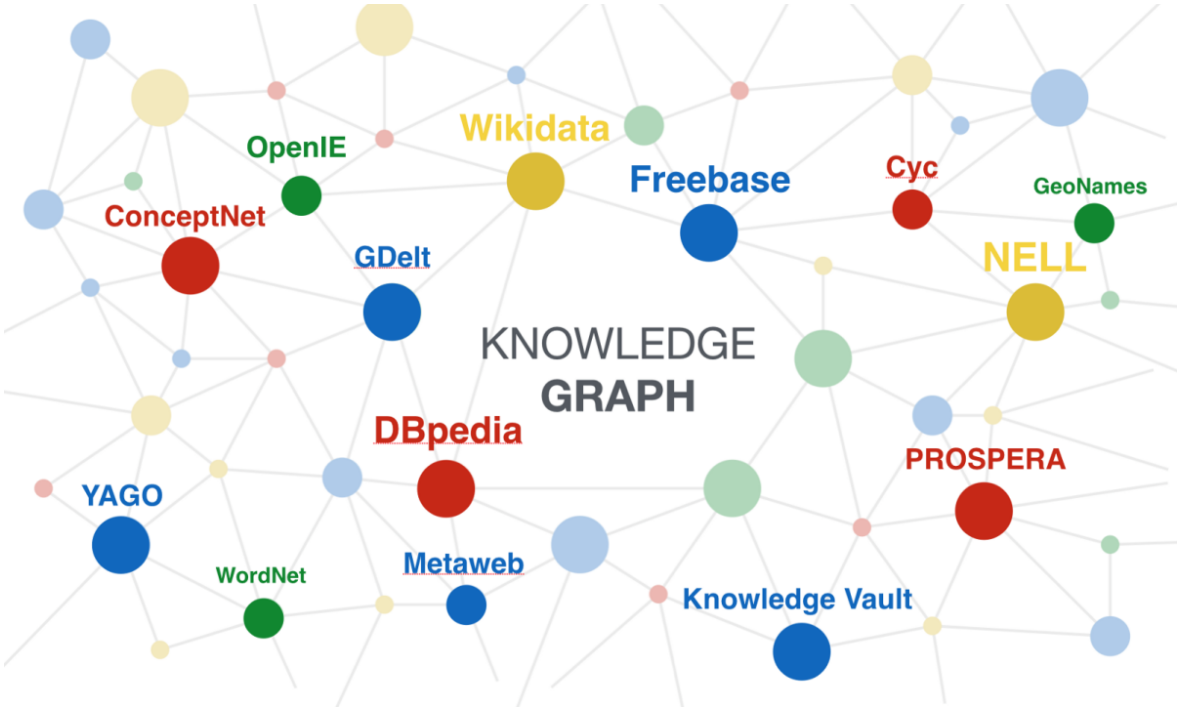
Knowledge Graphs and Natural Language: two faces of the same coin

Machine Learning Methods for Language Understanding

Andrea Papaluca



A thesis submitted for the degree of Doctor of Philosophy

ANU College of Engineering, Computing and Cybernetics
School of Computing

Australian National University

November, 2024

Declaration

This dissertation is an account of research undertaken between January 2021 and January 2025 at the College of Systems & Society (CSS, College of Engeneering, Computing and Cybernetics until December 2024), School of Computing, The Australian National University, Canberra, Australia.

The work presented in this thesis is that of the candidate alone, except where indicated by referencing literature or including acknowledgements in the text. It has not been submitted in whole or in part for any other degree at this or any other university.

This thesis is based on three papers published in refereed scientific journals or books. The development of ideas and research was undertaken with guidance from my supervisors Hanna Suominen, Sergio Rodríguez Méndez, Artem Lensky and Daniel Krefl and published papers presented in the thesis were written collaboratively with them. Below I describe my contributions to the research in each chapter and associated papers.

- Chapter 4 is the paper Papaluca, Krefl, Suominen et al. (2022). My contributions comprehend the formulation of the theoretical methods presented there and their implementation in practice, as well as the writing of the paper and the presentation of the results at the conference.
- Chapter 5 is the paper Papaluca, Krefl, Lensky et al. (2024). My contributions comprehend the formulation of the theoretical methods presented there and their implementation in practice, as well as the writing of the paper and the presentation of the results at the conference.
- Chapter 6 is the paper Papaluca, Krefl, Rodríguez Méndez et al. (2024). My contributions comprehend the formulation of the theoretical methods presented there and their implementation in practice, as well as the writing of the paper and the presentation of the results at the conference.

Andrea Papaluca
25 June 2025

Cover: Figure 1 from https://www.linkedin.com/pulse/knowledge-graphs-end-user-products-from-cyc-ai-part-daniel-kornev

Acknowledgements

First and foremost, I would like to express my deepest gratitude to The Australian National University (ANU) for awarding me the PhD scholarship that made this research possible. The financial support and academic resources provided in combination, by the Australian Government Research Training Program International Scholar ship and the Our Health in Our Hands (OHIOH) ANU initiative, have been fundamental in enabling me to pursue my studies and achieve my research goals.

A big appreciation goes to the Australian National Computational Infrastructure (NCI) as well, which provided me with the all the computational resources I needed for running the experiments in my research (NCI ANU project yv67).

I am also sincerely thankful to the Technology Innovation Institute (TII), in Abu Dhabi (UAE), for hosting me as an intern during my PhD journey. The opportunity to collaborate with outstanding researchers at TII allowed me to explore the field of research from a different perspective, which contributed to my growth as a person and a better researcher.

This PhD would not have been possible without the guidance of my advisors, whose support I deeply cherish. In particular, my gratitude goes to Hanna Suominen, Artem Lensky, Daniel Krefl and Sergio Rodríguez Méndez, whose expertise and mentorship have guided me throughout this journey.

Last but not least, I would like to thank my family and friends for their constant support and encouragement, which sustained me during the challenging times of my PhD route.

Abstract

Natural language forms one of the main means we humans use to communicate and can be therefore considered an important aspect of what we regard as "intelligence", as evidenced by, for example, the origins of founding Artificial Intelligence (AI) research in 1950s including Natural Language Processing (NLP) in the definition of AI. Thus, it is natural that, in the race to General AI, developing algorithms and models capable of understanding, processing and manipulating human language has acquired a pivotal role.

In recent years, the fields of NLP and Knowledge Graphs (KG) have witnessed remarkable advancements, each independently contributing to the enhancement of various AI applications. KGs, structured representations of knowledge, offer a powerful framework for organizing and connecting information, facilitating efficient knowledge retrieval and reasoning. On the other hand, NLP techniques enable machines to understand, generate and communicate in human language, bridging the gap between human and machine communication. The symbiotic relationship between KGs, NLP and Natural Language Understanding (NLU) is increasingly recognized as pivotal for advancing AI capabilities.

In my thesis, I delve into the interplay between these two domains, exploring how they complement each other to achieve deeper semantic understanding and more sophisticated reasoning, by proposing and evaluating machine learning methods that integrate them seamlessly. More specifically, the aim is to address the boundary connecting NLP/NLU to KGs, presenting which routes could be followed to achieve different levels of integration between the two and which degree of improvement has to be expected under each different scenario. The core of the thesis is constituted by three peer-reviewed papers (of which, two were best-paper awarded) that explore different aspects of the integration between KGs and Natural language.

Ranging from the more simple combination of graph and text embeddings through concatenation, to the deeper construction of a multi-modal aligned text-graph space and to the more high level usage of external KGs as reservoirs of commonsense knowledge, the thesis demonstrated how the integration of the two data modalities, and their corresponding encoder models, often enabled better modeling capabilities. This hybrid integration was beneficial in both language related tasks, such as Relation/Triplet Extraction (RE/TE) and Question Answering (QA), and graph associated tasks, such as Link Prediction (LP). Various standard datasets commonly used in literature on English NLP/NLU were adapted and enriched to allow for the joint processing of graph and text information, serving as benchmarks for quantitatively evaluating the improvement over the baseline. The set of tested models included Transformer-based architectures, ranging from their first iterations (e.g., Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT)), to the more recent Large Language Models (LLM) (e.g., Large Language Model Meta AI (LLaMA) and Falcon). On the graph side, instead, TransE embeddings as well as declinations of the Graph Convolutional Networks

(GCN), like Relational GCN (RGCN) and Compositional GCN (CompGCN), were considered.

Each one of these studies presented a different approach to achieve the integration and tested for different facets of reasoning and understanding natural language. However, they all demonstrated how pivotal the interaction of standard NLP models with KGs is for processing natural language. In particular, they evidenced as sometimes (specifically in the LLM case) the integration of an external KG both, leads to larger improvements and constitutes a cheaper approach, compared to, for instance, training and making use of larger, more complex language models. Therefore, the thesis offers a guideline for future research into how the integration of data modalities (and their corresponding encoder models) can enable better modeling capabilities in analysis tasks from answering questions to creating knowledge graphs.

Keywords: Knowledge Graphs, Natural Language Processing and Understanding, Knowledge and Language Representation Learning, Multi-modal Learning, Information Extraction, Link Prediction

List of Publications

- Papaluca, A., Krefl, D., Suominen, H. and Lenskiy, A. (2022). 'Pretrained Knowledge Base Embeddings for improved Sentential Relation Extraction'. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop. Dublin, Ireland: Association for Computational Linguistics, pp. 373–382. DOI : 10.18653/v1/2022.acl-srw.29
- Papaluca, A., Krefl, D., Lensky, A. and Suominen, H. (2024). 'Contrastive Language-Entity Pre-training for Richer Knowledge Graph Embedding'. In: Proceedings of the 4th International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI). Jeju Island, South Korea.
  - Invited as one of the 15 papers from this ICPRAI2024 conference for publication as a special issue in the Language Processing, Pattern Recognition and Intelligent systems series (ISSN (print): 2661-4316 | ISSN (online): 2661-4324) by World Scientific Publishing, Singapore.
- Papaluca, A., Krefl, D., Rodríguez Méndez, S., Lensky, A. and Suominen, H. (2024). 'Zero- and Few-Shots Knowledge Graph Triplet Extraction with Large Language Models'. In: Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024). Ed. by R. Biswas, L.-A. Kaffee, O. Agarwal, P. Minervini, S. Singh and G. de Melo. Bangkok, Thailand: Association for Computational Linguistics, pp. 12–23.

Code Realeases

The following code implementations were released under the MIT license as part of this thesis:

- https://github.com/BrunoLiegiBastonLiegi/Pretrained-KB-Embeddings-for-RE
- https://github.com/BrunoLiegiBastonLiegi/CLEP
- https://github.com/BrunoLiegiBastonLiegi/KG-TE-with-LLMs

Awards

- Best Paper Award at ICPRAI2024, third prize.
- Best Paper Award at KaLLM (ACL2024), first prize.

Talks and Poster Sessions

- Poster presentation at the ACL 2022 conference (Student Research Workshop), Dublin, Ireland.
- Talk at the ICPRAI 2024 conference, Jeju Island, South Korea.
- Talk at the ACL 2024 conference (KaLLM), Bangkok, Thailand.
- Poster presentation at the ACL 2024 conference (KaLLM), Bangkok, Thailand.

Contents

Figures

Tables

Abbreviations

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
CompGCN	Compositional Graph Convolutional Networks
EL	Entity Linking
GCN	Graph Convolutional Networks
GPT	Generative Pre-trained Transformer
IE	Information Extraction
KB	Knowledge Base
KG	Knowledge Graphs
LLaMA	Large Language Model Meta AI
LLM	Large Language Model
LP	Link Prediction
ML	Machine Learning
NED	Named Entity Disambiguation
NER	Named Entity Recognition
NLP	Natural Language Processing
NLU	Natural Language Understanding
OOD	Out of Distribution
QA	Question Answering
RE	Relation Extraction
RGCN	Relational Graph Convolutional Networks
TE	Triplet Extraction

Chapter 1

Introduction

Knowledge Graphs (KGs) offer a very compact, dense and efficient representation of natural language. As it will be discussed in more detail in Chapter 2, the core propositional information contained in a sentence (unstructured information) can be distilled and organized in a more concise graph form (structured information), which makes procedurally processing such information more convenient[1]. In these propositional terms, natural language and KGs implicitly offer two alternative views of the same abstract concept, message or piece of information: they are two faces of the same coin.

Communication, for instance, can be depicted schematically as the process of giving a physical realization to an abstract concept and, then, decoding it back to capture the original core information. We humans, are more used to natural language as a mean of encoding and decoding information, but, as it turns out, machines are not. Natural Language Processing (NLP), indeed, tries to provide computers with tools and techniques for understanding, and thus encoding and decoding, natural language.

KGs, however, thanks to the structured representation they offer, promise to be more natural to machines: all the information is explicitly presented and rigorously organized in the form of entities and relations between them. No room is left for ambiguities, interpretations or creative alternative structures, as each relationship is clearly stated and uniquely identified.

Furthermore, our communication is often imbued with large amounts of background knowledge, that can drastically shift the meaning of a message and which is complicated to capture solely with methods that model the grammatical and semantical aspects of language. Therefore, the large amount of information KGs store inside of them, could be very helpful to NLP models, for instance, to give a more accurate interpretation of natural language.

[1] A short disclaimer is due, however, for clarity and completeness. Naturally, the previous statement refers to the propositional content of language only. Natural language comprises a complex non-propositional structure as well. Nuances given by tone, formality, rhetorical formulations and many others, are very hard to capture through a KG representation. But even simple everyday phrases such as “How are you?”, “Please close the window.”, “Wow, that’s amazing!” do not assert facts and, thus, have no equivalent KG formulation. Therefore, here and in the rest of the thesis, especially for the KG-language parallelism, I will always implicitely refer to natural language in its propositional terms.

Indeed, the past few years have witnessed the flourishing of the field that lives on the boundary between NLP and KGs, and in the next section I will try to contextualize the aim of this thesis in that perspective.

1.1 Thesis Objectives & Key Contributions

This PhD thesis in Computer Science aims to delve into the intersection of these two vastly studied fields, NLP and KGs, which now more than ever, are increasingly recognized as pivotal in advancing Artificial Intelligence (AI).

In detail, the thesis revolves around the matter of how the two modalities can complement each other to achieve deeper semantic understanding and sophisticated reasoning, thus aiming to answer the following research questions:

- How can NLP and KG models be effectively integrated in AI systems?
- What are the comparative strengths and weaknesses of different integration approaches (shallow, deep, and retrieval-based) between NLP and KGs?
- How does the integration of KGs into NLP models (and vice versa) affect the performance of language- and graph-related tasks when using different model architectures and scales?

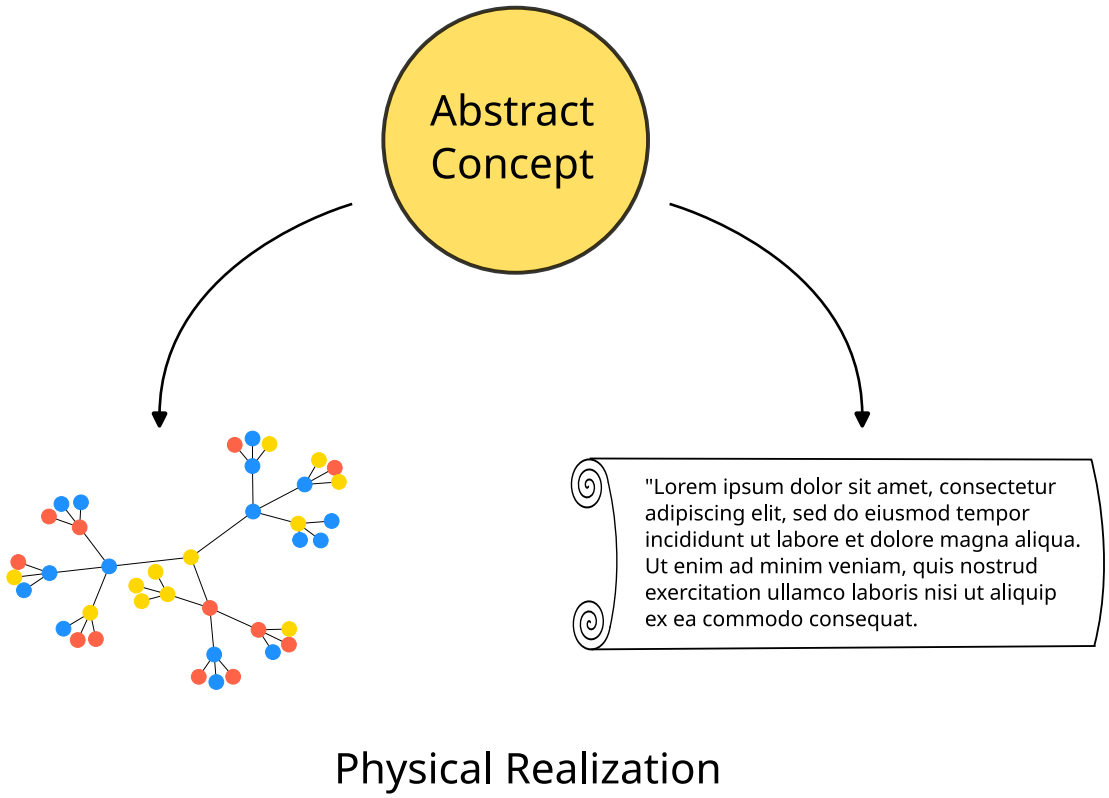


Figure 1.1: Physical realization of an abstract concept. KGs and natural language provide alternative views of the same core concept or message.

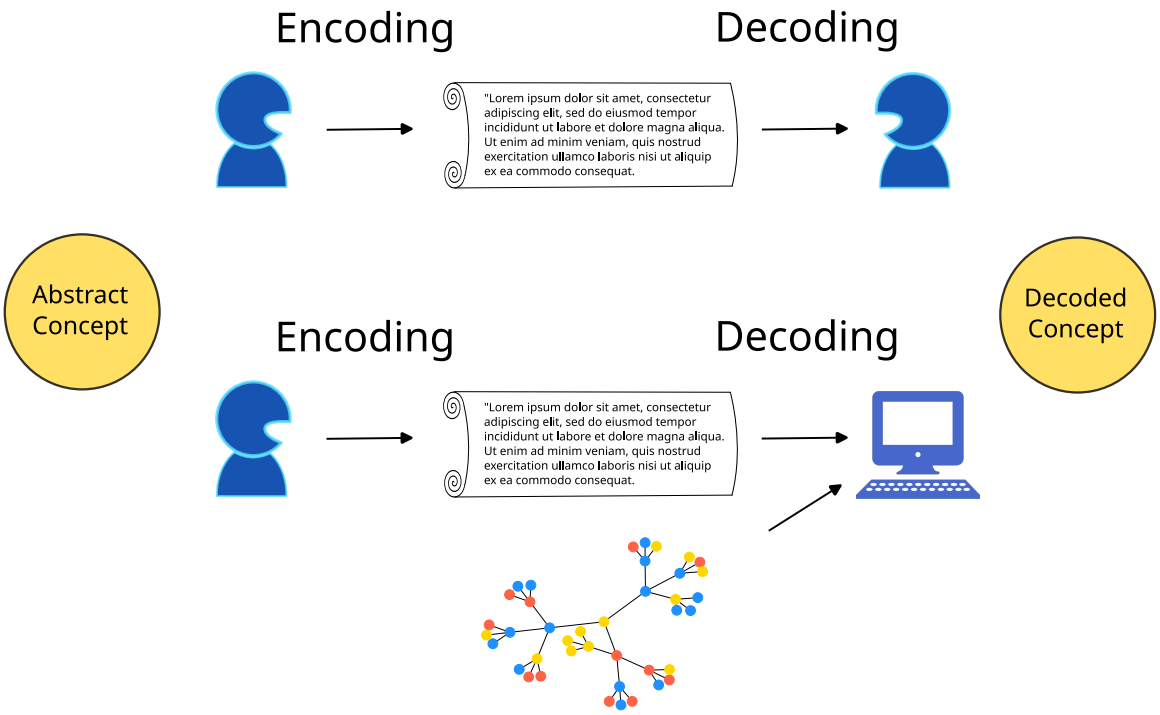


Figure 1.2: Communication modeled as the process of encoding an abstract concept in some physical form and decoding it back. Natural language is the mean we humans are more familiar with to encode a message we would like to communicate. Machines, however, are not naturally equipped to natively communicate in natural language, other physical forms, such as Knowledge Graphs for example, could be more suitable to them. In practice, communication between humans and machines could be mediated through a KG: the information encoded in Natural language could be decoded thanks to the combination of NLP techniques and KG resources.

- Can the integration of external KGs with language models offer a cost-effective alternative to scaling up model size in achieving high performance?

Or, in other words, the main objectives can be schematically summarized by the following key points:

- **Examine Integration Approaches**:

  Investigate various methods for integrating NLP/NLU models with KGs, ranging from deep low-level to more high-level approaches.

- **Evaluate Improvements in AI Tasks**:

  Quantitatively assess how integrating NLP and KG models enhances performance in real world problems, for both language-related and graph-associated tasks.

- **Develop Benchmarks and Datasets**:

  Adapt and enrich standard datasets for English NLP/NLU to support joint processing of textual and graph data, enabling thorough evaluation of integration approaches.

- **Analyze Model Effectiveness**:

  For each different integration approach, evaluate how the choice of the base model affects the result, both from the NLP side and from the KG perspective.

- **Demonstrate Cost-Effective Integration**:

  Highlight how integrating external KGs with language models often leads to significant performance gains at a relatively reduced computational cost compared to solely relying on larger, more complex models.

- **Provide Future Guidelines**:

  Offer a roadmap for advancing research on NLP and KG integration, emphasizing their synergy to improve diverse AI applications like reasoning, question answering, and KG construction.

In practice, the list above translated to the following main contributions:

- **Integration Approaches and their impact on NLP/KG tasks**:

  Three different architectures were proposed to incorporate KGs in NLP pipelines (and vice versa): a shallow embedding-based integration (Chapter 4), a deep multi-modal alignment procedure (Chapter 5), and an augmented retrieval approach where the KG acts as a reservoir of commonsense knowledge (Chapter 6). A thorough evaluation of the impact provided by each approach was performed on real world AI problems: Relation Extraction (RE), Triplet Extraction (TE), Question Answering (QA), and Link Prediction (LP). Several baseline models were tested to validate the effect on architectures with a different complexity and number of parameters: the Bidirectional Encoder Representations from Transformers (BERT (Devlin et al., 2018)), General Pretrained Transformer (GPT (Radford, Wu et al., 2019a)) and advanced Large Language

Models (LLMs) (Large Language Model Meta AI (LLaMA (Touvron et al., 2023)) and Falcon (Penedo et al., 2023)) from the NLP side, whereas translational graph embeddings (TransE (Bordes, Usunier et al., 2013)) and Relational and Compositional Graph Neural Networks (RGCN (Schlichtkrull et al., 2017) and CompGCN (Vashishth, Sanyal et al., 2019)) from the KG side.

- **Enriched Datasets**:

  Popular NLP and KG datasets have been extended (and made available under the MIT license) to allow for testing and benchmarking the integration:

  - the NYT (Riedel et al., 2010a) and Wikidata (Sorokin and Gurevych, 2017) datasets were included with the graph embeddings of each one of their entities and relations
  - descriptions were added to all the entities of the FB15k-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018) and YAGO3-10 (Mahdisoltani et al., 2015) datasets
  - a graph structure was extracted and attached to both the WebNLG (Gardent et al., 2017) and NYT (Riedel et al., 2010a) corpora

In the next section, a brief signpost outlining the content of the chapters is provided to facilitate the navigation through the contents of the thesis.

1.2 Thesis Outline

In the upcoming chapters (4, 5, 6), three works are presented that experiment with the mutual integration of Language modeling and KGs methods for various tasks. Before that, however, I provide a brief overview in Chapter 2 of the background they rest on.

I shortly introduce the reader to the problem of processing natural language, by presenting some core NLP concepts and retracing the history of the field going through some of its most relevant models. The concept of "Knowledge Graph" is also more formally introduced in the chapter, presenting its main characteristics and its role with respect to natural language. Two practical example cases of natural language adjacent tasks, that make use of the structure KGs provide, are also showcased.

Then, in Chapter 4, a model for Relation Extraction (*c.f* Section 2.7) is proposed, that combines a transformer based language model (*c.f* Section 2.3) with some pre-trained KG embeddings. The proposed architecture demonstrates the benefit of supplementing the purely textual representation of language, with topological features that come from the graph structure of a KG. A detailed study is also conducted, investigating under which circumstances the additional graph features are expected to be more impactful and to which degree they contribute depending on the data used for training. This work provides a first evidence of the symbiotic relation that connects natural language to KGs, showing how the features they separately encode can be transferred and made mutually available to allow for better language understanding capabilities.

Chapter 5 builds on the preceding work and pushes the concept of graph and text integration to the next level. An architecture inspired by the CLIP (Radford, Kim et al., 2021) model is proposed there, that aligns a KG encoder and a text encoder to learn a multi-modal joint graph-text space. The alignment is obtained by learning a mapping that enforces the closeness of KG nodes and their corresponding textual descriptions in the embedding space. This, among other things, is shown to enable link prediction out of the box, without the need of any additional finetuning. Furthermore, the proposed procedure of alignment represents an effective method for pre-training a graph encoder and grants it access to some of the semantical features learnt by the text encoder, as models pre-trained this way demonstrate stronger link prediction capabilities.

The problem of KG triplet extraction, instead, is explored in Chapter 6. The task consists in extracting *subject-predicate-object* relations from sentences in the form of KG triplets, *i.e.* with subject and object being entities of the graph and predicate one of its supported edges. This therefore condenses NER, EL and RE (*c.f* Sections 2.5-2.7) in a single tasks posing a formidable challenge and enabling direct KG construction from unstructured text. In particular, the work assesses the capabilities of a wide range of LLMs to perform the task in the zero-shots and few-shots settings, namely without any finetuning involved and with no (zero-shots), or just a few (few-shots), examples provided to them at inference time. Experiments indicate, however, that LLMs struggle to accurately extract the triplets in these settings. Nevertheless, the work demonstrates that when they are coupled to a KG that acts as a reservoir of background knowledge, their accuracy substantially improves. Indeed, simply providing them with the triplets found in the Knowledge Base that are most relevant to the input sentence, allows them to achieve zero-shots performance comparable to some of the standard fully trained baselines. Moreover, organizing the information coming from the KG as pairs of sample sentences and triplets to be extracted from them, helps the LLMs further and pushes them closer to finetuned *State-of-the-Art* models. The study concludes with an analysis of the impact the model's size has on the final triplet extraction capabilities and how they scale with the quality of the supplied Knowledge Base information.

The final remarks are presented in Chapter 7, together with the limitations of the current work and some ideas for future developments.

Chapter 2

Background

This Chapter starts by showcasing the capabilities of modern language models. Then, to better understand their functioning and how we reached such impressive results, I provide a brief overview of NLP. Ranging from the basic concepts of vocabularies and tokenization, passing through the more rudimentary n-grams language model and the first context-based embedding models, such as Word2Vec, until the introduction of the transformer model that represented a corner stone in the evolution of language processing. I thus try to retrace here, the history of the developments that brought us here where we stand today, and build some intuition of the foundations that power the core of modern models.

In the second half of the Chapter, instead, I shift the focus more to the information and knowledge processing theme, introducing the concept of KGs. These data structures are closely related to language, as they provide, broadly speaking, a compact and dense representation of it, but more precisely of the information language carries. First, the NLP tools are presented, which allow for the distillation of a KG structure starting from an unstructured text. Namely, the Named Entity Recognition and Linking tasks are discussed as well as the Relation Extraction one. Second, two example use cases for KGs, Link Prediction and Question Answering, are illustrated to showcase the potential of this representation connected to standard language reasoning problems.

Finally, this concludes with an overview of the chapters that will follow, guiding the reader through different approaches to the integration of KGs and NLP methods.

2.1 Natural Language Processing

In the vast landscape of human communication, few mediums rival the richness, complexity, and subtlety of natural language. A product of centuries of cultural evolution and societal development, natural language serves as the cornerstone of human interaction, facilitating the exchange of ideas, emotions, and knowledge with unparalleled nuance. From the fervent debates of ancient philosophers to the rapid-fire conversations of modern social media platforms, natural language stands as a testament to humanity's innate ability to convey meaning through words.

However, despite its ubiquity and indispensability, natural language remains a formidable challenge for computational systems to comprehend and manipulate effectively. The intricate interplay of syntax, semantics, pragmatics, and context presents a daunting barrier for machines seeking to navigate the complexities of human language. This challenge is further compounded by the inherent ambiguity, variability, and idiosyncrasy inherent in natural language, rendering traditional rule-based approaches inadequate for capturing its nuanced intricacies.

Enter NLP, a burgeoning field at the intersection of computer science, linguistics, and artificial intelligence. At its core, NLP aims to bridge the chasm between human language and machine understanding, empowering computers to analyze, interpret, and generate natural language text with human-like proficiency. Through a diverse array of algorithms, techniques, and methodologies, NLP endeavors to unlock the wealth of knowledge encoded within textual data, enabling machines to comprehend, reason about, and interact with human language in meaningful ways.

The importance of NLP in today's digital age cannot be overstated. With the exponential growth of digital content across various domains, ranging from social media streams and online forums to scientific literature and news articles, the ability to harness and extract insights from vast volumes of textual data has become a strategic imperative for individuals, organizations, and societies alike. From facilitating information retrieval and sentiment analysis to powering virtual assistants and machine translation systems, NLP underpins a myriad of applications that shape our daily lives and drive innovation across diverse fields.

In recent years, the synergy between NLP and KGs has emerged as a potent paradigm for unlocking the latent potential of textual data. By leveraging the structured representation of knowledge embodied in knowledge graphs and the linguistic richness of natural language, researchers have made significant strides in advancing the frontiers of information extraction, semantic understanding, and knowledge discovery. This symbiotic relationship between NLP and KGs holds the promise of revolutionizing how we organize, access, and reason about information in an increasingly interconnected and data-driven world.

In this thesis, we embark on a journey to explore the nexus of NLP and KGs, delving into the intricacies of their integration and the transformative impact it holds for advancing our understanding of textual data. Through a combination of theoretical insights, experimental methodologies, and practical applications, we seek to unravel the mysteries of human language and pave the way for a new era of intelligent

Figure 2.1: A simple tokenization example, credit for the figure to Utkarsh Kant.

systems capable of harnessing the full spectrum of human knowledge encoded within the vast expanse of textual data.

Sounds good, right? Yes, maybe the style is a bit pompous for my taste and many sentences are quite generic to the point they sound like a cliché, but overall I think the previous paragraphs provide a nice glimpse into what NLP is and why it is important. What if I told you that all the text preceding this paragraph, in this section, was generated by a machine prompted with the following input:

```
I am writing the introduction of my computer science PhD thesis
about Natural Language Processing and Knowledge Graphs. I want
to start the introductory chapter talking about what is natural
language and why Natural Language Processing is important.
Can you provide some inspiration?
```

It is impressive to witness how much progress was made in such a short time over the past recent years, especially considering, as the machine correctly reported as well, how complicate human language is for computers. Therefore, in the next few sections I would like to provide a brief outline of some of the core NLP concepts and techniques that contributed to bring us to the stage we are today. This quick glimpse by no means intends to be complete nor exhaustive, but rather tries to build an intuition of how language modeling works. The Stanford NLP book by Jurafsky and Martin (2024) has been a large source of inspiration for me in writing the next sections. The book represents a standard resource for all those looking for a comprehensive discussion on NLP and, thus, I would redirect the reader there for a more detailed presentation of what I discuss here.

2.2 A Machine Readable Representation of Language

As it happens for numbers, computers need to have available an internal representation of the single units of information we want to process (Shannon, 1948). Numbers, for example are represented as sequences of the binary units $\{0, 1\}$ and operations between them are reproduced as compositions of binary operations on these *bit-strings*. Similarly, in order to process natural language, we first need to understand which

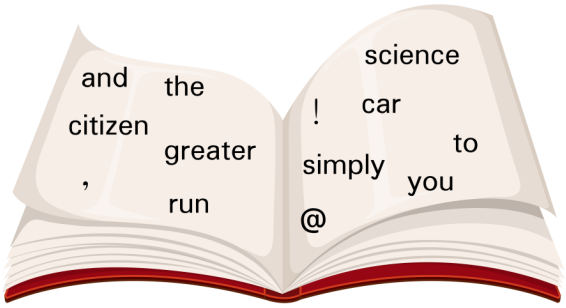


Figure 2.2: A vocabulary defines the lexicon a language model has available. Credits for the figure to graphicsrf.

are the single units of information composing it, and then construct a paradigm to build a machine compatible representation of them.

The easier and most natural way to split a natural language textual snippet, is by separating all of the particles divided by a white space that constitute it (Salton and Lesk, 1965). This way, we isolate each single unit that is contributing to form the original message which can then be processed singularly, one by one, to elaborate the text as a whole. This process of separating the basic units forming a text is commonly known as "Tokenization" and lies at the basis of natural language processing. Clearly, this white space separation is a quite trivial, and often insufficient, method for tokenizing sentences and additional rules can be implemented on top of it, like splitting special characters and punctuation. Additionally, even more sophisticated methods might also contemplate, for instance, a lemmatization based separation, *i.e.* where the root of each word (lemma) is separated from its inflected form, or character level tokenization (Chomsky, 1957; Fellbaum, 1998).

However, regardless of the tokenization technique used, the overall idea remains unchanged: isolating and identifying the fundamental units of a text. The identified units can then be mapped to unique numerical identifiers in a vocabulary (Fellbaum, 1998; Salton and Lesk, 1965), effectively allowing for the translation of a natural language message into a sequence of integer numbers. Naturally, vocabularies are taken to be large but still finite in size, meaning that *out-of-vocabulary* tokens might be encountered. Still, they can be addressed via sub-token or character level decomposition (similar in spirit to lemmatization) or, as a last resort, marking them as "unknown" with a special identifier (UNK).

