Inductive Learning on Heterogeneous Graphs Enhanced by LLMs for Software Mention Detection

Gabriel Silva
IEETA, DETI, LASI, Univ. Aveiro, PT
grsilva@ua.pt

Mário Rodriges
IEETA, ESTGA, LASI, Univ. Aveiro, PT
mjfr@ua.pt

António Teixeira
IEETA, DETI, LASI, Univ. Aveiro, PT
ajst@ua.pt

Marlene Amorim
GOVCOPP, DEGEIT, Univ. Aveiro, PT
mamorim@ua.pt

Abstract

This paper explores the synergy between Knowledge Graphs (KGs), Graph Machine Learning (Graph ML), and Large Language Models (LLMs) for multilingual Named Entity Recognition (NER) and Relation Extraction (RE), specifically targeting software mentions within the SOMD 2025 challenge. We propose a methodology where documents are first transformed into heterogeneous KGs enriched with linguistic features (Universal Dependencies) and external knowledge (entity linking). An inductive GraphSAGE model, operating on PyTorch Geometric's 'HeteroData' structure with dynamically generated multilingual embeddings, performs node classification tasks. For NER, Graph ML identifies candidate entities and types, with an Large Language Model (LLM) (DeepSeek v3) acting as a validation layer. For RE, Graph ML predicts dependency path convergence points indicative of relations, while the LLM classifies the relation type and direction based on entity context. Our results demonstrate the potential of this hybrid approach, showing significant performance gains post-competition (NER Phase 2 Macro F1 improved to 43.6% from 29.5%, RE Phase 1 33.6% Macro F1), which are already described in this paper, and highlighting the benefits of integrating structured graph learning with LLM reasoning for information extraction.

1 Introduction

Advancing the capabilities of Natural Language Processing (NLP) often requires moving beyond the surface level of plain text to leverage richer, more structured information. While raw text provides the foundation, incorporating details like semantic relationships, external world knowledge, or task-specific metadata can significantly boost performance on complex understanding (Safuan and Ku-Mahamud, 2025). This necessity, however, introduces a significant challenge: developing frameworks capable of seamlessly handling diverse data formats and integrating multiple layers of annotations – ranging from word-level tags (like part-of-speech) to sentence-level labels (like sentiment) and document-level classifications (like topic).

KGs provide a notably flexible and powerful paradigm to address this complexity. By representing information as nodes (entities, concepts) and edges (relationships), KGs offer an inherently structured way to capture intricate connections within and beyond the text. This structure facilitates the coherent integration of various annotation types across different textual granularities, ensuring that, for instance, word-level syntactic information can coexist and relate to document-level semantic themes. Crucially, KGs excel at maintaining the explicit connections between linguistic units and associated knowledge, preserving context that might be lost in purely sequential models. Furthermore, the KG paradigm benefits from a mature and growing ecosystem of established standards, databases, and software tools for creation, querying, and reasoning.

Despite the clear advantages offered by KGs for representing rich, multi-level information, the recent trajectory of mainstream NLP research has largely centered on models that process raw text sequences directly. This past decade has been marked by significant breakthroughs: the fundamental contribution of distributional representations via Word Embeddings (Mikolov et al., 2013), the development of powerful sequential models like Bi-LSTMs (Lample et al., 2016), arguably the most impactful release with the Transformer architecture (Vaswani et al., 2017), followed by large pre-trained models such as BERT (Devlin et al., 2019) and BART (Lewis et al., 2020), and more recently, generative AI and LLMs such as ChatGPT (OpenAI, 2023). Although these methods have pushed the state-of-the-art by learning complex patterns from vast textual data, their primary focus on sequential text

Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025), pages 164–172

input means the potential synergies of explicitly integrating structured knowledge, as offered by KGs, remain relatively underexplored in many application areas.

The use of graphs has been previously demonstrated in adjacent fields, such as Open Information Extraction, particularly for the Chinese language, where graph-based approaches have yielded favorable results (Lyu et al., 2021). Initial research has also investigated the potential of integrating graphs and LLMs. The study by (Chen et al., 2024) examines the role of graphs as both enhancers and predictors. An example of the combined capabilities of graphs and LLMs is Microsoft GraphRAG (Edge et al., 2025), which aims to leverage their synergistic effects.

The research presented here is part of the 2025 Software Mention Detection (SOMD) competition. The primary contribution of this work is a multilingual NER and RE system that utilizes KGs, Graph ML, and LLMs.

