# REBL: Entity Linking at Scale

Chris Kamphuis[1], Faegheh Hasibi[1], Jimmy Lin[2] and Arjen P. de Vries[1]

[1]Radboud University, Nijmegen, The Netherlands

[2]University of Waterloo, Waterloo, Canada

**Abstract**

REBL is an extension of the Radboud Entity Linker (REL) for Batch Entity Linking. REBL is developed after encountering unforeseen issues when trying to link the large MS MARCO v2 web document collection with REL. In this paper we discuss the issues we ran into and our solutions to mitigate them. REBL makes it easier to isolate the GPU heavy operations from the CPU heavy operations, by separating the mention detection stage from the candidate selection and entity disambiguation stages. By improving the entity disambiguation module we were able to lower the time needed for linking documents by an order of magnitude. The code for REBL is publicly available on GitHub.

## 1. Introduction

Entity linking concerns the task of automatically identifying entity mentions in the text and linking them to the corresponding entities in a knowledge-base (KB). It fulfils a key role in knowledge-grounded understanding of text and has been proven effective for diverse tasks in information retrieval [1, 2, 3, 4, 5, 6, 7], natural language processing [8, 9], and recommendation [10]. Utilizing entity annotations in these downstream tasks depends upon the annotation of text corpora with a method for entity linking. Due to the complexity of entity linking systems, this process is often performed by a third-party entity linking toolkit, examples including DBpedia Spotlight [11], TAGME [12], Nordlys [13], GENRE [14], and REL [15].

A caveat in existing entity linking toolkits is that they have not been designed for batch processing large numbers of documents. Existing entity linking toolkits are primarily optimised to annotate individual documents, one at a time. This severely restricts utilization of state-of-the-art entity linking tools such as REL and GENRE, that employ neural approaches and require GPUs for fast operation. Annotating millions of documents incurs significant computational overhead, to the extent that annotation of a large text corpus becomes practically infeasible

*DESIRES 2022 – 3rd International Conference on Design of Experimental Search & Information REtrieval Systems, 30-31 August 2022, San Jose, CA, USA*

chris@cs.ru.nl (C. Kamphuis); f.hasibi@cs.ru.nl (F. Hasibi); jimmylin@uwaterloo.ca (J. Lin); arjen@cs.ru.nl (A. P. de Vries)

http://chriskamphuis.com (C. Kamphuis); https://hasibi.com (F. Hasibi); https://cs.uwaterloo.ca/~jimmylin/ (J. Lin); https://cs.ru.nl/~arjen (A. P. de Vries)



CEUR Workshop Proceedings (CEUR-WS.org)

using modest computational power resources. Batch entity linking is however necessary to build today's data-hungry machine learning models, considering large text corpora like the new MS MARCO v2 (12M Web documents) [16].

This paper describes our experience with optimizing the Radboud Entity Linking (REL) toolkit for batch processing large corpora. REL annotates individual documents efficiently, requiring only modest computational resources, while performing competitively when compared to the state-of-the-art methods on effectiveness. It considers entity linking as a modular problem consisting of three stages:

**(i) Mention detection.** The goal of this step is to identify all possible text spans in a document that might refer to an entity. If a text span that refers to an entity is not identified properly in this stage, the system will not be able to correctly link the entity in later stages.

**(ii) Candidate selection.** For every detected mention, REL considers up to $k_1+k_2(=7)$ candidate entities. $k_1(=4)$ candidate entities are selected based on their prior occurrence probability $p(e|m)$ (for entity $e$ given mention $m$). These priors are pre-calculated from Wikipedia hyperlinks and the CrossWiki [17] corpus. The other $k_2(=3)$ entities are chosen based on the similarity of their embeddings to the contextual embedding of the mention (considering a context of maximum 200 word tokens).

**(iii) Entity disambiguation.** The goal of this final step is to map the mention to the correct entity in a knowledge base. The candidate entities for each mention are obtained from the previous stage and REL implements the Ment-norm method proposed by Le and Titov [18].

This paper explains the challenges of batch processing in REL and presents the approaches we found to overcome these challenges. We show that our updated REL toolkit, REBL, improves REL efficiency 9.5 times, decreasing the processing time per document (excluding mention detection) on a sample of 5000 MS MARCO documents from 1.23 seconds to 0.13 seconds. We demonstrate that REBL enables the annotation of a large corpus like MS MARCO v2, given modest computational resources. We discuss potential improvements that can be made in order to further improve efficiency of batch entity linking. The REBL code and toolkit are available publicly at https://github.com/informagi/REBL.