Nonetheless, numbers alone cannot help in assigning a meaning to the words. The conversion to numbers is useful for making processing easier, but how do we actually build a sensible representation for the single tokens?

The answer to this appears to be found in the concept of "context". Namely, there is a clear correlation between the meaning assigned to a word and the context it appears in, *i.e.* the words surrounding it (Deerwester et al., 1990; Firth, 2020; Harris, 1981; Mikolov et al., 2013). Consider for instance the sentence

```
"I ____ myself eggs for dinner."
```

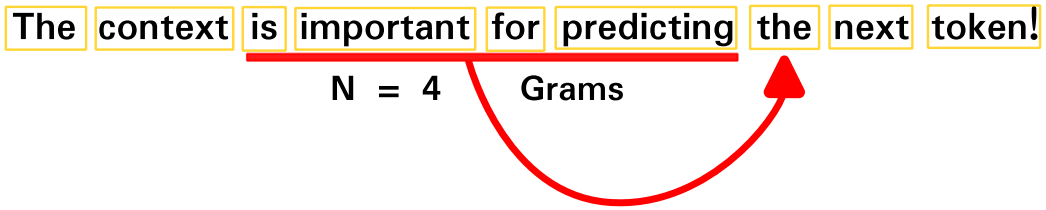


Figure 2.3: A N-grams model with $N = 4$, the four highlighted tokens are used to predict the following one.

It is not hard to imagine which words might be adequate to fill the empty space: the missing word is directly following a pronoun, suggesting it might be a verb, and appears to be connected with eggs and dinner, making it is easy to imagine it to be related to some kind of food preparation, such as "cook", "prepared", "make", etc... . Therefore, this hints at making use of the context to build the token representation we need.

However, we are able to infer the missing word because we are aware of the meaning of the surrounding words. In the case of a language model, that does not possess our semantical background knowledge, we need instead to recur to some statistics. For instance, one of the earliest class of models developed for language generation are the so-called Bayesian N-grams models (Brown, Cocke et al., 1990; Shannon, 1948). These are statistical models that assign a probability to all the words w in a vocabulary given some history h, consisting of N other preceding words

$$P(w|h) = P(w|w_1, w_2, \ldots, w_N)\,. \tag{2.1}$$

By iterating over a large enough corpus, we can model the distribution P by simply updating a Maximum Likelihood Estimator (MLE) to predict the next token given the preceding ones. The statistical co-occurrence of words in sentences will gradually update the estimator, which will develop an internal representation that embeds words with similar contexts, and thus meanings, closer.

The final estimated distribution P then encodes, in some sense, the "meaning" and provides a quantitative representation for the words that enables language processing: sentence completion and generation, next word suggestion, similarity calculation, etc... , can all be achieved through some manipulation of the embedding vectors associated with each word.

In particular, next token prediction is just one of the most naive methods to leverage context for word embedding. Two other well known and widely used approaches, are the so called Continuous Bag of Words (CBoW) and Skip-gram paradigms that were introduced with the Word2Vec model by Mikolov et al. (2013). They build on top of the naive N-grams model, but with the fundamental difference that they do not only look at the history of the current token, but also at its "future". Namely, having defined a window of size $2n$, the context is composed by, both, the n preceding and following tokens to the current one.

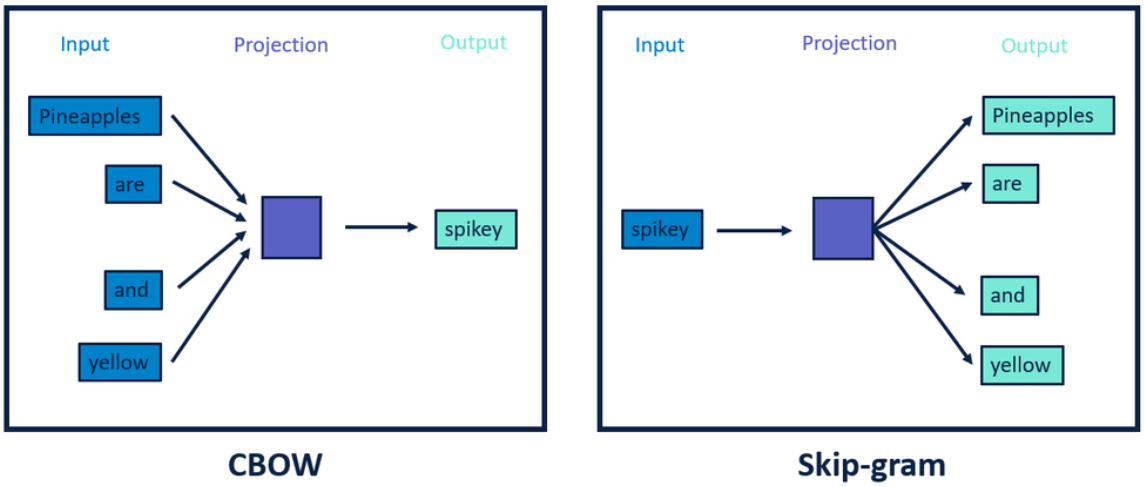


Figure 2.4: A comparison of the CBoW and Skipgrams paradigms: the former predicts a word starting from the surrounding words in the context, whereas the latter infers possible context words starting from an input. Credits for the figure to Sydney Firmin.

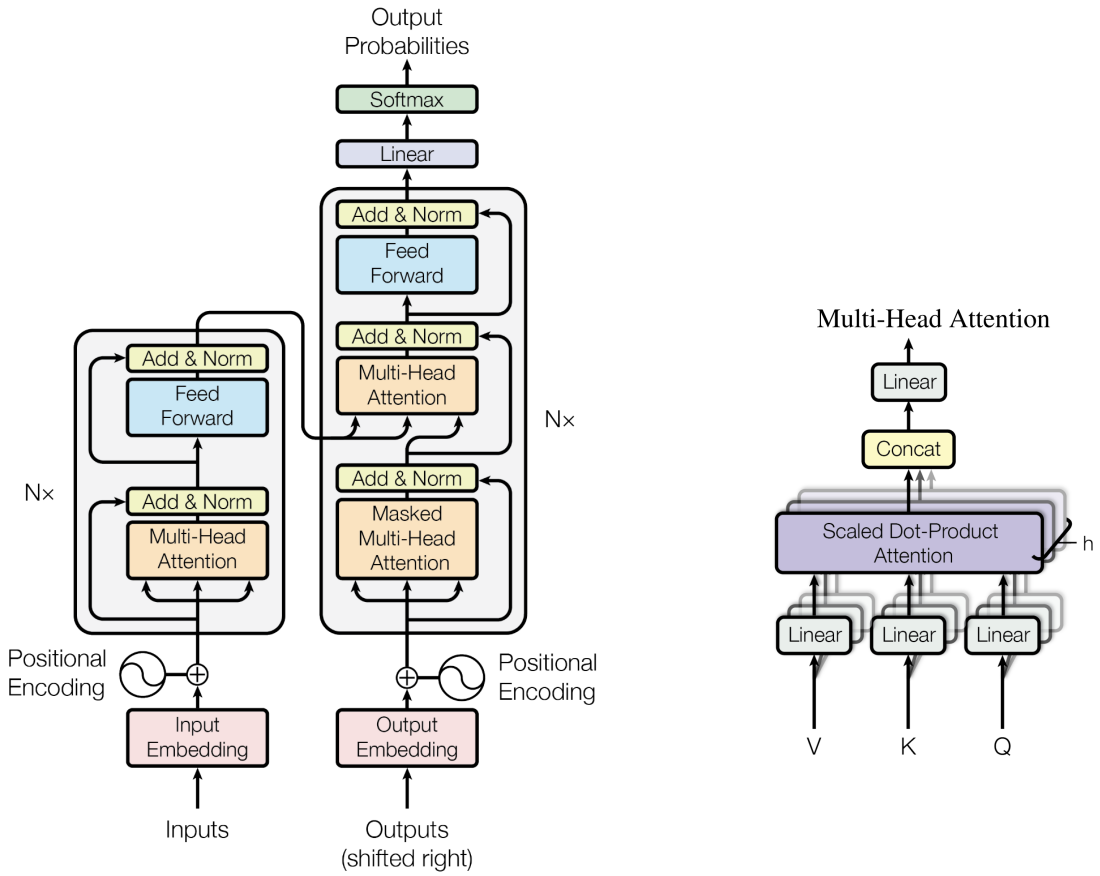
In the CBoW scheme, then, the current token is masked and has to be predicted back starting from the tokens contained inside of the window, taken in mixed order. Whereas, Skip-gram implements the opposite mechanism. The complete context window is masked and the task is to predict which token belonged to the context in the first place, given the current one.

All these word embedding techniques proved very successful in building a semantics aware representation of language, and embodied the *State-of-the-Art* in language modeling for several years. However, they suffer from a quite relevant shortcoming: the quantitative embedding they produce is fixed. In detail, each word in the predefined vocabulary is assigned a unique multidimensional array that represents it in the semantic embedding space, regardless of the context it was found in. The same word, appearing in two separate sentences, which, maybe, refer to completely different circumstances, is going to always take the same value. The value will reflect the average over all the contexts the word was encountered in through the corpus, surely, but still this kind of behaviour is suboptimal.

If you think about it, one of our main assumptions, in the first place, was indeed that the meaning of a word had been strictly dependent on its context. Whereas, this mechanism seems to go a little bit in the opposite direction. Just to give an example, think about the word "right" and the two sentences "I think I am right." and "You have to take the third right after the crossing.". Clearly, the use of "right" is different in the two sentences and we would like the semantic embedding space to reflect this.

2.3 Paying Attention to the Context with Transformers

Few years after the introduction of the Word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) models, a new architecture called "transformer" was introduced by Vaswani et al. (2017), which accidentally contained a solution to this context-independence problem. It was originally proposed for dealing with the task



(a) transformer architecture **(b)** attention mechanism

Figure 2.5: The transformer model: (a) the complete transformer architecture consisting of an encoder and decoder modules based on Feed Forward Neural Network and Attention layers alternated, (b) in detail representation of the *query-key-value* mechanism powering the attention layers. Credits for the figures to Vaswani et al. (2017).

of language translation, from this comes the origin of its name, but was very soon adapted in general for word embedding and, thus, declined to basically all the other NLP tasks. The majority of the modern Large Language Models, indeed, are still making use of the same underlying architecture, that was tremendously scaled up in size to reach the impressive results we are witnessing today.

In Figure 2.5 it is illustrated the transformer architecture. The model is composed of an encoder block (left) and a decoder block (right), which share a very similar inner architecture: some standard Feed Forward Neural Networks (FFNN) combined with what has been called an "attention" layer. Notably, a positional encoding is needed to inject information about the order of the tokens to the model. The FFNNs comprise two linear transformations chained through a nonlinear activation, which learn a gradually deeper representation of the tokens. However, the key to the success of the transformer lies, indeed, in the newly introduced attention mechanism.

The basic idea is that these layers allow the model to learn, for each input token, which of the of the other tokens in the sentence is more relevant in processing the current one, *i.e.* where to shift the attention. Indeed, if in the previous models the "attention"[1] was distributed homogeneously across the whole context, in this case each token in the context is going to be assigned an attention weight $w^{(a)}$ that weights its contribution to the final embedding. As a consequence, the same word, but appearing in two different sentences, and thus contexts, will be embedded differently in general, hence enabling a more flexible and fine-grained semantic representation.

To provide a little more details, a *query* (Q), *key* (K) and *value* (V) mechanism was proposed to power these attention layers. Namely, alongside the standard semantic representation, which is the final objective, three other embeddings for Q, K and V are learnt during training. The query Q provides a representation of the current input we are trying to calculate the attention scores for, whereas the key K measures the relevance of a given token in relation to the current query Q. The combination of the two $Q \cdot K$, therefore, computes the "activation" for each input, that weights its actual attention value V.

For instance, suppose that we have some input tokens

$$(t_0, t_1, \ldots, t_n) \tag{2.2}$$

and we would like to compute the attention weights related to the token t_i. We will have, as query, a vector q_i and, as keys, the vectors

$$K = (k_0, k_1, \ldots, k_n)\,. \tag{2.3}$$

If we multiply each key for the query we obtain the sequence of activations

$$q_i \cdot K = (q_i \cdot k_0, q_i \cdot k_1, \ldots, q_i \cdot k_n)\,, \tag{2.4}$$

[1] Here attention is meant in its wider term, as the weight or importance associated to each token in the context, clearly the notion of attention came with the Transformer and was not available to previous models.

that, once normalized through some function σ, can then be used to weight the contribution of the values

$$V = (v_0, v_1, \ldots, v_n) \tag{2.5}$$

and obtain the actual attention to the token t_i for all the inputs (including t_i itself)

$$w_i^{(a)} = \sigma(q_i \cdot K) \cdot V\,. \tag{2.6}$$

In practice, this whole procedure can be easily vectorized by organizing Q, K and V in matrices

$$W^{(a)} = Softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \cdot V\,, \tag{2.7}$$

where σ is taken to be the $Softmax$ function in order to generate a categorical distribution that sums to 1. Furtermore, the activation is divided by the square root of the dimension of the key vectors, d_k, to improve the stability of the gradients.

Moreover, the mechanism just presented is often replicated in parallel over multiple "heads", *i.e.* multiple separated and different versions of the query, key and value matrices. This leads to a set of different attention weights $W^{(A)}$, one for each head, which provide alternative views of the input that are finally combined to produce the global attention. The main advantage of this approach is to reduce the risk of obtaining attention weights highly biased towards a single, or very few, tokens. As the Q, K and V matrices are randomly initialized, this promotes in some sense a larger diversity and allows to highlight different features or aspects of the input separately.

As mentioned in the beginning of this section, the transformer architecture was originally proposed for the language translation task only, but its impact on the whole field has gone way beyond that. Indeed, shortly after its introduction, people started realizing how effective it was in processing language. Its capability of capturing long range correlations in sentences, which has always been one of the major shortcomings of Recurrent Neural Networks (Jordan, 1986; Rumelhart and McClelland, 1987), granted it access to richer word representations. For this reason, two famous models that build on the same transformer architecture were proposed few years later, the *Bidirectional Encoder Representations from Transformers* (BERT) (Devlin et al., 2018) and the *Generative Pre-trained Transformer* (GPT) (Radford, Wu et al., 2019a).

The former makes use of the encoder block of the transformer, thus making it more suitable for language representation alone, whereas the latter relies on its decoder block, which natively allows for language generation. They both are trained on large corpora to learn the underlying distribution of language (*c.f* 2.1), either by Masked Token Prediction (BERT) or Next Token Prediction (GPT). The robust language embedding capabilities they acquire through this large training step, allows them to easily deal with any language related task by the means of some "finetuning". Namely, training, on limited amount of data, some task-specific layers on top of the token embeddings they provide yields impressive results already, eliminating the need to train totally new models from scratch for each specific objective. This computationally and data inexpensive approach, indeed, has been the standard

approach to NLP for years. Only recently, the advent of generative Large Language Models (LLM), which are still transformer-based, started shifting the center of mass of the field.

This short introduction, hopefully, provided an intuition of which kind complications are faced when trying to process human natural language through a computer programme, and further, which routes can be taken to address the task. The unstructured nature of language and the large amount of background knowledge required to correctly extract the information it carries, however, still pose a serious challenge for machines. For this reason, simultaneously with the development of improved models for language processing, other forms of organization of the information, affine to language but more suitable for machine processing, have been explored. The so called "Knowledg Graph" is one of these, and in the next sections I will provide an outline of its characteristics and the new possibilities it enables.

2.4 Efficient Knowledge Representation with Knowledge Graphs

We have just discussed, in the previous sections, as natural language features an intrinsically unstructured form. The information it contains is not easily accessible recursively following an algorithm, and this makes it harder for machines to interprete it. Therefore, it would be useful to have available a structured representation of the information contained in the sentences. This is provided by the concept of a KG (Bollacker et al., 2007; Vrandečić, 2012).

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a structure made of vertices $\mathcal{V}$ and edges $\mathcal{E}$. Each vertex $v \in \mathcal{V}$, also called node, may represent any generic object and each edge $\epsilon \in \mathcal{E}$ connecting a pair of nodes (v_1, v_2) indicates that a relationship between them exists. In the most simple case the vertices do not possess any attribute making them differ from each other. Their only characteristic is the numerical index used to uniquely identify them. Similarly, a unique type of edge exists and no direction is specified in the simple case (*undirected graph*), which only assesses the presence of a connection between two vertices.

KGs are a particular category of graphs. Their vertices represent real world entities such as geographical locations, organizations or companies and historical or public figures for instance, but also more abstract concepts like time or altitude. Furthermore, additional attributes can be associated with each entity. For example, it is quite common for them to possess a short description and their translation in different languages.

KGs also allow for a more detailed description of the edges. Firstly, the connections are directed in this case, which means that the relationships have a well defined intrinsic order and they are not symmetric under the exchange of entities anymore. It is quite common, indeed, to define for each directed relation r in the graph, the corresponding inverse relation r^{-1}. The latter provides a backward view of the relation, which could also be seen as a passive form of the original relationship in

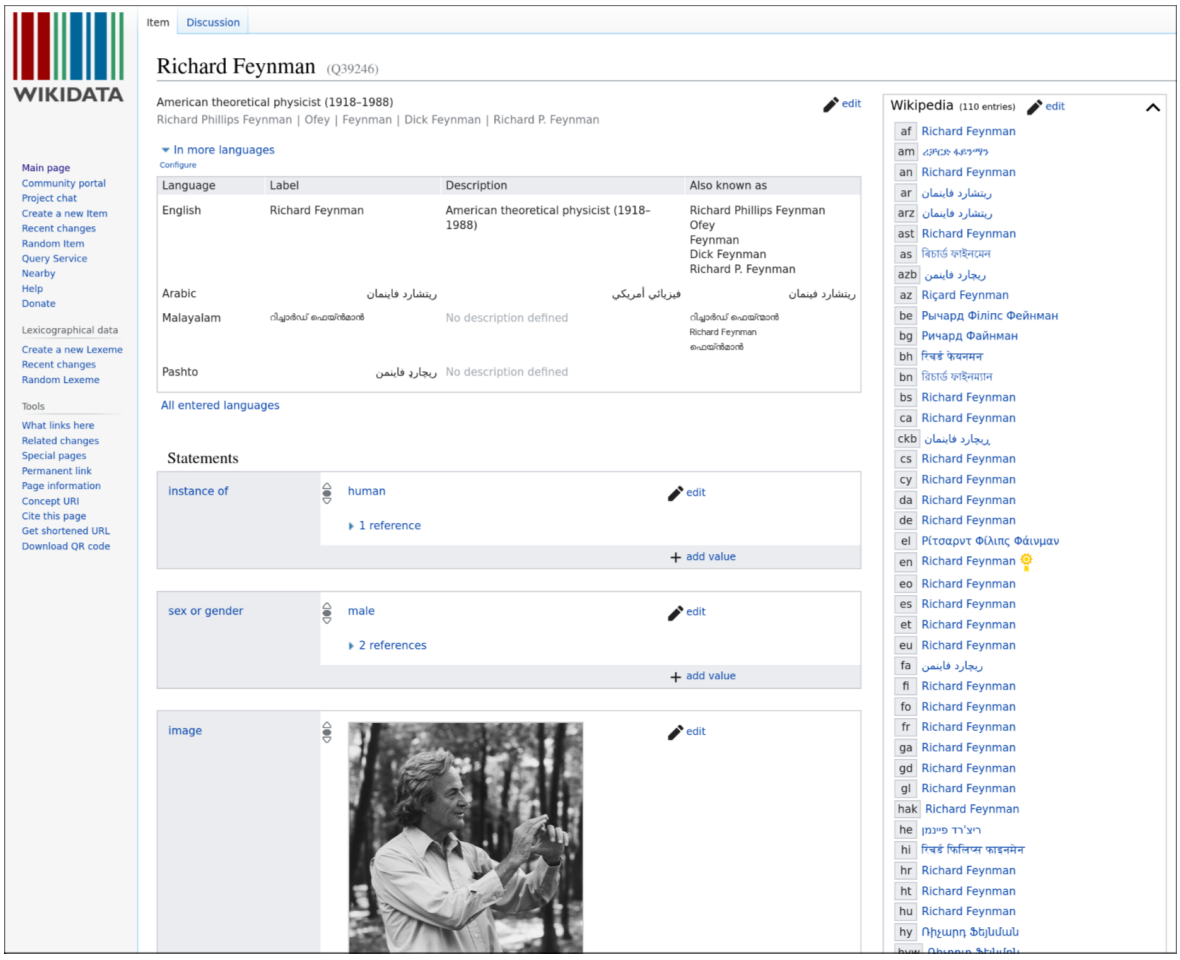
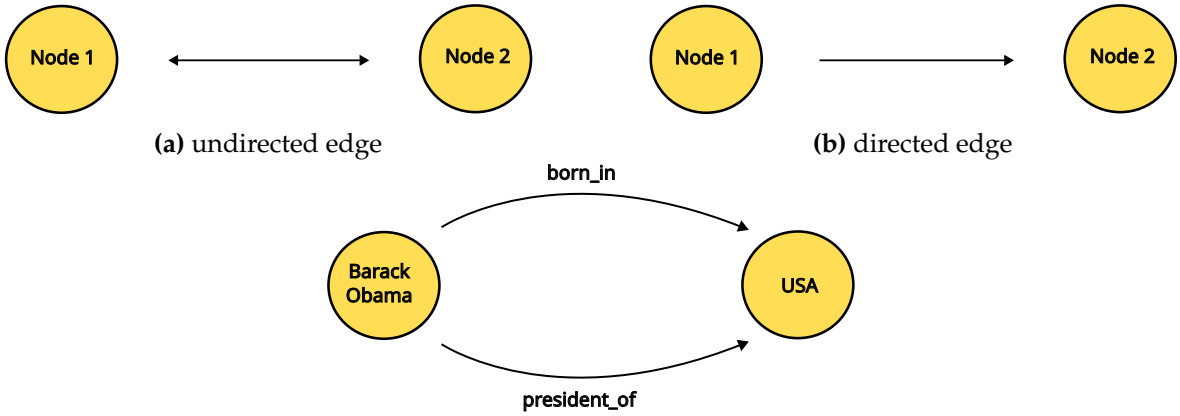


Figure 2.6: Wikidata web page for the entity "Richard Feynman", famous american physicist of the 20th century.



(a) undirected edge **(b)** directed edge

(c) Knowledge Graph edges

Figure 2.7: Different kind of graph edges.

some sense. For instance, the relationship

$$(barack\_obama, was\_president\_of, USA) \tag{2.8}$$

has a well defined direction and cannot be reversed as

$$(USA, was\_president\_of, barack\_obama)\,. \tag{2.9}$$

However, by defining $was\_president\_of^{-1} := was\_governed\_by$ we can provide a backward view of the original relationship as

$$(USA, was\_governed\_by, barack\_obama)\,. \tag{2.10}$$

As it probably became clear from the previous example, a second intrinsinc feature of the edges in KGs is that they also contain information about the type of relationship they express. While in a generic graph an edge simply represents the existence of a connection between two nodes, be it some sort of correlation in *undirected* or a cause-effect relationship in *directed* graphs, KG edges specify a distinct kind of connection between two entities. This in particular implies, for example, that the same two entities can be connected by multiple edges of different types.

Therefore, the richer and more fine-grained structure provided by KGs, makes them capable of representing any unstructured textual (propositional) information in structured form. However, the conversion of a text to the corresponding KG form is all but trivial. In the upcoming sections I am going to analyze the three main NLP tasks that have to be addressed to successfully achieve this.

2.5 Named Entity Recognition

In order to build a KG we first need the vertices, *i.e.* the entities. Therefore, the task we need to be able to solve consists in identifying the entity mentions that appear in a text. This is commonly known in literature as Named Entity Recognition (NER) (Grishman and Sundheim, 1996). Namely, given a sentence decomposed in

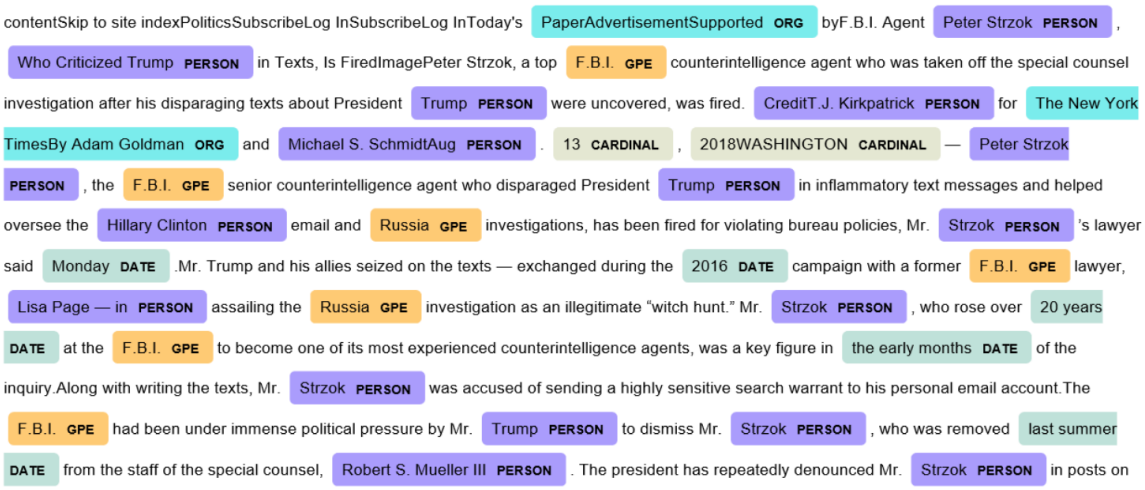


Figure 2.8: An example of Named Entity Recognition from a text paragraph. Each entity is assigned a type in the set {Organization (ORG), Person, Geopolitical Entity (GPE), Date}. Credits for the figure to Matthew Mcmullen.

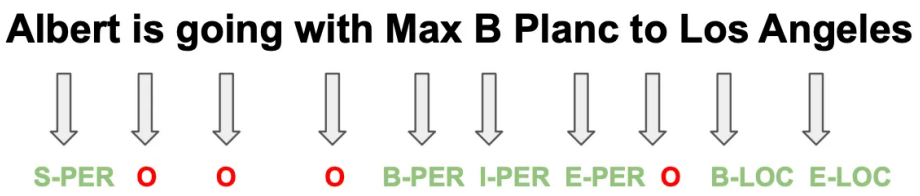


Figure 2.9: An example of the "BIOES" tagging scheme, each token in the sentence is assigned a specific tag depending whether it belongs to an entity or not. Credit for the figure to Muskaan Maurya.

its tokenized representation $S = (t_1, t_2, \ldots, t_N)$, we would like to individuate the subsequences $(t_i, t_{i+1}, \ldots, t_{i+n_k})$, each one defining the span of the k-th entity of length n_k. In addition to the span, also a type, selected from a predefined closed set, is usually assigned to each extracted entity. In particular, the assigned type can also be seen as a $(entity, is\_instance\_of, type)$ connection in the final KG.

The task of identifying the token subsequences usually translates to assigning to each token in the sentence a tag. In particular, the tokens composing an entity will be assigned a special tag. For instance, a popular Named Entity Recognition tagging scheme is the so-called IOBES (or BIOES) one (Tjong Kim Sang and De Meulder, 2003; Tjong Kim Sang and Veenstra, 1999). Namely, the scheme contemplates five different tags: "O" for outside of an entity, "B" for a beginning token of an entity, "I" for an inside token of an entity, "E" for an ending token of an entity, and finally "S" for an entity consisting of one single token. However, simpler schemes can be used as well, like for example tagging each token composing an entity as "I" and all the others "O", thus ignoring any information about the entity's beginning and end.

This means that Named Entity Recognition can be mapped to a token classification task and, therefore, can be evaluated with the standard classification metrics: False & True Positives, and False & True Negatives, whose combinations yield the popular

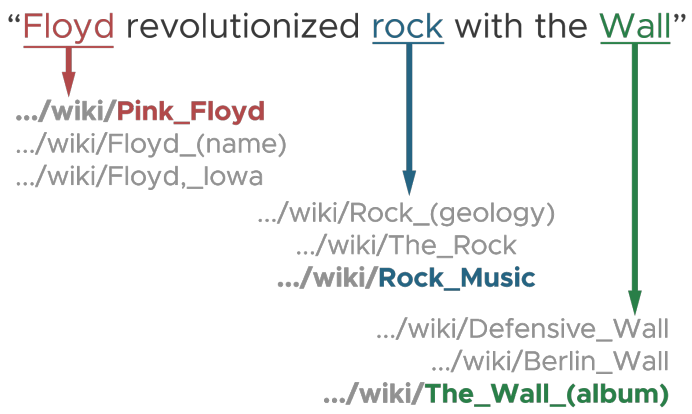


Figure 2.10: Linking and disambiguating some entity mentions contained in a sentence. Credit for the figure to Alberto Parravicini.

Precision, Recall and F1 scores (Rijsbergen, 1979).

2.6 Entity Linking

Named Entity Recognition, therefore, allows for the identification of the entities, thus the objects of interest, in a text. However, what is identified by a NER model in reality is not the actual entity, *i.e.* the abstract concept to which it is associated to, but rather its textual mention. In detail, the same general concept might manifest in several different textual forms in practice. For example, the entity "USA" which is associated to the concept of the homonymous North-American country, might appear straight as "USA" in text, but also in its extended form as "United States of America" or shortened as "US". All these are separate textual entity mentions from the perspective of the NER model, but they actually refer to the same underlying entity.

Conversely, the same textual form might refer to different abstract concepts, therefore, leading to ambiguities. Imagine, for instance, that there were a company somewhere in the world whose acronym was "US", for instance "Underground Solutions" a construction corporation specialized in underground metro lines realization. How would you disambiguate a "US" entity mention in a text? In other words, how can you know whether the mention refers to the North-American country or the construction company?

Therefore, in practice, the goal of Entity Linking (EL) (Bunescu and Paşca, 2006) is to resolve textual entity mentions to unique identifiers in a knowledge repository. More formally, given a tokenized sentence $S = (t_1, t_2, \ldots, t_N)$, an entity mention $m_i = (t_i, t_{i+1}, \ldots, t_{i+n_k})$ and a Knoweledge Graph or Knowledge Base (KB) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the task consists in mapping the mention to the corresponding graph vertex $v_i \in \mathcal{V}$ that uniquely identifies it. Hence, learning a function f that maps:

$$f : m \in S \longrightarrow v \in \mathcal{V} \ . \tag{2.11}$$

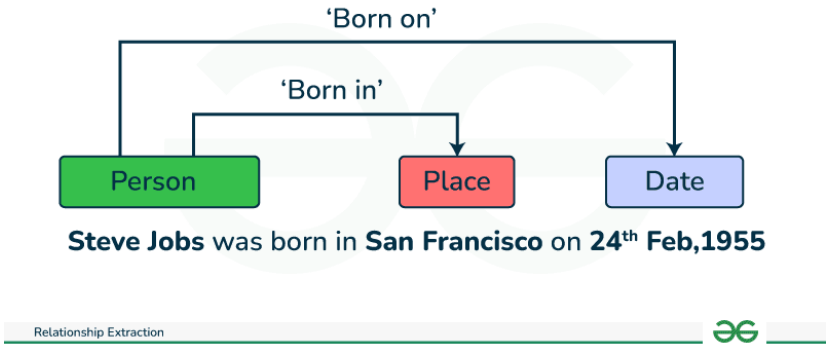


Figure 2.11: Relation Extraction exemplified: the relations between pair of entities in a sentence are extracted and identified. Credit for the figure to geeksforgeeks.com.

The whole task can be represented in two alternative forms. Either as a classification problem, where each node in the selected KB constitutes a separate target class and the aim is to "classify" the textual entity mentions accordingly. Or, more frequently in recent years, as a ranking task: a ranked list of the nodes contained in the KB has to be produced, from the most likely candidate to disambiguate the entity in object to the least one.

The latter approach, has been prevailing lately as the more sensible approach, due to its better scalability (Kolitsas et al., 2018). A cardinal requirement when dealing with KBs that constantly increase in size. While the classification problem would in general require models' predictions of the same size as the total number of classes, the ranking method would just need to assign a score to each node individually and sort the nodes according to it.

Concerning the metrics used to evaluate the Entity Linking task, standard classification metrics could be used with both the approaches, as, in fact, the ranking problem can be easily converted to a classification instance by taking the first ranked candidate as the predicted class. Nonetheless, metrics native to ranking problems would be more natural in the second case, such as the *hits@k* which measures how often the correct prediction lands in the top k ranked candidates provided by the model (Bordes, Usunier et al., 2013).

2.7 Relation Extraction

Named Entity Recognition and Entity Linking thus allow us to identify the vertices of the KG, Relation Extraction (Grishman and Sundheim, 1996), instead, takes care of reconstructing eventual edges existing between them starting from relationships expressed in text. Namely, given a sentence $S = (t_1, t_2, \ldots, t_N)$ and two entity mentions $m_1 = (t_{i_1}, t_{i_1+1}, \ldots, t_{i_1+n_1})$ and $m_2 = (t_{i_2}, t_{i_2+1}, \ldots, t_{i_2+n_2})$, the task consists in predicting the relationship r connecting them. A label is associated with r, *e.g.*:

$$\{born\_in, leader\_of, located\_in, \ldots\} ,$$

that specifies the type of relationship occurring between the two entities, a special negative *no_relation* label is often included as well. The label can, both, be chosen within a predefined closed set of possible relations or generated at runtime in an open fashion (also known as *open* Relation Extraction) (Angeli et al., 2015).