2 Method

In this section, the methodology employed is described. We will outline the overall approach, describe the processing of the dataset and finally how to generate our predictions for both NER and RE.

2.1 Overall Process

The methodology presented involves converting input documents into a structured graph format. This intermediate representation is specifically designed to serve as input for various Graph ML algorithms. The overall system architecture that facilitates this process is illustrated in Figure 1.

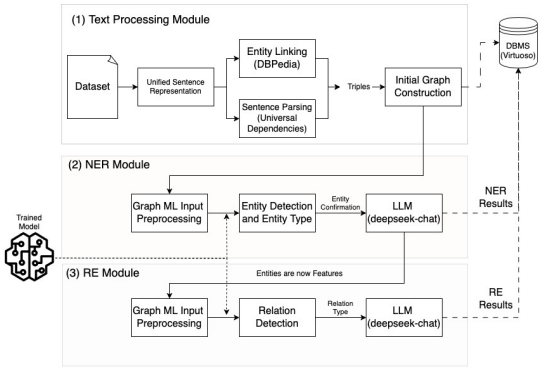

Figure 1: Overview of the current framework architecture

There are 3 crucial steps in this architecture. The first one is the Text Processing Module where we convert the dataset into our initial graph representation, we enrich our data by using making use Entity Linking and Universal Depedencies. The second step is the NER Module. In this module, the Graph ML algorithm performs two tasks: identifying entities and determining the type of each entity. The LLM serves as a confirmation layer when the Graph ML algorithm is uncertain about which type of entity to assign to a given word. The last step is the RE module, in this module our Knowledge Graph (KG) already has knowledge about the predicted entities and identifies where a relationship is present, then the LLM will decide which type of relationship exists between these two words based on their entity types.

2.2 Dataset Processing

In this work the only dataset used was the one provided by the competition. This dataset consists of 1,150 sentences for algorithm development, followed by two distinct testing phases. The first testing phase contains 203 sentences, while the second phase includes an additional 220 sentences. The first phase test set more closely resembles the training data, whereas the second phase serves as an out-of-domain evaluation.

As previously described, the text processing module is responsible for processing the dataset. Initially, texts are converted into a unified representation, ensuring consistent spacing around punctuation across all sentences and proper formatting of URLs, among other standardizations. This process may add or remove tokens from sentences, therefore, an additional attribute is maintained for each word to represent its mapping in the original sentence, enabling reconstruction at a later stage.

The second step of this text processing module involves performing Entity Linking. We query DBpedia for concepts identified in the sentences and establish links in our KG to the corresponding DBpedia nodes, this query is shown in Listing 1. We query DBPedia Software sub-set for concepts that contain the given word in English. Each DBPedia concept that is found is then added as a Class on our Graph with the connections found in the "class" query variable. Ideally, this step would engage with the entire DBpedia instance rather than this limited subset. However, at the time of preparing this work, access to a DBpedia dump was unavailable due to maintenance. In this step, we also parse the sentence using a Universal Dependencies (de Marneffe et al., 2014) parser to extract the syntactic depen-

dencies and morphological features of each word.

Listing 1: DBPedia SPARQL Query. "word" is replaced with the term to query.

```
SELECT DISTINCT ?s ?label ?class WHERE {
    ?s rdf:type dbo:Software .
    ?s rdfs:label ?label .
    FILTER (lang(?label) = 'en') .
    ?label bif:contains "word" .
    ?s rdf:type ?class .
}
```

Each word has attributes including feats, their dependency graph, lemma, edges, and other characteristics defined in Universal Dependencies. Additionally, each word can be linked to the original sentence to which it belongs. We form each triple and upload this data into a triplestore (Virtuoso). For a more in-depth look at the process of building the graph from text can be read at (Silva et al., 2023, 2024). An example of the connections in the graph can be seen in Figure 2. We start at a sentence and navigate the graph through its dependency tree (Sentence -> ROOT word -> dependents). An example of what a "word" node looks like can be seen in Figure 3. This word did not have any connections to DBpedia, as such, the edges are not present.

