Assisted Knowledge Graph Authoring: Human-Supervised Knowledge Graph Construction from Natural Language

Marcel Gohsen
Bauhaus-Universität Weimar
Weimar, Germany

Benno Stein
Bauhaus-Universität Weimar
Weimar, Germany

ABSTRACT

Encyclopedic knowledge graphs, such as Wikidata, host an extensive repository of millions of knowledge statements. However, domain-specific knowledge from fields such as history, physics, or medicine is significantly underrepresented in those graphs. Although few domain-specific knowledge graphs exist (e.g., Pubmed for medicine), developing specialized retrieval applications for many domains still requires constructing knowledge graphs from scratch. To facilitate knowledge graph construction, we introduce WAKA: a Web application that allows domain experts to create knowledge graphs through the medium with which they are most familiar: natural language.

CCS CONCEPTS

• **Computing methodologies** → **Semantic networks**; **Information extraction**; • **Information systems** → *Information integration.*

KEYWORDS

Knowledge Graph Construction, Semantic Web, Information Extraction

ACM Reference Format:
Marcel Gohsen and Benno Stein. 2024. Assisted Knowledge Graph Authoring: Human-Supervised Knowledge Graph Construction from Natural Language. In *Proceedings of the 2024 ACM SIGIR Conference on Human Information Interaction and Retrieval (CHIIR '24), March 10–14, 2024, Sheffield, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3627508.3638340

1 MOTIVATION AND BACKGROUND

Knowledge graphs are a way to encode and store real world knowledge by specifying semantic relationships (edges) between entities (nodes). These datastructures have been used effectively for various information retrieval applications like question answering [5], recommendation systems [17], or search engines [10]. Knowledge graphs are also popular for domain-specific applications in areas such as cybersecurity, education, finance, medicine, or news [20], which typically require domain-specific ontologies or knowledge.

CHIIR '24, March 10–14, 2024, Sheffield, United Kingdom

ACM ISBN 979-8-4007-0434-5/24/03.
https://doi.org/10.1145/3627508.3638340

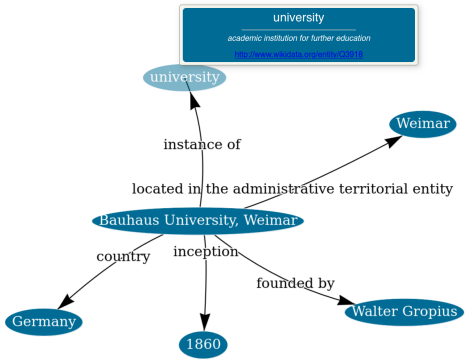

Figure 1: Visualization of a knowledge graph in the WAKA frontend in which the entity `university` is highlighted.

Therefore, besides encyclopedic (e.g., Wikidata[1]) and commonsense (e.g., OpenCyc[2]) knowledge graphs, there are also domain-specific graphs, for example for medicine (e.g., Pubmed [3]) or linguistics (e.g., WordNet[4]). However, domain-specific knowledge graphs exist only for few domains. The remaining domains require a considerable amount of effort to create knowledge graphs in order to use them in domain-specific information systems.

The manual creation of knowledge graphs and associated ontologies from scratch is a complex process and usually requires much expertise and effort [1]. To tackle this, automatic methods for constructing knowledge graphs from unstructured text have been realized using open information extraction [12], a mixture of information extraction pipelines [8] or recently large language models [4]. Since there was no sufficient ground truth during the development of mentioned methods, the quantitative evaluations of these methods have been limited to components of the construction algorithm (e.g. relation extraction, entity linking), but have never been extended to the entire construction algorithm. Qualitative evaluations by Martinez-Rodriguez et al. [12] and Kertkeidkachorn and Ichise [8] have shown a precision of the resulting triples of 50%, which underlines the need for human intervention in the construction process. Unfortunately, the above approaches do not provide the source code for the above knowledge graph construction approaches and therefore cannot be included in our experiments.

