LINTEXT: A Visual Tool for Exploring and Modeling Knowledge in Text Documents

Riley Capshaw[1,*], Eva Blomqvist[1]

[1]Linköping University, 581 83 Linköping, Sweden

Abstract

A large part of knowledge is commonly encoded into text documents. While extracting this information into a Knowledge Graph (KG) is a common approach, it suffers from challenges when texts are added, removed, or changed, or when the schema of the intended KG changes. Instead we advocate an approach where text and models evolve together in an interactive manner. We present LINTEXT, a system accompanying a published method which allows users to jointly explore and model the information held within text documents. The modeling is accomplished by specifying fill-in-the-blank prompts along with some metadata which are then recorded as specifications for simple relations that can be used to generate an ontology. The exploration aspect is accomplished by having the system complete each prompt with entities identified from the text and presenting the completions as a ranked list to the user, allowing users to verify the quality of the extracted triples. By elevating the development of the ontology to a visual and interactive level, it has an immediate text connection and users can be more certain that the documents they wish to model contain the information they wish to extract or query. Additionally, our system is designed to support the development of relation extraction (RE) pipelines underlying the document analysis, with a particular focus on supporting methods for improving vector representations of the extracted entities. To this end, users can choose to analyze documents from pre-annotated RE data sets to understand how changes in different elements of the pipeline affect the results.

Keywords

Knowledge Graphs, Masked Language Models, Machine Reading, Entity Embedding, Document-level Relation Extraction, Interactive Knowledge Modeling.

1. Introduction

Knowledge Graphs (KGs) and their related ontologies are commonly recognized as key components of modern knowledge-based systems. However, a large body of knowledge still exists only in text documents. Such knowledge can be extracted and encoded into a KG, but if the document corpus is prone to changing, such extraction must be updated or possibly redone with every change. Even worse, when the underlying schema of the KG (e.g., the ontology) changes, the whole process may have to be repeated from scratch. Given such requirements on flexibility and evolution of the knowledge, hand-crafting KGs is not scalable, particularly due to the rate at which knowledge changes in many real-world applications. As such, partially automating the process of extracting and even modeling knowledge has been a subject of research for many years. Nevertheless, accurate and reliable automatic KG construction from natural language documents still remains a difficult task with many challenges, even in light of the impressive recent advances in language modeling. Therefore we advocate a different approach, where text and models (KG and ontology) can evolve together in an interactive manner.

While automated ontology extraction from text has been investigated in the area of ontology learning for several decades, with some systems accompanied by user interfaces, these have not been focused on the combination of text exploration, fact extraction, and ontology modeling that we are presenting here. The approaches closest to ours include end-user focused ontology modeling systems such as the enterprise Wiki approach in [1] and Protégé plugins like [2], although both pre-date the recent major advances in language modeling. Also related are systems which have recently emerged for linking entities between text and images, connected to a knowledge base, such as [3]. However, none of these

EKAW 2024: EKAW 2024 Workshops, Tutorials, Posters and Demos, 24th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2024), November 26-28, 2024, Amsterdam, The Netherlands.

*Corresponding author.

riley.capshaw@liu.se (R. Capshaw); eva.blomqvist@liu.se (E. Blomqvist)

examples support the exploration and interactive evaluation of the effects of different *modern* extraction techniques and components, together with schema (ontology) updates by the user.

In this paper, we present Linköping University's Text Exploration Tool (LINTEXT), a first demonstration of the interactive part of such a vision. LINTEXT is designed with a prompt-first approach to text exploration in mind. This means that the act of exploring documents becomes a modeling task that interactively defines a schema, which itself may be the starting point of an ontology. Users have the option of specifying more information, such as what types of entities are relevant or whether a relation can be symmetric. LINTEXT then provides visual feedback in the form of a ranked list of statements which could possibly be extracted from the document. This allows a user to understand whether the document they are modeling contains instances of the relations they define, and how well a particular relation extraction (RE) pipeline extracts correct triples. In addition to exploring and modeling documents, LINTEXT is built on top of our recent work on zero-shot approaches for RE with masked language models (MLMs) [4, 5], as well as our work to appear in EKAW 2024 on improving vector representations for entities [6]. As such, it is also designed to help researchers explore the effects of changes to components of RE pipelines, such as the named entity recognition software, the MLM, the scoring metrics, or the approach for generating vector representations for the entities. The system is made publicly available via our GitHub repository[1].