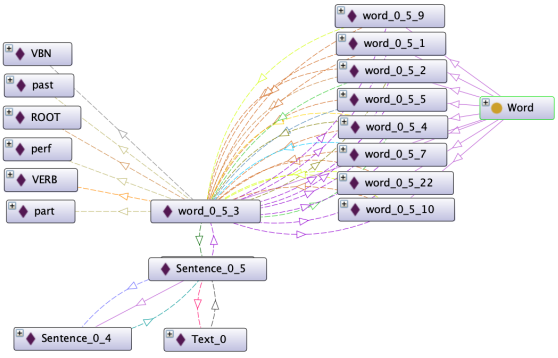

Figure 2: Example of the connections in a graph starting from a sentence.

2.2.1 Named Entity Recognition

For NER we add the entity tags without the BIO part to each word in our graph as an attribute. Words that do not represent an entity are simply tagged with "Nothing".

2.2.2 Relation Extraction

To represent relations between entities while treating the problem as a node classification task, we did not label the relations as edges in our graph. Instead, we made use of the dependency graph. For

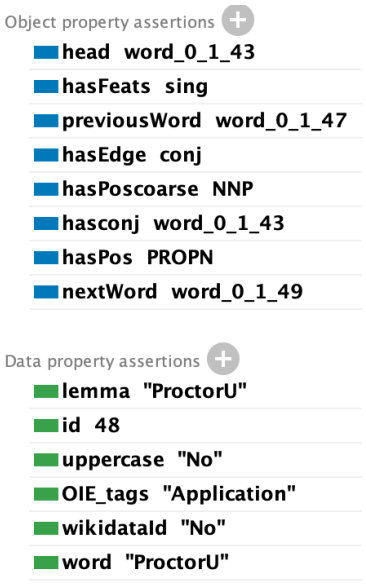

Figure 3: Example of a word node for the word "ProctorU"

each pair of entities that form a relation, we traverse their dependency graph until we reach the word at which they converge. At this convergence point, we create an attribute and designate it as a relationship. To backtrack we simply look for the beginning of identities that converge in that word.

In Figure 4 we have an example of how we do this. The sentence from the test set "Standardized regression coefficients (SCR) were calculated using the sensitivity package of the R - project [50] ." has a relationship (sensitivity, PlugIn_of, R). In this example we can see that they both converge on the word "package" by following their dependency graphs:

sensitivity - package

R - project - of - package

This allows for a relation to be marked at the "package" word.

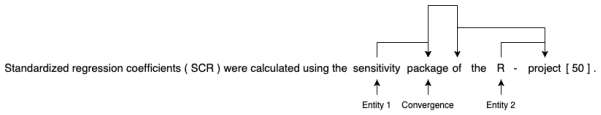

Figure 4: Using the Dependency Graph to get the convergence between two entities.

2.3 Prediction Process

In this subsection, we will elaborate on the process used to derive NER and RE results from the initial text.

2.3.1 Named Entity Recognition

Following the construction of the initial graph the format has to be adapted from triples to the Pytorch Geometric (PyG) (Fey and Lenssen, 2019) format. As part of this preparation pipeline, embeddings are generated for each word node. These embeddings are created using the "intfloat/multilingual-e5-large-instruct" model(Wang et al., 2024) from HuggingFace[1]. We chose this model due to its multilingual capabilities, performance and size according to the MTEB benchmark[2] (Muennighoff et al., 2023). These are intentionally not stored in the graph due to their large vector size. Instead, they are computed dynamically when required by the Machine Learning pipeline, just prior to converting the augmented graph into the PyG format.

Given the heterogeneous nature of the graph, which contains nodes and edges with diverse types and attributes, representing the complete edge information within a single tensor is not feasible. As a result, we utilize the HeteroData object[3] to structure the graph data for model training. Additionally, we adopted an inductive learning approach (Lachaud et al., 2023), selected for its improved applicability and generalizability to real-world scenarios where graph structures may evolve or be unseen during training. We use a GraphSAGE (Hamilton et al., 2018) based architecture. Our model for both tasks consists of 8 layers where each layer is a GraphSAGE layer, followed by normalization, ReLU and applying dropout with a learning rate of 0.01 and dropout of 0.3.

We train two distinct models: one that predicts whether a word corresponds to an entity, and another that predicts the type of the previously identified entities. If the output of the entity type prediction falls below a specified threshold, we validate the result using a LLM, specifically, we utilize DeepSeek v3 (deepseek-chat) (DeepSeek-AI et al., 2025) with queries adapted from the Microsoft RAG (Edge et al., 2025) GitHub repository. The NER query can be seen on Table 1 We utilize the five entity types with the highest likelihood identified by the Graph ML algorithm as grounding for our query.