## 2. From REL to REBL

The objective that led to this paper was to link the MS MARCO v2 collection [16]. This collection contains 11,959,635 documents split into 60 compressed files, totalling roughly 33GB in size. Decompressed, these files are in JSON line format (where every line represents a JSON document). Documents have five fields: *url*, *title*, *headings*, *body*, and *docid*. For our experiments we wanted to link the title, headings, and body of the documents. We use the 2019-07 Wikipedia dump to link to, which is one of the two dumps REL was initially developed on. It is, however, straightforward to take another dump of Wikipedia and develop another REL instance.

In order to ease linking this size of data, we separated the GPU heavy mention detection stage from the CPU heavy candidate selection and entity disambiguation stages; the modified code

can be found on GitHub.[1] The inputs for mention detection are the compressed MS MARCO v2 document files, and its output consists of the mentions found and their location in the document, in Apache Parquet format.[2] These files together with the source text are the input for the subsequent phases (candidate selection and entity disambiguation). The final output consists of Parquet files containing spans of text and their linked entities. In the following, we discuss what is changed for mention detection, candidate selection, and entity disambiguation steps to make REL more suited to link the MS MARCO v2 collection.

## 2.1. Mention Detection

REL [15] uses Flair [19] for mention detection, a state-of-the-art named entity recognition system. Flair uses the `segtok`[3] package to segment an (Indo-European) document in sentences, internally represented as `Sentence` objects. These sentences are split into words / symbols represented as `Token` objects. When creating these representations however, it is not possible to recreate the source text properly, as Flair removes multiple whitespace characters when occurring after each other. REL corrects for this to preserve the correct span data with regard to its location in the source text, which is an inefficient process. We set out to construct the underlying data structures ourselves for REBL. To do this, we used the `syntok`[4] package, a follow-up version of `segtok`. The author of both packages claims that the `syntok` package segments sentences better than `segtok`.

When constructing the sentences from the token objects, we ran into another issue originated from data handling procedure in Flair: Flair removes various zero width Unicode characters from the source text: zero width space (`U+200B`), zero width non-joiner (`U+200C`), variation selector-16 (`U+FE0F0`), and zero-width no-break space (`U+FEFF`). These characters occur rarely, but in a collection as big and diverse as MS MARCO v2, these characters are found in some documents. When encountering these characters, the token objects were constructed such that the span and offset of the token still referred to that of the source text.

For the case of the zero width space, we updated the `syntok` package; although zero width space is not considered a whitespace character according to the Unicode standard, it should be considered a character that separates two words. For the other Unicode characters removed by Flair, we manually update the span in the `Token` objects created by Flair such that they refer correctly to the positions in the source text. Now, when Flair identifies a series of tokens as a possible mention, we can directly identify the location in the source text from the `Token` objects.

Flair supports named entity recognition in batches; this way multiple batches of text can be sent to the GPU for faster inference time. Because REL had been designed to tag one document at a time, it did not use this functionality. REBL exploits this feature, allowing the user to specify the number of documents to be tagged simultaneously.

[1] https://github.com/informagi/REBL
[2] https://github.com/apache/parquet-format
[3] https://github.com/fnl/segtok
[4] https://github.com/fnl/syntok

### 2.2. Candidate Selection and Entity Disambiguation

REL makes use of a $p(e|m)$ prior, where $e$ is an entity, and $m$ is a mention. These priors are saved in a (SQLite) database, and up to 100 priors per mention are considered. Data conversion between client and the representation stored in the database incurred however a large serialization cost. We updated this to a format that is faster to load, with the additional benefit of a considerably decreased database size.[5] We experimented with data storage in the DuckDB column oriented database as an alternative, but found that SQLite was (still) more efficient as key-value store, at least in DuckDB's current state of development.

We found that the entity disambiguation stage took much longer than reported in the original REL paper. This difference is explained by the length of the documents to be linked. The documents evaluated by van Hulst et al. [15] were on average 323 tokens long with an average of 42 mentions to consider. The number of tokens in an MS MARCO v2 document is on average 1800, with 84 possible mentions per document.[6] Per mention, 100 tokens to the left, and 100 tokens to the right are considered as the context for the disambiguation model. The larger documents result in a larger memory consumption per context and per document, with higher processing costs as a consequence.