With this paper, we introduce WAKA: a convenient Web application to construct and author knowledge graphs from unstructered

[1] https://www.wikidata.org/
[2] https://sourceforge.net/projects/opencyc/
[3] https://pubmed.ncbi.nlm.nih.gov/
[4] https://wordnet.princeton.edu/

text.[5] WAKA takes advantage of the high-quality ontology and the large number of entities in Wikidata by linking entities and relations to corresponding entries in the knowledge base but also allows adding new entities and relations. The application offers an intuitive interface that allows to supervise, correct, extend, and visualize the automatically proposed knowledge graph. Finally, the authored knowledge graph can be stored in a standard data model for Semantic Web—the Resource Description Framework[6] (RDF).

2 AUTHORING INTERFACE

The interface of WAKA consists of two main components: a text editor and an interactive graph visualization (see Figure 1). A user may write or import a text into the editor from which knowledge is to be derived. Based on the text in the editor, pressing a button triggers the automatic construction of a knowledge graph (see Section 3), which can then be corrected and extended by the user.

The proposed graph is shown to the user in the visualization and the corresponding entities are annotated in the text editor. For each of the entities that participate in a relation, all the mentions in the text are annotated. Since relations are not always explicitly mentioned but can be inferred from the text, we do not annotate relations in the editor. The editor and the visualization represent linked data views. Hovering over a node in the graph highlights the node and all corresponding annotations and vice versa. Since the resulting graphs can get quite large and the visualization may get cluttered, WAKA supports navigation and graph manipulation techniques such as zooming, camera movement or node dragging.

A user can use several interactions to validate and correct the proposed knowledge graph via WAKA's interface. Hovering over an entity annotation or over a graph node reveals a label, a description, and a link to the corresponding Wikidata entity in a tooltip. Hovering over edges in the knowledge graph, displays the label, description, and Wikidata link of the associated Wikidata property.

To correct a wrong link to a Wikidata entity or property, a user can click either on one of the corresponding annotations or on the node in the visualization. An overlay menu will be opened which shows the label and description of the currently linked entity or property. Additionally, a list of other proposed entities or properties is provided that originate from giving the mention span to the entity retrieval pipeline described in Section 3.1. Clicking on any of the proposals will establish the connection of the entity mention or relation to the Wikidata entry and the annotations and visualization will be updated accordingly. If the correct entity or property is not part of the proposals, a search box allows to freely enter a query to retrieve the correct Wikidata entry from the entity retrieval or relation retrieval pipelines, respectively. For the deletion of entities or relations a button is provided in the overlay menu.

Entities can be added to the graph by highlighting a text span in the editor. When a text span is highlighted a button appears that opens the overlay menu to let the user select an entity from Wikidata (or leave it unlinked if the entity does not exist yet). Analogously, relationships can be added by selecting two entities and pressing a button or connecting two nodes in the visualization.

[5]Demo video: https://youtu.be/CwkW3Xwb5zg
Code: https://github.com/webis-de/waka
[6]https://www.w3.org/RDF/

The user can download the resulting knowledge graph by clicking on a download button in the interface. It is planned to make the download format configurable. However, WAKA currently supports to download the authored knowledge graph in N-Triples format (i.e., a plain text serialisation for RDF triples).

The knowledge graph visualization has been realized with vis-network.[7] All other interface components are implemented in native Javascript without any required dependencies.

3 KNOWLEDGE GRAPH CONSTRUCTION

Our approach to automatically construct knowledge graphs from unstructered text consists of three main components: an entity discovery pipeline, a relation extraction pipeline, and final knowledge fusion and evaluation steps. Figure 2 gives an overview of the steps in the automatic knowledge graph construction algorithm. For efficiency reasons, the entity discovery and relation extraction pipelines are executed in parallel.