[1] https://huggingface.co/intfloat/multilingual-e5-large-instruct

[2] https://huggingface.co/spaces/mteb/leaderboard

[3] https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.data.HeteroData.html

Both predictions are then incorporated into our graph and transmitted to the RE Module. If two or more connected words are identified as entities we can add the respective BIO tags in conjunction with the entity tag, the one that comes first in the sentence will be tagged with the B and the following linked ones with I.

2.3.2 Relation Extraction

The RE process is similar to the NER process with the model architecture being the same. As discussed in Section 2.2.2 we identify the convergence point of each pair of words that form a relationship so in this step we want to predict which words are a convergence point. Once this convergence point is established, we just have to go through the identities that were identified in the previous step for a given the sentence.

The LLM in this prediction module is used to determine both the direction and type of the relationship. We ground the LLM by giving it the Entity type of each identified word and ask it to classify the relation type. The model used is the same as previously mention, DeepSeek V3. In instances where no convergence word is identified, we provide the LLM with the entities and ask it to identify if there are any relations present in the sentence. Table 2 shows the query used.

3 Results

This section presents the results of our training on the validation set and the performance on the test set for both phases of the competition. Every run was documented using the Weights & Biases platform (Biewald, 2020).

3.1 Named Entity Recognition

As previously outlined in the methods section, NER was categorized into two models: a binary model and an entity classification model.

3.1.1 Binary Model

The Binary Model performed well when identifying entities and non-entities. The model achieved an F1-Score of 93.6% with a recall and precision of 93.8% and 93.3% respectively on the validation set. Additionally, we plotted the ROC and Precision-Recall curves, as presented in Figures 56. The accuracy is not reported as it is not a relevant metric for this problem due to the imbalance in the dataset (number of words classified as entity vs non-entity).

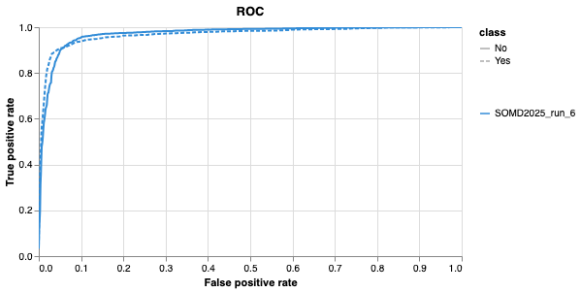

Figure 5: ROC Curve for the Entity Binary classification model.

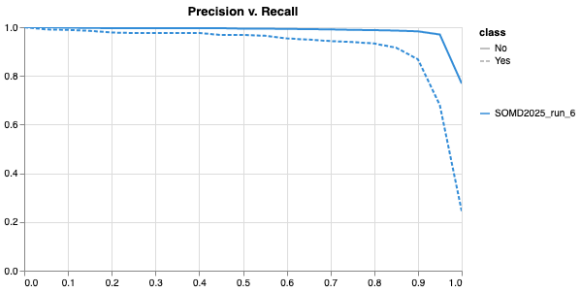

Figure 6: Precision-Recall curve for the Entity Binary classification model.

The F1-Score and corresponding curves indicate that the model demonstrated strong performance in accurately identifying whether a word is an entity.

3.1.2 Entity Type Model

Using the output from the previous model, the subsequent step was to add this prediction as an attribute for each word to identify the type of entity. Consequently, we trained a model with this additional attribute, which indicates whether a word is an entity, and aims to predict the type of entity present in the word. The training results yielded F1, Precision, and Recall scores of 70.5%, 70.7%, and 71.6%, respectively, on the validation set. A normalized confusion matrix is presented in Figure 7.

The model exhibited the greatest difficulty in identifying entities classified as "AlternativeName" and "SoftwareCoreference.". Additionally, it frequently confused "PlugIn" with "Application," with a misclassification rate exceeding fifty percent. This observation is further supported by the ROC curve presented in Figure 8.

During Phase 1, we evaluated the performance of the Graph ML model by submitting results without validation from the LLM. This approach resulted in a decrease in performance, thereby supporting our hypothesis that utilizing the LLM as a confirmation tool in cases of model uncertainty represents a viable strategy.