Quite commonly, the predefined set of relations is taken to be a subset of the relations supported by a target KB, thus providing a direct mapping of the extracted relationships to the edges of the KG. Open Relation Extraction, on the other hand, has the advantage of not being restricted to a closed set of relations, thus enabling exploration of other domains for instance. However, as a consequence, a further linking step is required to embed the extracted relationships into a KG structure.

2.8 Working with the Knowledge Graph

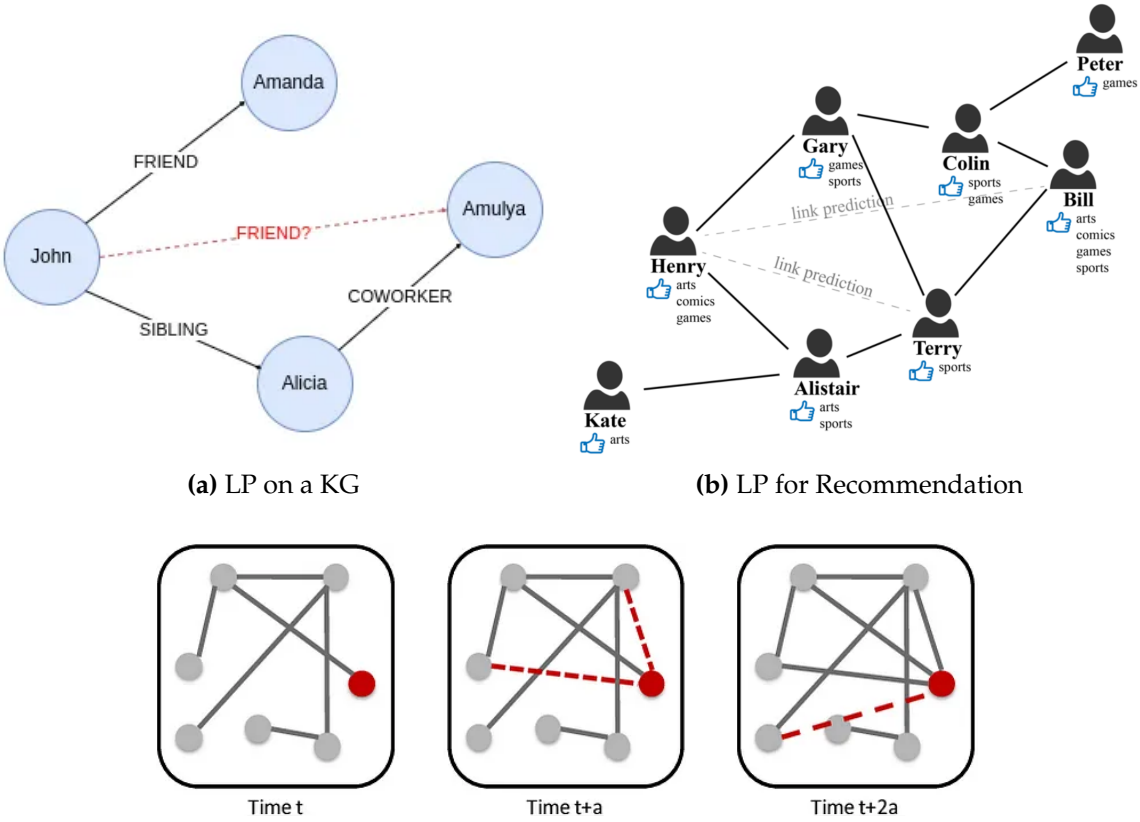
The combination of the three tasks just described, therefore, provides the means for capturing an extremely compressed representation of the information contained in a text and, for this reason, it is often referred to as Information Extraction (IE). In practice, an Information Extraction pipeline allows for the conversion of a textual snippet to its corresponding equivalent KG representation. Despite it being an efficient way for compressing and organizing information in a more structured form, one might wonder whether the KG representation brings any other upside to the table.

The answer is yes. Even without considering the three tasks just discussed above, which are clearly deeply entangled with the concept of KGs, but that are still well defined in their absence, or at least do not necessarily require the complete KG structure, the utility of such a graphical representation is evident. The structure provided by KGs allows for a more rigorous organization of the real world knowledge, which makes it substantially easier to process the information in an algotihmic and mathematical way. This means that algorithms can be designed to elaborate on real world facts, effectively disentangling the task from the complication of interpreting natural language and thus allowing for a higher level of abstraction.

As a consequence, the advent of KGs stimulated the birth of new NLP related tasks that build on them. In the following I briefly outline some of those that I consider of major relevance.

2.9 Link Prediction

A very well known and studied problem proper to network or graph based architectures in general, and not exclusively to KGs, is predicting the presence of new links between their nodes (Bordes, Usunier et al., 2013; Grover and Leskovec, 2016; Liben-Nowell and Kleinberg, 2003; Nickel, Murphy et al., 2016). Indeed, graphs often suffer from incompleteness. The absence of an edge between two nodes can happen for several reasons. Maybe, because the information was not there in the first place, *i.e.* the link was missing in the original data that was used as a source for the graph. Either due to the incompleteness of the source data itself, or just because that



(a) LP on a KG

(b) LP for Recommendation

(c) LP for time evolution

Figure 2.12: Link Prediction in three different kind of graph structures. Figure 2.12a shows the standard Link Prediction on KGs, where also the type of the link has to be predicted. Whereas, in 2.12b it is illustrated a social kind of network, where common interests can be used for inferring possible links and, for instance, improving recommendations for a user. Finally, 2.12c demonstrates a general homogeneous and undirected graph that evolves over time be creating new connections. Credit for figure 2.12a to Tomaz Bratanic. Credit for figure 2.12b to Yanwei Cui and Will Badr. Credit for figure 2.12c to Mark Needham.

information was not yet known or discovered. Some graphs might even possess an intrinsically mutable nature themselves. For instance, graphs representing proteins interactions normally account for the possibility of new edges to form and others to disappear over time.

Therefore, the capability of predicting the likelihood of the new edges formation can be useful in several ways: as a mean of enriching or completing the graph providing for the incompleteness of the original source data, as a method for Knowledge Discovery (KD) hinting at new probable unknown relationships, or even for modeling the evolution of a system over time.

To give a formal definition of the task, I make use of the standard graph notation introduced in the previous sections already, and consider the more simple case of an undirected homogeneous graph first. Given the usual graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ composed of vertices $v \in \mathcal{V}$ and known edges $\epsilon = (v_i, v_j) \in \mathcal{E}$, the problem consists in constructing a score function

$$f_s : \tilde{\mathcal{E}} \longrightarrow \mathbb{R} \,, \tag{2.12}$$

which assigns larger scores to the most probable missing edges $\tilde{\epsilon} \in \tilde{\mathcal{E}}$ in the graph, where, of course, $\tilde{\mathcal{E}}$ and $\mathcal{E}$ have no overlap $\tilde{\mathcal{E}} \cap \mathcal{E} = \varnothing$. Then, the set of new inferred edges can be constructed, for instance, by setting a threshold s^* and taking all the edges $\tilde{\epsilon}^* \in \tilde{\mathcal{E}}$ with score larger than s^*

$$\tilde{\mathcal{E}}^* = \{\tilde{\epsilon}^* \in \tilde{\mathcal{E}} \mid f_s(\tilde{\epsilon}^*) > s^*\} \,. \tag{2.13}$$

The problem, however, is that normally it is not known in advance which missing edges are true and which are not, making it impossible to evaluate the quality of the scoring function f_s and the corresponding predicted edges $\tilde{\mathcal{E}}^*$.

For this reason, a common choice is to hold out and remove a portion $\mathcal{E}_{test} \subset \mathcal{E}$ of the original edges composing the graph in order to then verify the capability of the link prediction model to recover them. In detail, each one of these held out test edges is going to be evaluated in the following way. Given an edge $\epsilon_{i,j} = (v_i, v_j) \in \mathcal{E}_{test}$, the tail of the edge, v_j, is corrupted by replacing it with any other vertex $v_k \in \mathcal{V} \setminus \{v_j\}_{j \neq k}$ in the graph, such that the modified edge obtained this way does not belong to $\mathcal{E}$, neither to $\mathcal{E}_{test}$:

$$\mathcal{E}_{false} = \{\epsilon_{i,k} = (v_i, v_k) \mid \epsilon_{i,k} \notin \mathcal{E} \cup \mathcal{E}_{test}\}. \tag{2.14}$$

The edge set obtained this way then presumably contains all the edges that are known not to belong to the graph and that, therefore, are false by construction. Naturally, as discussed above, it is often hard to say whether a specific edge in the graph was missing due to the graph being incomplete or because it was actually false. However, the zero level assumption here, is that the graph used for testing is a small scale graph whose degree of completeness we have under control and that, therefore, we can be reasonably confident that all the edges not present in $\mathcal{E} \cup \mathcal{E}_{test}$ are false.

To evaluate the quality of the scoring function f_s, we then evaluate it both on the original test edge $\epsilon_{i,j} = (v_i, v_j)$ and the complete false set $\mathcal{E}_{false}$ constructed this way. The score assigned to the test edge $f_S((v_i, v_j))$ is then compared to the scores assigned to all the corrupted edges $\{f_s(\epsilon)\}_{\epsilon \in \mathcal{E}_{false}}$ to obtain the rank r of the prediction, *i.e.* the position of the true edge in the complete list of scores ordered from the largest to the

smallest. The lower the rank, the higher the position, the better is the quality of the prediction. Ideally, all the test edges should be ranked in the first position.

The exact same procedure can be followed by replacing the head v_i of a test edge $\epsilon_{i,j} = (v_i, v_j) \in \mathcal{E}_{test}$, instead of its tail, leading to two separate corrupted edge sets $\mathcal{E}_{false}^{(head)}$ and $\mathcal{E}_{false}^{(tail)}$. The evaluation of the scoring function f_s on these two sets originates two distinct ranks $r^{(head)}$ and $r^{(tail)}$, which will be different in general.

Iterating this procedure over all the test edges in $\mathcal{E}_{test}$ will produce two sets of ranks $R^{(head)} = (r_0^{(head)}, r_1^{(head)}, \dots)$ and $R^{(tail)} = (r_0^{(tail)}, r_1^{(tail)}, \dots)$ that can be used to provide a global evaluation of the performance of the link prediction model.

The *Mean Rank*, for instance, is the most naive way to provide an indication of the overall performance of the model

$$\mathrm{MR} = \frac{1}{2\mathrm{N}} \sum_{1}^{\mathrm{N}} \mathrm{r}_\mathrm{i}^{(\mathrm{head})} + \mathrm{r}_\mathrm{i}^{(\mathrm{tail})} \,. \tag{2.15}$$

However, since we often want to give more emphasis to the capability of the model to rank its predictions in the top positions, other metrics are preferred, such as the *Mean Reciprocal Rank*

$$\mathrm{MRR} = \frac{1}{2\mathrm{N}} \sum_{1}^{\mathrm{N}} \frac{1}{\mathrm{r}_\mathrm{i}^{(\mathrm{head})}} + \frac{1}{\mathrm{r}_\mathrm{i}^{(\mathrm{tail})}} \,, \tag{2.16}$$

or the likelihood to have a prediction ranked inside the top n positions, also called *Hits@n*,

$$\mathrm{Hits@n} = \frac{1}{2\mathrm{N}} \Big(\; \big| \; \{ \mathrm{r} \in \mathrm{R}^{(\mathrm{head})} | \, \mathrm{r} < \mathrm{n} \} \; \big| + \big| \; \{ \mathrm{r} \in \mathrm{R}^{(\mathrm{tail})} | \, \mathrm{r} < \mathrm{n} \} \; \big| \; \Big) \,. \tag{2.17}$$

Those described here are all the elements defining the link prediction task for homogeneous undirected graphs, but the whole picture can be easily extended to KGs. Indeed, one can rederive the whole tractation by taking into account the presence of different edge types, thus unlocking an additional degree of freedom $\epsilon = (v_i, r_k, v_j) \in \mathcal{E}$. This means, in particular, that the scoring function becomes sensitive to the edge type $f_s = f_s(v_i, r_k, v_j)$ and that the construction of the edge sets $\tilde{\mathcal{E}}$, $\mathcal{E}_{test}$, $\mathcal{E}_{false}$ changes accordingly, as two edges sharing the same head and tail vertices may differ just for the relation connecting them now. Link Prediction evaluation can be performed in the same way as well, *i.e.* by alternately corrupting the head and the tail of an edge while keeping fixed the relation and the other vertex.

The sensitivity of f_s to the edge types brings an additional positive consequence. Link Prediction is often achieved in the Machine Learning field through training Neural Network based models to learn an embedding space where head entities are mapped into tail entities through a transformation that depends on the relation type (*e.g.* translation) (Bordes, Usunier et al., 2013; Kipf and Welling, 2017; Nickel,

Murphy et al., 2016). This means, in particular, that we gained the ability to predict which vertex (entity) is obtained if a specific relation is applied to another vertex (entity). This kind of structure, indeed, can often be used for representing the process of answering questions, as it will be discussed in the following section.

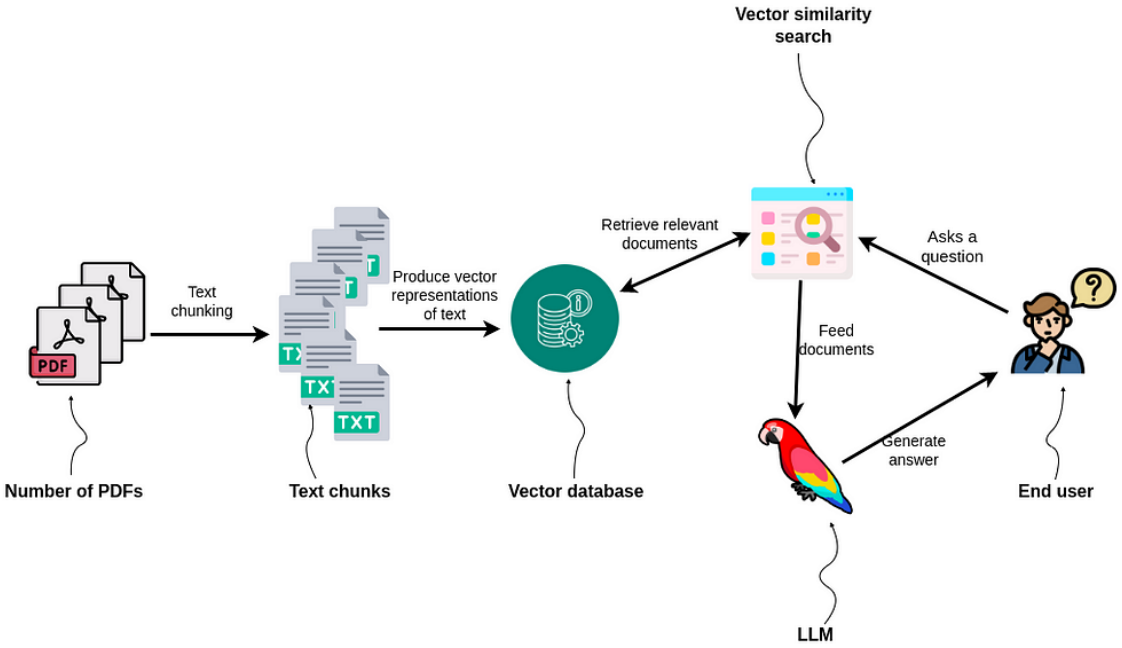
2.10 Question Answering

Question Answering refers to the problem of finding an answer to a question formulated in natural language. In detail, given a question Q and some context C containing the relevant information to answer the question, the task consists in extracting from the context C or generating from scratch the correct answer. Indeed, three slightly different Question Answering settings are often identified, based on how the final answer is produced and where the information relevant to answer the question is found.

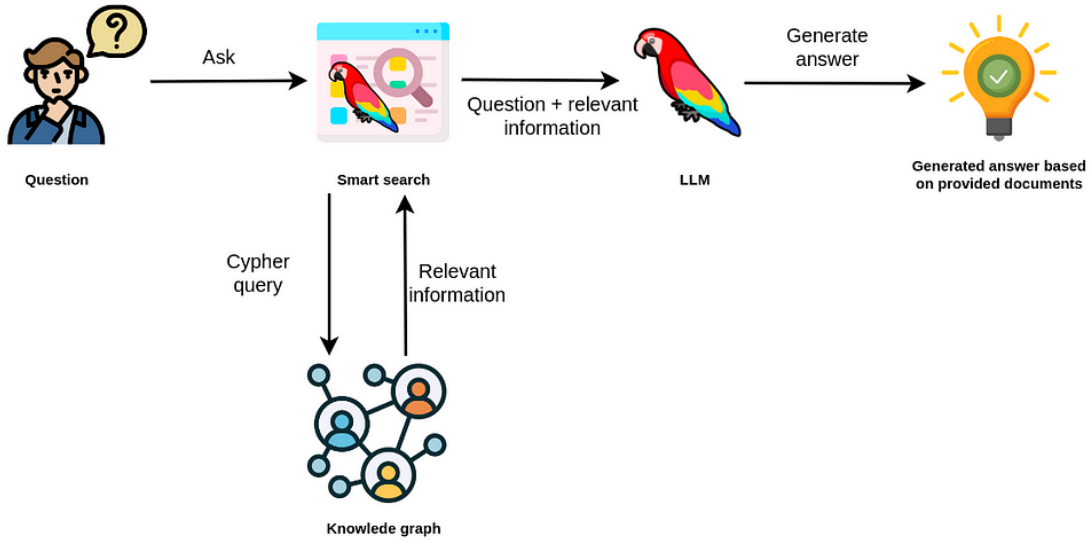
Extractive Question Answering (Devlin et al., 2018; Rajpurkar et al., 2016), for instance, is the simplest variant of the task, where the context C containing the exact answer already is provided as input to the model at inference time. Therefore, in this setting, the task is limited to highlighting the answer inside of the context, and can be formulated as a token classification task, similar to Named Entity Recognition. Considering the context as a sequence of tokens $C = (t_0, t_1, \ldots, t_N)$, the model has to correctly classify the tokens corresponding to the answer $(t_i, t_{i+1}, \ldots, t_{i+n})$ with some answer label, for instance "ANS", while marking all the others with a negative label, *e.g.* "O". Being a classification task, the performance of the model can be evaluated with the usual P, R and $F1$ metrics.

Slightly more complicate is the *Open Generative Question Answering* task (Chen et al., 2017). In this case, the model still has access to an external context in the form of a corpus or a set of documents, but the answer is not simply extracted word by word from it, rather it is generated from scratch. Therefore, both the input question Q and the context C are combined into a prompt that serves as some sort of initial seed for the generative model, which, then, will recursively generate tokens conditioned on the context and the question. A closed variant of the task exists as well, *Closed Generative Question Answering* (Brown, Mann et al., 2020a; Radford, Wu et al., 2019b), where no context is provided to the model which, thus, has to uniquely resort to its internal knowledge. Namely, the data it was trained on and its capability to recall it at need when relevant.

Therefore, the ability to provide the QA model with the relevant context to answer the question is a key feature for the success of the task. A common way to extract and select the context given an input question, is to perform some similarity search among a database of prepared documents or chunks of documents. At inference time, *i.e.* when the question is asked to the model, the database is searched for text shards similar to the input question, within which the context is chosen, as depicted in Figure 2.13a. However, this context selection process can be troublesome. The quality of the retrieved context can be very sensitive to the sharding strategy adopted for the documents in the database, *e.g.* how they were cut to obtain the chunks. Moreover,



(a) Document-based Question Answering



(b) KG-based Question Answering

Figure 2.13: An Open Generative Question Answering pipeline. In 2.13a the classic documented based approach is illustrated. A database of documents is separated into chunks that are then embedded and stored in a vector database. The input question is similarly embedded and compared to the elements in the vector database to retrieve the chunks that are more relevant to find the answer. The retrieved context is then fed, together with the input question, to the generative model for answer generation. In 2.13b, instead, the document database is replaced with a KG. The input query is embedded into the KG and the most relevant triplets are retrieved and combined in the context fed to the generative model. Credits for the images to Tomaz Bratanic.

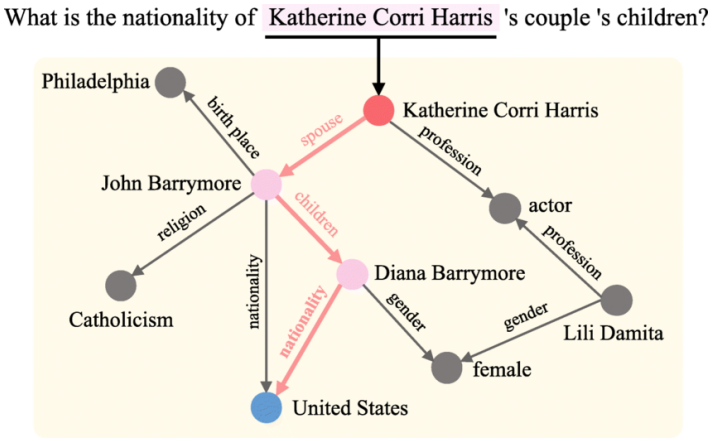


Figure 2.14: An example of KG Question Answering. The input question is embedded into a Knowledge graph structure, that is then traversed to reach the vertex providing an answer to the question. Credits for the image to Cui et al. (2022).

the problem of finding a, representative enough, vector embedding for the complete shard, that is capable of capturing its complete essence is all but trivial.

For this reason, having available a more condensed source of information, such as a KB, could help in making the context retrieval process easier, leading in turn to more relevant contexts (Bordes, Chopra et al., 2014; Cui et al., 2022; Lukovnikov et al., 2017). As discussed in the previous sections, indeed, natural language text can be embedded into a KG structure through the Named Entity Recognition, Entity Linking and Relation Extraction processes. Hence, a question can, in principle, be translated into a query to a KB and the relevant context to answer the question will consist of the nodes and edges obtained through the embedding process. Moreover, for simple closed questions that do not involve extensive elaboration, one might not even need to rely on a generative model, but rather the answer is found directly through this *question-to-graph* embedding process. For instance, the question "Where was Barack Obama's wife born?" could be answered by taking the entity corresponding to "Barack Obama's wife", "Michelle Obama", and looking for the node that is connected to it through a "birth_place" edge. If this connection is not present, one can always try to predict it through a Link Prediction mechanism like the one described in the previous section. Namely, the answer to the question might be found by looking for the vertex v that provides the highest score

$$f_s(Michelle\_Obama, birth\_place, v)\,. \tag{2.18}$$

Knowledge graphs, therefore, represent a promising approach to store, retrieve and dynamically process the information about the real world that is commonly found in text. The models for language processing introduced in Sections 2.2-2.3, indeed, appear to be natural partners of the KG formalism introduced here for the common objective of Natural Language Understanding and Processing. For this reason, the works presented in the upcoming chapters of this thesis (4, 5 and 6) explore the mutual integration of graph and language models for different language

and KG related tasks, introducing new architectures and pipelines that leverage the advantages of both data modalities.

Chapter 3

Global Literature Review

This short chapter aims at grouping in a unique place all the literature that inspired my work. This attempts to summarize and logically join the more detailed and specific reviews that can be found in the chapters that follow. Hence, for a more detailed presentation of the works reported here I remand the reader to the corresponding Chapters: 4, 5 and 6.

3.1 Relation Extraction with Knowledge Graphs

The task of Relation Extraction (RE) is certainly closely related to the concept of Knowledge Graphs (*cf.* 2.7). For this reason, several works can be found in literature that experimented with the use of KGs for textual RE. In some sense, KGs define a set of prototypes of relations that can be aligned to those found in text. Indeed, this is the underlying idea driving the "RE by Distant Supervision" approach that was first proposed by Mintz et al. (2009). They put forward to leverage the KG as a source of supervision for RE models. Namely for a given pair of KB entities connected through a target relation, a corpora is searched to find all the sentences that contain those two entities. Each sentence found this way is going to be labeled as an example of that precise relation. Therefore, a supervised model for RE can then be trained using the constructed labeled examples. Naturally, the assumption that a sentence containing the two entities in question is expressing exactly the relation we seek, is purely heuristic and, in facts, is probably going to be straight wrong in many cases. However, the idea is that, statistically speaking, such an assumption will hold true more often than not, and thus, that the true positives are going to largely outweigh the false positives.

Riedel et al. (2010b) iterated on this by trying to mitigate the presence of false positives. In detail, they come up with a Bayesian graph model to make decisions on whether two entities are related and, in case, whether this relation is expressed in a given sentence. They applied this approach to the NYT (Riedel et al., 2010a) registering a significant error reduction. Vashishth, Joshi et al. (2018) followed a similar research direction, and proposed to leverage the side information contained in KGs, *e.g.* entity types and relation aliases, to improve on the distantly supervised RE approach.

Bastos, Nadgeri, Singh, Mulang' et al. (2021) also made use of KGs to aid the sentential RE task, but they considered the canonical supervised approach instead. They also proposed to leverage side information coming from entity properties, but combining

it with the topological information coming from the neighborhood of the entity. A similar route was taken by Nadgeri, Bastos, Singh, Mulang' et al. (2021a), who put forward once again to make use of entities' ancillary information, but this time, it is suggested to dynamically select only the bits that are relevant for the RE instance at hand.

3.2 Multi-modal Learning extended to Knowledge Graphs

The concept of multi-modal learning is surely well-established in the ML community since the early 2000s. Examples of such an idea could be found, for instance, in attempts to jointly process audio and visual data for speech recognition, like the introduction of the CUAVE dataset (Patterson et al., 2002) and the multi-modal Hidden Markov Models (Bengio, 2004), or, more recently, in tasks requiring to connect image and textual data, such as Image Classification (IC) (Frome et al., 2013; Vinyals et al., 2014) and Visual Question Answering (VQA) (Antol et al., 2015).

CLIP (Radford, Kim et al., 2021) belongs to this latter class of models, as it consists in a contrastive learning procedure to align images with their textual captions. This very specific objective, however, enabled a zero-shot transfer in a wide range of image related tasks, and moreover, provided the means to learn an aligned multi-modal embedding space that was crucial for successive breakthroughs in the field of multi-modal learning. For instance, Mokady et al. (2021) made use of the CLIP embeddings to develop a model capable of generating captions for input images and, vice versa, Rombach et al. (2021) showed how CLIP could be used for the inverse process as well. Namely, the combination of CLIP and a latent diffusion procedure allowed for beautiful images to be generated out of a textual prompt.

All the work on RE with the aid of KGs discussed in the previous section, can be regarded, in a wider sense, as a form of multi-modal learning involving graph and textual data modalities. In recent years, even the opposite direction was pursued. Namely, the impact of side entity information, such as descriptions (*i.e.* textual data), was evaluated not only for NLP tasks, like RE, but also for graph associated tasks, as Link Prediction (LP).

Xie et al. (2016) first proposed to leverage entity descriptions, and combine them with the canonical topological information, in LP. Similarly, an implicit alignment between the topological and textual information was enforced by Yao et al. (2019), who tried to train a BERT (Devlin et al., 2018) model to perform LP by feeding it with the description of entities and relations in the form KG triplets. This was followed by other proposals (Shen et al., 2022; Wang, Zhao et al., 2022) that explored variations of the same underlying architecture and pushed the performance to the *State-of-the-Art*.

A more explicit example of this *graph-text* multimodality, instead, can be found for instance in the proposal by Yasunaga et al. (2022). They devised a pipeline to fuse in a unique representation a textual snippet and its corresponding KG diagram, which was then shown to be very effective in various commonsense reasoning tasks.

3.3 Knowledge Graph construction through Triplet Extraction

A natural extension to the RE problem is the (KG) Triplet Extraction task (TE), which again focuses on extracting relations from unstructured text, but also requires to identify the entities participating in the relation and to map them to a reference KG. Hence, TE's closeness to the KG field is quite obvious and the role it covers is quite fundamental: good TE capabilities allow for automatic KG construction from unstructured text.

The standard approach to the task has been for a long time the canonical NLP paradigm, *i.e.* training a supervised model. Indeed, several Long Short Term Memory (LSTM) based models have been proposed (Fu et al., 2019; Zeng, He et al., 2019; Zeng, Zeng et al., 2018; Zheng et al., 2017) to tackle TE, which steadily pushed the *State-of-the-Art*. In particular, Zheng et al. (2017) adapted and revised the WebNLG Gardent et al. (2017) challenge to make it a TE dataset, that has become one of the standards in literature. Transformer based models (Tang et al., 2022) currently hold the best performance and, as their LSTM counterpart, are normally either fully supervised or on-task finetuned.

The advent of Large Language Models (LLM), however, opened new possibilities for solving the TE task. Namely, lifting the need for supervision or finetuning and enabling zero and few-shots TE, through the use of carefully engineered prompts. This, in particular, is very helpful in those cases where the availability of labeled training example is scarce. Chia et al. (2022) and Kim et al. (2023), for instance, experimented with the automatic crafting, through LLMs, of new examples for training supervised models. Whereas, Wadhwa et al. (2023), Wei, Cui et al. (2023), and Zhu, Wang et al. (2023) proposed alternative prompting techniques to improve the LLMs capabilities in the zero and few-shots TE problem. In particular, Wadhwa et al. (2023)'s approach not only involved a smart prompt choice, but also tried to incorporate some relevant contextual information in the prompts. This procedure of providing LLMs with additional external resources goes under the name of Retrieval Augmented Generation (RAG) and embodies one of the most promising avenue for future research in NLP. Furthermore, it represents an alternative, more high level and less resource intensive, approach to the integration of language models and KGs. The latter methods are effectively used as oracles to query in this case, partially mitigating the need of developing a deeply entangled representation of text and graph.

3.4 Research gap in current literature

After this short outline of the most relevant literature, I would like to highlight here the research gaps that this thesis aims to cover. Once again, more details and one-to-one comparisons between my work and the reported literature can be found in the following Chapters: 4, 5 and 6.

First, while a variety of approaches have been proposed for combining textual and KG modalities, ranging from shallow embedding techniques to deep multimodal

alignments and retrieval-based strategies, there is a lack of a systematic and comparative evaluation across different integration paradigms, which makes it difficult to assess their relative strengths and weaknesses. Additionally, although multimodal learning has advanced significantly in domains such as vision-language alignment, the explicit and semantically grounded integration of textual and graph-based knowledge remains only superficially explored. This gap is further stressed by the scarcity of standardized benchmarks and datasets that support joint processing of text and graph data, limiting reproducibility and fair comparison across methods. Moreover, while Large Language Models (LLMs) have enabled remarkable performance on tasks like Triplet Extraction, their success often comes at a high computational cost, highlighting the need for more cost-effective solutions, such as leveraging external KGs to enhance smaller or less resource-intensive models. Finally, there is an absence of generalizable design principles or guidelines to assist researchers in choosing or developing integration strategies tailored to specific AI tasks. This thesis addresses these shortcomings by proposing and rigorously evaluating multiple integration architectures, extending existing datasets to support multimodal benchmarking, exploring bidirectional use cases, and offering practical insights for the development of efficient, scalable, and synergistic NLP-KG systems.

Chapter 4

Pretrained Knowledge Base Embeddings for improved Sentential Relation Extraction

This chapter was published as Papaluca, Krefl, Suominen et al. (2022) and presented at the Student Research Workshop at the Association for Computational Linguistics (ACL) conference 2022 in Dublin, Ireland.

In this chapter it is put forward to combine pre-trained knowledge base graph embeddings with transformer based language models to improve performance on the sentential Relation Extraction task in natural language processing. The proposed model is based on a simple variation of existing models to incorporate *off-task* pre-trained graph embeddings with an *on-task* finetuned BERT encoder. I perform a detailed statistical evaluation of the model on standard datasets. Evidence is provided, that the added graph embeddings improve the performance, making such a simple approach competitive with the state-of-the-art models that perform explicit on-task training of the graph embeddings. Furthermore, I observe for the underlying BERT model an interesting power-law scaling behavior between the variance of the F1 score obtained for a relation class and its support in terms of training examples.

4.1 The Role of Knowledge Bases in Natural Language Processing

Relation Extraction is certainly a very relevant tasks in the field of Language Modeling. The ability to automatically individuate and recognize relationships expressed in natural language sentences, both, requires a profound understanding of the language, and constitutes a pivotal ingredient for many other language related problems, such as Structured Search and Question Answering. Furthermore, as discussed in the previous chapter, Chapter 2, Relation Extraction also covers a cardinal role in the process of automatic Knowledge Graph construction, and, more recently, even the opposite implication has been demonstrated (Bastos, Nadgeri, Singh, Mulang' et al., 2021; Mintz et al., 2009; Nadgeri, Bastos, Singh, Mulang' et al., 2021a; Vashishth, Joshi et al., 2018). Namely, the aid of external Knowledge Bases proved to be effective in improving the Relation Extraction capabilities of Language Models.

Indeed, besides large quantities of unstructured textual data, also structured data has become widely available to machine learning researchers in recent years. Knowledge Bases (KBs), such as Wikidata (Vrandečić, 2012) (formerly Freebase Bollacker et al., 2007; Pellissier Tanon et al., 2016), Yago (Suchanek et al., 2007) and UMLS (Bodenreider, 2004), organise various kinds of information in structured form and constantly grow in size and richness of included information.

These Knowledge Bases are represented in terms of relations between entities, forming a graph structure that makes retrieval and processing of the included information easier and finds particular application in various language related tasks, such as question answering (QA), search engine development and knowledge discovery. Distant supervision (Mintz et al., 2009) is another notable example that employs a KB to improve Relation Extraction (RE). For each pair of entities found in a sentence, distantly supervised models check whether a link between the entities in the KB graph exists, and, if there is a match, the sentence is then used as a training example for supervised learning.