3.1 Entity Discovery

The entity discovery pipeline aims to detect all named entities in the text, link them to entities in Wikidata and rank them according to their relevance. We calculate the relevance of an entity based on the findings of Kasturia et al. [7], who have found that the relevance of an entity depends equally on how well it fits a mention and a context. The set of discovered named entities forms the pool from which subjects and objects are drawn in the knowledge fusion step, rendering a high recall preferable to high precision.

Named Entity Recognition. The first step in the entity discovery pipeline is to recognize named entities in the given text along with their mention spans and entity types. In addition, we extract concepts as noun phrases that were not recognized as named entities. We use established named entity recognition methods that offer a good tradeoff between effectiveness and efficiency.

According to a comparative study of open named entity recognition frameworks [15], the Stanford NLP Toolkit [11] achieved the highest recall on the CoNLL 2003 dataset [14]. The second-highest recall was achieved by SpaCy from which we use the en_core_web_sm pipeline. Both models take features at sentence-level into account, and thus we employ the FLERT model [16] through the Flair framework [2] which considers document-level features for the named entity recognition. From all three frameworks, we employ model versions that were trained on OntoNotes 5 [18], and thus classifies found entities into an 18-class ontology.

Based on the determined type of recognized entity, an entity must be either linked to an entry in the knowledge base or represents a literal. We distinguish between numeric literals, consisting of `PERCENT`, `MONEY`, `QUANTITY`, `CARDINAL`, and `ORDINAL` and temporal types, such as `DATE` and `TIME` from the OntoNotes ontology. Entities of the remaining entity types are linked to an entity in Wikidata.

Entity Retrieval. To leave the entity disambiguation to the user, we model entity linking as a retrieval task. We build an Elasticsearch index of entities from Wikidata (i.e., subjects and objects of RDF statements). To keep Wikidata meta-information and irrelevant entities out of the index, we define the following filtering rules:

[7]https://visjs.github.io/vis-network/docs/network/

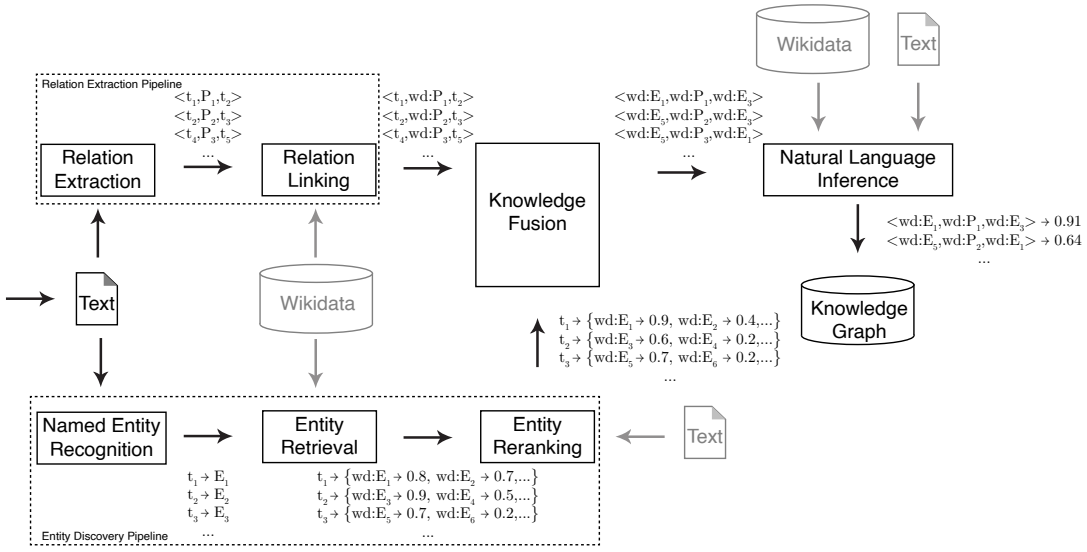

Figure 2: Architecture of the automatic knowledge graph construction approach.

(1) An entity has a valid URI.
(2) An entity has at least one property (i.e., outoing edge).
(3) An entity is not a Wikimedia category.
(4) An entity is not a Wikimedia disambiguation page.