The results for Phase 1 of this model was a Macro-average F1 score of 44.6%, with Precision at 52.7% and Recall at 40.2%. In Phase 2, the results showed a decline, with an F1 score of 29.5%, Precision at 35.8%, and Recall at 27.3. Following the conclusion of Phase 2, we have successfully improved these metrics, resulting in current values of 43.6% for F1, 43.7% for Precision, and 45.2% for Recall, which are now comparable to the results of Phase 1.

3.2 Relation Extraction

The last part of the competition was the RE which we could only do after having identified the entities. As was previously described for RE we identify the word where two entities converge and tag it as a "relation", as such, the goal of this model is to find those convergence words. When these convergence words are found we can go through the identities and find which ones converge to that predicted word.

This model achieved an F1 score of 87.4%, precision of 88.1% and recall of 86.8%. Similarly to the binary model we can see the ROC and Precision vs Recall curves in Figure 9 and Figure 10.

Unfortunately we only managed to obtain relation results for the first phase of the competition with the scores being: 33.5% F1-Score, 38.4% Precision and 32.1% Recall. However, with the model for entities having significant improvements after the competition is ended, our hypothesis is that these better results will also show themselves in the RE portion of the work.

4 Conclusion

Although the results in the competition were not optimal compared to those of other participants, we have made significant improvements to the model during the ongoing open phase which are here presented, enhancing its competitiveness on the test set. In the NER task, the Phase 2 results yielded a Macro F1 score of 29.5% and a Micro F1 score of 37.4%. In this Open Submission phase, the scores have improved to 43.6% and 58%, respectively, increasing their competitiveness. Unfortunately, for RE, we were unable to obtain results for Phase 2 in a timely manner. Thus, the Phase 1 results were as follows: 33.6% F1-Score, 38.4% Precision and 32.1% Recall. With the improvements to the en-

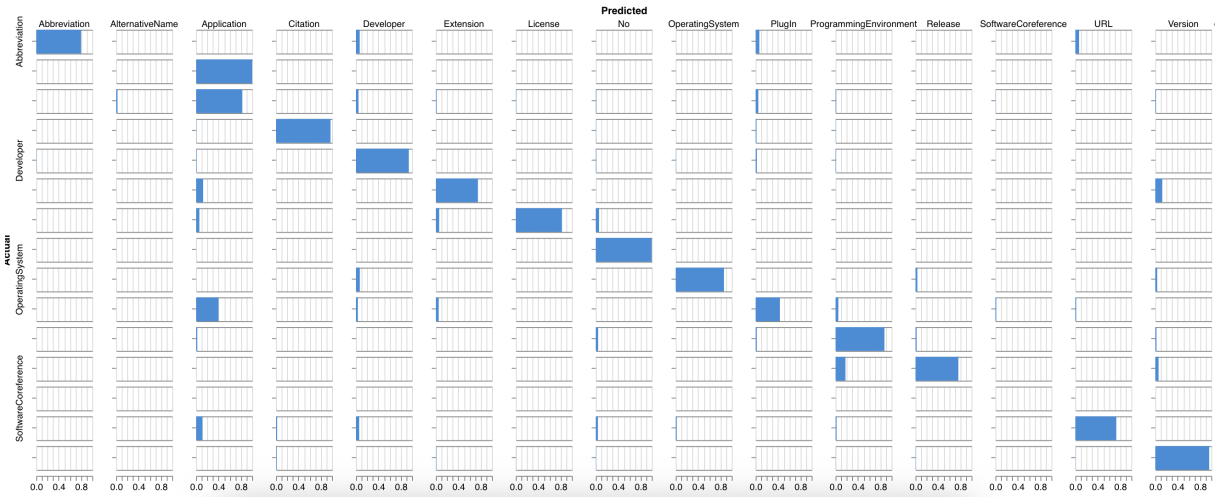

Figure 7: Normalized confusion matrix for the entity prediction model.

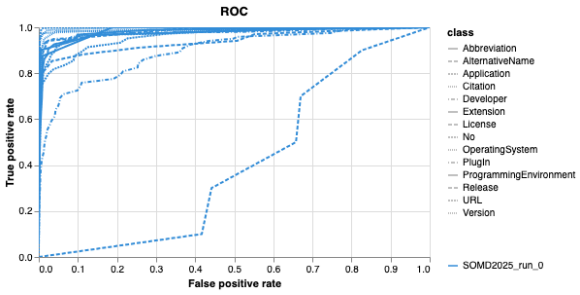

Figure 8: ROC Curves for the entity prediction model.

