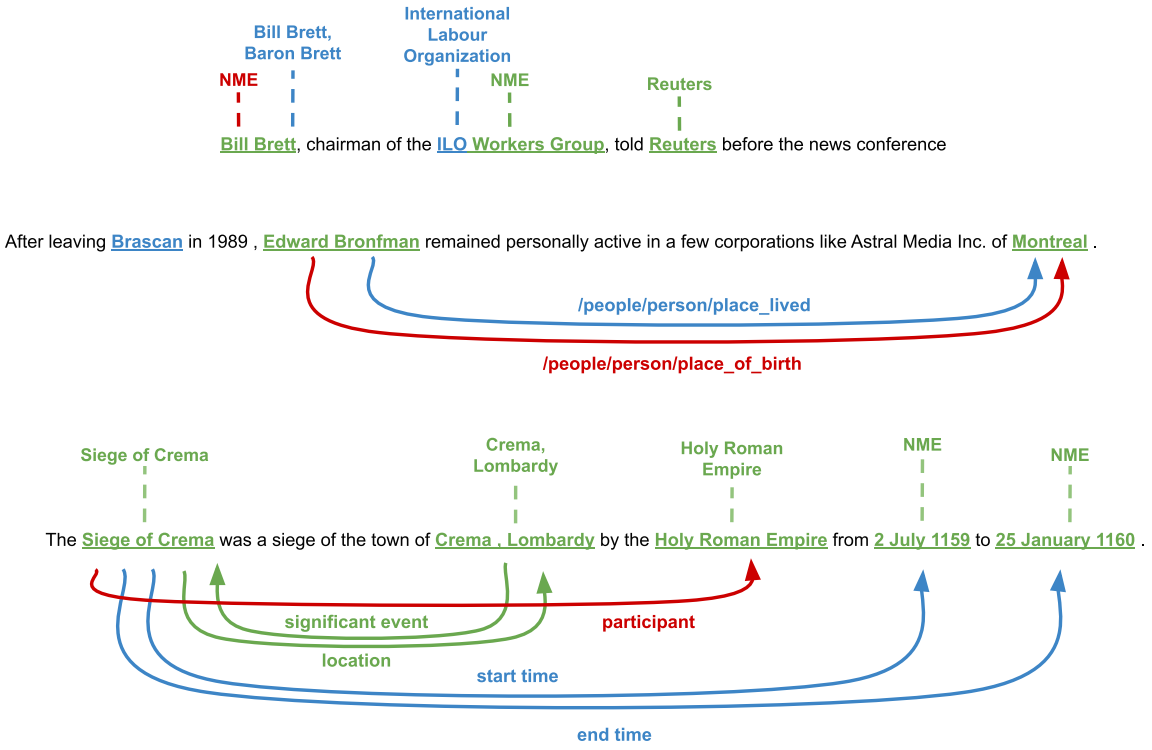

Figure 2: Example predictions by ReLiK$_L$ on AIDA (top), NYT (middle), and REBEL (bottom) for EL, RE, and cIE respectively. Green stands for true positive, blue for false positive, and red for false negative.

and a relation type) are assumed to be independent and factorized such that they are computed separately, in a pairwise fashion. Mentions are linked as subjects to the spans that share a triplet (blue lines in Figure 3) and to the relation type label (green lines). Finally, labels are linked to the object entity (red lines). In most cases, these are sufficient to decode each triplet but we want to point out a shortcoming of this strategy. The decoding is done by pairs. First mention-mention, i.e. in Figure 3 (Jack, Malaga), (Jack, New York), (John, Malaga) and (John, New York); then label-mention (birth place, Malaga), (birth place, New York), (live in, Malaga) and (live in, New York); and finally mention-label (Jack, birth place), (Jack, live in), (John, birth place), (John, live in). At this point, the issue should be clear. From this set of pairs, one cannot retrieve the correct triplets, even though the model would have not made any mistake in its predictions. It is worth pointing out that these phenomena do not happen on either test set for NYT or CONLL04, therefore it doesn't affect reported performance.

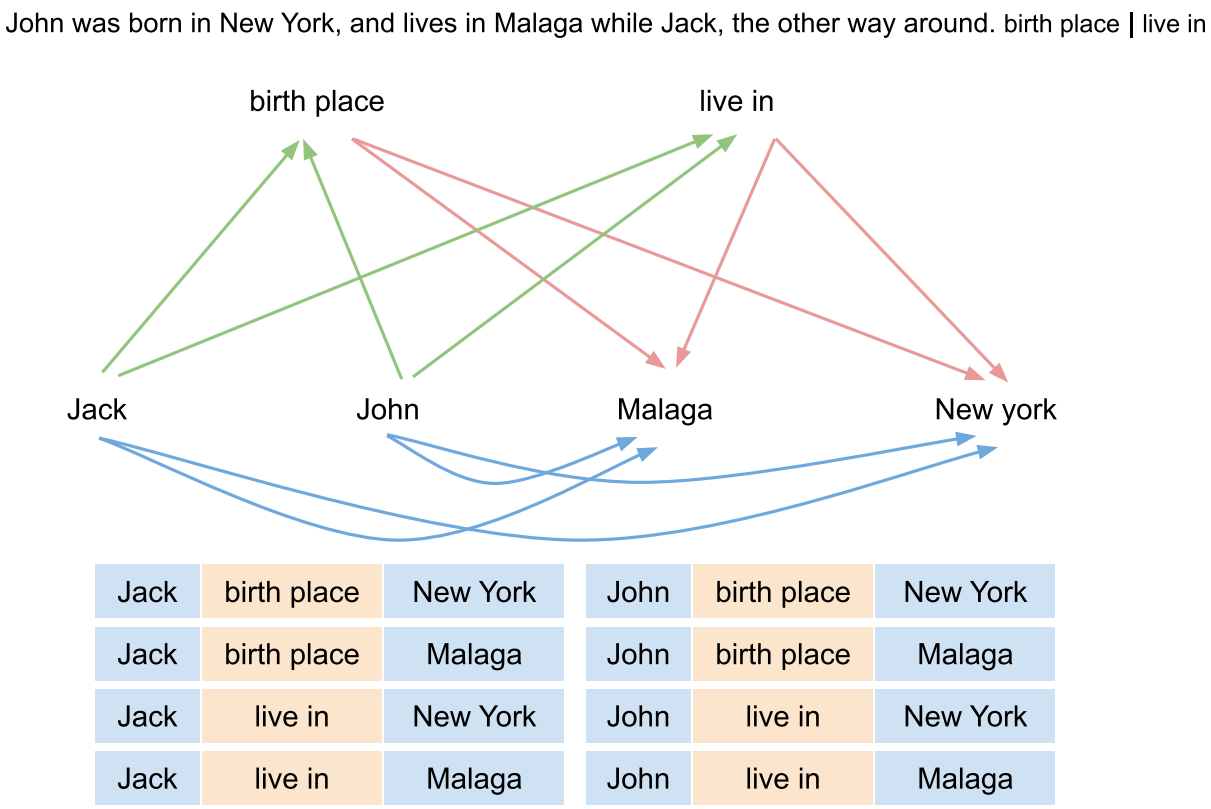

Figure 3: Example of a sentence as input to USM where their token-linking strategy would fail even if the model made the right predictions.