We improve the efficiency of the entity disambiguation step such that it could be run in a manageable time. REL recreates database cursors for every transaction. We rewrote the REL database code such that one database cursor is created for the entity disambiguation module. Within a document, the same queries were issued to the database multiple times. This happens for example when a mention occurs multiple times within a document. By caching the output of these queries, we were able to significantly lower the number of database calls needed. We cached all database calls per every segment in the collection, as we ran the process for every segment separately.

The default setting of REL is to keep embeddings on the GPU after they are loaded. This, however, slowed down disambiguation when many documents are being processed consecutively, because operations like normalization were carried out over all embeddings on the GPU. By clearing these embeddings as soon as a document is processed, a significant speed up has been achieved.

Finally, after retrieving the embeddings from the database, REL puts them in a python list. We rewrote the REL code such that the binary data is directly loaded from NumPy, a data format that Pytorch operates on.

## 3. Effects on Execution

In the mention detection stage, we improved tokenization and applied batching. In the MS MARCO v2 collection, 411,906 documents have tokens that were automatically removed by Flair, which are 3.4% of all documents in the collection. The MS MARCO v1 collection does not have documents that contained these characters; the documents in that version of the

[5] The table that represents the priors shrank from 9.6GB to 2.2GB.

[6] These figures are calculated over the body field; we also tagged the shorter title and headers fields.

collection are (probably) sanitized before publishing. Batching documents in the mention detection stage decreased the average time for finding all named entities. We used batches of size 10, as the documents are relatively large. The optimal batch size will depend on the available GPU memory.

A few documents in the MS MARCO v2 collection could not be linked. This happened only in extraordinary cases, where linking with entities did not make sense in the first place; an example being a document consisting of numbers only.[7] Here, the `syntok` package created one long `Sentence` object from this file that could not fit in GPU memory.

Table 1 shows the improvements we made to the candidate selection and entity disambiguation step, and describes how much time is saved in REBL. The code improvements to create the database cursor only once and to load the data directly from NumPy had no noticeable effect on the overall run time of entity disambiguation and are not reported in this table. Note that the large standard deviations are primarily due to the differences in processing costs between long and short documents.

**Table 1**
Efficiency improvements for Candidate Selection and Entity Disambiguation. Improvements are calculated over a sample of 5000 documents using a machine with an Intel Xeon Silver 4214 CPU @ 2.20GHz using 2 cores, that has 187GB RAM memory, and a GeForce RTX 2080 Ti (11GB) GPU. Improvements are cumulative; the times shown include the previous improvement as well.

| Improvement | Seconds | Explanation |
|---|---|---|
| Old Candidate Selection + Entity Disambiguation | $1.23 \pm 2.09$ | Average time it takes to select candidates and disambiguate per document |
| No embedding reset | $0.26 \pm 1.60$ | The default setting of REL was to keep embeddings in GPU memory after they were loaded, by clearing them from GPU memory after every document a speed up was achieved. |
| Cache database calls | $0.15 \pm 1.31$ | When an entity occurs within a document, there is a high probability of it occurring multiple times. By caching the calls, we increase the memory usage but are able to lower the time needed for candidate selection + entity disambiguation. |
| Representation change candidates | $0.13 \pm 1.19$ | By representing the candidates better in the database, we were able to save on conversion time lowering the time needed for candidate selection. |

## 4. Conclusion and Future work

We introduced REBL, an extension for the Radboud Entity Linker. We utilize REL's modular design to separate the GPU heavy mention detection stage from the CPU heavy candidate selection and entity disambiguation stages, as many researchers have dedicated GPU and CPU

[7] The source document was a price list in PDF format.

machines. The mention detection module has been made more robust and reliable, using a better segmenter and preserving location metadata correctly. The candidate selection step and entity disambiguation step were updated to improve their runtime, especially for longer documents.

Although it is now possible to run REL [15] on MS MARCO v2 [16] in a (for us) somewhat reasonable time, we identified further improvements to implement, that we work on actively.

Found mentions are compared to all other mentions during the candidate selection step, the complexity of this step is $O(n^2)$, with $n$ being the number of mentions found in a document, which is especially problematic for longer documents. As we are only interested in mentions that are similar, we expect that it might be worthwhile to implement a locality sensitive hashing algorithm to decrease the number of comparisons needed in this stage. However, we would need to run additional experiments to ensure the effectiveness of the model does not suffer.