2. The LINTEXT System

LINTEXT enables users to experiment with different configurations of prompt-based zero-shot relation extraction pipelines for short documents (around 512 tokens after tokenization). It supports reading documents from established RE data sets (e.g., DocRED [7] and BioRED [8]) as well as user-provided documents. Users additionally can load relation definitions from these data sets or define new relations. These definitions are then used to analyze the documents, allowing users to visually confirm whether some information in the document can be modeled by the schema. The analysis is presented to the user as a sorted list of statements that could be extracted from the document. This aspect is discussed in more detail in Section 2.2. Users can adjust elements of the schema and re-analyze the document until they are satisfied with the list's ordering, then save the results to a file to be shared.

2.1. User Interface

LINTEXT's user interface as seen in Figure 1 is broken into two parts. On the left is the document view, which shows the text of the current document after preprocessing. Users can either select a document from a data set or "custom" to enter their own text to process. Preprocessing may involve tokenization specific to a particular language model (LM), such as the word-piece tokens particular to BERT-like models [9]. Users can hover over each word or entity in the document view to see how it was tokenized, or they can check the "Show tokens" box to display detailed tokenization information for the entire document. This allows users to understand when tokenization might affect their results such that they may choose a different MLM with a more tailored tokenizer (such as BioBERT [10] for biomedical texts). Entity mentions are color-coded, with the outline color representing the entity's type and the fill color representing when mentions refer to the same entity. For example, "cardiomyopathy," "myocardial dysfunction," and "myocardial necrosis" are all mentions of the same entity, each with the type `DiseaseOrPhenotypicFeature`. When loading documents from a data set, these annotations are provided as the ground truth, or could be provided by a pre-processing step.

On the right is the schema view, which shows the current schema being explored or constructed. This can be loaded from file, loaded with a document from a known data set, or a user can create this from scratch. In order for a user to add a relation to the schema, at a minimum they need to add an identifier and a fill-in-the-blanks prompt written in natural language (with variables). The relation name is meant to be a human-readable name (i.e., label) while the relation identifier could simply be a code, as occurs