Note that the utility of KBs for RE extends beyond being just a useful source of supervision labels. A natural question arises whether one can develop models which combine unstructured textual data with structured information to further improve performance, for general natural language processing tasks.

One particular class of approaches that has been gaining momentum is based on the idea of dynamically learning representations of KB entities simultaneously with word representations (Bastos, Nadgeri, Singh, Mulang' et al., 2021; Nadgeri, Bastos, Singh, Mulang' et al., 2021a; Vashishth, Joshi et al., 2018). The motivation behind this class of methods is whether such combined representations could improve performance for a downstream NLP task, due to a more representative embedding.

However, although in theory this could result in optimally finetuned word and graph representations for the downstream task, it might be challenging in practice. On-task training of the graph embeddings requires significantly more complex models, and therefore increases the training cost and is more prone to overfitting.

Therefore, instead of training both, graph and word embeddings, I investigate in this chapter whether combining static pre-trained graph embeddings, such as those provided in Lerer et al. (2019), with *on-task* learned word embeddings already achieves a significant performance boost for downstream tasks. The underlying question being whether a neural model is able to transfer the topological information contained in the pre-trained graph embeddings in a useful manner to the task at hand.

4.2 Related Work

In the following, I would like to briefly review three particular works in the literature that make use of the information contained in a KB to improve classification performance on the RE task, and explain how the approach presented here relates to these previous works.

In Vashishth, Joshi et al. (2018) the authors propose to use KBs as a supplementary source of information to improve on the multi-instance learning paradigm for RE. Note that in multi-instance RE one aims at identifying the relation between two targeted entities, for a given bag of sentences. In particular, the authors of Vashishth, Joshi et al. (2018) match the relation predicted by the Stanford OpenIE (Angeli et al., 2015) pipeline with the set of relation aliases found in the KB. Out of this they obtain a *matched relation embedding*, h^{rel}, that is then concatenated to the sentence representation. Similarly, they build an *entity type embedding*, h^{type}, using the entity type found in the KB, which is concatenated as well to the sentence encoding.

In Bastos, Nadgeri, Singh, Mulang' et al. (2021) the authors make use of the KB information to improve on the sentential RE task. They propose to construct an *Entity Attribute Context* embedding, h^o, by processing several entity properties found in the KB with the help of a BiLSTM (Schuster and Paliwal, 1997) encoder. Additionally, a *triplet context* embedding, h^r, is learned for each relation triplet by imposing the translational property in the embedding space (Bordes, Usunier et al., 2013) to the triplet and its KB neighbours. Finally, the two different representations obtained are aggregated with the sentence encodings by a GP-GNN (Zhu, Lin et al., 2019) and fed to a classifier for relation prediction.

The authors of Nadgeri, Bastos, Singh, Mulang' et al. (2021a), however, suggest that statically adding all the available KB information might be counterproductive in some case. They rather propose to dynamically select the useful information. To do so, for each entity they extract and encode several KB properties with a BiLSTM (Schuster and Paliwal, 1997), that are then combined together with the sentence encoding to form a *Heterogenous Information Graph*. The relevant context information is then obtained by pruning this graph with the help of a combination of graph convolutional neural network (Kipf and Welling, 2017), pooling and self-attention layers, and finally aggregated and fed to the relation classifier, similarly to what is done in Bastos, Nadgeri, Singh, Mulang' et al. (2021).

Note that the KB embedding is dynamically learned in all above cited works. Furthermore, in these examples, it derives from a wide variety of entity properties and information retrieved from the KB, but not directly from the graph structure itself. The only exception being the *triplet-context* embedding h^r of Bastos, Nadgeri, Singh,

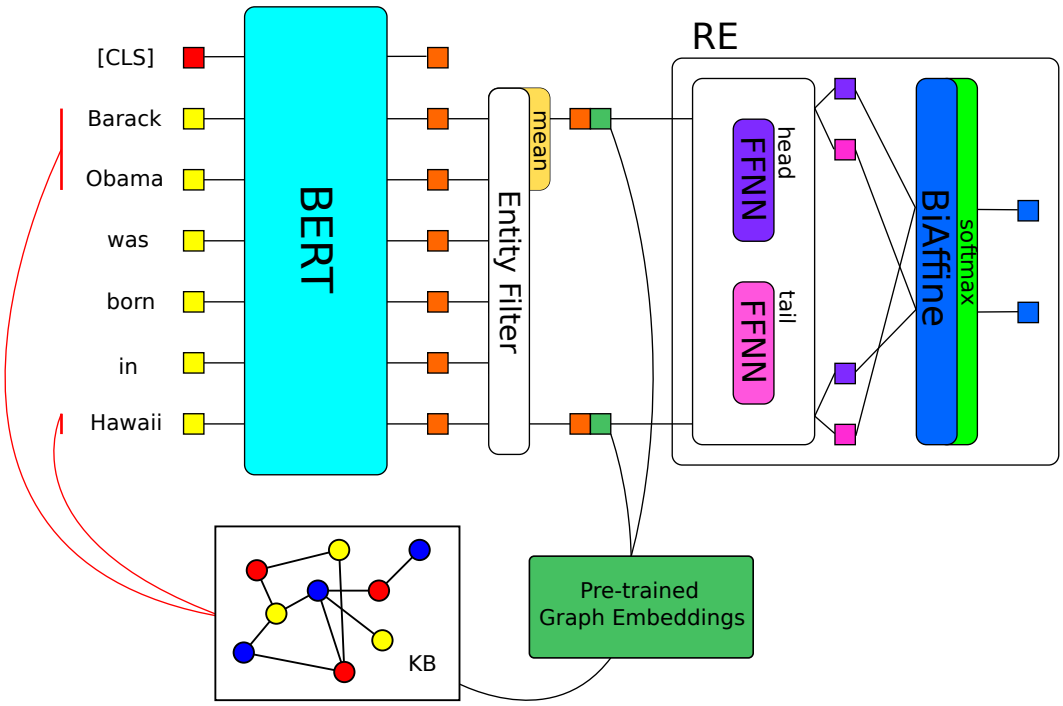


Figure 4.1: The RE model I make use of, inspired by Giorgi et al. (2019). The initial encodings of the sentences are provided by a pre-trained language model. The encodings corresponding to named entities are extracted, and averaged in case of multi-token entities. For the graph embedding augmented model the graph embeddings are in addition concatenated to the relative entity encodings. Each entity is then decomposed in a *head-tail* representation to form the (h, t) candidate pairs passed to the Biaffine Attention Layer (Dozat and Manning, 2017) for relation classification.

Mulang' et al. (2021), which comes closest to the graph embedding (Lerer et al., 2019) I am going to make use of. However, their embedding is trained only on the downstream task, but not on the full KB.

Therefore, what is so far missing in the literature, is the basic baseline of simply adding a pre-computed *topology-based* KB embedding in order to improve the RE task performance. In this chapter I aim to fill this gap. A detailed evaluation of such a model, which takes pre-computed KB embeddings into account, is provided. As I will show in the following, such a simple model extension is in fact competitive with the more advanced models mentioned above on standard benchmark datasets.

4.3 A Model for Relation Extraction

To perform Relation Extraction, I make use of the model introduced in Giorgi et al. (2019); Nguyen and Verspoor (2019). The only difference being that I do not perform Named Entity Recognition (NER), but instead train the model using gold entities, i.e. the correct span of the entity mentions is fed to the model as input. Thereby, it is possible to evaluate the RE performance without contamination with wrongly predicted entities. I chose this model for its relative simplicity, while still providing close to state-of-the-art performance in RE and its high modularity that

allows for easy modifications and extensions. Other models for text embedding and relation classification could be used, nevertheless, as the focus here is more directed to investigating the impact of the additional Knowledge Base embeedings, the relevance of the specific language model used is only marginal in the experiments.

The RE classifier I use resembles the one introduced in Nguyen and Verspoor (2019), except that it is combined with a pre-trained transformer-based (Devlin et al., 2019; Vaswani et al., 2017) encoder, as proposed in Giorgi et al. (2019). For illustration, the complete model is sketched in Figure 4.1.

The information flows from left to right and the errors are free to backpropagate through all the preceding layers. The input of the model is a sentence of N tokens, $w_1, w_2, ..., w_N$, containing two or more known entities, $e_i = w_j, w_{j+1}, ..., w_{j+k}$. The output is one or more triplets (h, t, r) representing the relations found in the input sentence. h and t represent the head and the tail of the relation respectively, while r is the type of relationship between h and t.

A more detailed description of the model pipeline is given in the remainder of this section. For reproducibility, I made the model source codes available on Github [1].

4.3.1 Pre-trained Language Model

The pre-trained language model provides the initial encodings for the tokens contained in the sentence. I opted to use variants of the BERT (Devlin et al., 2019; Liu et al., 2019) model (specifically, the implementation of *bert-base-cased* and *roberta-base* by Huggingface (Wolf et al., 2020a)), but in general the encoder can be replaced by any other encoder capable of providing context-dependent embeddings of a sentence. The encoder is left unfrozen during training such that the gradients can propagate through it. Thereby its parameters are fine-tuned to optimize the token representation for the specific task at hand.

In more detail, I start with the encoder completely frozen and gradually unfreeze the last four layers over the first epochs of training, as suggested for instance in Araci (2019). It has been shown that such training procedure does not necessary yield a decrease in accuracy, but it does significantly reduce the computational burden, c.f., Araci (2019).

4.3.2 Entity Filter

The purpose of the entity filter is to filter out each token that does not compose an entity. Two common choices for multi-token entities are to keep either the last token or the average of the tokens as an identifier of the complete entity. I tested both cases and did not find any evidence of one being superior to the other in this application. Therefore, I opted to take the average encoding among the tokens composing the entity as identifier.

Note that for the model augmented by a graph-embedding, I concatenate to the average encoding obtained for each entity, $\mathbf{x}_i^{BERT}$, the relative graph embedding

[1] https://github.com/BrunoLiegiBastonLiegi/Pretrained-KB-Embeddings-for-RE

$\mathbf{x}_i^{graph}$ of Lerer et al. (2019), such that I obtain for the final input $\mathbf{x}_i$ to the RE module

$$\mathbf{x}_i = [\mathbf{x}_i^{BERT}, \mathbf{x}_i^{graph}] \,. \tag{4.1}$$

4.3.3 RE Module

For details about the RE module I refer to the original works (Dozat and Manning, 2017; Giorgi et al., 2019; Nguyen and Verspoor, 2019). In a nutshell, the RE module consists of two steps. Firstly, the *head-tail* decompositions of the inputs are constructed:

$$\mathbf{x}_i^{h|t} = \mathcal{F}_{h|t}\left([\mathbf{x}_i^{BERT}, \mathbf{x}_i^{graph}]\right) , \tag{4.2}$$

where $\mathbf{x}_i^h$ and $\mathbf{x}_i^t$, are the representation of the inputs viewed as the head or the tail of a relation triplet (h, t, r). The projection is performed by two simple feed forward neural networks, $\mathcal{F}_{head}$ and $\mathcal{F}_{tail}$, composed of two linear layers separated by ReLU activation and dropout ($p = 0.1$) for regularization.

Secondly, all pairs $\{(\mathbf{x}_j^{head}, \mathbf{x}_k^{tail})\}_{j \neq k}$ are constructed by combining all the heads with each possible tail (Giorgi et al., 2019; Miwa and Bansal, 2016; Nguyen and Verspoor, 2019), and fed to a biaffine attention layer (Dozat and Manning, 2017; Giorgi et al., 2019; Nguyen and Verspoor, 2019) for relation classification. The biaffine layer $\mathcal{B}(\cdot, \cdot)$ performs a combination of a bilinear transformation and a linear projection:

$$\mathcal{B}(\mathbf{x}_1, \mathbf{x}_2) := \mathbf{x}_1^\top \mathbf{U} \mathbf{x}_2 + \mathbf{W}(\mathbf{x}_1 \| \mathbf{x}_2) + \mathbf{b} \,, \tag{4.3}$$

$$\mathbf{x}_i^{RE} = \mathcal{B}(\mathbf{x}_j^{head}, \mathbf{x}_k^{tail}) \,, \tag{4.4}$$

where I indicate with $\|$ the column-wise concatenation. A final softmax activation layer provides the scores for each of the relation classes. The RE loss is taken to be the crossentropy,

$$\mathcal{L}_{RE} = \sum_{i=1}^{M} \frac{\exp \mathbf{x}_{i,T}^{RE}}{\sum_{j \neq T} \exp \mathbf{x}_{i,j}^{RE}} \,, \tag{4.5}$$

where $\mathbf{x}_{i,T}^{RE}$ represents the score assigned to the correct class.

Note that if a sentence contains n_e entities, the RE module is going to provide $2\binom{n_e}{2} = \frac{n_e!}{(n_e-2)!}$ relation predictions. One for each of the possible entity combinations

$$(e_i^{head}, e_j^{tail})_{i \neq j} \quad i, j = 1, ..., n_e$$

and, therefore, allowing for multiple relations extracted from a single sentence. For each (e_i^{head}, e_j^{tail}) pair that does not correspond to any existing relationship, the model is expected to still provide a prediction, but with a specific "NO_RELATION" label assigned.

Dataset	train	$\textbf{train}_{cut}$	test	relations	KB entities	batchsize	epochs
Wikidata	372,059	369,577	360,334	353	464,535	8	1-2
NYT	455,771	$\sim$ 453-455,000	172,448	58	63,601	8	3

Table 4.1: Statistics and training settings for all the datasets considered. $\textbf{train}_{cut}$ indicates the actual number of training sentences used after discarding part of the data, as described in the main text. Note that for the NYT dataset the $\textbf{train}_{cut}$ is given as a range, as for each repetition I sampled a different subset of the training and validation set, as described in Section 4.4. For the Wikidata dataset, I experimented with training for 1 and 2 epochs.

4.4 Experiments

In order to quantify the benefit of adding pre-trained graph embeddings to a RE model, I have considered two popular RE datasets: the *Wikidata* (Sorokin and Gurevych, 2017) and *NYT* (Riedel et al., 2010b) corpora.

Since I make use of pre-trained graph embeddings, I decided to discard all sentences in the training set containing entities not present in the graph embedding (Lerer et al., 2019). Note that in order to be able to fairly compare the models with and without graph embeddings, I also exclude these sentences in training the basic model without graph embeddings. For the test set, I keep all the sentences regardless of the available embeddings. In case no embedding for the entity is available, I simply substitute the graph embedding with an identically zero tensor.

For each dataset I train the model with a different random initialization ten times with and without the addition of the pre-trained graph embeddings. Hence, in total I train twenty models per experiment. Note that I always use the *AdamW* (Loshchilov and Hutter, 2019) optimizer with learning rate $2 \cdot 10^{-5}$. For the implementation of the optimizer and of the whole model depicted in Figure 4.1 I rely on the Pytorch Library (Paszke et al., 2019).

For evaluation, I compare the performance of the two models (with and without graph embeddings) using Precision, Recall and F1 score (Powers, 2008). These are standard metrics used to benchmark classification tasks, as they provide a well rounded view of: how confident a model is in its positive predictions (Precision), how likely is for it to rightly provide positive predictions (Recall) and a combination of the two (F_α score). This is done, both, for each single relation class, and on average with micro- and macro-averaging, *i.e.* taking either the global or *per-class* mean. In detail I rely on the two following evaluation methods.

First, for each relation class, I collect the F1 score obtained by the ten different trained models and I plot the complete distribution of the results as a violin plot. The mean of the ten F1 values obtained is also reported as a colored dot inside the violin. The same is done for the global micro- and macro-average. I do this for both, the model with added graph embeddings and the baseline model without graph embeddings.

Then, I generate the micro *Precision-Recall* curve for each of the ten models trained. The ten curves obtained are interpolated and averaged to form a single mean PR

			Micro			Macro		
			P	R	F1	P	R	F1
Wikidata	Ours	RECON	87.24	87.23	87.23	**63.59**	33.91	44.23
		KG-Pool	**88.60**	**88.59**	**88.60**	-	-	-
		bert-base	85.43	78.37	81.74	51.42	37.24	40.02
		roberta-base	85.50	80.30	82.81	49.33	36.77	39.22
		roberta-base$_{1e}$	83.71	83.01	83.35	46.09	34.52	36.38
	Ours$_{ge}$	*bert-base*	88.34	79.19	83.51	55.72	39.62	43.24
		roberta-base	87.66	82.07	84.77	54.42	**41.64**	**44.33**
		roberta-base$_{1e}$	86.83	83.93	85.36	50.88	38.52	40.57
NYT	Ours	*bert-base*	47.13	75.57	57.98	28.35	45.27	33.05
	Ours$_{ge}$	*bert-base*	**51.13**	**76.46**	**61.24**	**31.66**	**47.31**	**36.20**

Table 4.2: Summary of the P, R and F1 scores (averaged over ten runs) obtained in the experiments for the three datasets considered. I indicate with the subscript *ge* the results obtained by the model with the graph embeddings added. For the *Wikidata* experiment I even specify with the subscript 1*e* the models that have been trained for 1 single epoch instead of 2. If available, I include the results obtained by others on the same dataset, it is left blank (-) otherwise. In particular, for the *NYT* dataset, I am not aware of other works in the literature measuring the performance under the F1 metric.

	P@10	P@30
RESIDE	73.6	59.5
RECON	87.5	74.1
KG-Pool	92.3	86.7
Ours	96.6	83.1
Ours$_{ge}$	**97.5**	**86.8**

Table 4.3: Comparison of the P@10 and P@30 for the NYT dataset. I indicate with the subscript *ge* the results obtained by the model with the graph embeddings added.

curve both, for the model provided with the graph embeddings and the baseline model. At each value of recall I compute the standard deviation of the precision over the ten different curves. The deviation is shown in shaded color around the mean curve.

4.4.1 Pre-Trained Graph Embeddings

For all the examples, I rely on the Wikidata KB (Vrandečić, 2012) with pre-trained graph embeddings of the entities provided by Lerer et al. (2019). In detail, the authors of Lerer et al. (2019) propose a memory efficient and distributed implementation of several popular graph embedding methods, such as RESCAL (Nickel, Tresp et al., 2011), TransE (Bordes, Usunier et al., 2013) and DistMult (Yang et al., 2015). This new implementation was specifically developed for dealing with very large graphs while being competitive with the original implementation of the state-of-the-art models aforementioned.

The pre-trained embeddings I use were trained on the so-called "truthy" Wikidata dump dated 2019-03-06, making use of the TransE (Bordes, Usunier et al., 2013) approach with embedding dimension of size 200.

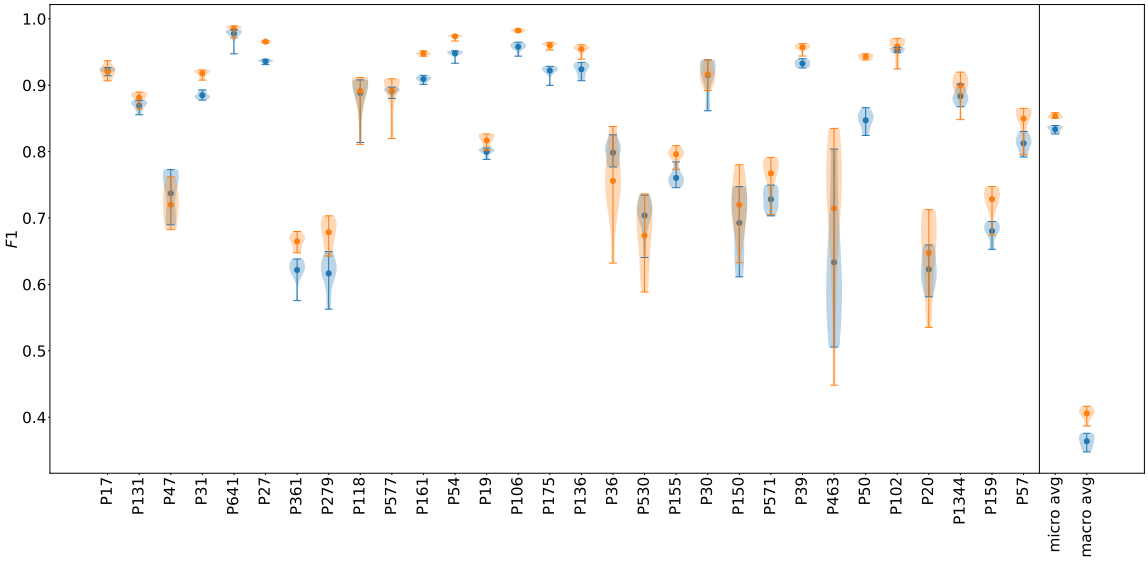
4.4.2 Wikidata

The *Wikidata* (Sorokin and Gurevych, 2017) dataset is a RE corpus built by distant supervision with entities aligned to the Wikidata Knowledge Base (Vrandečić, 2012). I rely on the dataset version provided by Bastos, Nadgeri, Singh, Mulang' et al. (2021). Some statistics of the dataset are shown together with my training configuration in Table 4.1. Note that in the evaluation I ignore the results for the NA relation class (i.e. the "NO_RELATION" class), as is usually done in the literature (Bastos, Nadgeri, Singh, Mulang' et al., 2021; Nadgeri, Bastos, Singh, Mulang' et al., 2021a).

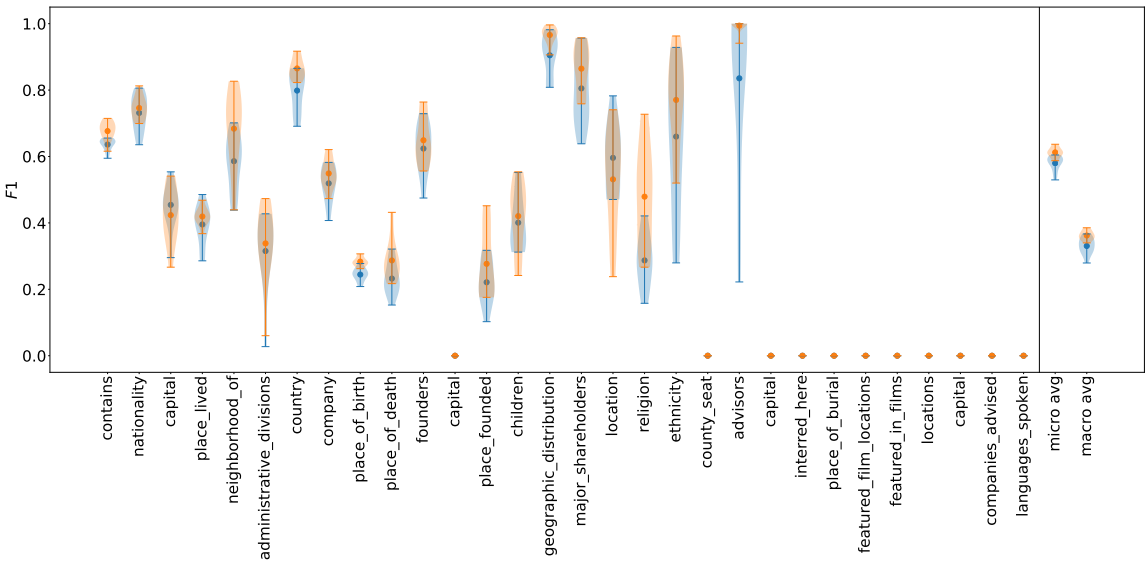
For this dataset, in addition to the *bert-base-cased* (Devlin et al., 2019) model, I also tested the *roberta-base* (Liu et al., 2019) language model. For the latter, I also experimented with training for just one epoch instead of two, obtaining slightly better micro F1, at the cost of lower macro F1.

The F1 score violin plots for the top 30 relation classes in the dataset are shown in Figure 4.2a. The illustrated results were obtained with the *roberta-base* model as sentence encoder, the whole model was trained for 1 epoch. Although some relations do not benefit from the inclusion of graph embeddings, the majority of them show a consistent improvement. On average, it is measured a performance boost, both, in micro- and macro-averaged F1 score, of around $\sim$ 2 - 4% depending on the specific language model, c.f. Table 4.2.

Even so the addition of graph embeddings yields a significant performance boost to the base model, as discussed above, the boosted model does not outperform the current state-of-the-art Nadgeri, Bastos, Singh, Mulang' et al. (2021a) in the micro-averaged metrics. In particular, only the micro precision metric (*bert-base*$_{ge}$: 88.34 ± 0.33), is statistically in range of the state-of-the-art model. For reference, I observed standard deviations in the range $\sim$ 0.1 - 1 for the results reported in



(a) Wikidata



(b) NYT

Figure 4.2: F1 scores on the Wikidata and NYT corpora for each relation. The micro- and macro-averages are also shown. Only the top 30 relations are plotted, ordered by decreasing support. The F1 score distribution of the ten different trained models is illustrated via violin plots. The baseline model is shown in blue and the model with added graph embeddings in orange. The means of the distributions are indicated by bullet points inside the violin plots.

Table 4.2. However, for macro-averaged scores, the model I trained (*roberta-base*$_{ge}$) surpasses the current state-of-the-art Bastos, Nadgeri, Singh, Mulang' et al. (2021), both, in recall (R) and F1.

4.4.3 NY Times

Another popular RE dataset built by distant supervision is the *NYT* corpus (Riedel et al., 2010b), obtained by aligning a set of over 1.8 million articles from the *NY Times* journal to the *Freebase* Knowledge Base. I made use of the *Wikidata SPARQL* query service [2] to obtain the mapping from the old Freebase id scheme to the new Wikidata one. I make use of the dataset version provided by Bastos, Nadgeri, Singh, Mulang' et al. (2021). In Table 4.1 are reported some statistics of the dataset together with the settings I used in my experiments.

I train on the validation (114,317 sentences) and training sets (455,771 sentences) by randomly discarding 20% of the sentences for each one of the ten runs, such that I obtain a number of train sentences comparable to Bastos, Nadgeri, Singh, Mulang' et al. (2021); Nadgeri, Bastos, Singh, Mulang' et al. (2021a); Vashishth, Joshi et al. (2018). Note that I had to further discard between 1,000 to 3,000 train sentences, depending on the run, due to the missing graph embeddings. I train the model with the settings listed in Table 4.1, and evaluate the performance at P@10 and P@30 (precision at fixed recall 10% and 30%), as proposed in the literature. As before, I ignore the score of the NA relation class in averaging.

Figure 4.3b illustrates the comparison of the performance of the two models trained. I observe that the model with the added graph embeddings shows a slower decay of the precision with increasing recall and, in particular, an improved precision on average and through the whole curve. The F1 score calculation confirms this trend, as both micro- and macro-F1 exhibit an improvement of about $\sim 5\%$ to $\sim 10\%$, as reported in Table 4.2 and Figure 4.2b.

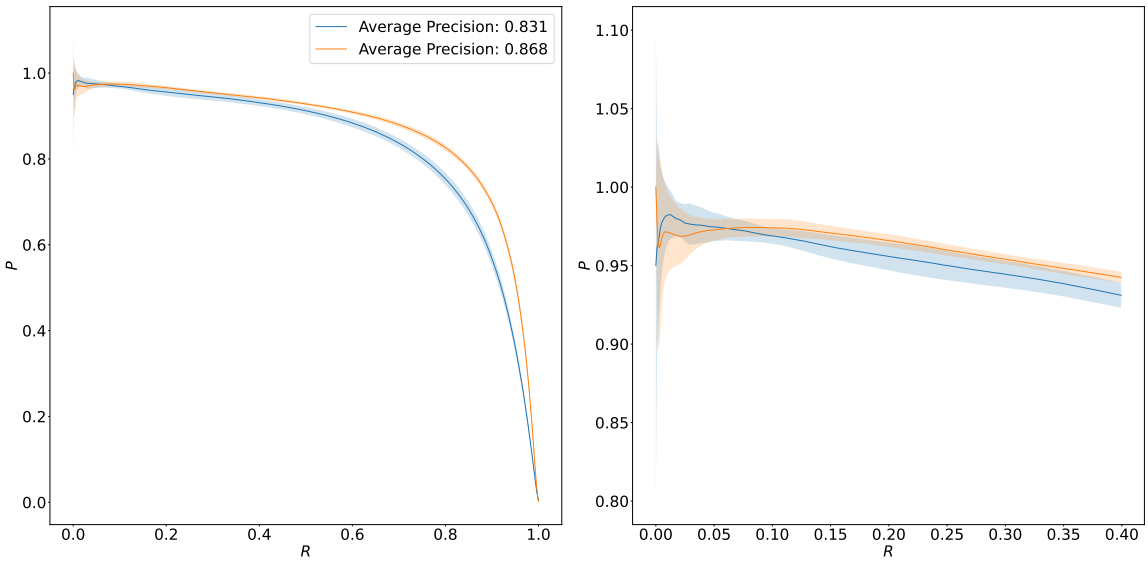
The comparison with the results obtained by others in the literature, reported in Table 4.3, demonstrates the solid performance of my model. In particular, the model is able to surpass the current state-of-the-art Nadgeri, Bastos, Singh, Mulang' et al. (2021a), both, under P@10 and P@30.

4.4.4 Statistical Analysis

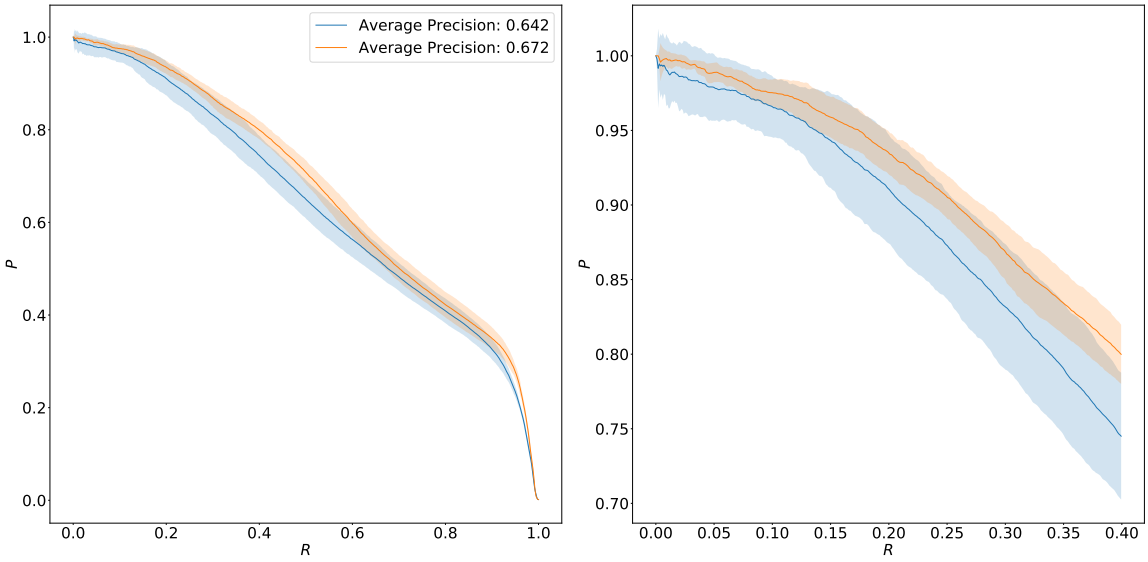
To better understand under which circumstances the additional information contained in the graph embeddings is beneficial, some of the results given above are analyzed below. I am going to consider each relation separately, and I will look at four different variables characterizing them:

- $\sigma^2(F1)$: Variance of the F1 score obtained across the ten runs.
- $\overline{F1}$: Mean F1 score obtained across the ten runs.
- S: Support, i.e. the amount of training examples available.

[2] https://query.wikidata.org/



(a) Wikidata



(b) NYT

Figure 4.3: Micro Precision-Recall curves for the baseline model (blue) and the model augmented with graph-embeddings (orange) trained on the *Wikidata* and *NYT* datasets. The right plot gives a more detailed view into the region with Recall ≤ 0.4. The solid lines represent the average P-R curve for the ten trained models. The shaded regions represent the corresponding standard deviations of the Precision at given Recall.

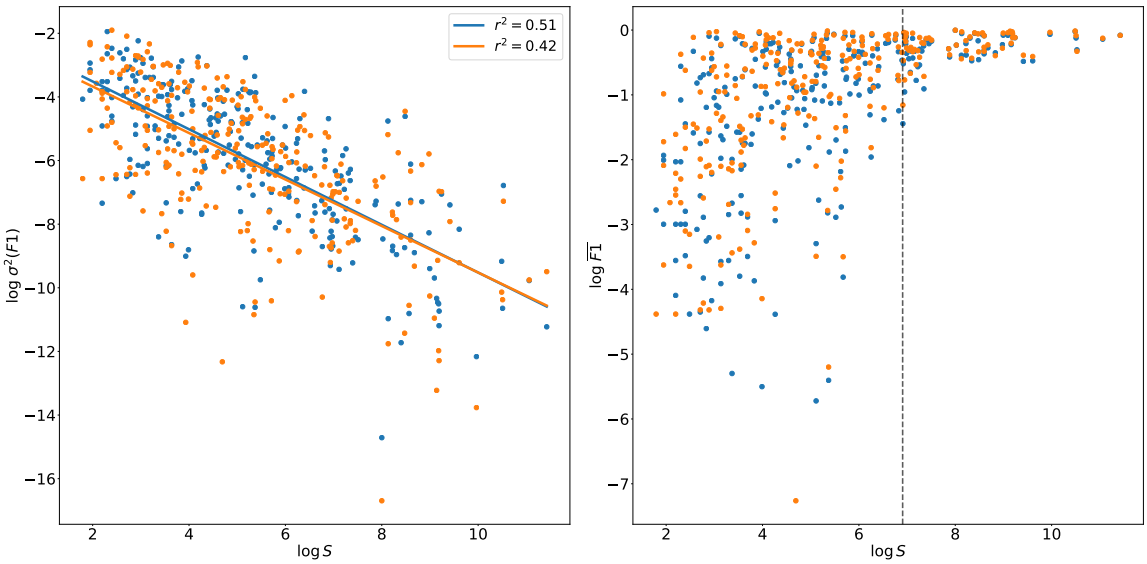


Figure 4.4: Distribution of the *Wikidata* dataset relations according to their score variance $\sigma^2(F1)$ (*left*) and their average score $\overline{F1}$ (*right*) against relation support S. Note that each point represents a different relation. The color of the points indicates from which model the datapoint originates. Blue for the baseline and orange for the graph embedding augmented model. Both axis are plotted in logarithmic scale. The distributions in the (*left*) plot are fitted with the linear regression (4.6) with the r^2 value of the fits reported in the legend. The support $S = 1000$ threshold is indicated as a dashed line in the (*right*) plot.