REBL now implements a two step approach that writes intermediate results to the file system in Parquet format. A streaming variant would be preferable. We have also kept SQLite as database backend, but will consider specialized key-value stores to speed up candidate selection and entity disambiguation. We will revisit DuckDB upon progress in the implementation of zero-cost positional joins.

The candidate selection stage considers the context of a mention. This context has to be constructed from the source document. As a result, we load the source data a second time during candidate selection. Alternatively, we may output mention context in the mention detection stage, which could then speed up the remaining. However, this would significantly increase the size of the mention detection output. More experiments are needed to strike the right balance here.

Overall, it has become clear that a data processing oriented perspective on entity linking is necessary for efficient solutions. Having made explicit quite a few implicit design choices, re-evaluating these might lead to more effective entity linking as well. The last word on entity linking at size has not been written!

## Acknowledgments

This work is part of the research program Commit2Data with project number 628.011.001 (SQIREL-GRAPHS), which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

## References

[1] E. J. Gerritse, F. Hasibi, A. P. de Vries, Entity-Aware Transformers for Entity Search, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, 2022, pp. 1455–1465.

[2] E. J. Gerritse, F. Hasibi, A. P. de Vries, Graph-Embedding Empowered Entity Retrieval, in: Proceedings of the 42nd European Conference on Information Retrieval, 2020, pp. 97–110.
[3] C. Xiong, J. Callan, T.-Y. Liu, Word-Entity Duet Representations for Document Ranking, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, 2017, p. 763–772.
[4] F. Hasibi, K. Balog, S. E. Bratsberg, Exploiting Entity Linking in Queries for Entity Retrieval, in: Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval, ICTIR '16, 2016, p. 209–218.
[5] K. Balog, H. Ramampiaro, N. Takhirov, K. Nørvåg, Multi-Step Classification Approaches to Cumulative Citation Recommendation, in: Proceedings of the 10th Conference on Open Research Areas in Information Retrieval, OAIR '13, 2013, p. 121–128.
[6] R. Reinanda, E. Meij, M. de Rijke, Mining, Ranking and Recommending Entity Aspects, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, 2015, p. 263–272.
[7] S. Chatterjee, L. Dietz, BERT-ER: Query-specific BERT Entity Representations for Entity Ranking, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, 2022, p. 1466–1477.
[8] T. Lin, Mausam, O. Etzioni, Entity linking at web scale, in: Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX), 2012, pp. 84–88.
[9] D. Ferrucci, Introduction to "This is Watson", IBM Journal of Research and Development 56 (2012) 1:1–1:15. doi:10.1147/JRD.2012.2184356.
[10] Y. Yang, O. Irsoy, K. S. Rahman, Collective Entity Disambiguation with Structured Gradient Tree Boosting, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 777–786.
[11] P. N. Mendes, M. Jakob, A. García-Silva, C. Bizer, DBpedia Spotlight: Shedding Light on the Web of Documents, in: Proceedings of the 7th International Conference on Semantic Systems, I-Semantics '11, 2011, p. 1–8.
[12] P. Ferragina, U. Scaiella, TAGME: On-the-Fly Annotation of Short Text Fragments (by Wikipedia Entities), in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, 2010, p. 1625–1628.
[13] F. Hasibi, K. Balog, D. Garigliotti, S. Zhang, Nordlys: A Toolkit for Entity-Oriented and Semantic Search, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, 2017, p. 1289–1292.
[14] N. D. Cao, G. Izacard, S. Riedel, F. Petroni, Autoregressive Entity Retrieval, in: International Conference on Learning Representations, 2021.
[15] 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: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, p. 2197–2200.
[16] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, B. M. Andrew McNamara, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, T. Wang, MS MARCO: A Human Generated MAchine Reading COmprehension Dataset, in: InCoCo@NIPS, 2016.
[17] V. I. Spitkovsky, A. X. Chang, A Cross-Lingual Dictionary for English Wikipedia Con-

cepts, in: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12), European Language Resources Association (ELRA), 2012, pp. 3168–3175.

[18] P. Le, I. Titov, Improving Entity Linking by Modeling Latent Relations between Mentions, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 1595–1604.

[19] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, R. Vollgraf, FLAIR: An easy-to-use framework for state-of-the-art NLP, in: Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), 2019, pp. 54–59.