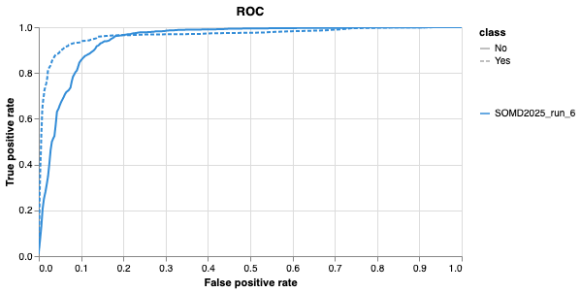

Figure 9: ROC Curves for the relation model.

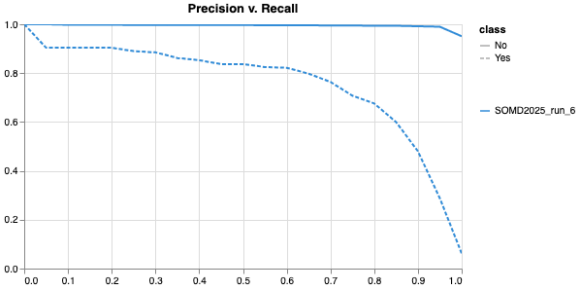

Figure 10: Precision-Recall curves for the relation model.

tity model, these numbers are expected to be even better.

Our model possesses several advantages, including being lightweight, easily adaptable to other problems (as no specific code or preprocessing was conducted for this dataset), and supporting multiple languages (as long as a parser is available).

Despite the promising methodology and its positive aspects, considerable work remains to enhance these results. The work here developed is available at: https://github.com/gabrielrsilva11/SOMD2025.

Future Work

- Incorporate additional external knowledge into our graph to enhance its ability to generalize knowledge. Furthermore, integrate DBpedia (Auer et al., 2007) into our KG.

- Testing various Graph ML models, including transformers (Hu et al., 2020), has the potential to significantly influence the results.

- Obtain additional data or a pre-trained model. The dataset utilized for training the graph machine learning algorithms was exclusively provided by the competition and comprises a total of 1,150 sentences.

5 Acknowledgements

This work was funded by FCT - Fundação para a Ciência e a Tecnologia (FCT) I.P., through national funds, within the scope of the UIDB/00127/2020 project (IEETA/UA, http://www.ieeta.pt/) and the scholarship UI/BD/153571/2022.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007.

Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, page 722–735. Springer-Verlag.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. 2024. Exploring the potential of large language models (llms)in learning on graphs. *SIGKDD Explor. Newsl.*

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592, Reykjavik, Iceland. European Language Resources Association (ELRA).

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, and Chenyu Zhang et al. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From local to global: A graph rag approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive representation learning on large graphs. *Preprint*, arXiv:1706.02216.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, WWW '20, page 2704–2710, New York, NY, USA. Association for Computing Machinery.

Guillaume Lachaud, Patricia Conde-Cespedes, and Maria Trocan. 2023. Comparison between inductive and transductive learning in a real citation network using graph neural networks. In *Proceedings of the 2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '22, page 534–540. IEEE Press.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Zhiheng Lyu, Kaijie Shi, Xin Li, Lei Hou, Juanzi Li, and Binheng Song. 2021. Multi-grained dependency graph neural network for chinese open information extraction. In *Advances in Knowledge Discovery and Data Mining*.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. *Preprint*, arXiv:2210.07316.

R OpenAI. 2023. Gpt-4 technical report. *ArXiv*, 2303.

Safuan and Ku Ruhana Ku-Mahamud. 2025. Handling semantic relationships for classification of sparse text: A review. *Engineering Proceedings*, 84(1).

Gabriel Silva, Mário Rodrigues, António Teixeira, and Marlene Amorim. 2023. A Framework for Fostering Easier Access to Enriched Textual Information. In *12th Symposium on Languages, Applications and Technologies (SLATE 2023)*, volume 113 of *Open Access Series in Informatics (OASIcs)*, pages 2:1–2:14, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Gabriel Silva, Mário Rodrigues, António Teixeira, and Marlene Amorim. 2024. First assessment of graph machine learning approaches to Portuguese named entity recognition. In *Proc. Int. Conference on Computational Processing of Portuguese*, pages 563–567. ACL.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *Preprint*, arXiv:2402.05672.