- $\Delta\overline{F1} := \overline{F1}_{ge} - \overline{F1}_{b}$: Gap between the average scores obtained by the model with the additional graph embeddings and the baseline model.

Note that the *Wikidata* corpus provides the largest number of relation classes and thereby features a wider statistical variety. Therefore, I focus on the results for the *Wikidata* corpus in the remainder of this section, in particular the ones obtained by my *roberta-base* based model trained for 1 epoch. For some relations I have always obtained $\overline{F1} = 0$ with zero variance $\sigma^2(F1) = 0$, and, surprisingly, not all of them had close to zero support S. Those with higher support, however, were usually semantically similar to relations provided with much larger support that were constantly preferred by the model. I exclude all these $\overline{F1} = 0, \sigma^2(F1) = 0$ relations since their identically null variance is an artifact and thus they do not provide a good representation of the $\sigma^2(F1)$ behaviour.

As shown in Figure 4.2a I observe a large variance of results for several relations across the ten different runs. The relationship between the variance, $\sigma^2(F1)$, and the support S of each relation is plotted in Figure 4.4 (*left*). Note that I take the log scale for both axis. Both, the baseline, and the graph embedding augmented models, show an approximately linear relationship of $\sigma^2(F1)$ with increasing support, under the $\log - \log$ transformation. I infer the following linear relationships (with and without

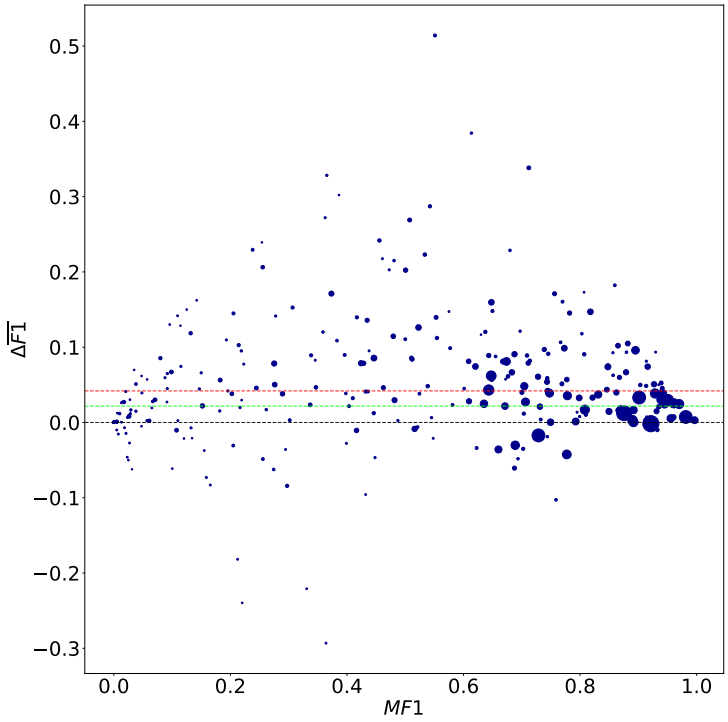


Figure 4.5: Gap $\Delta\overline{F1}$ for each relation in the *Wikidata* dataset plotted against the average score $MF1$ (4.8) obtained across the ten runs. The red and green dashed lines indicate the average and the median gap, respectively, whereas the gray dashed line a gap of zero. The size of each dot is given by the square root of the support of its corresponding relation.

graph embeddings):

$$\begin{aligned} y &= -0.75x - 2.01\,, \\ y &= -0.73x - 2.21\,, \end{aligned} \tag{4.6}$$

with standard errors of 0.05 for the slopes and 0.26, respectively, 0.30, for the intercepts (note that I have taken $y = \log \sigma^2(F1)$ and $x = \log S$). All the coefficients are rounded off to the first two decimals.

This implies a *power-law* scaling behavior of $\sigma^2(F1)$:

$$\sigma^2(F1) \propto \mathrm{S}^{-0.74}\,, \tag{4.7}$$

where I took as exponent the mean of the two regression coefficients given above.

In Figure 4.4 (*right*) the $\overline{F1}$ against the support is plotted using logarithmic scale, as for $\sigma^2(F1)$. However, a linear relationship is not observed in $\log - \log$ space as before, but rather some non-linear dependence. In particular, it appears that the larger the support of a given relation is, the better the model is able to learn it, as one would naively expect. The plot indicates that a support of at least $\mathrm{S} \approx 1000$ is a sufficient condition for good classification performance with an expected low variance of performance.

The gap $\Delta\overline{F1}$ is plotted against the averaged $\overline{F1}$ score,

$$MF1 = \frac{1}{2}\left(\overline{F1}_{ge} + \overline{F1}_{b}\right), \tag{4.8}$$

for each relation in Figure 4.5. Note first that a mean and median gap of $\sim +0.05$, respectively $\sim +0.025$, are observed. The performance boost is inline with the micro- and macro- average based observation in the previous section. In the plot also the size of support is indicated for each relation. It is clearly observable that relations with large support are clustered at high $MF1$, in accord with the discussion above.

It is interesting to notice that, both, for very high $MF1$ as well as very low $MF1$ the augmentation with graph embeddings only gives mild performance gains with low variance. In contrast, for $MF1 \sim 0.1$ - 0.9 a larger variance of $\Delta\overline{F1}$ is observed, that leads to gaps in the wider range ~ -0.3 - 0.5.

4.5 Pre-trained Knowledge Base Embeddings: a Useful Tool

In both the experiments discussed in Section 4.4, I measured on average a noticeable improvement over the baseline for the model with included pre-trained graph embeddings. Tables 4.2 and 4.3 summarize the numeric outcomes of our experiments, and also include for comparison results of the state-of-the-art methods obtained by others on the same datasets. In particular, the model is able to reach performance close to the current state-of-the-art in the *Wikidata* dataset under the micro-averaged

metrics and sets a new state-of-the-art under the macro F1 metric. Similarly, for the *NYT* dataset the model achieves a new state-of-the-art under the P@10 and P@30 metric.

I also analyzed in detail the performance of my model for each relation of the *Wikidata* dataset, finding an interesting *power-law* scaling of the variance of the F1 score with increasing support of the relation. In particular, this study provided an estimate of around $\sim$ 1000 training occurrences per relation needed for good prediction performance with small uncertainty. However, further investigation is needed to explore the validity of this finding in more generality.

Therefore, I would like to highlight that not only I was able to verify that the inclusion of general pre-trained graph embeddings is helpful for the RE task, but also that such a simple model extension is competitive with other state-of-the-art models that directly perform on-task training of those embeddings. This implies that the inclusion of such pre-trained graph embeddings might be helpful across a wider spectrum of language related tasks to improve performance at a relatively low additional cost of complexity and computational burden.

I see this finding as giving further support to the wider adoption of pre-computed graph embeddings in natural language processing tasks. I envisage that their adoption may become comparable to the popular Glove (Pennington et al., 2014) and Word2vec (Mikolov et al., 2013) pre-trained word embeddings.

However, in common with related works in the literature, the model presented here rests on the assumption of having the correct entity identification at hand (gold entities). Identifying entities in a sentence, however, is itself a challenging task, usually referred to as *Named Entity Recognition*. This task is further complicated by the need to map the entities to corresponding nodes in the KB (*Entity Linking*). This currently limits the practical applicability of mine, and models akin to it in the literature, and warrants further research.

Furthermore, even though the use of pre-trained graph embeddings proposed here represents a quite effective and inexpensive approach, it allows for only a limited degree of flexibility. The pre-trained graph embeddings are fixed and disconnected from their textual counterpart. Any, even small, modification to the text and corresponding embedding is by no means transferrable to the graph representation, which remains fixed. Similarly, comparison between textual and graphical data, which could be of use for instance in Entity Linking or Question Answering, is not viable as the two embedding spaces are completely separated. Using the pre-trained graph embeddings as an initial representation to later finetune on a downstream task, like the Relation Extraction problem presented here, is a viable approach, in theory, but not trivial in practice. A large amount of additional KG information might be needed for the considered task, which has to also be reformulated to simultaneously make room for a meaningful *topology-oriented* target.

For this reason, in the next Chapter, I explore in more detail a possible approach to generate more deeply entangled and aligned text and graph representations. As it will be shown, this method grants similar benefits to the ones recorded here, but with

the advantage of a more flexible representation that allows for easier manipulation of graphical and textual information.

Chapter 5

Contrastive Language-Entity Pre-training

This chapter was published as Papaluca, Krefl, Lensky et al. (2024) and presented at the International Conference for Pattern Recognition and Artificial Intelligence (ICPRAI) 2024 in Jeju Island, Korea, and won the third prize for the best paper award.

The Contrastive Language-Image Pre-training (CLIP) model has demonstrated the benefits of using natural language supervision for embedding images. A similar line of research can be imagined for Knowledge Graphs as well, but this is currently underexplored (if not unexplored) in literature. In this chapter, it is proposed a pretraining procedure that aligns a graph encoder and a text encoder to learn a common multi-modal graph-text space. The alignment is obtained by training a model to predict the correct associations between Knowledge Graph nodes and their corresponding descriptions. The procedure is tested with three popular Knowledge Bases: Wikidata (formerly Freebase), WordNet, and YAGO, measuring the degree of alignment by looking at the Euclidean distance between the node and description embeddings learnt. The results indicate that such a pretraining method allows for link prediction without the need for additional fine-tuning. Furthermore, I demonstrate that a graph encoder pre-trained on the description matching task allows for improved link prediction performance after fine-tuning, without the need for providing node descriptions as additional inputs. The code used in these experiments has been released at GitHub (https://github.com/BrunoLiegiBastonLiegi/CLEP) under the MIT license.

5.1 Multi-modal Learning with Knowledge Graphs

In recent years, significant progress has been achieved in developing models capable of learning a joint representation of text and image modalities, and interest in such models has been growing steadily in the machine learning community (Ionescu et al., 2022; Mokady et al., 2021; Radford, Kim et al., 2021; Rombach et al., 2021). One of the reasons being that such models, which are able to connect representations of textual and image data modalities, have proven incredibly powerful in image caption and generation tasks. A notable example is the *Contrastive Language-Image Pre-Training* (CLIP) procedure (Radford, Kim et al., 2021) that lies at the core of two popular models, CLIP-cap (Mokady et al., 2021) and latent diffusion (Rombach et al., 2021). The former model yielded substantial improvement in performance on the image captioning task, while the latter on the image generation task.

This success behooves me to ask if there are other data modalities that are suitable for learning a joint representation in a similar fashion. In this chapter, I will explore the replacement of the image modality with a graph modality. In particular, Knowledge Graphs (KG), also known as Knowledge Bases (KB), seem suitable for such joint learning for the following reason: In this kind of graphs, the nodes represent entities and the links describe the relations between them. Many KG can be viewed as a graphical representation or summarization of a textual corpus, with the entities being the common entities occurring in the corpus and the relations the factual relations described in the text. Hence, most KG are natural descendants from natural language corpora, yielding a very distilled representation of the information contained therein.

This also explains why KG play a fundamental role in several language-related tasks, such as question answering and semantic search. In particular, the mutual inclusion of graph and textual information has been proven to beneficial for both text and knowledge representation learning. Language models utilizing KG as additional sources of information have demonstrated stronger performance in language-related tasks, such as relation extraction (Bastos, Nadgeri, Singh, Mulang et al., 2021; Nadgeri, Bastos, Singh, Mulang' et al., 2021b; Papaluca, Krefl, Suominen et al., 2022; Vashishth, Joshi et al., 2018) and question answering (Yasunaga et al., 2022). At the same time, the inclusion of textual information (*e.g.*, entity descriptions) has been proven to be helpful in Knowledge Representation Learning tasks, such as Link Prediction (Xie et al., 2016; Yao et al., 2019).

Therefore, I investigate in this chapter whether a graph encoder pre-trained to learn such a multi-modal graph-text representation, similarly manifests enriched node embedding capabilities. This would represent an effective way to inject textual information into graph embeddings without the need of explicitly providing it to the graph model in downstream tasks. Furthermore, such a model, capable of interchangeably using graph and text as inputs, might serve as a foundation for dealing with natural language and graph-related tasks, similar to what CLIP (Radford, Kim et al., 2021) represents for text and images. Therefore, opening the way to new research directions that build on it, as happened for CLIP-cap (Mokady et al., 2021) and stable diffusion (Rombach et al., 2021).

In order to achieve such a joint representation of text and graph modalities, I propose

to train a model to predict the correct association between Knowledge Base nodes, *i.e.*, entities, and their descriptions found in the KB. This is similar to what has been introduced in (Radford, Kim et al., 2021), but with the fundamental difference that image data is replaced with graph-structured information.

The outline is as follows. First, in Section 5.2 a summary of the relevant literature is provided.

After introducing the model for caption prediction in Section 5.3, I demonstrate in Section 5.5 that alignment of the resulting KG entities and textual descriptions embedding is achievable for three popular Knowledge Bases: Wikidata, WordNet, and YAGO.

Furthermore, it is observed in Section 5.5.2 that a model trained in this entity-description matching task is capable of performing also link prediction using hybrid triples, composed of head entities (graph) and tail descriptions (text) (see Section 5.5.2). The model is competitive with baseline models without the need for additional finetuning under common link prediction metrics.

It is provided in Section 5.6 experimental evidence to indicate that the pre-training procedure allows the graph encoder to learn richer node representations. Namely, the performance in the link prediction task increases further under additional fine-tuning, without the need to explicitly provide descriptions as additional input (see Section 5.6). Furthermore, in Section 5.7 it is demonstrated that the text encoder shares similar improvements after the pre-training, as it shows stronger performance in the Question Answering task under certain circumstances.

I conclude in Section 5.8 providing a summary of the main outcomes of the methods illustrated in this chapter.

5.2 Related Work

The starting point of this chapter has been the CLIP procedure by Radford, Kim et al. (2021). The authors of this previous work proposed to leverage natural language supervision to learn a multi-modal image-text embedding space that yields richer and more general image representations by entangling language semantics with image features. Here, natural language supervision refers to letting the model learn the correct associations between images and corresponding captions. It turned out that such a pre-training procedure enabled zero-shot transfer in a plethora of tasks involving image representation learning, such as geo-localization, optical character recognition (OCR), facial emotion recognition and activity recognition in videos (Radford, Kim et al., 2021). Furthermore, this aligned image-text representation has served as a foundation for other tasks involving language and image processing. Two examples being image captioning (Mokady et al., 2021) and generation (Rombach et al., 2021). The method proposed here is based on a similar pre-training pipeline, but with images replaced by KG entities.

The aim of finding a multi-modal embedding space to increase the ability and performance of models is not unique to tasks involving image modalities. Other attempts

to find a joint embedding of data modalities, which are related to the proposal illustrated in this chapter, can be found in approaches to incorporate KG embeddings into word representations. In particular, the models introduced in Bastos, Nadgeri, Singh, Mulang et al. (2021); Nadgeri, Bastos, Singh, Mulang' et al. (2021b); Papaluca, Krefl, Suominen et al. (2022); Vashishth, Joshi et al. (2018); Yasunaga et al. (2022) are of relevance.

Yasunaga et al. (2022) have proposed a pipeline capable of fusing the language and graph representations of a text and its corresponding extracted subgraph, to achieve *state-of-the-art* performance in several commonsense reasoning benchmarks.

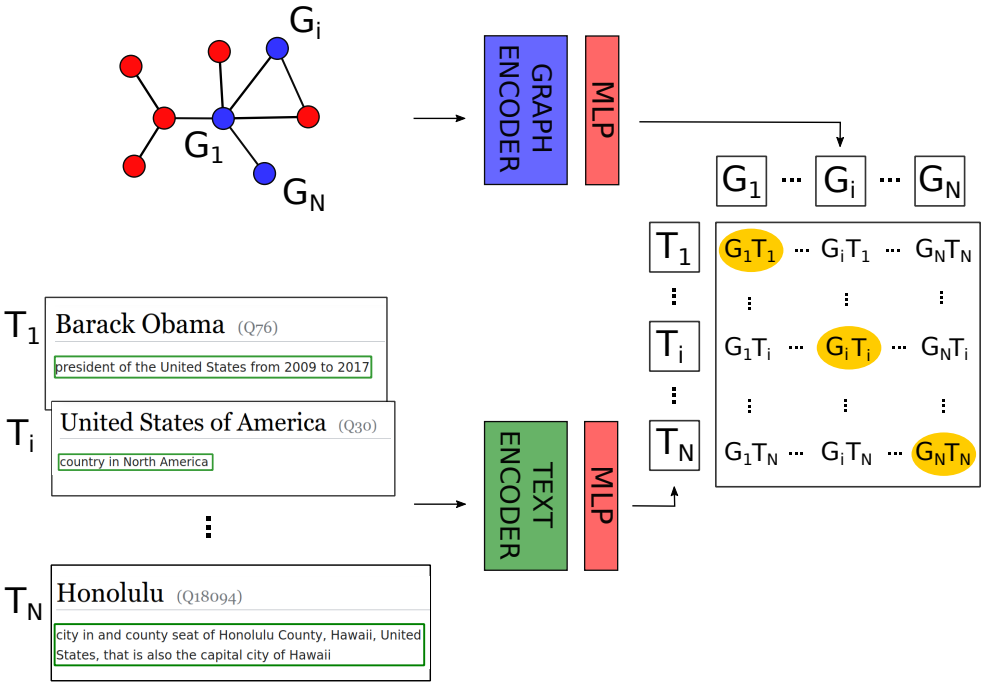
The methods introduced by Bastos, Nadgeri, Singh, Mulang et al. (2021); Nadgeri, Bastos, Singh, Mulang' et al. (2021b); Papaluca, Krefl, Suominen et al. (2022); Vashishth, Joshi et al. (2018) led to improved performance under the Relation Extraction task in Natural Language Processing. In particular, ranging from the dynamic inclusion of the relevant information contained in the KG (Nadgeri, Bastos, Singh, Mulang' et al., 2021b), to the more simple static augmentation of word encodings with pre-trained KG node embeddings (Papaluca, Krefl, Suominen et al., 2022). For a thorough evaluation of the conditions under which an improvement can be expected, and its magnitude, see Papaluca, Krefl, Suominen et al. (2022).

Similarly, augmentation of graph embeddings with KG entity descriptions has been considered in Shen et al. (2022); Wang, Zhao et al. (2022); Xie et al. (2016); Yao et al. (2019) and tested in Knowledge Representation Learning tasks. In detail, Xie et al. (2016) proposed to simultaneously minimize a *structure-based* and a *description-based* energy to learn coupled graph-description embeddings optimal for link prediction. The other works, instead, are based on varying approaches to finetune a pre-trained BERT model (Devlin et al., 2018) in the link prediction task. In detail, the models are fed with entity and relation descriptions in the form of *head-relation-tail* triples, and their truthfulness is predicted.

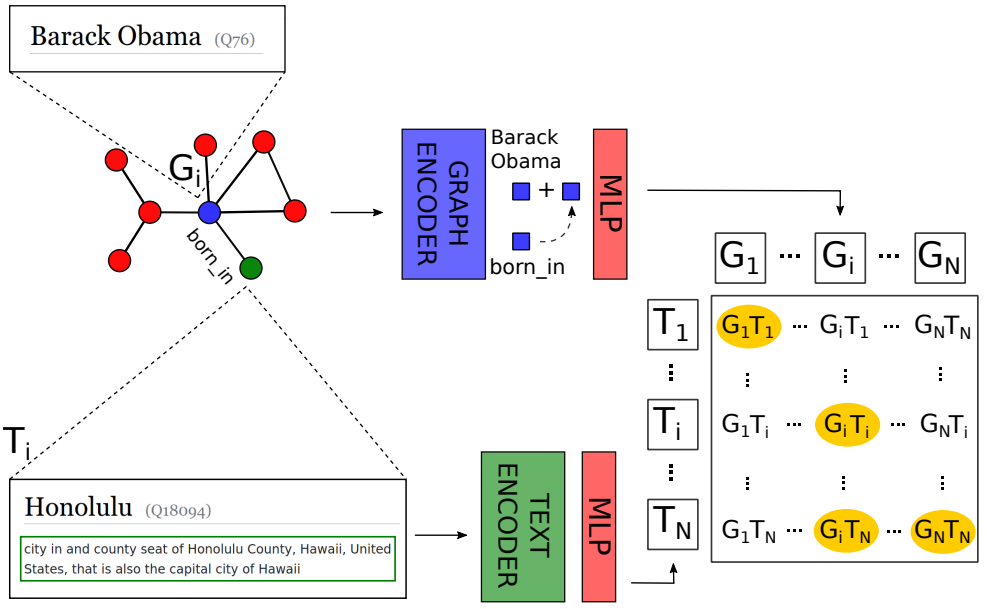
All these previous works, however, considered the case of both textual and graph information being provided to the model at inference time in downstream tasks. In this work, I rather aim at achieving a deep alignment of the two underlying representations to obtain a truly multi-modal graph-text latent space. This space will integrate features coming from both the modalities, possibly mitigating the need of providing them at inference time. Similarly to Shen et al. (2022); Wang, Zhao et al. (2022); Xie et al. (2016); Yao et al. (2019), I test this hypothesis by exploring how this joint training impacts the graph representation for link prediction.

5.3 CLEP: Aligning Text and Knowledge Base Nodes

Inspired by the language-image pre-training procedure by Radford, Kim et al. (2021), I propose here a similar pipeline to train a model to predict the correct association between Knowledge Graph entities and their textual descriptions. The goal of the procedure is to learn coupled graph and description embeddings such that the positions in the respective embedding spaces are close, thereby encoding features coming, both, from the topological structure of the graph and the textual meaning of the entities.



(a) entity-to-entity



(b) head-to-tail

Figure 5.1: Entity-description matching pre-training procedure. The graph and text encoders provide the embeddings for the node-description pairs, which are then projected to the joint multi-modal latent space by the two mapping networks. The complete model is trained to maximize the cosine similarity of the correct pairs (highlighted elements of the matrix shown) while minimizing that of the incorrect ones. (a) The association of each entity with its own description is considered. (b) The head entity of a triple is matched with the description of the tail entity.

The data batches consists of KB entities $\{e_i\}_{i=1,..,n}$ and corresponding description sentences $\{d_i\}_{i=1,..,n}$. Hence, the input to the model is a batch

$$\{(e_1, d_1), \ldots, (e_n, d_n)\}, \tag{5.1}$$

of correct entity-caption pairs.

I make use of a *graph encoder* and a *text encoder* to provide a latent representation of the graph entities ($x_i^{(g)}$) and their descriptions ($x_i^{(t)}$):

$$x_i^{(g)} = \text{GraphEncoder}(e_i), \tag{5.2}$$
$$x_i^{(t)} = \text{TextEncoder}(d_i). \tag{5.3}$$

Mapping networks are stacked on top of both encoders to provide a mapping from the text, respectively graph, latent space to a common multi-modal embedding space:

$$\tilde{x}_i^{(g)} = \text{MLP}_g(x_i^{(g)}), \tag{5.4}$$
$$\tilde{x}_i^{(t)} = \text{MLP}_t(x_i^{(t)}), \tag{5.5}$$

where it is indicated with MLP a standard multi-layer feed-forward network with identical architecture for MLP_g and MLP_t, but with independent weights.

The two encoders and mapping networks are trained to maximize the cosine similarity of the correct node-description associations, while minimizing the cosine similarity of the incorrect ones. This is achieved by considering a matrix M with elements $m_{i,j}$ given by

$$m_{i,j} = \frac{\tilde{x}_i^{(g)} \cdot \tilde{x}_j^{(t)}}{\|\tilde{x}_i^{(g)}\| \|\tilde{x}_j^{(t)}\|} \cdot e^{\tau}.$$

with τ, temperature parameter initialised as $e^{\tau} = 0.07$ and learnt during training. I define the row-wise cross-entropy of M as

$$\text{CE}(M) = -\frac{1}{n} \sum_{i=1}^{n} \log \frac{e^{m_{i,i}}}{\sum_{j=1}^{n} e^{m_{i,j}}}, \tag{5.6}$$

and take as loss function

$$\mathcal{L} = \frac{1}{2}\left(\text{CE}(M) + \text{CE}(M^{\top})\right). \tag{5.7}$$

Note that the transpose of M is included in the loss function in order to enforce the minimization of the off-diagonal elements simultaneously in rows and columns. The training procedure is illustrated in Figure 5.1a.

I also consider a variation of this scheme that should give more emphasis to the topological structure of the KB. For a KB triple (e^{head}, r, e^{tail}), composed of the head and tail entities and the relation between them, head entities e^{head} are matched with

	Entities	Relations	Train Triples	Test Triples
FB15k-237 (Toutanova and Chen, 2015)	14,541	237	272,115	20,466
WN18rr (Dettmers et al., 2018)	40,943	11	86,835	3,134
YAGO3-10 (Mahdisoltani et al., 2015)	123,182	37	1,079,040	5,000

Table 5.1: Statistics of the original datasets, see Section 6.4.1 for all the details.

	Entities	Relations	Train Triples	Test Triples
FB15k-237 (Toutanova and Chen, 2015)	14,296	235	256,862	19,062
YAGO3-10 (Mahdisoltani et al., 2015)	110,250	32	774,091	3,547

Table 5.2: Statistics of the pruned datasets, see Section 6.4.1 for all the details.

the respective tail descriptions d^{tail}. Therefore, instead of data batches as in (5.1), where each entity is associated with its own description, I consider

$$\left\{ \left(e_1^{head}, r_1, d_1^{tail}\right), \ldots, \left(e_n^{head}, r_n, d_n^{tail}\right) \right\} . \tag{5.8}$$

Then, provided a *graph encoder* capable of embedding the different relations in addition to the entities, the representations of the head entity ($h_i^{(g)}$) and the relation ($\rho_i^{(g)}$) are obtained as

$$\left(h_i^{(g)}, \rho_i^{(g)}\right) = \text{GraphEncoder}\left(e_i^{head}, r_i\right) , \tag{5.9}$$

and combined via the proper composition mechanism defined by the encoder. For instance, in this chapter I consider translation (Bordes, Usunier et al., 2013):

$$x_i^{(g)} = h_i^{(g)} + \rho_i^{(g)} . \tag{5.10}$$

Note that the combination via translation can be replaced with any other composition operation compatible with the *graph encoder*, *e.g.*, multiplication (Yang et al., 2014).

The encoding of the description of the tail is obtained as before in eq. (5.3), *i.e.*,

$$x_i^{(t)} = \text{TextEncoder}\left(d_i^{tail}\right) . \tag{5.11}$$

The mapping step and loss calculation are performed in the exact same way as defined in equations (5.4) to (5.7). However, since different head entities can be mapped to the same tail entity via the same relation in the KB, in general the correct entity-description associations will not all be on the diagonal of M only, as before. The procedure is summarized in Figure 5.1b.

5.4 Datasets and Methodology

In the experiments I considered three popular link prediction datasets: FB15k-237 (Toutanova and Chen, 2015), YAGO3-10 (Mahdisoltani et al., 2015) and WN18rr

(Dettmers et al., 2018). The descriptions of entities were provided by corresponding Knowledge Bases, as detailed below. Unfortunately, not for all entities a description was available. Therefore, those entities with missing descriptions were discarded. As a result, I worked with slightly cut-off versions of the original datasets, as reported in Table 6.2.

In detail, in order to obtain the descriptions for the FB15k-237 (Toutanova and Chen, 2015) dataset I mapped the entities to the Wikidata KB (Vrandečić, 2012), with only 245 entities missing. Instead, for the YAGO3-10 (Mahdisoltani et al., 2015) dataset the YAGO KB (Mahdisoltani et al., 2015) provided already the majority of entity descriptions needed. For the missing cases, I again fell back to the Wikidata KB. However, I still had to discard roughly 10% of the entities (12933). Finally, for the WN18rr (Dettmers et al., 2018) dataset I was able to keep all the entities because all the descriptions needed were present in the WordNet KB (Fellbaum, 1998).

In order to train the model for the entity-description matching task illustrated in Figure 5.1a, I generated a 80-20% train-test split of the entities contained in each dataset. For the *head-to-tail* matching depicted in Figure 5.1b instead, the original train-validation-test split of the triples provided by the dataset itself was used. Note that the graph encoder is always provided with a graph built only out of the training triples. No test or validation triple has been fed to the model at this stage.

In the experiments detailed below, I tested two different graph encoders, RGCN (Schlichtkrull et al., 2017) and CompGCN (Vashishth, Sanyal et al., 2019). For both encoders, I used an identical configuration: two layers and an encoding dimension equal to 200. For the entity descriptions, I used GPT2 (Radford, Wu et al., 2019a) and DistilBert (Sanh et al., 2019) by taking respectively the last and first token encodings as a representation of the descriptions. The language model is pre-trained and I allow only the last 4 layers to adapt during training in order to keep intact the text representation it originally learnt. Finally, for the MLP part, eq. (5.4)-(5.5), I decided to use a simple linear feed-forward layer, since it provides a good trade-off between the flexibility of the mapping and the capability to preserve the structure of the original graph and language spaces. For example, if the graph encoder allowed for the composition of entities and relations via translation, I would like to preserve this even after the mapping.

For the implementation[1] of the whole pipeline, I relied on the general-purpose machine learning library Pytorch (Paszke et al., 2019). I use the Hugging Face (Wolf et al., 2020b) version of the GPT and DistilBert language models and the implementation provided by the DGL library (Wang, Zheng et al., 2019) for the graph encoders.

5.5 The Multi-modal Graph-Language Space

In order to verify that the graph and language latent spaces are indeed aligned after training, I performed for each of the two variations of the (pre)-training procedure outlined in Section 5.3 a test of alignment.

[1] I make available the code here for reproducibility.

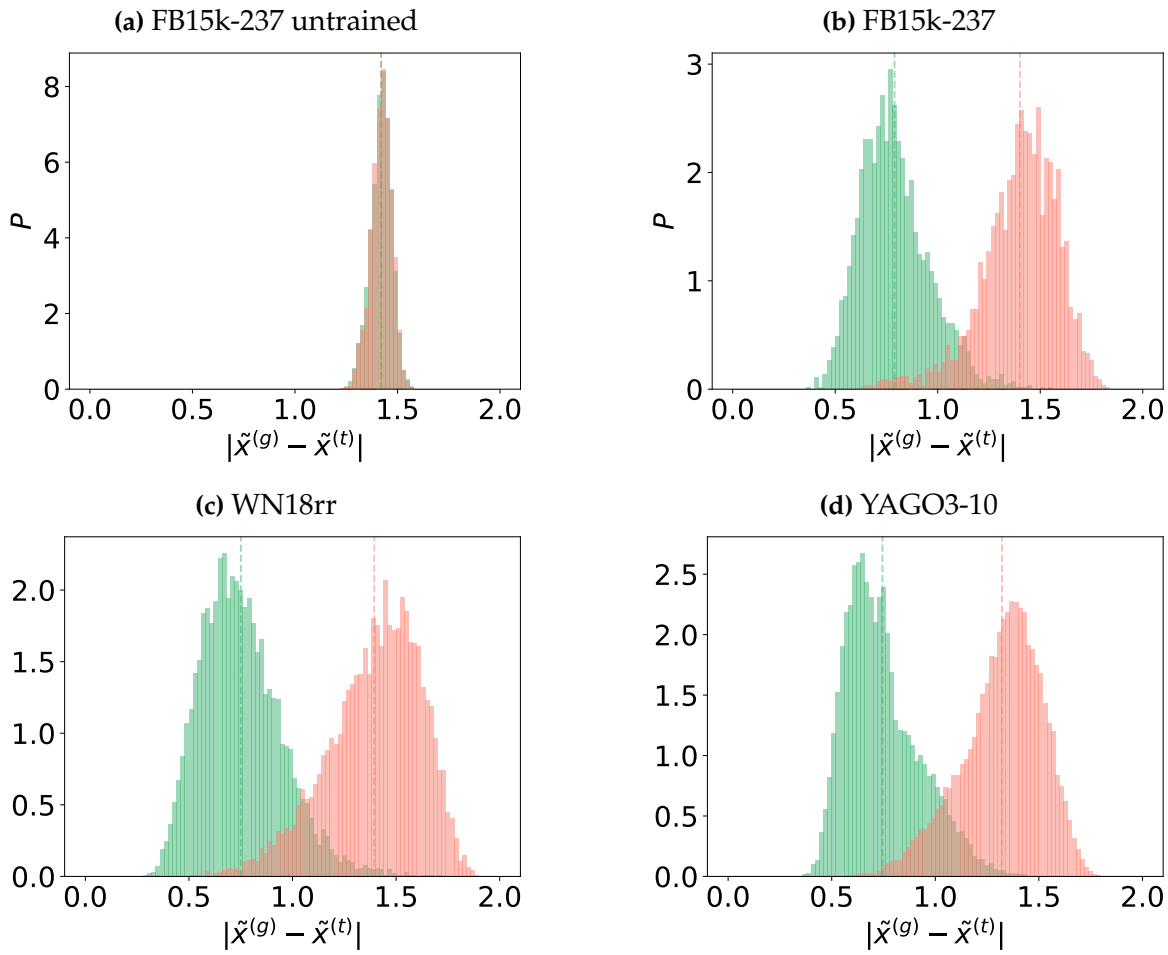


Figure 5.2: Distribution of the Euclidean distances of the correct (5.12) and incorrect (5.13) entity-description associations represented in green and red, respectively. Each embedding vector was normalized before calculating the Euclidean distance. The mean of each distribution is marked by a vertical dashed line. For reference, I also include, in Figure (a), the results obtained by an untrained model.