[1] https://github.com/LiUSemWeb/LINTEXT

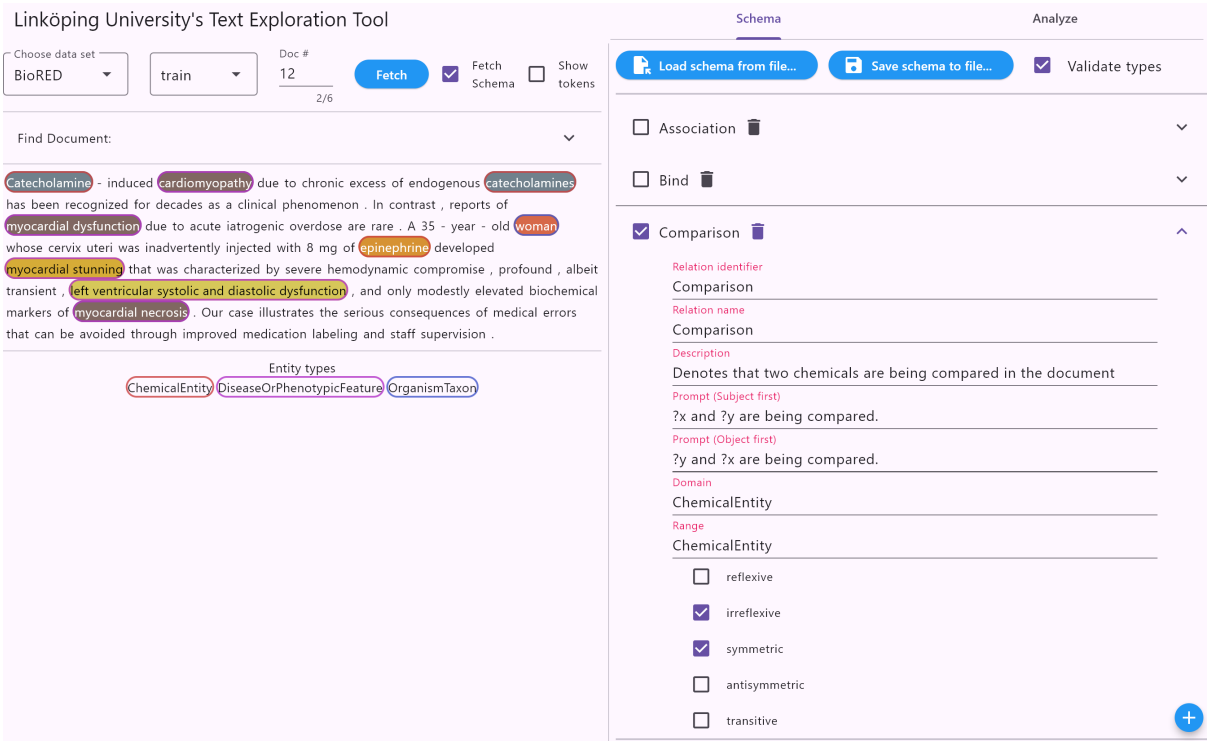

Figure 1: (Left) **Document View**. In addition to providing their own documents, users can choose from a selection of standard document-level relation extraction datasets, such as BioRED (shown above) and DocRED. Annotated information includes tokenization, entities, entity mentions, and entity types. (Right) **Schema View**. Users are able to load an existing set of relations from a data set, then edit and save them for use in the extraction phase. The supported fields are shown above. The checkboxes are used to specify which relations will be used when analyzing the current document in the Document View.

in many ontologies and datasets. For example, DocRED assigns all relations WikiData [11] property identifiers. This field can be used to clarify that the relation `P569` is also known as `date of birth`. The domain and range fields, when not empty, limit statements extracted for a relation to only include entities which have the types listed, i.e., basically applying a closed-world semantics to these notions by assuming that all entity types are disjoint[2]. In Figure 1, `Comparison` therefore only applies to and from `ChemicalEntity` mentions. Users can also select whether some relation properties apply, such as (ir)reflexivity, (anti)symmetry, and transitivity. BioRED explicitly mentions that its relations should be considered undirected, which implies that they are all symmetric. Additionally, it might be useful to mark that a `ChemicalEntity` cannot be compared with itself, and thus `Comparison` is irreflexive.

2.2. Pipeline

The pipelines we focused on primarily support our ongoing work in the area focused on smaller MLMs [4, 5]. Figure 2 shows an overview of the standard pipeline. The pipeline preprocesses an input document by recognizing all entities, their mentions, and their types, then tokenizing the text. Given a user-defined prompt as the query (`?x is in ?y.` in the above example), all unique pairs of entity mentions are gathered. These pairs are then filtered by user-specified restrictions, such as domain, range, and the relation properties seen in Figure 1 (right). For example, mention pairs which do not satisfy the domain and range are discarded, or for antisymmetric relations, mention pairs corresponding to the same entity are discarded. Each mention pair is then used to fill in the two blanks from the prompt to generate a candidate statement, which is then ranked using a MLM according to some scoring

[2]While this is not the semantics that would be applied in a resulting OWL ontology modeled through the interface, it has been shown in previous work [4, 5] to be a useful way of improving relation extraction results, and is also a bit more user-friendly for users who are not as familiar with the open-world assumption.