Following this rule set, we have collected about 109 million entities from Wikidata which is well within range of Wikidatas self-reported number of content pages of 107 million.[8] For each entity, we store its URI, its label (i.e., the entity name displayed on its Wikidata page), its description, and a concatenation of the label, description, and any available alternate names in English as a search key. To approximate the 'commonness' of an entity for the retrieval ranking, we store the number of incoming edges for each entity.

Each text span that represents an entity according to the named entity recognition models is queried against our entity index and ranked by BM25. However, we take into account that direct matches of the entity label are more relevant than a match of the description or alternate names. We compute the relevance score of an entity e according to an entity mention t as the maximum of BM25 according to the label scaled by a parameter α with $\alpha > 1$ and the (unscaled) BM25 according to the search key of an entity.

$$rel_t(e) = max\left(\alpha \cdot BM25\left(t, label\left(e\right)\right), BM25\left(t, key\left(e\right)\right)\right) \quad (1)$$

Pilot experiments have shown, that $\alpha = 3$ yielded best results. In addition to an entities' relevance, we take its commonness into account. We scale the relevance of an entity by its commonness $comm(e)$. To tune down the weight of the commonness, which can reach into the millions, we take the common logarithm and add one to account for cases where the commonness is zero.

$$score_t(e) = rel_t(e) \cdot log(comm(e) + 1) \quad (2)$$

We retrieve at most 20 entities per mention and define a minimum score threshold of at least 20 for considered entities.

[8] https://www.wikidata.org/wiki/Special:Statistics

Entity Reranking. The retrieval score of an entity approximates how well an entity fits to a span of a text. However, the context in which an entity is mentioned does not influence the retrieval. To remedy this, we perform a reranking that evaluates how well an entity fits semantically with the sentence it is mentioned in.

To calculate the semantic similarity between an entity and a sentence from the text, we construct a short descriptive sentence for the entity using the following template *"{label} is a {description}"*. For example, if we follow the template for the named entity `Germany`, we obtain the descriptive sentence *"Germany is a country in Central Europe"*. We embed the descriptive sentence and the sentence from the text with DistilRoBERTa used through the Sentence Transformer framework [13]. This model has been chosen since it provides semantically representative embeddings in a reasonable inference time. The semantic similarity is calculated by the cosine similarity between the embedding vectors and is then multiplied by the normalized retrieval score.

3.2 Relation Extraction and Linking

Relation extraction is the task of detecting semantic relations between entities that can be inferred from a text. In contrast to open information extraction, these relations are usually grounded in an existing ontology. The result of the relation extraction are triples representing edges in a knowledge graph.

To solve relation extraction, we use mREBEL [6], a state-of-the-art relation extraction model, which is a multilingual extension to the original REBEL model [3]. Although we focus on extracting knowledge graphs from English texts, we found that mREBEL performs better and supports more relation types than the original model. The creators of mREBEL defined relation extraction as a seq2seq task and solved it by fine-tuning BART.

A major advantage of mREBEL is that it has been trained using aligned pairs of Wikipedia abstracts and relations from Wikidata.

Consequently, it is trivial to link the extracted relations to the corresponding relation in Wikidata.

We link the extracted relations to the corresponding Wikidata relations by using the same retrieval pipeline than for the retrieval of entities (cf., Section 3.1) on a separate relation index. However, no reranking is necessary, and we disambiguate retrieved Wikidata relations by choosing the best according to its retrieval score.

3.3 Knowledge Fusion

The knowledge fusion components of the algorithm aims to combine knowledge triples from the relation extraction pipeline with the linked entities from the entity discovery pipeline to build Wikidata grounded RDF triples. It does so by picking subject and object entities for each extracted relation from the pool of linked entities.

Given a ranked list of entities for each recognized mention span and a set of triples describing which entity mention spans are in relation (e.g., <*Weimar*, `wd:country`, *Germany*>), we replace the mention spans with their associated entity. We build a set of RDF triple candidates by applying the cartesian product out of all mentioned subject and object entities for each extracted relation.