5.5.1 Distance Between the Mappings

In the first case I consider as measure of alignment the Euclidean distance between the embedding of the entities (5.4) and descriptions (5.5) after normalization. I compare the empirical distance distributions P of

- Correct entity-description associations:

$$P\left(\|\tilde{x}_i^{(g)} - \tilde{x}_i^{(t)}\|\right). \tag{5.12}$$

- Incorrect entity-description associations:

$$P\left(\|\tilde{x}_i^{(g)} - \tilde{x}_j^{(t)}\|_{i\neq j}\right). \tag{5.13}$$

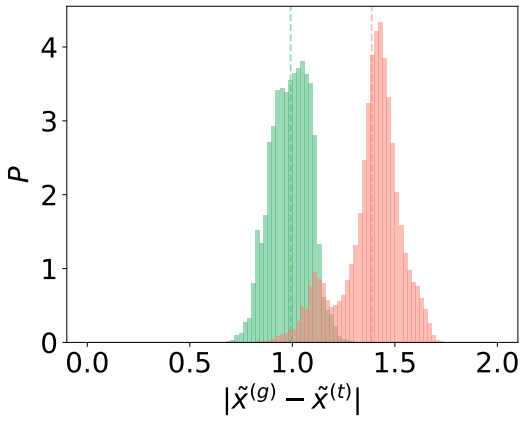


Figure 5.3: Distribution of the Euclidean distances of the correct (5.12) and incorrect (5.13) entity-description associations represented in green and red, respectively. Each embedding vector was normalized before calculating the Euclidean distance. The mean of each distribution is marked by a vertical dashed line. In this case a Compgcn (Vashishth, Sanyal et al., 2019) pre-trained with the *head-to-tail* strategy (Figure 5.1b) on the FB15k-237 dataset was used.

For alignment of the two latent spaces one would expect a clear separation between the two distributions (5.12) and (5.13), with the first (correct associations) shifted more to the left, *i.e.*, closer to zero, than the second (incorrect associations). In Figure 5.2 empirical histograms are plotted for the distributions (5.12) and (5.13) for the three datasets introduced in Section 6.4.1. They were all obtained by training an RGCN encoder alongside the GPT2 language model following the procedure of Figure 5.1a Similar outcomes were obtained when using the pre-training scheme depicted in Figure 5.1b and the CompGCN encoder, as illustrated in Figure 5.3. As expected, there is a clear separation between the correct and incorrect entity-description associations visible. In particular, the mean of the incorrect association distribution is for all datasets almost twice as far away. However, it is interesting to note that a certain degree of overlapping between the distributions (5.12) and (5.13) is observable for all tested datasets. As a consequence, sporadically some of the incorrect associations are located closer than the correct ones. A possible explanation of this may be found in the fact that a similar overlap exists between entity descriptions too. In particular, short descriptions such as "Music genre" or "American actress" are often shared over a wider range of different entities.

5.5.2 Entity-Description Similarity as Link Prediction

As another test of alignment, I tested whether the *head-to-tail* strategy (Figure 5.1b) allows for link prediction. Being able to predict links between head entities and corresponding tail descriptions from the embedding spaces, *i.e.*, across the graph and text embedding space, would mean that graph information can be substituted with textual information in my representation and, as a consequence, provide strong evidence that an alignment between the embedding spaces exists.

In general, link prediction is intended as follows. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ composed of vertices $v \in \mathcal{V}$ and edges $\epsilon \in \mathcal{E}$ of the form $\epsilon = (h, r, t)$, with $h, t \in \mathcal{V}$

	FB15k-237				WN18rr			
	MR	MRR	hits@1	hits@10	MR	MRR	hits@1	hits@10
$\text{CompGCN}_{\text{CLEP}}$	**198**	0.222	0.137	0.396	**1211**	0.305	0.199	**0.508**
RGCN+Distmult	315	**0.237**	**0.156**	**0.407**	5688	**0.408**	**0.393**	0.435

Table 5.3: Link prediction performance obtained by a CompGCN (Vashishth, Sanyal et al., 2019) trained with the procedure depicted in Figure 5.1b and using the scoring function (5.14). For comparison, I also include the results obtained by a RGCN (Schlichtkrull et al., 2017) baseline. Note that, while the RGCN has access to the complete graph information (both head and tail nodes are provided), the CompGCN here relies on head nodes and tail descriptions, as illustrated in Section 5.5.2.

and relations $r \in \mathcal{R}$, the link prediction task consists of assigning a score $f(h, r, t)$ to edges for some defined scoring function f. A successful model would assign lower scores to the non-existent edges and higher scores to existing ones out of a never-before-seen sample. For evaluation, each test edge $\epsilon = (h, r, t) \in \hat{\mathcal{E}}$[2] is corrupted by substituting the tail t with every other possible node in the graph. The corrupted edges are then scored against the original edge to obtain the ranking.

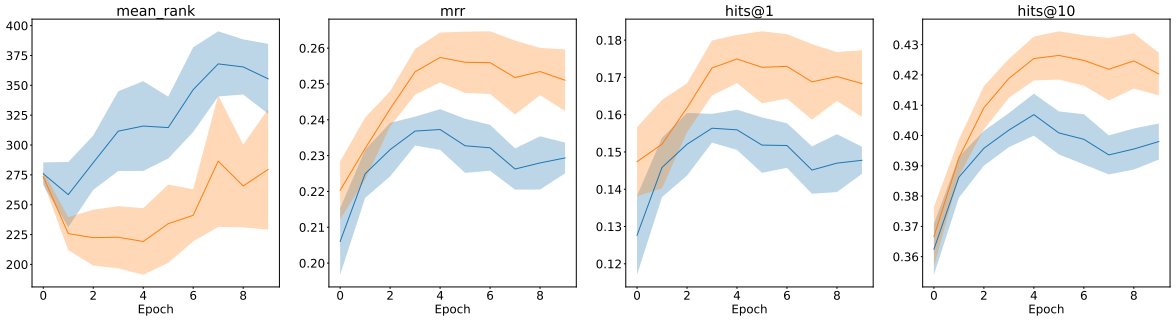
In my case, however, instead of the tail node t, its description d^{tail} is considered. The score function f is defined to be the cosine similarity

$$f(h, r, t) = \frac{\text{MLP}_g\left(x^{(g)}_{head}\right) \cdot \text{MLP}_t\left(x^{(t)}_{tail}\right)}{\|\text{MLP}_g\left(x^{(g)}_{head}\right)\| \, \|\text{MLP}_t\left(x^{(t)}_{tail}\right)\|}, \tag{5.14}$$

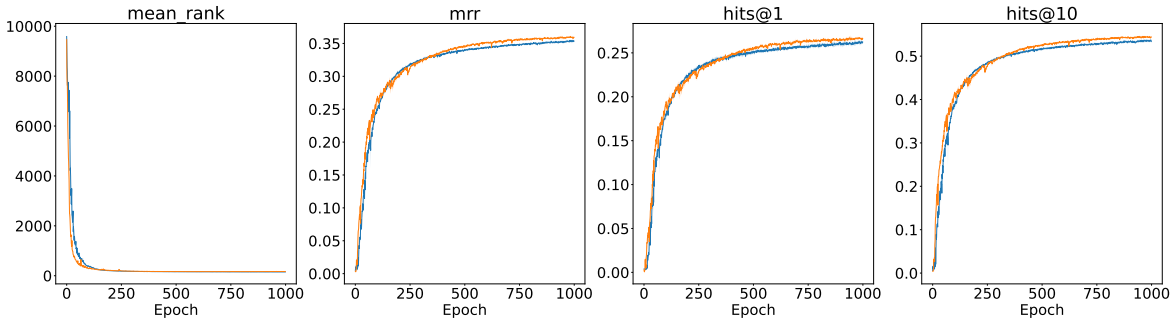
with $x^{(g)}_{head}$ and $x^{(t)}_{tail}$ obtained as in (5.10) and (5.11), representing the graph embedding of the head translated by the relation and the text encoding of the tail's description. Rankings are obtained by calculating the similarity score (5.14) for each description d_i in the dataset and comparing it with the score obtained using the correct tail description d^{tail}.

In Table 5.3 several common link prediction metrics are reported: *Mean Rank* (MR), *Mean Reciprocal Rank* (MRR), *hits@1* and *hits@10*, obtained by a CompGCN (Vashishth, Sanyal et al., 2019) trained with the procedure depicted in Figure 5.1b. In the two datasets considered, the model was able to be competitive with an RGCN (Schlichtkrull et al., 2017) baseline model trained on the standard link prediction task (i.e. making use of both head and tail entities coming from the graph) and significantly outperformed it for the *Mean Rank* metric. Note that due to GPU memory limitations of available hardware, I was not able to perform the above link prediction test on the YAGO3-10 dataset (Mahdisoltani et al., 2015).

[2]Note that the notation used here is still the same adopted in Chapter 4, namely h and t are the entities and r the relation, but the order here is taken to be reversed. This, in any case, has no impact on the on the tractation and just represents an alternative, equally valid, choice.



(a) RGCN



(b) CompGCN

Figure 5.4: Comparison of several link prediction metrics for the FB15k-237 dataset, between the baseline model (blue) and the one that was pretrained on the caption prediction task (orange). An RGCN (Schlichtkrull et al., 2017) encoder ($batchsize = 128$) combined with a Distmult (Yang et al., 2014) head was used in (a), whereas (b) reports the results obtained by a CompGCN (Vashishth, Sanyal et al., 2019) encoder ($batchsize = 1024$) combined with a ConvE (Dettmers et al., 2017) head. The relative metric calculated on the validation set is reported for each epoch of the training. Note that I plot the mean over the performed repetitive runs. The standard deviation is indicated by the shaded regions.

5.6 Link Prediction Finetuning

In the previous section, evidence was found that the procedure introduced in Section 5.3 is able to yield an alignment of the graph and text spaces, making it possible to use to some extent graph and language information interchangeably (cf. 5.5.2). Moreover, since the graph and text encoders are unfrozen and free to change during the alignment process, information can flow from one to the other. As a result, it is expected that the graph encoder might feature richer node representations, which would allow for improved performance on downstream tasks. In particular, without the need to explicitly provide entity descriptions as additional inputs, as for instance proposed in (Xie et al., 2016; Yao et al., 2019).

In order to verify that this is indeed the case, I compare the link prediction performance of a baseline graph encoder and an identical model that has been pretrained in the entity-description matching task. In both cases, on top of the encoder is stacked either a Distmult (Yang et al., 2014) or a ConvE (Dettmers et al., 2017) head to perform link prediction as proposed in (Schlichtkrull et al., 2017; Vashishth, Sanyal et al., 2019). If my hypothesis is correct, the pretrained model should be able to finetune

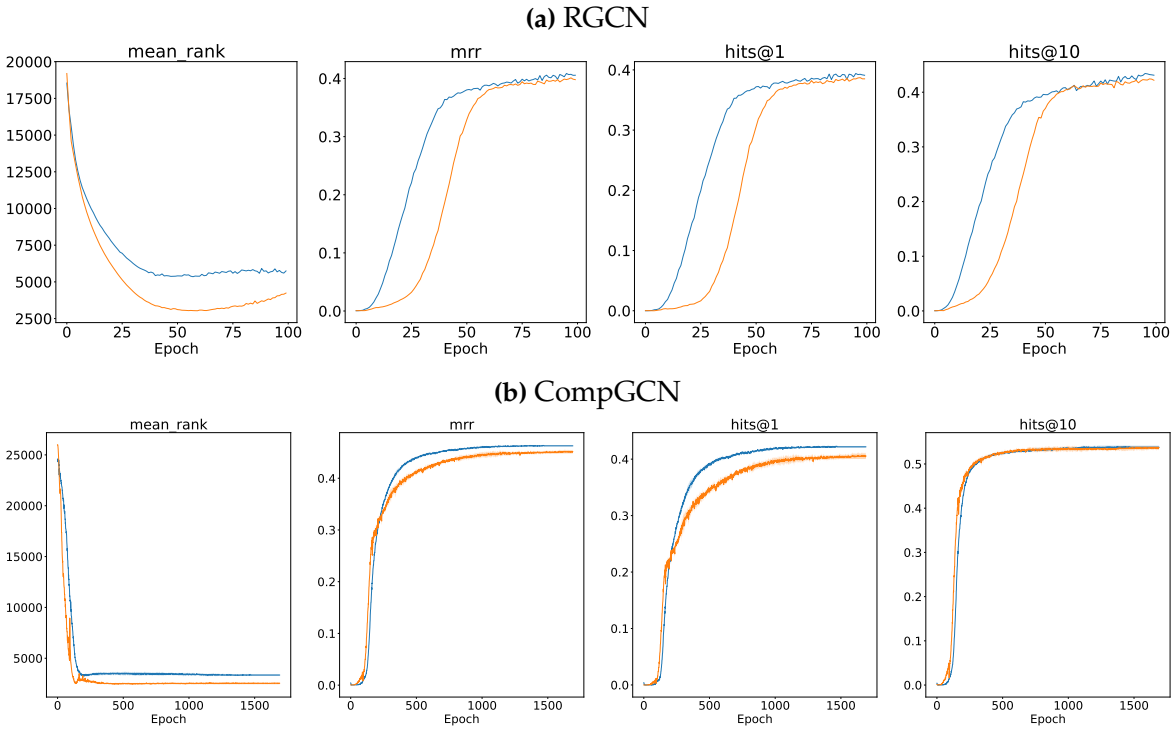


Figure 5.5: Comparison of several link prediction metrics for the WN18rr dataset, between the baseline model (blue) and the one that was pretrained on the caption prediction task (orange). An RGCN (Schlichtkrull et al., 2017) encoder (*batchsize* = 128) combined with a Distmult (Yang et al., 2014) head was used in (a), whereas (b) reports the results obtained by a CompGCN (Vashishth, Sanyal et al., 2019) encoder (*batchsize* = 1024) combined with a ConvE (Dettmers et al., 2017) head. The relative metric calculated on the validation set is reported for each epoch of the training. Note that I plot the mean over the performed repetitive runs. The standard deviation is indicated by the shaded regions.

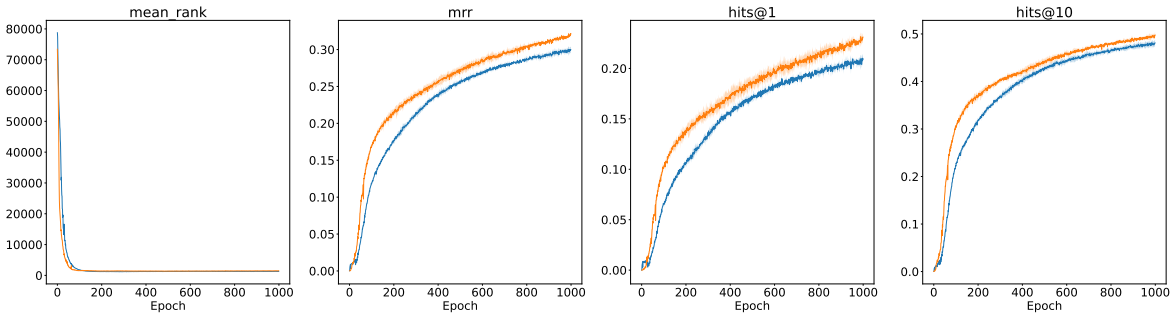


Figure 5.6: Comparison of several link prediction metrics for the YAGO3-10 dataset, between the baseline model (blue) and the one that was pretrained on the caption prediction task (orange). A CompGCN (Vashishth, Sanyal et al., 2019) encoder (*batchsize* = 1024) combined with a ConvE (Dettmers et al., 2017) head was used here. The relative metric calculated on the validation set is reported for each epoch of the training. Note that I plot the mean over the performed repetitive runs. The standard deviation is indicated by the shaded regions.

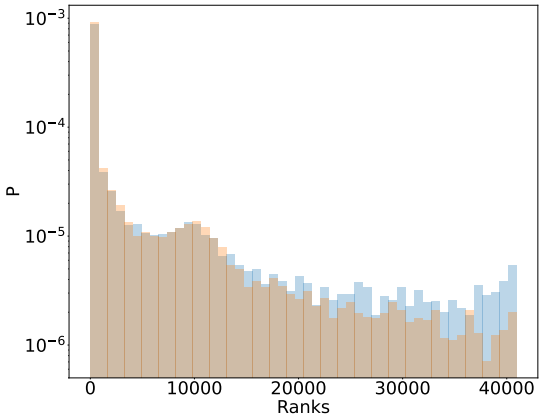


Figure 5.7: Distribution of the ranks obtained in the link prediction task for the WN18RR dataset. I report in orange and blue the results obtained respectively by a CompGCN (Vashishth, Sanyal et al., 2019) pre-trained on the description prediction task and a randomly initialized CompGCN. In both cases, a ConvE (Dettmers et al., 2017) head was used to perform link prediction. The log scale is used for the y-axis for better visualization.

for the task by making use of the additional information learned during pretraining, resulting in stronger performance. Note that this differs from what is presented in Section 5.5.2 as I consider here the standard link prediction scheme in the graph embedding space, i.e. both head and tail entities come from the graph and no entity description is used at this stage.

In Figure 5.4, 5.5 and 5.6 are reported the results obtained for the three datasets considered: FB15k-237 (Toutanova and Chen, 2015), WN18rr (Dettmers et al., 2018) and YAGO3-10 (Mahdisoltani et al., 2015). The baseline model (shown in blue) and the one that has undergone the description matching pretraining (in orange) are compared under the commonly used link prediction metrics: *MR*, *MRR*, *hits@1* and *hits@10*. All the metrics are calculated on the validation set and plotted for each training epoch. The average metric value across several different runs is represented as a solid line and the standard deviation as a shaded area around the mean. In detail, I run three different experiments for each CompGCN (Vashishth, Sanyal et al., 2019) example and ten of them for the RGCN (Schlichtkrull et al., 2017). I opted for a smaller number of runs for the CompGCN due to the larger computational cost.

5.6.1 FB15k-237

For the FB15k-237 dataset, I present the results obtained with an RGCN encoder (Schlichtkrull et al., 2017) combined with a Distmult (Yang et al., 2014) head in Figure 5.4. Number of training epochs equal to 10 together with a batch size of 128 proved to be sufficient to reach convergence. The model that has gone through the caption prediction pretraining easily outperforms the vanilla model in all metrics considered by a significant margin, quite notably in the 10% range for *mrr*. The separation in performance between the two models is clearly visible in Figure 5.4, as

		MR	MRR	hits@1	hits@10
FB15k-237	RGCN *	-	0.249	0.151	0.41
	RGCN+Distmult	315	0.237	0.156	0.407
	CompGCN *	199	0.352	0.264	0.530
	CompGCN+ConvE	151	0.354	0.262	0.534
	$\text{CompGCN}_{\text{CLEP}}$+ConvE	168	**0.359**	**0.266**	**0.544**
	KG-BERT *	153	-	-	0.42
	KG-BERT	137	-	-	0.427
	LASS *	**108**	-	-	0.533
	SimKGC *	-	0.336	0.249	0.511
WN18rr	RGCN+Distmult	6263	0.417	0.397	0.450
	$\text{RGCN}_{\text{CLEP}}$+Distmult	5688	0.408	0.393	0.435
	CompGCN *	3533	0.479	0.443	0.546
	CompGCN+ConvE	3435	0.465	0.422	0.544
	$\text{CompGCN}_{\text{CLEP}}$+ConvE	2620	0.450	0.403	0.539
	KG-BERT *	97	-	-	0.524
	LASS *	**35**	-	-	0.786
	SimKGC *	-	**0.666**	**0.587**	**0.800**
YAGO3-10	CompGCN+ConvE	**1299**	0.302	0.211	0.480
	$\text{CompGCN}_{\text{CLEP}}$+ConvE	1369	**0.323**	**0.233**	**0.498**

Table 5.4: Link prediction performance of the RGCN (Schlichtkrull et al., 2017) and CompGCN (Vashishth, Sanyal et al., 2019) graph encoders with, respectively, a Distmult (Yang et al., 2014) and a ConvE (Dettmers et al., 2017) LP head after finetuning, under the four metrics: MR, MRR, hits@1 and hits@10. The models for which the entity-description matching pretraining has been performed are marked with a *CLEP* subscript. For reference, I also include the original results reported in (Schlichtkrull et al., 2017; Vashishth, Sanyal et al., 2019) for RGCN and CompGCN, as well as several other *state-of-the-art* models that make use of descriptions: KG-BERT (Yao et al., 2019), LASS (Shen et al., 2022) and SimKGC (Wang, Zhao et al., 2022). I indicate with a * all the results that I only report, but did not reproduce. In particular, as I have been using a slightly cut-off version of the FB15k-237 and WN18rr datasets, direct comparison with those models is not possible.

the deviation is larger than the standard deviation obtained across the ten different runs.

Similarly, for a CompGCN encoder (Vashishth, Sanyal et al., 2019) an improvement is observed for the pretrained model over the baseline (it is not easily spot from Figure 5.4 but the numbers reported in Table 5.4 confirm this). However, in this case the difference is smaller, around the $\sim 1-5\%$ range, but still consistent.

The comparison with the original results reported by Schlichtkrull et al. (2017); Vashishth, Sanyal et al. (2019) demonstrate how the reduced datasets maintained the original complexity of the task unaltered. Other *State-of-the-Art* models that make use of descriptions are reported as well, which in this case are slightly topped in the majority of the metrics by the baseline CompGCN encoder and, as a consequence, by my CLEP pretrained one.

5.6.2 WN18rr

In Figure 5.5 are shown the results obtained for a CompGCN encoder (Vashishth, Sanyal et al., 2019) combined with a ConvE head (Dettmers et al., 2017) on the WN18rr dataset. In this case I opted for batches of size 1024. This choice provided much lower variance across the different runs, but required a larger number of epochs to reach convergence ($\sim$ 2000). Unfortunately, for this dataset, it appears evident that the majority of metrics do not benefit from the pretraining procedure. In particular, in the worst case it loses roughly 5% of the *hits*@1 performance. However, it is interesting to note that the *mean rank* still shows an improvement similar to that observed for the FB15k-237 dataset (actually even larger, approaching 20%). In general, this indicates that the pretraining procedure has not the effect of increasing the probability of ranking an entity in a top position in this case, but rather that it helps to decrease the likelihood of having very low rankings. This behavior is easily observable in Figure 5.7, as the pre-trained model exposes a notably lower probability to rank entities in the 20,000-40,000 range.

The results obtained by the RGCN encoder show a very similar pattern, but unfortunately I experienced a large instability during training in several occasions, which led me to report only the result where I was able to obtain a consistent training from the beginning to the end. Even in this case, comparison to Vashishth, Sanyal et al. (2019) demonstrate as the reduced dataset did not decrease the complexity of the LP task, if not even made it slightly harder.

I believe that one possible explanation for this poor performance and discrepancy with the FB15k-237 case, might lie in the type of relations found in the WN18rr dataset. In particular, many of them give information about the grammatical relationship (*e.g.*, *_hypernym* and *_member_meronym*), and therefore rather express the grammatical relation between the words than the relation between actual entities. Hence, the pretraining based on the entity descriptions is not expected to give significant additional information for predicting the (grammatical) relations between words.

This might also explain to some extent the large gap that exists between BERT based models, KG-BERT (Yao et al., 2019), LASS (Shen et al., 2022) and SimKGC (Wang,

Zhao et al., 2022), and the GNN based ones I opted for (*cf.* Table 5.4). Namely, the more "grammatical" nature of the WN18rr dataset, might be more suitable for language based models that were originally trained to understand the structure of sentences and, only later, declined to the LP task.

5.6.3 YAGO3-10

For the YAGO3-10 (Mahdisoltani et al., 2015) dataset I recorded results similar to those obtained in the FB15k-237 (Toutanova and Chen, 2015) dataset, as illustrated in Figure 5.6. Once again, a CompGCN (Vashishth, Sanyal et al., 2019) encoder with a ConvE head (Dettmers et al., 2017) was used and it was trained for 1000 epochs with 1024 batch size. However, the three curves for the different metrics *MRR*, *hits@1* and *hits@10*, suggest that convergence had not been reached yet and that both models probably would have benefited from longer training. Unfortunately, since training for 1000 epochs had been already quite time intensive for this dataset, I could not afford to train further due to computational cost. Nonetheless, since the separation between the two models remains consistent over the first 1000 epochs, I expect that longer training would keep the relative performance between the two models intact, but yield larger absolute values for the metrics.

5.7 Question Answering Finetuning

In the previous section I found evidence of the improved link prediction performance of a graph encoder pre-trained in the entity-description matching task of Section 5.3. This was the result of some of the information contained in the descriptions being able to flow from the language model to the graph encoder during pre-training. Therefore, it is legitimate to imagine whether such process is bidirectional, *i.e.* whether, at the same time, the language model gains additional information from the graph representation learnt by the graph encoder during pre-training.

To answer this question, I will consider in the following the popular NLP task of Question Answering (QA). In this task, a language model is provided with a pair (Q, C) composed of a question Q and a context paragraph C that may or may not contain the answer to the question Q. Then, the model has to highlight the part of the context paragraph C that answers the question Q, if present, or a blank string answer otherwise. Note that it is a common practice to include as many questions with missing answers as those that have one, both during training and at evaluation time especially. The reason being that the capability to refrain from giving plausible, but still wrong, answers is an important feature that a good model should expose, as it provides a strong evidence of the deep understanding of both the question and the context paragraph. For performance evaluation, the canonical precision P, recall R and $F1$ score are commonly used, both separately for the question with and without an answer available and globally overall.

Therefore, to verify what impact the pre-training of Section 5.3 has on the language model, I will compare here two identical text encoders, out of which only one has undergone the entity-description matching pre-training, in the SQuAD (Rajpurkar

	SQuAD					
	Has Ans		No Ans		Overall	
	P	F1	P	F1	P	F1
$\text{DistilBert}_{\text{CLEP (FB15k-237)}}$	**56.44**	**65.00**	55.04	55.04	55.74	60.00
$\text{DistilBert}_{\text{CLEP (YAGO3-10)}}$	51.16	59.73	62.66	62.66	56.92	61.20
$\text{DistilBert}_{\text{CLEP (WN18RR)}}$	50.13	57.87	**73.22**	**73.22**	**61.69**	**65.55**
DistilBert (Sanh et al., 2019)	50.32	58.04	69.13	69.13	59.74	63.60

Table 5.5: Question Answering (QA) performance obtained in the SQuAD (Rajpurkar et al., 2016) by a DistilBert model (Sanh et al., 2019) pre-trained with the procedure depicted in Section 5.3 coupled with a CompGCN (Vashishth, Sanyal et al., 2019) encoder. The Knowledge Graph used in the entity-description pre-training is reported in parenthesis, *e.g.* (FB15k-237). For comparison, the results obtained by a standard DistilBert (Sanh et al., 2019) baseline are also included.

et al., 2016) QA task. In more detail, note that both the language models were pre-trained for learning a text representation first and one of them is further pre-trained as illustrated in 5.3. A span prediction head consisting of a simple linear layer is stacked on top of both models to predict the span of the correct answers. The complete model obtained this way is then finetuned *end-to-end* on the SQuAD (Rajpurkar et al., 2016) QA task.

Note, however, that the SQuAD dataset does not come with an associated Knowledge Graph structure readily available, as it happened for the FB15k-237, WN18rr, and YAGO3-10. This means, in particular, that the text encoder which is finetuned on Question Answering here was actually pre-trained using one of the latter datasets. The larger is the overlapping between the entities contained in the SQuAD sentences and those belonging to one of the other three datasets, the more likely is for the pre-training to be impactful. Unfortunately, measuring the degree of overlap is a non-trivial task. A full entity recognition and linking pipeline would have to be run on the SQuAD dataset, which will eventually result in several errors and, as a consquence, uncertain evaluations.

For the moment, therefore, I can only limit myself to test out text encoders pre-trained on all the three datasets used so far and comment the results obtained. Any deeper analysis, would require more control on the available data.

In Table 5.5 is reported the summary of the performance obtained in the SQuAD (Rajpurkar et al., 2016) task. A DistilBert (Sanh et al., 2019) language model was used in all cases and finetuned for 3 epochs. The models indicated with a *CLEP* subscript were pre-trained on the entity-description matching task of Section 5.3, coupled with a CompGCN (Vashishth, Sanyal et al., 2019) encoder and using one of the three different Knowledge Graphs associated with the corresponding datasets: FB15k-237 (Toutanova and Chen, 2015), YAGO3-10 (Mahdisoltani et al., 2015) or WN18rr (Dettmers et al., 2018).

In two cases out of three, namely when using FB15k-237 and YAGO3-10, the pre-training allowed the model to improve the ability to find the correct answers, at the cost, however, of worse predictions for the questions that do not have one. The two

models, indeed, seem more keen to provide one of the plausible answers, which leads to better performance in the answerable questions, but of course undermines their ability to refrain from answering. Surprisingly, when considering the WN18rr dataset and the associated Knowledge Graph, this behaviour ir reversed. Namely, the performance in the answerable questions are similar to the DistilBert baseline, whereas the model exposes a much improved capability to refrain from providing plausible answers.

5.8 A Model for joint Graph-Text Processing

In summary, in this chapter I proposed an architecture to learn a multi-modal joint graph-text embedding by training a graph encoder and a language model to generate aligned representations of KB entities and their descriptions. I was able to verify successful alignment for three popular Knowledge Bases (Wikidata, Wordnet and YAGO) by measuring the Euclidean distance between embeddings generated for matching entity and description pairs. I also demonstrated that a model trained on the entity-description matching task is able to perform link prediction across the graph and text spaces, providing additional strong evidence of successful alignment.

Furthermore, the experiments on the link prediction task showed that a graph encoder pre-trained with the proposed scheme achieves better performance compared to a randomly initialized encoder in the majority of cases. This indicates that the entity-description matching allows the encoder to learn richer node representations. I suspect that this is due to some of the information contained in the descriptions being able to flow from the language model to the graph encoder during pre-training. Therefore, the graph encoder gained additional textual information, which can be of use for downstream tasks.

This warrants further research to verify that, firstly, the graph encoder manifests the same improvements in other graph-related tasks, such as node classification, and vice versa, that the language model similarly benefits from the joint pre-training. A test was conducted, indeed, on the Question Answering task for the SQuAD dataset which demonstrated mixed results depending on the Knowledge Graph used in the pretraining, *i.e.* derived either from the FB15k-237, WN18rr or the YAGO3-10 dataset. However, since no annotation for the entities was available in the dataset, no precise degree of overlapping or similarity with the entities contained in the pre-training graph could be estimated and, therefore, no clear conclusions could be made on the matter. More suitable candidate datasets for Question Answering, or other similar *language-related* tasks, should probably be tested to deeper investigate the impact of the pre-training on the text encoder.

Moreover, another interesting direction to pursue, could be to test whether the combined use of the two aligned encoders enables zero-shot classification capabilities as demonstrated by CLIP (Radford, Kim et al., 2021). Zero-Shot Entity Linking, for instance, appears to be a perfect candidate for future experiments. Thanks to the text-graph alignment developed here, textual entity mentions could be compared to the graph entities in order to generate a set of possible candidates for the disambiguation, as sketched in Figure 5.8.

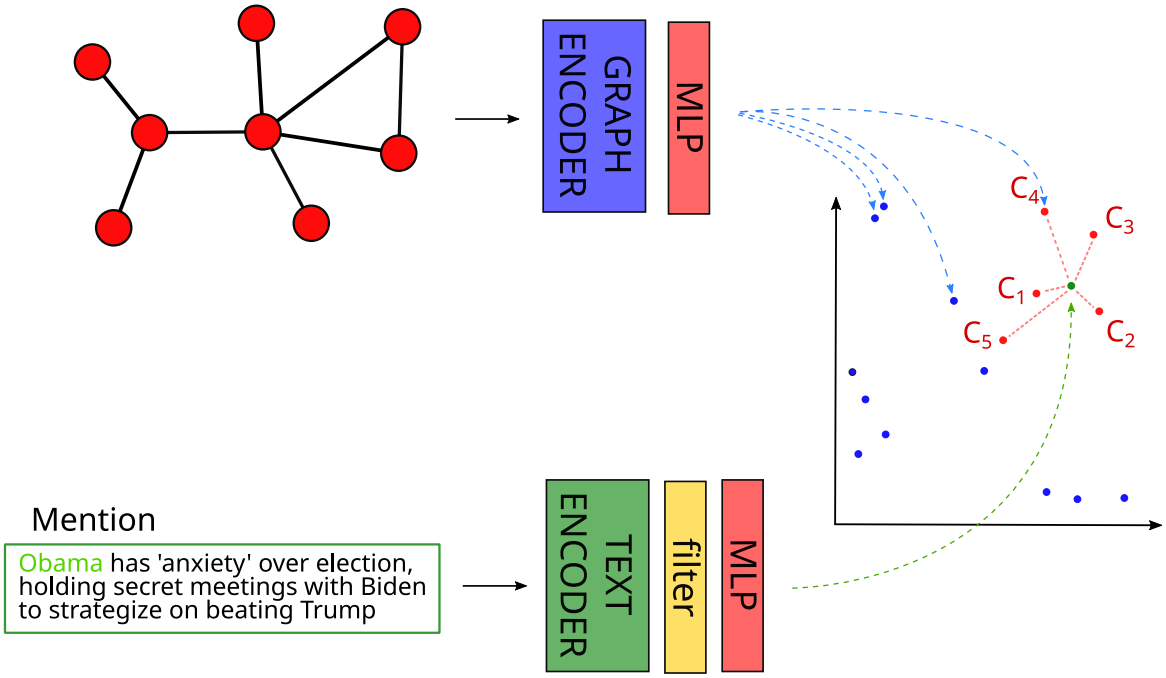


Figure 5.8: A possible zero-shot Entity Linking pipeline based on CLEP. Each textual entity mention is, first, processed by the text encoder, filtered to select the tokens forming the actual entity surface form and, finally, mapped into the aligned text-graph space. At the same time, each node of the selected Knowledge Base is processed by the graph encoder and similarly mapped into the multi-modal space. The set of candidate nodes to disambiguate the entity of interest, is obtained by retrieving the top-n nearest neighbors to the projected entity mention in the embedding space, under a suitable metric.