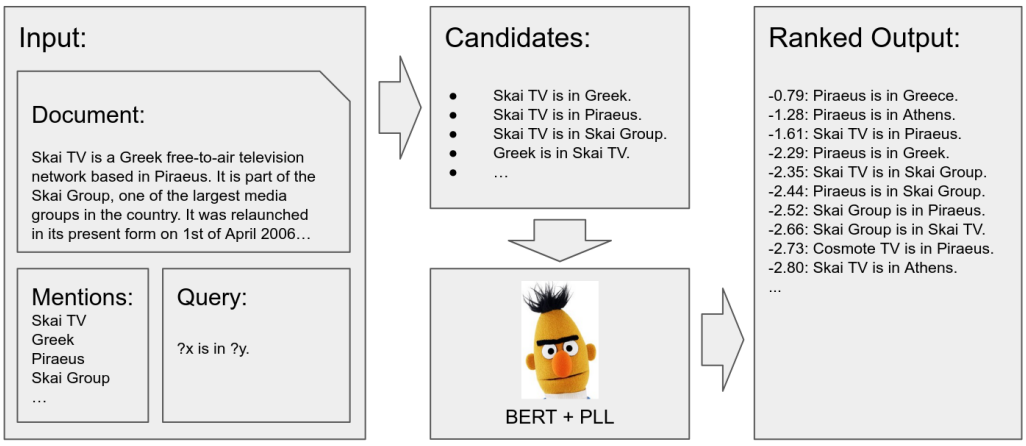

Figure 2: Given a preprocessed document, a MLM such as BERT, RoBERTa, or BioBERT is used to rank candidate statements, by default using PLL scores. These statements are constructed by taking a user-written prompt and replacing two blanks (`?x` and `?y`) with all pairs of entity mentions from the document. The exact mechanism for generating the scores can be changed by the user, enabling an effective search for pipeline improvements.

metric. The default metric is pseudo-log-likelihood (PLL) [12, 13] as used in our earlier work [4], but this pipeline has been used to support work involving PLL approximations such as cosine similarity [5]. In support of our work accepted to EKAW 2024, this pipeline (and user interface) additionally allows users to experiment with advanced contextualization techniques specific to MLMs.

2.3. Implementation

LINTEXT is composed of two separate processes. The first is the visual front end developed using the cross-platform user-interface SDK Flutter[3], allowing it to run on most major operating systems and web browsers with minimal effort. The second process is the web server back end developed in Python and based heavily on the tools provided by the Hugging Face transformers library[4] for easy swapping of LMs and tokenizers. While some form of hardware acceleration is suggested (e.g., CUDA support), most models can be run on CPU-only hardware, though with a noticeable increase in response times. The two components communicate using a simple JSON API specification, ensuring their independence and allowing for greater flexibility, which in turn better enables the swapping of RE pipeline components.

3. Use Case: Extending the BioRED Schema

Consider as a use case the example from Figure 1. To load this screen, first select 'BioRED' from the 'Choose data set' drop-down, select the 'train' subset in the next drop-down menu, and enter 12 as the document number. Ensure that the 'Fetch Schema' checkbox is checked before clicking the 'Fetch' button so that the system also loads the BioRED schema[5] and displays it in the Schema View (right).

The BioRED schema includes eight relations, most of which encode multiple distinct sub-relations. For example, according to the annotation guidelines, `Positive Correlation` between two entities can represent `Upregulation`, `Exhibition`, `Response`, `Sensitivity`, `Induce`, or `Increase` relations, each with a different domain and range. At the start, a user might enter a single prompt for this relation into the 'Prompt (Subject first)' field, such as `?x is positively correlated with ?y.` Since BioRED explains that all relations are undirected, the user checks the checkbox for 'symmetric' and leaves other boxes unchecked. The user can additionally say that an entity should not be considered positively

[3] https://flutter.dev/
[4] https://huggingface.co/
[5] BioRED itself does not provide a file with a schema definition. We constructed one manually based both on the data and the annotation guidelines, then associated it with the data set in a way that the system can handle. We provide a copy of this schema in the GitHub repository, including simple prompts for each relation.