A Prompts Used

Table 1: Prompt Structure for entity identification

Section	Prompt Details
Target Activity	You are an intelligent assistant that helps a human analyst to identify the entity type of words in a Sentence.
Goal	Given a word or words identify their given entities based on a given list.
Steps	1. You are given a word or words. If there is more than one word identify if they belong to the same entity or are separate entities. - Each word is separated by a space, even if it is punctuation count it as a word. - You must ONLY classify the given words. DO NOT CLASSIFY ANY OTHER WORDS IN THE SENTENCE. 2. For the word or words given identify which entity they belong to from the list of given words. - Make sure to take into account the previous words into your classification. 3. Return output as a single list of all the word entity pairs in steps 1 and 2. Use **{record_delimiter}** as the list delimiter. DO NOT ADD ANY EXPLANATION. 4. Format each pair as <word> {delimiter} <entity_type>

Examples	`Example 1:`
	Words: IMB SPSS Inc .
	Entity types: No, Application, Developer, URL, Version, PlugIn, Citation, Extension, ProgrammingEnvironment, OperatingSystem, Release, Abbreviation, License, SoftwareCoreference, AlternativeName
	Sentence: The Pearson correlation coefficient between the two analyses was calculated using IBM Statistical Package for Social Sciences software (SPSS , ver. 21 ; IMB SPSS Inc . , Chicago , IL , USA) and differences were considered as statistically significant if the p - Value was < 0.05 .
	Output:
	IMB {delimiter} Developer
	{record_delimiter}
	SPSS {delimiter} Developer
	{record_delimiter}
	Inc {delimiter} Developer
	{record_delimiter}
	. {delimiter} Developer
	`Example 2:`
	Words: GNU
	Entity types: No, Application, Developer, URL, Version, PlugIn, Citation, Extension, ProgrammingEnvironment, OperatingSystem, Release, Abbreviation, License, SoftwareCoreference, AlternativeName
	Sentence: FamSeq is a free software package under GNU license (GPL v 3) , which can be downloaded from our website : http://bioinformatics.mdanderson.org/main/FamSeq , or from SourceForge : http://sourceforge.net/projects/famseq/ .
	Output:
	GNU {delimiter} No

Table 2: Prompt Structure for relation identification between entities

Section	Prompt Details
Target Activity	You are an intelligent assistant that helps a human analyst to identify relations between entities in a sentence.
Goal	Given a pair of words identify if a relationship exists between them and the type of relationship based on a list of options.
Steps	1. You are given a list of words by / in the form of word / word / ...) a sentence and a token count list which contains the ids of each token. Start by identifying if there is a relationship between the words. 2. In case there is a relationship, from the list of options given in Relationship Possibilities, choose the type of relationship that best suits these two words. You must ONLY choose a relationship from the relationship list. - Make sure to take into account the full sentence to identify the type of relationship. - A relationship CAN NOT have itself as the head token and the tail token. Ex: URL_of \t6\t6 3. In case a relationship is found the output should be in the format of: relationship \ttoken_id\ttoken_id - make sure the token_id count starts at 0 and has a maximum equal to the number of spaces in the sentence. 4. However if no relationship is found between the entities the output should be "None". DO NOT ADD ANY EXPLANATION.

Examples	`Example 1:`
	Words: Remote / Software / http://softwaresecure.com / ProctorU / http://proctoru.com
	Relationship Possibilities: Abbreviation_of, AlternativeName_of, Citation_of, Developer_of, Extension_of, License_of, PlugIn_of, Release_of, Specification_of, URL_of, Version_of
	Sentence: Depending on the course , instructor , and exam type , DE MCS students have taken exams at a regional location monitored by a paid proctor or have taken exams using commercial online proctoring services such as Remote Proctor Now from Software Secure (http://softwaresecure.com) or ProctorU (http://proctoru.com) .
	Tokens: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
	Output: Developer_of \t41\t37;URL_of \t44\t37;URL_of \t49\t47
	`Example 2:`
	Words: Mutation / 3.10 / SoftGenetics
	Relationship Possibilities: Abbreviation_of, AlternativeName_of, Citation_of, Developer_of, Extension_of, License_of, PlugIn_of, Release_of, Specification_of, URL_of, Version_of
	Sentence: Sequence data were imported as AB 1 files into Mutation Surveyor v 3.10 (SoftGenetics , State College , PA) .
	Tokens: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
	Output: Developer_of \t14\t9;Version_of\t12\t9