3.4 Natural Language Inference

The natural language inference step ranks the RDF triple candidates by whether (1) the triple exist in Wikidata and (2) the triple can be inferred from the text. To also take the entity retrieval score into account, we first assign the mean retrieval score of subject and object entities as score for a triple candidate. If a triple candidate does exist in Wikidata, it is most likely true, and thus we boost the triples score by multiplying it with a constant factor. Prior experiments revealed that the most effective factor is three.

To assess whether a triple can be inferred from the text is a more challenging task. We base this computation on Facebook's BART-large-MNLI model[9]—a zero-shot natural language inference (NLI) model based on BART-large [9] that was tuned on the multi-genre NLI dataset MultiNLI [19]. The model predicts probabilities of whether a text is about a set of freely selectable class labels. In our case, we generate the class label of a triple by concatinating the labels of the subject, predicate, and object. To disambiguate the subject and object, we add the description in brackets. We scale the score of each triple candidate by their corresponding probability. For each extracted relation, we select the highest ranked triple candidate as the final set of triples.

4 EVALUATION

To test the performance of the automatic knowledge graph construction algorithm, a ground truth of aligned texts and Wikidata grounded triples is required. There has been a lack of sufficiently large knowledge graph construction benchmark datasets. However, a new dataset for evaluating relation extraction, which also contains links to their entities and relations in Wikidata, have been released recently: the RED^{FM} dataset [6]. The test set of this corpus contains 446 aligned pairs of Wikipedia abstracts and sets of Wikidata-linked knowledge triples. With this dataset we compute precision, recall and F1 of all components of the construction algorithm.

[9]https://huggingface.co/facebook/bart-large-mnli

Table 1: Precision, Recall, and F1 of the knowledge graph construction components on the test set of the RED^{FM} dataset.

Task	Precision	Recall	F1
Named Entity Recognition	0.067	0.912	0.121
Entity Retrieval	0.004	0.765	0.007
Entity Reranking	0.011	0.739	0.020
Relation Extraction	0.303	0.778	0.407
Relation Linking	0.303	0.778	0.407
Knowledge Fusion	0.147	0.309	0.180
Natural Lanugage Inference	0.182	0.325	0.206

For named entity recognition, we consider a 'hit' if the correct mention span is recognized. For entity retrieval and entity reranking, we compare the mention span and the Wikidata link. To evaluate relation extraction and relation linking, we compare the presence of the corresponding relation or Wikidata property, respectively. A triple is considered a hit if there is a corresponding triple with an identical subject, predicate, and object in the ground truth. To evaluate the knowledge fusion, we select the best triple by score for each set of triple candidates.

Table 1 shows the macro-averaged precision, recall, and F1 values of each step in the knowledge graph construction pipeline. The entity discovery pipeline yields solid recall values which is important for the knowledge fusion. We can also see an increase in precision after reranking, without affecting recall too much.

The relation extraction and linking pipelines also scored high recall values. However, high precision would have been desirable as this pipeline controls how many triples are returned and therefore sets an upper limit on the precision that can be achieved in the final fusion and inference steps. Consequently, the precision values for knowledge fusion and natural language inference are quite disappointing. However, the natural language inference step is able to increase both, precision and recall values of the algorithm.

In an error analysis, we found that some 'errors' are actually related to the test set and not to the algorithm. With an average 2.7 triples per text with an average length of 464 characters, many of the inferrable triples are not part of the ground truth which might be one of the reasons for the low precision. The low performance of the knowledge graph construction algorithm underlines the difficulty of the problem and emphasizes the need for human-supervision.