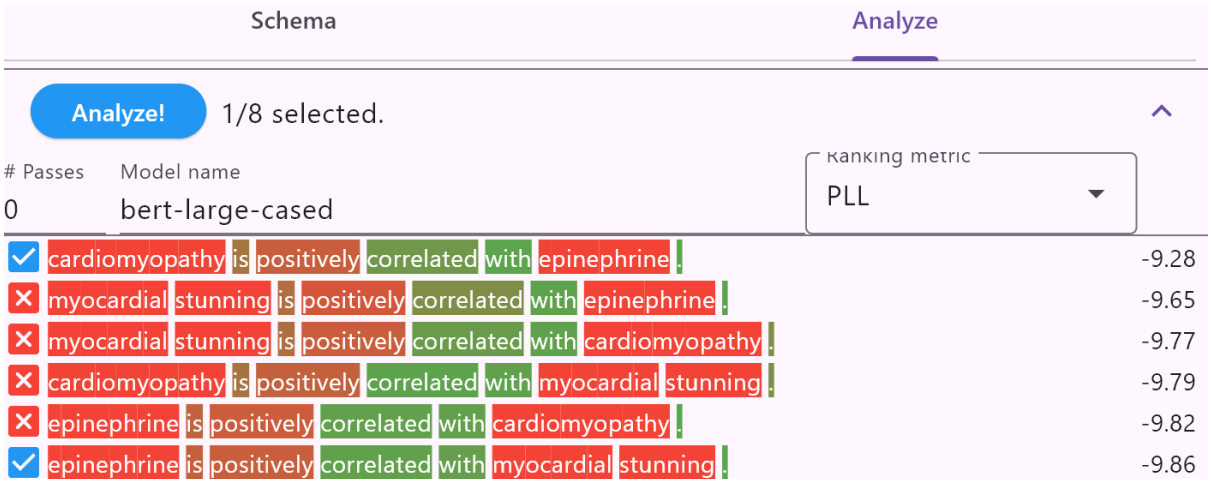

Figure 3: The **Analyze View** can be accessed by clicking the Analyze tab at the top right, replacing the Schema View. There, users can select details about the RE pipeline, such as the MLM, and view the analysis output as a ranked list of statements. Statements which are true according to the data set are marked with a blue checkmark. The background color for each token is a heat map of its individual score given the chosen ranking metric, from red as 0 (lowest) to green as 1 (highest). The exact value is shown by hovering over a token.

correlated with itself by checking the box for 'irreflexive'. Next, the user checks the box beside `Positive Correlation` and leaves all others unchecked to inform the system that it is the only relation to be considered. To start analyzing documents, the user switches to the 'Analyze' tab (Figure 3) and, by not changing any other options, clicks the 'Analyze' button to run the default RE pipeline. The interface now shows a list of candidate statements that could be extracted from the current document.

The user might find that the results are not particularly good since the highest-ranked candidates are not supported by the text for many documents. This could be because most documents discuss the sub-relations mentioned before rather than explicitly saying that two entities are positively correlated, making it difficult to relate the prompt to the text. Alternatively, the user might realize that they are only interested in finding documents which say that some `ChemicalEntity` (drug) will `Induce` a particular `DiseaseOrPhenotypicFeature` (disease). In both cases, the user can improve their results by adding a new relation to the schema. To do so, the user returns to the Schema View and presses the blue plus button in the bottom right, then fills in the fields accordingly to define the `Induce` relation. Now, the user has the option of either using this new relation on its own or jointly analyzing the `Induce` and `Positive Correlation` relations. The former allows the user to narrow their results to just this sub-relation, while the latter broadens their search, using more specific evidence of the `Induce` relation as evidence for when a document contains instances of `Positive Correlation` relations. Finally, this process can be repeated for all other sub-relations, resulting in a much more fine-grained schema which better models the actual content of the documents.