To conclude, I believe that the multi-modal graph-text embedding presented in this chapter might form a foundation for the joint processing of tasks involving graphical and textual information, similar to what CLIP (Radford, Kim et al., 2021) achieved for text and images. In particular, one example of a such task may be found in graph generation from textual inputs. This is similar in spirit to the popular stable diffusion model for image generation (Rombach et al., 2021), which is based on the multi-modal image-text embedding provided by CLIP (Radford, Kim et al., 2021). The envisaged mechanism of generating a graph from textual inputs can be viewed as an alternative way to perform information extraction from unstructured text and will be another interesting direction for future research.

Nonetheless, the pretraining procedure that I just illustrated here is often quite delicate in practice. In case the considered graph is relatively small, one easily risks to incur in overfitting problems and, to minize the risk, has to necessarily make use of very large batch sizes to grant a wider diversity to each batch and produce a more solid *description-entity* matching task. However, it is easy to understand that this rapidly increases the demand for computational resources, specifically GPU memory (which is expensive), as batches of dimension $\sim 10^4 - 10^5$ quickly translate in the order of hundreds of *Giga Bytes* (GB) of memory needed, depending on the length of the descriptions.

Even in the case those resources were available, though, if the graph information is too incomplete, for instance given a very sparse or poorly connected Knowledge Graph, enlarging the batch size will only partially help to mitigate the risk of poor generalization in the alignment. This in particular is problematic, as Knowledge Graphs and Knowledge Bases are known to naturally suffer from incompleteness. It would therefore be useful to have available methods for automatic Knowledge Graph Completion (KGC), both starting from graph information, such as the Link Prediction task discussed in this Chapter, but also by processing directly unstructured text to extract the relevant knowledge triplets, as presented in the following Chapter.

Chapter 6

Zero- and Few-Shots Knowledge Triplet Extraction with Large Language Models

This chapter was published as Papaluca, Krefl, Rodríguez Méndez et al. (2024) and presented at the Workshop on Knowledge Graphs and Large Language Models (KaLLM) at the Association for Computational Linguistics (ACL) conference 2024 in Bangkok, Thailand, and won the first prize for the best paper award.

In this chapter, I tested the Triplet Extraction (TE) capabilities of a variety of Large Language Models (LLMs) of different sizes in the Zero- and Few-Shots settings. In detail, I proposed a pipeline that dynamically gathers contextual information from a Knowledge Base (KB), both in the form of context triplets and of (sentence, triplets) pairs as examples, and provides it to the LLM through a prompt. The additional context allowed the LLMs to be competitive with all the older fully trained baselines based on the Bidirectional Long Short-Term Memory (BiLSTM) Network architecture. I further conducted a detailed analysis of the quality of the gathered KB context, finding it to be strongly correlated with the final TE performance of the model. In contrast, the size of the model appeared to only logarithmically improve the TE capabilities of the LLMs.

6.1 A dense Representation of Sentences

The task of Triplet Extraction (TE) (Nayak et al., 2021) is of fundamental importance for Natural Language Processing (NLP). This is because the core meaning of a sentence is usually carried by a set of $(subject, predicate, object)$ triplets. Therefore, the capability to identify such triplets is a key ingredient for being able to understand the sentence.

Currently, the State-Of-The-Art (SOTA) for TE is achieved by models that approach the TE task in an *end-to-end* fashion (Fu et al., 2019; Tang et al., 2022; Zeng, He et al., 2019; Zeng, Zeng et al., 2018; Zheng et al., 2017). That is, they are trained to perform all the TE sub-tasks, namely, Named Entity Recognition (NER (Yadav and Bethard, 2018)), Entity Linking (EL (Alam et al., 2022)), and Relation Extraction (RE Detroja et al. (2023)), together. These SOTA models follow the classic NLP paradigm, *i.e.*, they are trained by supervision on specific TE datasets. However, this dependence on labeled data restricts their generality and, therefore, limits the applicability of such models to the real world.

While several labeled datasets for the TE task are publicly available Gardent et al. (2017); Riedel et al. (2010a), these cover only part of the spectrum of possible entities and relations. This means that a supervised model trained on this public data will be restricted to the closed set of entities and relations seen during training, implying that it may lack generalization capabilities. Producing a tailored dataset for training a model for particular applications, is, however, in general expensive Johnson et al. (2018).

For this reason, the recent language understanding and reasoning capabilities demonstrated by Large Language Models (LLMs), such as the Generative Pre-trained Transformer 4 (GPT-4) (OpenAI, 2023), LLM Meta AI (LLaMA) (Touvron et al., 2023), and Falcon (Penedo et al., 2023) to name a few, have led researchers (Chia et al., 2022; Kim et al., 2023; Wadhwa et al., 2023; Wei, Cui et al., 2023; Zhu, Wang et al., 2023) to investigate whether they represent a viable option to overcome the limitations imposed by supervised models for TE. In detail, the new approach being that at inference time the LLMs are prompted to extract the triplets contained in a sentence, while being provided with only a few labeled examples (or no example at all in the Zero-Shot setting). This LLM approach largely limits the amount of data needed to perform the task, and, in particular, lifts the restriction of adhering to a predefined closed set of relations. However, the investigations so far indicated that the Zero and Few-Shots performance of the LLMs appears to be often underwhelming compared to the classic fully trained NLP models (Wadhwa et al., 2023; Wei, Cui et al., 2023; Zhu, Wang et al., 2023).

In order to enhance the abilities of LLMs in the TE task, I propose in this work to aid them with the addition of a Knowledge Base (KB). I demonstrate that augmenting LLMs with KB information, *i.e.*, dynamically gathering contextual information from the KB, largely improves their TE capabilities, thereby making them more competitive with classic NLP baselines. In particular, it is shown that when the retrieved information is presented to the LLMs in the form of complete TE examples, relevant to the input sentence, their performance gets closer to the fully trained SOTA models.

In the next section (Section 6.2), I first provide an outline of those works that experimented with LLMs for TE before. I then proceed, in Section 6.3, to present a pipeline combining LLMs and KBs for the TE task. The proposed pipeline is tested on two standard TE datasets in Section 6.4, and the results of an ablation study are reported in Section 6.4.6. Finally, the closing remarks are presented in Section 6.5.

6.2 Related Work

Classical *end-to-end* fully supervised models currently hold the best performance in the TE task. Starting from the older baseline, Zheng et al. (2017), which also introduced the revised version of the WebNLG dataset for Natural Language Generation (NLG) that is commonly used, several other architectures based on the bidirectional Recurrent Neural Networks (RNNs) (Fu et al., 2019; Zeng, He et al., 2019; Zeng, Zeng et al., 2018) have steadily improved the SOTA over the years. More recently, Transformer-based models achieved a big leap forward in performance, with the recent UniRel model being the current SOTA (Tang et al., 2022).

With the advent of LLMs, Chia et al. (2022) and Kim et al. (2023) tested the use of such models for those TE cases where the availability of examples to train on is low. The first work proposed to use a LLM to generate training examples to finetune a *Relation Extractor* model to recognize relations for which labels were not available. The latter work, instead, suggested using relation templates of the form $\langle X \rangle$ *relation* $\langle Y \rangle$ and finetune a LLM to fill out $\langle X \rangle$ and $\langle Y \rangle$ with the entities appearing in the sentence.

Wadhwa et al. (2023),Wei, Cui et al. (2023), and Zhu, Wang et al. (2023) investigated the general TE task in both Zero- and Few-Shots settings. These studies proposed different approaches based on LLM prompting. The first work tested the Few-Shots performance of GPT-3 (Brown, Mann et al., 2020b) and Text-to-Text Transfer Transformer (T5) (Raffel et al., 2023) under the inclusion of manually-crafted and dataset-dependent contextual information in the prompt. The second work proposed to perform TE by sequentially prompting ChatGPT in two stages: asking to individuate the possible relation types first and then extracting the entities participating in each relation. The procedure demonstrated better results than a one-stage approach where the model is prompted to extract the triplet directly. Finally, the third work, evaluated GPT-3 (Brown, Mann et al., 2020a) and GPT-4 (OpenAI, 2023) on some standard benchmarks in the Zero- and One-Shot settings. However, classical fine-tuned models proved to be superior in the majority of the cases.

In my study, I similarly test the Zero- and Few-Shots capabilities of LLMs in two standard TE datasets that have not been covered by these previous works. In contrast to Wadhwa et al. (2023) that manually crafted static dataset-specific context to be fed to the LLM, I propose here to dynamically gather contextual information useful for extracting the triplets from a KB. This makes my approach more flexible and less data-dependent, as the KB does not require any manual operation and can be easily switched depending on the need. Also, in contrast to other works, I investigate a wide range of Language Models with varying sizes. This allowed me to provide an in-depth analysis of the scaling of the performance, both, from the perspective of

the model chosen, and the quality of the contextual KB information included in the prompt.

6.3 A Pipeline for Triplet Extraction

In this section, I provide a detailed illustration of the pipeline used to test the TE capabilities of LLMs.

6.3.1 Task Formulation

Given a sentence composed of tokens $(t_1, t_2, \cdots, t_N)$, the TE task consists of identifying all the relations expressed in it and extracting them in the form of triplets (s, p, o). Here, $s = (t_i, \cdots, t_{i+n_s})$ and $o = (t_k, \cdots, t_{k+n_o})$ represent a subject and an object of length n_s and n_o tokens, and p is the predicate. Usually, the task is related to a specific KB, *i.e.*, a graph of the form $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, composed of entities $e \in V$ as vertices and relations $r \in \mathcal{E}$ as directed edges. Therefore, s and o of the sentence correspond to vertices $e_s, e_o \in \mathcal{V}$. The predicate p is mapped to a relation included in the closed set of possible edge types of the KB.

6.3.2 LLMs as Triplet Generators

In order to perform TE, I can prompt LLMs to generate for a given input sentence a sequence of tokens corresponding to the set of entity-relation triplets $\left\{(e_s^i, r_p^i, e_o^i)\right\}_{i=1}^{n}$. As demonstrated by Wadhwa et al. (2023), Wei, Cui et al. (2023), and Zhu, Wang et al. (2023), LLMs are, in principle, able to extract the knowledge triplets contained in a text without a need for task-specific training, under a suitable choice of prompt. In general, successful LLM prompts follow a fixed schema that provides a detailed explanation of what the task consists of, a clear indication of the sentence to process, and some hints or examples for the desired result.

In this work, I tested the use of three different prompts: a simple baseline illustrated in Figure 6.2 and the two slight variations of it of Figure 6.3.

However, preliminary testing in TE showed no significant difference in the F1 scores among them, as summarized in Table 6.1. Therefore, I opted for using only the base prompt reported of Figure 6.2 in the main experiments.

6.3.3 Retrieving Information from a KB oracle

In order to support LLMs in the TE task, I propose the pipeline illustrated in Figure 6.1. The pipeline augments the LLM with external KB information. In detail, for each input sentence, relevant context information contained in the KB is retrieved and attached to the LLM prompt described above. The context-enriched prompt is then fed to the LLM for the knowledge triplet generation.

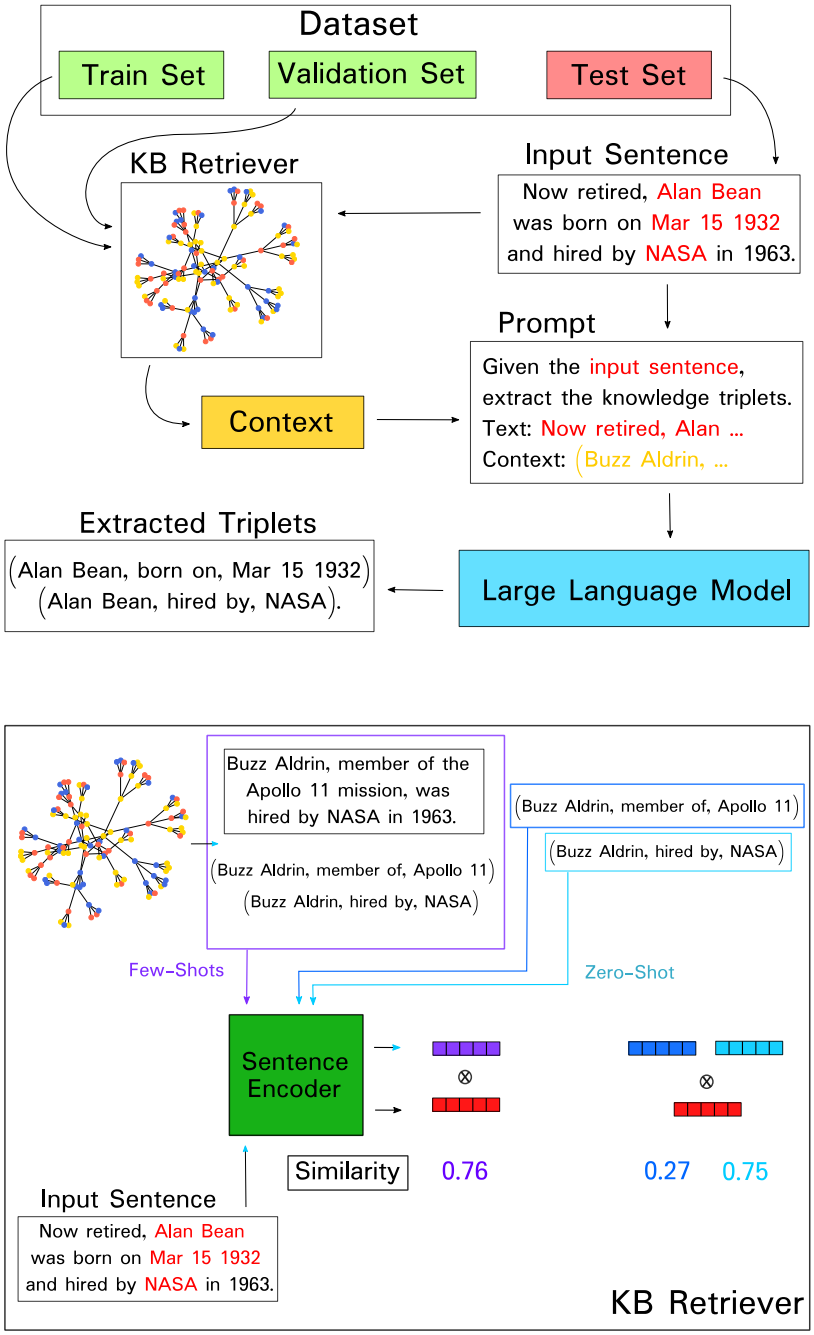


Figure 6.1: The TE pipeline. Top: illustration of the pipeline. A KB is constructed from the training and validation splits of a given dataset. For each test sentence, the relevant contextual information is retrieved from the KB and included in the prompt for a LLM-based TE. Bottom: summary of information retrieval from the KB. Either the sentence-triplets pairs or the single triplets alone are encoded by a sentence encoder and compared to the encoding of the input sentence by cosine similarity.

I prepare the information coming from the KB in two different forms: either as simple context triplets

$$T_c = \left\{ (e_s^i, r_p^i, e_o^i) \right\}_{i=1}^{N_{KB}} \in \mathcal{G} \,, \tag{6.1}$$

or as sentence-triplets pairs

$$E_c = \left\{ (S_c^i, T_c^i) \right\}_{i=1}^{N_{KB}} \,. \tag{6.2}$$

The latter provides factual examples of triplets to be extracted for specific sentences. Note that I indicate with N_{KB} the number of triplets, respectively, sentence-triplets pairs retrieved from the KB.

In the first case, augmentation is achieved by simply attaching the retrieved triplets T_c as an additional "Context Triplets" argument to the base prompt reported in Figure 6.2. For the second approach, instead, I substitute the two static examples

Triplet Extraction Prompt

```
Some text is provided below. Extract up to {max_triplets} knowledge
triplets in the form (subject, predicate, object) from the text.
---------------------------------------------------------------------------------------------------
Examples:

Text: Abilene, Texas is in the United States.
Triplets:
(abilene texas, country, united states)

Text: The United States includes the ethnic group of African
      Americans and is the birthplace of Abrahm A Ribicoff
      who is married to Casey Ribicoff.
Triplets:
(abrahm a. ribicoff, spouse, casey ribicoff)
(abrahm a. ribicoff, birth places, united states)
(united states, ethnic group, african americans)
---------------------------------------------------------------------------------------------------
Text: {text}
Triplets:
```

(a) Base static 2-shots prompt

0.5-Shots Prompt

```
Some text and some context triplets in the form
(subject, predicate, object) are provided below.
Firstly, select the context triplets that are relevant to the input text.
Then, extract up to {max_triplets} knowledge triplets in the form
(subject, predicate, object) contained in the text taking inspiration
from the context triplets selected.
---------------------------------------------------------------------------------------------------
Text: {text}
Context Triplets:
{context_triplets}
Triplets:
```

(b) base KB-aided prompt

Figure 6.2: The base prompt I experimented with. At inference time the $\{text\}$ and $\{max\_triplets\}$ variables are substituted with the sentence to process, respectively, the maximum number of triplets found in a sentence in the corresponding dataset. (a) The plain base prompt with two static examples provided (2-shots), (b) KB-aided adaptation of the base prompt (a), an additional $\{context\_triplets\}$ argument is included to accommodate for the KB triplets retrieved from the KB.

Chain-of-Thought Prompt

```
Some text is provided below. Procede step by step:
- Identify a predicate expressed in the text
- Identify the subject of that predicate
- Identify the object of that predicate
- Extract the corresponding (subject, predicate, object)
  knowledge triplet
- Repeat until all predicates contained in the text have been extracted,
  but no more than {max_triplets} times
---------------------------------------------------------------------------------------------
Text: {text}
Triplets:
```

(a) Chain-of-Thoughts

Documented Prompt

```
Some text is provided below. The text might contain one or more
predicates expressing a relation between a subject and an object.
The subject is the entity that takes or undergo the action expressed by
the predicate.
The object is the entity which is the factual object of the action.
The information provided by each predicate can be summarized as a
knowledge triplet of the form (subject, predicate, object).
Extract all the information contained in the text in the form of
knowledge triplets. Extract no more than {max_triplets} knowledge
triplets.
---------------------------------------------------------------------------------------------
Text: {text}
Triplets:
```

(b) Documented

Figure 6.3: Two variations of the base prompt of Fig. 6.2: (a) a prompt implementing the Chain-of-Thoughts approach of Wei, Wang et al. (2023), and (b) one providing more details about the core components of the TE task, namely, including definitions of subject, object, predicate, and triplet.

	Prompt	WebNLG			NYT		
		P	R	F1	P	R	F1
GPT 2 xl	*base*	0.044	0.031	0.037	0.0002	0.0001	0.0002
	documented	0.040	0.030	0.034	0.0003	0.0002	0.0003
	step-by-step	0.046	0.035	0.039	0.0005	0.0004	0.0004
		0.003	0.002	0.002	0.0001	0.0001	0.0001
LLaMA 65b	*base*	0.213	0.224	0.219	0.013	0.025	0.017
	documented	0.206	0.220	0.213	0.009	0.019	0.012
	step-by-step	0.211	0.227	0.219	0.012	0.023	0.015
		0.003	0.003	0.003	0.001	0.002	0.002

Table 6.1: Comparison of the Zero-Shot triplet extraction micro-averaged performance with the three different prompts 6.2 6.3a and 6.3b. The standard deviation of the performance across the 3 prompts is reported below each column.

provided in the base prompt, with the input relevant examples E_c retrieved from the KB.

The relevant context information to build the T_c triplets set for each input sentence is retrieved as follows. Given the KB, I isolate all the triplets $(e_s^i, r_p^i, e_o^i) \in \mathcal{G}$ contained therein, and store them in a node-based vector store index Liu (2022). In detail, each node of this index corresponds to one and only one of the triplets and stores the embedding obtained by running a small-scale sentence encoder, MiniLM (Wang, Wei et al., 2020), on the corresponding $(subject, predicate, object)$ string. In the first approximation, this should be enough to provide a meaningful embedding for each triplet. Other more sophisticated techniques can be imagined to improve the representation and, therefore, the retrieval. However, as the simple embedding method proved to perform well in my case, I did not investigate further.

During inference (*i.e.*, TE), I first encode the input sentence using the MiniLM. This is followed by comparing the obtained sentence embedding with all the triplet embeddings contained in the index to retrieve the top N_{KB} most similar triplets to the input sentence. Out of this N_{KB}-dimensional sample, I further select the first two triplets for each relation type present in the sample. In case only one triplet is present in the sample for a specific relation type, I only include that one. This is done to obtain a more diverse set of context triplets with a more homogeneous distribution over the relations. In some cases, indeed, the risk of obtaining a highly biased distribution towards a specific relation type exists, which is sub-optimal for those sentences that contain several different relationships.

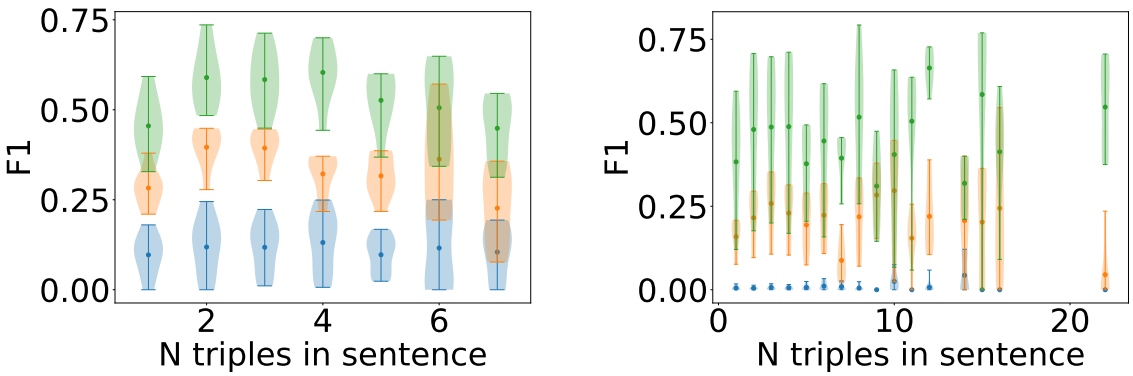
Note that a similar procedure can be followed to prepare the E_c examples set. However, in this case, the focus will be shifted to the example sentences I wish to include. Namely, each node of the vector store index is going to consist both of the example sentence and the KB triplets to be extracted from it. Then, the embedding vector is obtained by running the sentence encoder on either, the example sentence alone, or the sentence and triplets combined. As before, at inference time the top N_{KB} most similar (sentence, triplets) pairs to the input sentence are retrieved and included in the prompt as Few-Shots examples.

6.4 Large Language Models to the Test

In this section, I first provide some details about the datasets and models that I tested. This is followed by the presentation of the main results for the TE task. A summary of the global results is shown in Figure 6.4, where the distribution of the TE F1 scores over the tested models is plotted as violin plots for the three different settings discussed in the upcoming sections.

6.4.1 Datasets and Models

In order to test the TE capabilities of a selected set of LLMs (see Table 6.3 for their comparison), I experimented with two standard benchmarks for the TE task: the aforementioned WebNLG (Gardent et al., 2017) and the New York Times (NYT) (Riedel et al., 2010a) dataset (see Table 6.2 for their basic statistics). The former was initially



(a) WebNLG **(b)** NYT

Figure 6.4: Global summary of the performance obtained by all the models tested in the three settings analyzed in Section 6.4: static 2-Shots (blue), 0.5-Shots (orange), 5-Shots (green). The distribution of the micro-averaged F1 across the LLMs is plotted against the number of triplets contained in a sentence for WebNLG (left panel) and NYT (right panel).

	Train	Validation	Test	Relations	Max	Avg
WebNLG	5,019	500	703	171	7	2.29
NYT	56,195	5,000	5,000	24	22	1.72

Table 6.2: Statistics of the WebNLG and NYT datasets. The number of training, validation, and testing sentences is reported, together with the number of relations types in the dataset and the maximum and average number of triplets contained in a sentence.

	Parameters [B]	Context
GPT-2 (Radford, Wu et al., 2019b)	0.1 | 1.5	1,024
Falcon (Penedo et al., 2023)	7 | 40	2,048
LLaMA (Touvron et al., 2023)	13 | 65	2,048
GPT-3.5 (Brown, Mann et al., 2020a)	175*	4,096
GPT-4 (OpenAI, 2023)	1,760*	8,192

Table 6.3: The number of parameters (in billions [B]) and context window size of the selected LLMs. I indicate with a * the numbers that are not officially confirmed.

proposed as a benchmark for the NLG task, but has been successively adapted to the TE task and included in the WebNLG challenge (Castro Ferreira et al., 2020). As the revision provided by Zheng et al. (2017) appears to be the most widely used in the literature, I decided to run the tests on that particular version of WebNLG. The NYT benchmark is a dataset created by distant supervision, aligning more than 1.8 million articles from the NYT newspaper with the Freebase KB. For each dataset, I used the training and validation splits to build the corresponding KB following the procedure outlined in Section 6.3.3.

I selected the eight different LLMs reported in table 6.3 that varied in their size and datasets they were trained on:

> **GPT-2** Radford, Wu et al. (2019b): The second iteration of the OpenAI LLM pretrained on over 7,000 books from the BookCorpus dataset Zhu, Kiros et al. (2015) and further trained on 8 million web pages. The model comes in different sizes; I used the *base* and *xl* versions with 100 million and 1.5 billion parameters, respectively.
>
> **Falcon** Penedo et al. (2023): Developed by the TII institute and trained on the Falcon RefinedWeb Penedo et al. (2023), a corpus consisting of 2.8TB of public web crawl data. I tested the 7 and 40 billion versions of the model.
>
> **LLaMA** Touvron et al. (2023): A family of LLMs released by Meta AI which has been trained on a large collection of data, including Wikipedia pages, books, scientific papers, StackExchange questions and general web crawl data, for a total of 1.2 trillion tokens. Out of the available sizes I selected the 13 and 65 billion models.
>
> **GPT-3.5-turbo/GPT-4** Brown, Mann et al. (2020a); OpenAI (2023): The most recent OpenAI chat-optimized models. Their number of parameters has not been officially disclosed but, in literature, it is commonly estimated to be around 175 billion and 1,760 billion, respectively.

I ran locally the GPT2, Falcon and LLaMA models in their 8-bit quantized version provided by the HuggingFace (Wolf et al., 2020b) library. For the OpenAI models, that I ran through the OpenAI API (GPT-3.5 and GPT-4), I was not able to find any information regarding the precision they used. The temperature was set to $\tau = 0.1$ for all the experiments. I experimented with higher temperatures but observed that they were detrimental to the TE performance of the model. The temperature parameter τ controls the shape of the Boltzmann distribution from which the generated tokens are sampled. Higher τ values lead to more flat distributions with heavier tails and therefore, a larger probability of sampling uncommon tokens. Hence, τ might be regarded as tuning the "creativity" of the model: with a large τ the same prompt could result in widely differing LLM outputs. However, for the TE task, "creativity" is not of relevance, as there usually exists only a unique solution.

Note that for Falcon and LLaMA LLMs, I also explored their *instructed* counterparts, *i.e.*, models that were fine-tuned for chat applications, either through Reinforcement Learning with Human Feedback (RLHF) Christiano et al. (2023) or supervision from other LLMs Taori et al. (2023). However, as the instructed models always performed on par, or worse, in the preliminary tests, I decided to present the base variants. In

Model	WebNLG			NYT		
	P	R	F1	P	R	F1
NovelTagging Zheng et al. (2017)	0.525	0.193	0.283	0.624	0.317	0.420
CopyRE Zeng, Zeng et al. (2018)	0.377	0.364	0.371	0.610	0.566	0.587
GraphRel Fu et al. (2019)	0.447	0.411	0.429	0.639	0.600	0.619
OrderCopyRE Zeng, He et al. (2019)	0.633	0.599	0.616	0.779	0.672	72.1
UniRel Tang et al. (2022)	**0.948**	**0.946**	**0.947**	**0.935**	**0.940**	**0.937**

Table 6.4: Micro averaged performance of some finetuned models selected from the literature.

Model		WebNLG		NYT	
		0-Shot	2-Shots	0-Shot	2-Shots
GPT-2	base	0.000	0.006	0.000	0.000
	xl	0.000	0.037	0.000	0.000
Falcon	7b	0.000	0.066	0.000	0.002
	40b	0.021	0.158	0.000	0.007
LLaMA	13b	0.006	0.129	0.000	0.002
	65b	**0.041**	**0.219**	0.000	**0.017**
OpenAI	GPT-3.5	0.000	0.144	0.000	0.008
	GPT-4	0.007	0.156	0.000	0.007

Table 6.5: Zero and 2-Shots micro-averaged F1 performance of the LLMs tested with the prompt of Figure 6.2 and without any context coming from the KB.

the case of GPT-3.5 and GPT-4, I used the instructed versions, as at the time being Open AI only provides the base model for GPT-3 (*text-davinci-002*).

I made use of the LlamaIndex (Liu, 2022), LangChain (Chase, 2022) and HuggingFace transformers Wolf et al. (2020b) python libraries for the implementation of the pipeline.

6.4.2 Zero- and 2-Shots without the KB

As a baseline, I test the Zero- and 2-Shots capabilities of the LLMs without any additional information supplemented from a KB.

As described in Section 6.3, I prompt the LLM with the base prompt of Figure 6.2 to extract all the triplets for a sentence in the form (subject, predicate, object). In particular, for the 2-Shots settings, two standard examples are included in the prompt but not changed over the different sentences (*c.f.* Figure 6.2).

In general, the LLMs queried by the base prompt do not seem capable of performing well in the TE task (Table 6.5 summarizes the performance obtained by the LLMs). The two static examples included in the 2-Shots setting help to clarify the task and improve substantially the performance over the Zero-Shot. However, all models struggle to achieve the performance of the classical baseline NLP models, shown

Model		WebNLG		NYT	
		0.5-Shot	5-Shots	0.5-Shot	5-Shots
GPT-2	base	0.249	0.430	0.175	0.375
	xl	0.297	0.517	0.193	0.448
Falcon	7b	0.381	0.567	**0.250**	0.519
	40b	0.345	0.615	0.226	0.547
LLaMA	13b	0.374	0.609	0.247	0.582
	65b	0.377	**0.677**	0.243	**0.647**
OpenAI	text-davinci-002	**0.403**	0.491	0.144	0.418
	GPT-3.5	0.336	0.520	0.088	0.184
	GPT-4	0.394	0.510	0.096	0.151

Table 6.6: 0.5 and 5-Shots micro-averaged F1 performance of the LLMs tested with the prompt of Figure 6.2 augmented with $N_{KB} = 5$ triplets, respectively, sentence-triplets pairs retrieved from the KB.

in Table 6.4. The sole exception is the LLaMA 65B model that achieves an F1 score close to the one obtained by Zheng et al. (2017) in the WebNLG dataset with 2-Shots. In particular, the NYT benchmark appears to be challenging for LLMs as they have difficulties even reaching a mere 1% F1 score. This discrepancy in performance between the datasets could potentially be explained as follows: In contrast to the WebNLG dataset, which features more linear and simple sentences, NYT articles often possess a quite complex structure, with several subordinate clauses and implicit relations. In particular, the triplet labels of the NYT dataset often cover only a subset of the actual relations found in the sentence. Therefore, without training examples available, LLMs cannot infer which relations are and are not supposed to be extracted.

6.4.3 Zero-shot with KB Triplets (0.5-Shots)

If I supplement the LLMs with context triplets retrieved from the KB, as described in Section 6.3.3 and illustrated in Figure 6.1, the performance of the LLM in the TE task increases substantially (see Table 6.6). I refer to this setting where only a set of context triplets, but no example sentence, is provided to the model as 0.5-Shots. The additional triplets hint at which relations and entities the LLM should expect, but they do not give any indication of which sentence pattern they could arise from.

In this case, the smallest model I tested, namely GPT-2 *base*, is competitive with the LLaMA 65B model without context triplets, both, for the WebNLG and the NYT dataset. Furthermore, the bigger models (Falcon, LLaMa, OpenAI) perform better or on par with some of the classical NLP baselines for the WebNLG dataset given in Table 6.4.

Even so for the NYT dataset a large improvement is obtained under the addition of context triplets, all the LLMs are not able to reach scores competitive with the classical NLP models. The reason behind this might be related to the lower capability of the KB retriever to gather relevant context for NYT (*c.f.* Figure 6.5) discussed

below and to the specific difficulties associated with the NYT dataset discussed in the previous section.

In general, it is interesting to observe that the performance with the addition of the context triplets appears to be less dependent on the particular LLM used in case of 0.5-Shot setting. Quite remarkably, the small GPT-2 *xl* is able to retain most of the performance of the larger models. This is particularly evident for the NYT dataset, where all the LLMs are not able to perform better than a 25% F1 threshold. This could be seen as a symptom of the TE accuracy being mainly driven by the added context triplets in this case. Indeed, I also tested this KB triplets augmentation combined with the inclusion of the two static examples used in Section 6.4.2, but no significant differences were observed.