5 CONCLUSION

In this paper, we introduced WAKA: a Web-based application to construct and author knowledge graphs from unstructered text through the convenience of an intuitive interface. As part of WAKA, we proposed a novel approach for automatically constructing linked knowledge graphs from unstructered text. However, a quantitative evaluation of the automatic approach revealed low F1 which indicates the complexitity of the task. The complexity of the task emphasizes the need for a method to supervise the construction and to support the correction of the resulting knowledge graphs.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work is supported by the Thüringer Ministerium für Wirtschaft, Wissenschaft und Digitale Gesellschaft (TMWWDG) under Grant 5575/10-5 (MetaReal).

REFERENCES

[1] Garima Agrawal, Yuli Deng, Jongchan Park, Huan Liu, and Ying-Chih Chen. 2022. Building Knowledge Graphs from Unstructured Texts: Applications and Impact Analyses in Cybersecurity Education. *Information* 13, 11 (Nov. 2022), 526.

[2] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 54–59.

[3] 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*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 2370–2381.

[4] Salvatore Carta, Alessandro Giuliani, Leonardo Piano, Alessandro Sebastian Podda, Livio Pompianu, and Sandro Gabriele Tiddia. 2023. Iterative Zero-Shot LLM Prompting for Knowledge Graph Construction. *CoRR* abs/2307.01128 (2023). arXiv:2307.01128

[5] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge Graph Embedding Based Question Answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, J. Shane Culpepper, Alistair Moffat, Paul N. Bennett, and Kristina Lerman (Eds.). ACM, 105–113.

[6] Pere-Lluís Huguet Cabot, Simone Tedeschi, Axel-Cyrille Ngonga Ngomo, and Roberto Navigli. 2023. RED^{FM}: a Filtered and Multilingual Relation Extraction Dataset. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 4326–4343.

[7] Vaibhav Kasturia, Marcel Gohsen, and Matthias Hagen. 2022. Query Interpretations from Entity-Linked Segmentations. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. ACM, Virtual Event AZ USA, 449–457.

[8] Natthawut Kertkeidkachorn and Ryutaro Ichise. 2017. T2KG: An End-to-End System for Creating Knowledge Graph from Unstructured Text. In *The Workshops of the Thirty-First AAAI Conference on Artificial Intelligence, Saturday, February 4-9, 2017, San Francisco, California, USA (AAAI Technical Report, Vol. WS-17)*. AAAI Press.

[9] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-Training for Natural Language Generation, Translation, and Comprehension. *CoRR* abs/1910.13461 (2019). arXiv:1910.13461

[10] Jiaying Liu, Jing Ren, Wenqing Zheng, Lianhua Chi, Ivan Lee, and Feng Xia. 2020. Web of Scholars: A Scholar Knowledge Graph. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 2153–2156.

[11] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, 55–60.

[12] Jose L. Martinez-Rodriguez, Ivan Lopez-Arevalo, and Ana B. Rios-Alvarado. 2018. OpenIE-based Approach for Knowledge Graph Construction from Text. *Expert Systems with Applications* 113 (Dec. 2018), 339–355.

[13] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 3980–3990.

[14] 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, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, Walter Daelemans and Miles Osborne (Eds.). ACL, 142–147.

[15] Xavier Schmitt, Sylvain Kubler, Jeremy Robert, Mike Papadakis, and Yves LeTraon. 2019. A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE, Granada, Spain, 338–343.

[16] Stefan Schweter and Alan Akbik. 2020. FLERT: Document-Level Features for Named Entity Recognition. *CoRR* abs/2011.06993 (2020). arXiv:2011.06993

[17] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, Anchorage AK USA, 950–958.

[18] Weischedel, Ralph, Palmer, Martha, Marcus, Mitchell, Hovy, Eduard, Pradhan, Sameer, Ramshaw, Lance, Xue, Nianwen, Taylor, Ann, Kaufman, Jeff, Franchini, Michelle, El-Bachouti, Mohammed, Belvin, Robert, and Houston, Ann. 2013. OntoNotes Release 5.0.

[19] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 1112–1122.

[20] Xiaohan Zou. 2020. A Survey on Application of Knowledge Graph. *Journal of Physics: Conference Series* 1487, 1 (March 2020), 012016.