4. Conclusions

This demonstration will showcase the possibilities of interactive knowledge extraction from text, coupled with ontology modeling. LINTEXT is a first attempt to develop a tool for interactive knowledge modeling and evolution based on text documents. It relies on an approach presented in our EKAW2024 paper [6] and previous work that does not require fine-tuning LMs or the use of expensive large generative LMs, while staying modular enough to support the exploration of those as alternative components in the pipeline. LINTEXT thus both allows for enabling end-users to model and explore texts, as well as for evaluating and improving current pipelines. Future work includes to conduct user studies on the usability of the proposed system interface, as well as to use it for evaluating our current pipeline with end-users (as in [6] it was only evaluated on existing standard RE datasets).

References

[1] C. Ghidini, B. Kump, S. Lindstaedt, N. Mahbub, V. Pammer, M. Rospocher, L. Serafini, Moki: The enterprise modelling wiki, in: L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, E. Simperl (Eds.), The Semantic Web: Research and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 831–835.

[2] V. Pammer, P. Scheir, S. Lindstaedt, Two protégé plug-ins for supporting document-based ontology engineering and ontological annotation at document level, in: 10th International Protégé Conference, 2007.

[3] S. Dost, L. Serafini, M. Rospocher, L. Ballan, A. Sperduti, Aligning and linking entity mentions in image, text, and knowledge base, Data & Knowledge Engineering 138 (2022) 101975. URL: https://www.sciencedirect.com/science/article/pii/S0169023X2100094X. doi:https://doi.org/10.1016/j.datak.2021.101975.

[4] R. Capshaw, E. Blomqvist, Towards tailored knowledge base modeling using masked language models, in: Proceedings of TEXT2KG, Co-located with ESWC 2023, CEUR-WS, 2023.

[5] R. Capshaw, E. Blomqvist, Understanding and estimating pseudo-log-likelihood for zero-shot fact extraction with masked language models, in: Proceedings of the 12th International Joint Conference on Knowledge Graphs, 2023.

[6] R. Capshaw, E. Blomqvist, Contextualizing entity representations for zero-shot relation extraction with masked language models, in: To appear in International Conference on Knowledge Engineering and Knowledge Management (EKAW'24), Springer, 2024.

[7] Y. Yao, D. Ye, P. Li, X. Han, Y. Lin, Z. Liu, Z. Liu, L. Huang, J. Zhou, M. Sun, DocRED: A large-scale document-level relation extraction dataset, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 764–777.

[8] L. Luo, P.-T. Lai, C.-H. Wei, C. N. Arighi, Z. Lu, BioRED: a rich biomedical relation extraction dataset, Briefings in Bioinformatics 23 (2022) bbac282. URL: https://doi.org/10.1093/bib/bbac282. doi:10.1093/bib/bbac282. arXiv:https://academic.oup.com/bib/article-pdf/23/5/bbac282/45936115/bbac282.pdf.

[9] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. URL: https://aclanthology.org/N19-1423. doi:10.18653/v1/N19-1423.

[10] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, J. Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, Bioinformatics 36 (2020) 1234–1240.

[11] D. Vrandečić, Wikidata: A new platform for collaborative data collection, in: Proceedings of the 21st International Conference on World Wide Web, WWW '12 Companion, Association for Computing Machinery, New York, NY, USA, 2012, p. 1063–1064. URL: https://doi.org/10.1145/2187980.2188242. doi:10.1145/2187980.2188242.

[12] J. Shin, Y. Lee, K. Jung, Effective sentence scoring method using BERT for speech recognition, in: W. S. Lee, T. Suzuki (Eds.), Proceedings of The Eleventh Asian Conference on Machine Learning, volume 101 of *Proceedings of Machine Learning Research*, PMLR, 2019, pp. 1081–1093. URL: https://proceedings.mlr.press/v101/shin19a.html.

[13] A. Wang, K. Cho, BERT has a mouth, and it must speak: BERT as a Markov Random Field language model, in: Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation, 2019, pp. 30–36.