A more general remark regards the underwhelming performance of the OpenAI GPT-3.5 and GPT-4 models on the NYT dataset both, in this 0.5-Shots setting, and in the Few-Shots setting discussed in the next section. A manual inspection of the triplets they provided as an answer suggested that they were less keen to adhere to the entities and relations appearing in the provided KB context, often paraphrasing or reformulating them in a more prolix form that lowered the accuracy. This might be a consequence of the instructed training they had gone through, as discussed in Section 6.4.1. In contrast, the results provided by the "text-davinci-002" model, *i.e.*, the base variant of GPT-3, were more in line with all the other LLMs.

6.4.4 Few Shots with KB Sentence-Triplets Pairs

To further aid LLMs in the TE task, I experiment with the inclusion in the prompt of input-specific (sentence, triplets) example pairs retrieved from the KB, as detailed in Section 6.3.3. Such updated prompts should provide a much stronger signal to the LLM as they not only suggest which entities and relations the LLM should expect, but also which kind of patterns in the sentence correspond to a specific relation. In particular, as it will be discussed in Section 6.4.5, the measured train-test overlapping seems to be large for both datasets (*c.f.* Figure 6.5) and, therefore, the updated prompts are likely to include examples of similar sentences. Therefore, performance improvements are expected, and in fact, looking at Table 6.6, I see that including 5 of these examples in the prompt makes the LLM competitive with most of the classical baselines reported in Table 6.4 (except the most recent SOTA from Tang et al. (2022)).

Interestingly, the performance gap between the two datasets narrowed under the updated prompt. In particular, the NYT corpus seems to have become far easier now for the LLMs. As discussed in Section 6.4.2, this dataset consists of sentences with a much more complex structure and more implicit relations. Therefore, having available examples of similarly constructed sentences might have helped the models to more easily identify the correct triplets. In addition, in this case, an underwhelming performance was observed for the OpenAI GPT-3.5 and GPT-4 models in the NYT dataset (see Section 6.4.3).

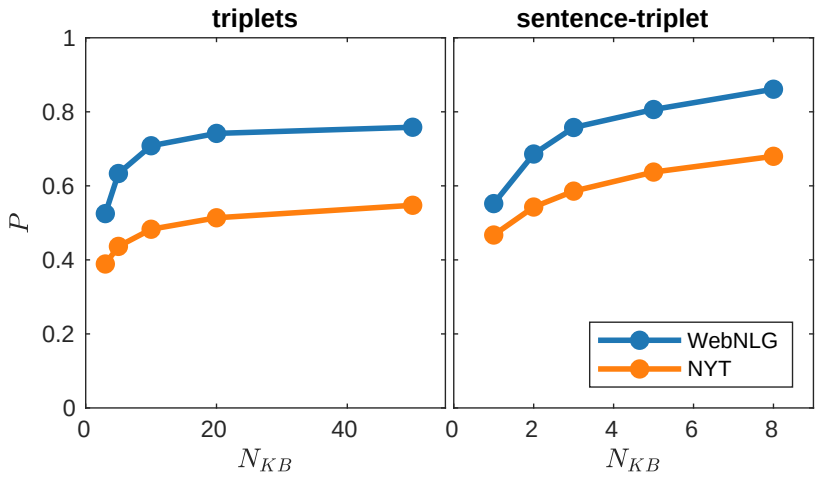


Figure 6.5: Probability that the correct triplet is present inside the retrieved KB context consisting of (left) triplets alone or (right) sentence-triplet example pairs, plotted against the amount of context gathered, N_{KB}.

6.4.5 Quality of the KB Context

To evaluate the effectiveness of the KB retriever and the quality of the included KB context, I plot in Figure 6.5 the probability of finding the correct triplets with increasing N_{KB}, *i.e.*, the solution to the TE task, inside the gathered KB context. Namely, for each test sentence contained in the two datasets, I looped over every labeled triplet and counted the number of times it was contained inside the context provided by the retriever. I repeated this procedure for different values of N_{KB}.

Figure 6.5(left) suggests that $N_{KB} \sim 10 - 20$ retrieved triplets almost maximize the probability of retrieving a useful context already, as, beyond that, the improvement is only marginal. However, as few as five triplets worked the best in my tests (*c.f.* Figure 6.6(left)), with just some exceptions for the sentences that contained a large number of triplets. Probably, a greater number of context triplets retrieved leads to a marginally increased likelihood of including relevant information, but at the cost of a larger dilution. Conversely, as illustrated by Figure 6.5(right), for the sentence-triplets augmentation convergence is not reached with $N_{KB} = 8$ yet. However, in my experiments, the final TE performance only marginally improved going from 5 to 8 sentence-triplets examples included (*c.f.* Figure 6.6(right)). Still, it is interesting to note that LLM performance increases with N_{KB} in this case, providing further evidence that the examples composed of sentence-triplets pairs are much more informative. Adding several of them does not lead to a dilution of useful information, but rather contributes to widening the spectrum of examples the LLM can take inspiration from.

In general, the probability of providing the correct triplet to the LLM through the context appears to be large: greater than 50% in the majority of the cases, and even approaching the $70 \sim 80\%$ for the WebNLG dataset. This is symptomatic of substantial overlap that exists between the training, validation, and test splits for both datasets.

In order to better understand the results obtained by the KB-augmented LLMs, I

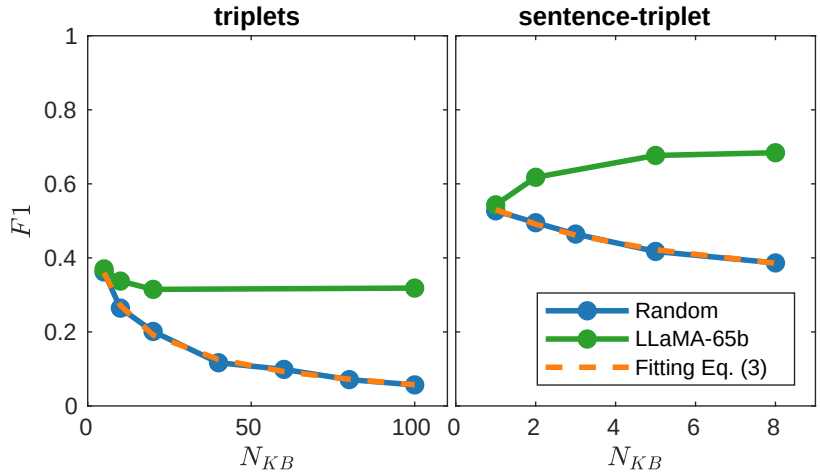


Figure 6.6: Degradation of the random model performance with the increase of the context information included, N_{KB}. The LLaMA-65b, instead, is able to retain most of its performance when more triplets are added (left panel), and sees a significant F1 rise with an increasing number of sentence-triplets pairs (right panel). For reference, I also report the fit of (6.3) as a dashed orange line.

considered the following simple random TE model: first, I randomly select the number of triplets $n \in [1, max\_triplets]$ to extract, with $max\_triplets$ indicating the maximum number of triplets contained in a sentence of the dataset. Then, I uniformly sample n triplets out of the retrieved KB context.

Surprisingly, the random model is very competitive with the KB-augmented LLM for small N_{KB} on the WebNLG dataset (see Figure 6.6), and similar results were observed for the NYT dataset. This can be explained by the hand of Figure 6.5. In detail, we can infer from the figure that the KB-augmented prompt has a large probability of containing the correct triplets to extract already, therefore, even randomly selecting a subset of them yields a relatively high accuracy. This provides further confirmation that the TE performance is largely driven by the KB retriever.

However, the performance of the random model decreases polynomially with N_{KB}, as the probability of randomly sampling the correct triplets follows the empirical scaling relation

$$F1_{rand}(N_{KB}) \sim \left(\frac{P(N_{KB})}{N_{KB}}\right)^n, \tag{6.3}$$

with n number of triplets to extract and $P(N_{KB})$ probability of retrieving the correct triplet from the KB (Figure 6.5).

In contrast, the LLM is able to retain much of its original performance for a larger number of triplets provided (*c.f.* Figure 6.6(left)) or even improve under the inclusion of more sentence-triplets examples (*c.f.* Figure 6.6(right)).

6.4.6 Scaling Down the KB

To further investigate the impact of the additional knowledge retrieved from the KB, I revisit in this section the performance of one of my best performing LLMs,

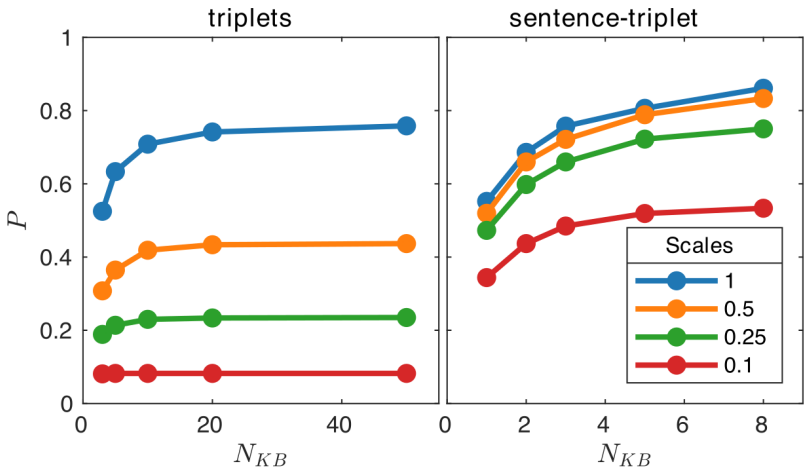


Figure 6.7: Probability that the correct triplet is present among the retrieved KB context information for the WebNLG dataset, with varying number N_{KB} of context triplets (left panel) or sentence-triplets pairs retrieved (right panel).

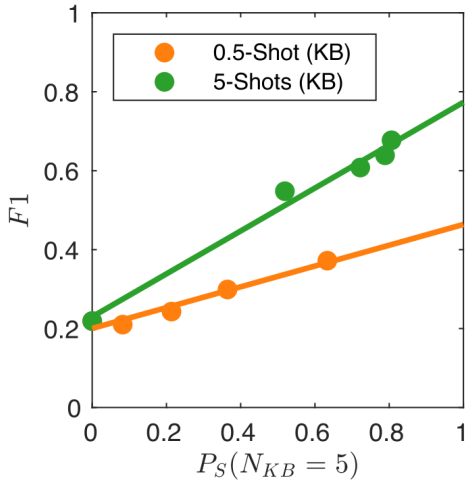


Figure 6.8: Triplets (orange) and sentence-triplets (green) KB augmented performance of the LLaMA-65b model with different scaled-down versions of the KB built for the WebNLG, $S = 0, 0.1, 0.25, 0.5, 1$. The F1 score is plotted against the probability of retrieving the correct triplet with $N_{KB} = 5$ for each S (namely $P(N_{KB} = 5)$ for each curve of Figure 6.7).

LLaMA-65b. In detail, I construct a scaled-down version of the KB via randomly sampling from the original training and validation splits, keeping only a fraction of the original sentences and triplets. For this reduced KB, the probability of having the correct triplet answer already within the retrieved information is reduced (*c.f.* Figure 6.7). This allows us to evaluate how the accuracy of the model is impacted by the quality of the retrieved data.

I decided to conduct this test on the WebNLG dataset. As $P(N_{KB})$ for the full-scale KB has been larger than for the NYT dataset, *c.f.* Figure 6.5, a wider range of values to be explored is allowed. Nonetheless, a preliminary test on the NYT dataset yielded similar results. In Figure 6.8 I report the variation of the final F1 score obtained by LLaMA-65b with prompts augmented by $N_{KB} = 5$ triplets and sentence-triplets pairs gathered from a KB of different scales $S = 0, 0.1, 0.25, 0.5, 1$. Here, the scale refers to the fraction of left-over data from the original KB. Note that $S = 0$ corresponds to the original prompt without any additional information from the KB. In order to have a more general metric, the F1 score is plotted against the probability $P_S(N_{KB} = 5)$ of having the correct triplet inside the retrieved data with $N_{KB} = 5$ for the different KB sizes. This corresponds to the probability curves of Figure 6.7 evaluated at $N_{KB} = 5$.

It is observed that the performance degrades as the probability $P_S(N_{KB})$ shrinks with decreasing S, as expected. In particular, the relation appears to be linear:

$$F1_{triplets} \sim 0.25 \cdot P_s(N_{KB} = 5) + 0.21. \tag{6.4}$$

$$F1_{\substack{sentence \\ -triplets}} \sim 0.55 \cdot P_s(N_{KB} = 5) + 0.21. \tag{6.5}$$

with measured determination coefficients $r^2 = 0.98$ and $r^2 = 0.96$, respectively. This suggests that there is a strong correlation between the TE capabilities of the model and the quality of the retrieved data. This is a further indication that the TE performance is influenced by the retrieved KB context at least as much as by the chosen LLM. Therefore, a natural question arises: *is it better to use larger models or larger KBs with more reliable information retrievers?*

To answer this question, I investigated how the final TE performance scales with the size of the model. In Figure 6.9, the F1 score is plotted against the number of parameters N_{par} in log scale for all the models I tested. The plot includes the results obtained for both the WebNLG and NYT datasets, for all settings considered. For each of the three settings, the models' performance grows linearly in log scale with respect to their sizes, up to the 2-shot NYT example, *c.f.*, related discussions above. However, note that I excluded the GPT-3.5 and GPT-4 0.5- and 5-Shots results obtained on the NYT dataset from the fit, as I consider them to be outliers and not representative, *c.f.* Section 6.4.3.

The scaling in the number of parameters N_{par} in log scale can be approximated by

$$F1_{norm} \sim m \cdot \log N_{par}. \tag{6.6}$$

The slope parameters of the linear fit for WebNLG are $m = 0.0456, 0.0304,$ and 0.0871

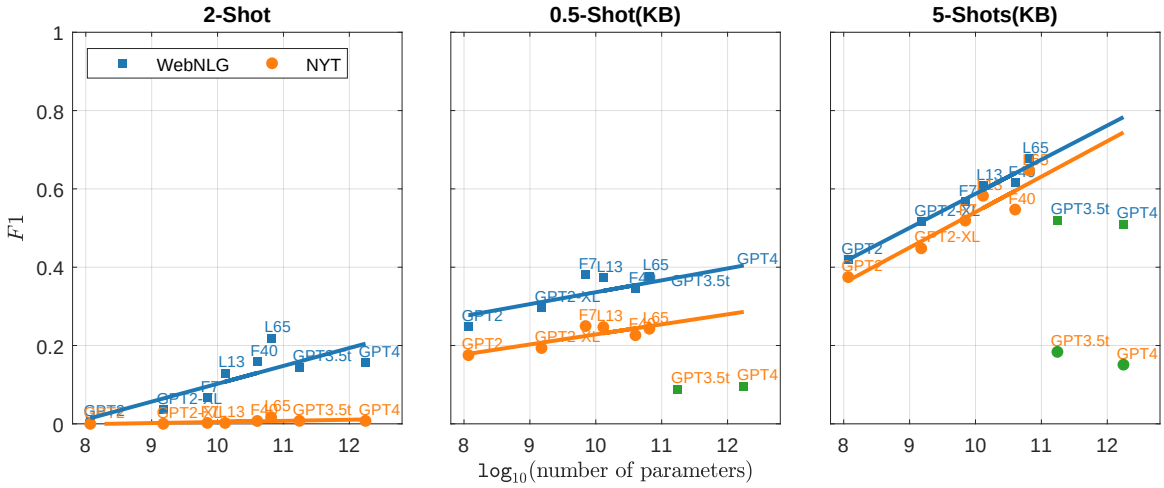


Figure 6.9: $F1$ score obtained by the tested models, plotted against their corresponding log of number of parameters, for WebNLG (blue) and NYT (orange) in the three settings: 2-shots, 0.5-shots with KB triplets ($N_{KB} = 5$), and 5-shots with KB sentence-triplets pairs ($N_{KB} = 5$). The outliers (GPT-4 and GPT-3.5 turbo) are shown in green.

for, respectively, 2-Shot, 0.5-Shot(KB), 5-Shots settings, and for the NYT the corresponding parameters are $m = 0.0028, 0.0257$ and 0.0906. The determination coefficients for the WebLNG and NYT datasets are, respectively, $r^2 = 0.67$, 0.62, and 0.97, and $r^2 = 0.18$, 0.7 and 0.90. Interestingly, the F1 score increase with the size of the model is steeper for the few-shots prompt (*c.f.* Figure 6.9 right). This suggests that larger models might be more capable in making use of several examples included inside of the prompt.

Therefore, the $F1$ score and thus the TE accuracy appears to scale linearly with the size of the KB (*c.f.* Figure 6.8), but only logarithmically with the size of the model (*c.f.* Figure 6.9). This suggests that it could be better to invest resources to improve the quality of the KB and its associated information retriever, rather than in training larger models.

6.5 A Challenging Task for Large Language Models

In this chapter, a pipeline for Zero- and Few-Shots TE from sentences was presented and tested for various LLMs. I showed that the inclusion of KB information into the LLMs prompting can substantially improve the TE performance. In particular, small models were often able to outperform their bigger siblings without access to the additional KB information. Furthermore, with the information from the KB organized as sentence-triplets pair examples relevant to the input sentence, the accuracy of the LLMs improved further. In this setting, the larger LLMs were getting closer to the classical SOTA models and outperformed most of the older baselines.

However, even for the largest models, TE remains a challenging task. Currently, LLMs are still no match for SOTA classical *end-to-end* models in the two standard benchmark datasets I tested as part of my work, in agreement with Wadhwa et al. (2023); Wei, Cui et al. (2023); Zhu, Wang et al. (2023).

Moreover, the performed investigation of the quality of the retrieved KB context in Section 6.4.6 raised questions about the generalization capabilities of LLMs for TE. The LLMs are capable of correctly individuating the relevant information in the context, but they do not shine, yet, in re-elaborating such information, generalizing and making use of it for different examples. The solution to the TE task, indeed, was often contained already in the retrieved KB context so that a model that randomly sampled triplets out of the context was competitive with a large LLM, such as LLaMA-65b, in some cases. The observed main advantage of the LLM being its robustness against dilution of the useful information, *i.e.*, the LLM was able to identify the relevant triplets among a larger amount of retrieved KB context information. On the contrary, the performance of the random model naturally decayed with increased context.

The competitive performance of the random model, which was also able to match some of the old BiLSTM baselines Fu et al. (2019); Zeng, Zeng et al. (2018); Zheng et al. (2017), led me to reconsider the generality of the WebNLG and NYT datasets to benchmark TE capabilities. A revised version of the two with reduced training, validation, and test overlapping might be useful to provide a more realistic evaluation of TE models.

I further investigated how the quality of the KB and of the context retrieved from it impacted the performance of the LLaMA-65b model. The experiment gave evidence of the accuracy of the LLM scaling linearly with the probability of finding the solution of the task within the context collected. At the same time, I found that the TE performance improved only approximately logarithmically with the size of the model. This suggests that improving the quality of the KB and the information retriever might be more effective than increasing the modeling power of the LLM for TE.

Chapter 7

Conclusion

This thesis explored the interplay between Knowledge Graphs and Natural Language, investigating how the two data modalities, and the corresponding models, can be seamlessly integrated to achieve better Natural Language Understanding capabilities. I would like, in the following, to summarize the main contributions of the thesis and highlight the limitations of the conducted work together with some ideas for future developments.

7.1 Thesis Summary and Contributions

Chapter 4 demonstrated how topology-based Knowledge Graph embeddings can be useful to enrich the textual representation learnt by a text encoder or a language model. Namely, it was shown as the simple concatenation of some pre-trained graph embeddings to corresponding textual embeddings of entities, yielded consistent improvements in extracting relations found in text that involved those same entities. To validate this claim, two popular RE datasets, NYT (Riedel et al., 2010a) and Wikidata (Sorokin and Gurevych, 2017), were extended to incorporate the graph embeddings of the entities and made available together with the implementation of the proposed approach under the MIT license on GitHub[1].

The model proved to be very competitive with the *State-of-the-Art* in both the datasets, despite its more cost-effective nature. Furthermore, compared to other related works (Bastos, Nadgeri, Singh, Mulang' et al., 2021; Nadgeri, Bastos, Singh, Mulang' et al., 2021a; Vashishth, Joshi et al., 2018), the presented method solely relied on the topological structure of the graph and did not leverage any ancillary information about the entities, such as the description or the type label. Another main contribution is embodied by the detailed study provided, which analyzed the circumstances under which the graph embeddings are more impactful.

However, despite its effectiveness and low computational cost, this shallow integration appeared to not be extremely flexible. The pre-trained embeddings are fixed, any composition or comparison with their textual counterparts is not possible out of the box, without training a model to specifically make use of both of them. A possibility could be to try and customize or finetune them for the objective of interest, but this is

[1] https://github.com/BrunoLiegiBastonLiegi/Pretrained-KB-Embeddings-for-RE

not straightforward, as a large amount of additional Knowledge Graph data may be required, together with a meaningful target to train for.

For this reason, in Chapter 5, I proposed a procedure to learn aligned multi-modal text-graph embeddings capable of going beyond the shallow integration. Namely, a text and a graph encoder were trained to learn aligned representations of Knowledge Graph nodes, *i.e.* entities, and their corresponding textual descriptions. Three different standard link prediction datasets, FB15k-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018) and YAGO3-10 (Mahdisoltani et al., 2015), were extended by including the descriptions of the entities that was possible to gather from publicly available sources, such as the Wikidata, Wordnet and YAGO Knowledge Bases themselves. The datasets and the implementation of the model were part of a code release[2] licensed under the MIT license.

The successful alignment was verified for all of the datasets and demonstrated how, on average, matching entity-description pairs sit twice as closer than non-matching ones in the multi-modal space. Furthermore, the aligned representation allowed for performing link prediction across the graph and text spaces, *i.e.* making use of head nodes and descriptions of the tails, with performance comparable to a fully-trained and finetuned RGCN (Schlichtkrull et al., 2017) model and without any specific training. This hinted at the fact that graph and textual information could be used interchangeably in the multi-modal space to some extent, providing further evidence of the good alignment achieved.

Furthermore, it was shown as the alignment pretraining often allowed both the graph and text encoders to incorporate features coming from the other modality. In detail, finetuning such encoders respectively on the LP and QA tasks, resulted in improved performance compared to an identical model, but randomly initialized, in the majority of the cases. Comparison with other *State-of-the-Art* LP models that make use of entity descriptions, also demonstrated as the proposed approach often yielded competitive performance, in particular, without requiring the descriptions to be provided explicitly at inference time. Furthermore, such an alignment of text and graph embeddings makes it possible to more easily combine, compare and manipulate information coming from the two different modalities, which opens up new interesting directions for future work, as it will be discussed in more detail in the next section.

This pretraining procedure, however, proved to be very delicate. A large amount of good quality Knowledge Graph data is required to obtain an alignment capable of generalizing well outside of the training data. The data should comprise a wide range of different entities and relations to favour a rich diversity, both from the perspective of the graph structure and of the variety of the textual descriptions. Moreover, the experiments evidenced that a densely connected Knowledge Graph is equally important.

However, Knowledge Graphs are known to suffer from incompleteness and, therefore, methods to automatically complete and enrich them are fundamental. For instance, running an information extraction, *i.e.* Triplets Extraction, pipeline on some

[2] https://github.com/BrunoLiegiBastonLiegi/CLEP

text is a viable way to possibly add new edges, or even nodes, to an existing Knowledge Graph. Nonetheless, this task is known to be non-trivial, and, in particular, because the models are often trained or finetuned on small datasets, they often lack generalization capabilities to *Out-of-Distribution* (OOD) samples, namely, when new entities and relations are encountered.

For this reason, Large Language Models, which are both, capable of solving a wide range of tasks in the zero or few-shot settings, thus without the need of any further training, and known to be in some sense "Open Knowledge Graphs" (Wang, Liu et al., 2020), represent perfect candidates to build Information Extraction, or Knowledge Graph Construction, pipelines in the real world.

In Chapter 6, indeed, I proposed such a pipeline for the extraction of Knowledge Graph triplets from text, based on Large Language Models. A wide range of models with varying number of parameters and amount of training data have been tested in the zero and few-shots Triplet Extraction task. In detail, two standard TE datasets, WebNLG (Gardent et al., 2017) and NYT (Riedel et al., 2010a), were used for testing: they were indexed and organized either in terms of their relations (triplets) or nodes (set of triplets) to provide a KG structure working as a background knowledge reservoir. The set of LLMs tested, moreover, included GPT2 (Radford, Wu et al., 2019b) (*base* and *xl*), LLaMA (Touvron et al., 2023) (*13b* and *65b*), Falcon (Penedo et al., 2023) (*7b* and *40b*), GPT3 (Brown, Mann et al., 2020a) and GPT4 (OpenAI, 2023). The extended datasets, the implementation of the pipeline and the results of the benchmark were all contained inside of the code release[3] that were made available under the MIT license.

The recorded *out-of-the-box* zero-shot performance of the LLMs on the two benchmark datasets proved to be underwhelming, however, and nowhere near the one of fully-trained *State-of-the-Art* models. Providing the LLMs with two additional static examples, 2-shot, slightly improved the situation but did not substantially change the overall figure.

On the other hand, pairing the LLMs with the aforementioned small scale local Knowledge Base, constructed starting from the train and development triplets of each dataset, proved to be very effective. All the LLMs, largely improved their Triplet Extraction capabilities, with several of them often outperforming most of the LSTM based fully trained models.

Nonetheless, a thorough analysis evidenced as the Triplet Extraction performance was largely driven by the quality of the information contained in the KB, and by the ability to contextually extract the relevant facts to be provided to the LLM. In detail, an ablation study with artificially reduced quality and richness of the KB was conducted and demonstrated a linear decrease of the LLM ability to extract the correct triplets. Whereas, the size of the model in terms of its number of parameters appeared to only logarithmically improve the performance, in a first approximation.

The obtained results were found to be in line with other similar works that tested the LLMs capabilities in dealing with relations (Wadhwa et al., 2023; Wei, Cui et al., 2023; Zhu, Wang et al., 2023). At the same time, compared for instance to Wadhwa

[3] https://github.com/BrunoLiegiBastonLiegi/KG-TE-with-LLMs

et al. (2023), that experimented with the addition of static, fixed, information in LLM prompting. This approach, which dynamically gathers the relevant knowledge from a reservoir, appeared to be more flexible: no human effort is required to manually design the prompt to be used, for example, in a new domain, but rather simply replacing or integrating the KB with the relevant information would suffice.

Broadly speaking, even Chapter 2 and Chapter 3, could be enumerated among the contributions of the thesis. Even though, they do not contribute to novel research, but rather consist in a reproposition of consolidated knowledge, I envisage them being an useful entry point to novices of the field. In detail, the former proposes a brief overview of Natural Language Processing, first, providing a little demonstration of the capabilities of modern language models and, afterwards, trying to build some intuition of the concepts and techniques that power them. From the basics, such as the definition of tokenization and vocabularies, going through the old n-grams models and the first context-based words embeddings, to the more recent attention-based transformer neural networks that power the modern Large Language Models. This is followed by an introduction to Knowledge Graphs that provides a brief description of their structure and characteristics. Their connection to Natural Language Processing is also presented by introducing the three tasks forming Information Extraction, which enable the *language-to-graph* transformation. Some examples of what to do with a Knowledge Graph and the structure it provides are illustrated as well: the two tasks of Link Prediction and Question Answering.

7.2 Limitations and Future Work

A main limitation of the work that I conducted in this thesis, concerns the relatively narrow spectrum of different problems or tasks I tested the *Natural Language - Knowledge Graph* integration on. What I mean by such statement is that the focus of the thesis has always been very much directed towards "Relations". Chapter 4 took the problem of Relation Extraction (RE) as the test bench for evaluating the integration. Chapter 5 had as one of its main themes Link Prediction (LP), which again is a form of RE but from the graph perspective. Finally, Chapter 6 focused on Triplet Extraction (TE), which represents a more generalized RE problem where also the entities enter the equation, but where the relation still plays a large role.

Nevertheless, entities are equally important both for Natural Language and in the Knowledge Graph structure, as the tasks centered around them are as well. Named Entity Recognition (NER) and Entity Linking (EL) are all but trivial problems and, as also illustrated in Chapter 2, they constitute a cardinal ingredient for Knowledge Graph Construction (KGC). However, with the exception of Chapter 6 where it was in some sense implicitly required, the capability of correctly extracting and linking the entities present in a text has always been given for granted. Whereas, it would be very interesting to verify what kind of impact the Knowledge Graph integration had on *entity-focused* tasks as well.

Entity Linking, in particular, seems a perfect candidate for this, being a problem that lives precisely on the boundary between Natural Language Processing and Knowledge Graphs. One might even say that it is in fact bridging the two worlds,

as the main objective is to build a mapping between textual entity mentions and their counterparts in the graph. For this reason, models such as the one introduced in Chapter 5, that are capable of jointly and coherently processing both textual and graph information, are promising candidates to push EL capabilities even further.

Future work in this direction, therefore, might vert into more extensively exploring the implications of such an aligned text-graph space. As briefly sketched at the end of Chapter 5, the achieved alignment between the graph and textual representations might be leveraged to generate a set of candidate entities disambiguating a taken textual entity mention. In more detail, through the definition of a suitable metric in the joint text-graph space, one might cluster, based on their reciprocal distance, the embedding of the input entity surface form with the embeddings of the KG nodes. Thus yielding a restricted set of the candidates that are most likely to disambiguate the entity in object.

Another relevant limitation concerns the scalability issues associated with Knowledge Graph based models. Namely, real world Knowledge Graphs often consist of millions of nodes and even more edges, and they are constantly growing in size, day by the day, the more information is added to them. Thus, scalability is a natural requirement for all the models that hope to harness the power of KGs and leverage the knowledge they contain in downstream tasks. However, the studies I conducted in this thesis have always considered the restricted Knowledge Graph associated with the corresponding dataset selected as a benchmark for the tests. Therefore, scalability has never been a major concern, as the total number of nodes and edges comfortably rested in every case several orders of magnitude below the complete graph dimension.

The only exception would be Chapter 4 and the TransE (Bordes, Usunier et al., 2013) based pre-trained graph embeddings I made use of. While it is still true that I restricted those embeddings to the subset of entities and relations relevant for the specific tasks in exam, they were actually coming from the complete Freebase graph, or at least the complete dump of the KG at the time they were trained. Whereas, the use of the full graph in Chapter 5, for instance, would have been impossible, or at best very expensive to aim for, due to the relatively poor scalability of the Graph Convolutional Networks (Schlichtkrull et al., 2017; Vashishth, Sanyal et al., 2019) I proposed to use there.

Therefore, a *trade-off* exists between the scalable but very broad fixed representation provided by the pre-trained graph embeddings, and the dynamic, more task-tailored but hardly scalable embedding allowed by the custom graph model. Further work will be needed to assess in more detail the terms of this trade-off: which approach is better under which circumstances, and, moreover, by which margin. An interesting research route to explore in the future, for instance, could be inspired by the textual analogue of this situation. Namely, the relationship between pre-trained transformer models, such as BERT (Devlin et al., 2018) and GPT (Radford, Wu et al., 2019a), and their finetuning process. These language models were trained on very large corpora with the broad objective of faithfully capturing the structure and semantics of sentences, only to later be combined with small custom layers and finetuned as a whole on the specific tasks of interest. In the same way, one could imagine that the more

simple, but scalable, graph models, as was for TransE in the previous example, could be broadly trained at mimicking graph structures first, and, afterwards, finetuned inside more complex pipelines to better fit the specific graph at hand.

7.3 Broader Impact Statement

The interplay between Knowledge Graphs (KGs) and Natural Language Processing (NLP) explored in this thesis represents a step toward advancing Artificial Intelligence (AI) systems that can reason, understand, and interact with humans more effectively.

From a societal perspective, this work has the potential to democratize access to knowledge by enabling AI systems to provide more accurate, context-aware, and interpretable answers in highly specialized domains. For instance, hybrid systems leveraging specialized KGs and NLP could improve diagnostic support in medicine or assist legal professionals in navigating complex legislation.

The demonstrated cost-efficiency of integrating external KGs with NLP models offers a sustainable alternative to building increasingly larger language models. By reducing computational demands while enhancing model capabilities, this work supports the development of more accessible AI solutions that can be deployed in resource-constrained environments. The utilization of KGs, furthermore, helps to enhance the interpretability and fairness of language models, mitigating their intrinsic lack of transparency

On the academic front, this thesis serves as a foundational guide for future research at the intersection of KGs and NLP, offering methodologies, benchmarks, and insights that could inspire innovations across disciplines.

In summary, the presented work not only had the chance to advance the technical *State-of-the-Art* in some cases, but, more generally, also contributed to shaping an AI landscape that is more efficient and sustainable, hopefully impacting in some part science and society.

7.4 Final Remarks

To conclude, I hope this thesis served as a further indication of the importance of the mutual integration of NLP models and Knowledge Graphs. Even LLMs, which represent the most advanced form of language processing that we have available nowadays, have difficulties in interpreting and elaborating all the facets and complexities of human language. Knowledge Graphs, therefore, help to unlock the full potential of language models. The compact and structured representation of knowledge they provide makes it possible to access, retrieve and elaborate information procedurally, thus enabling automatization. Hence, they serve as reservoirs of world and commonsense knowledge, which allow to, at least in some part, disentangle the complexity of modeling the grammatical structure from the interpretation of the semantics of words, entities and relations. On the other hand, as demonstrated in Chapter 5, textual information and its representation can be helpful to enrich a Knowledge Graph and improve the reasoning capabilities of models that rely on it.

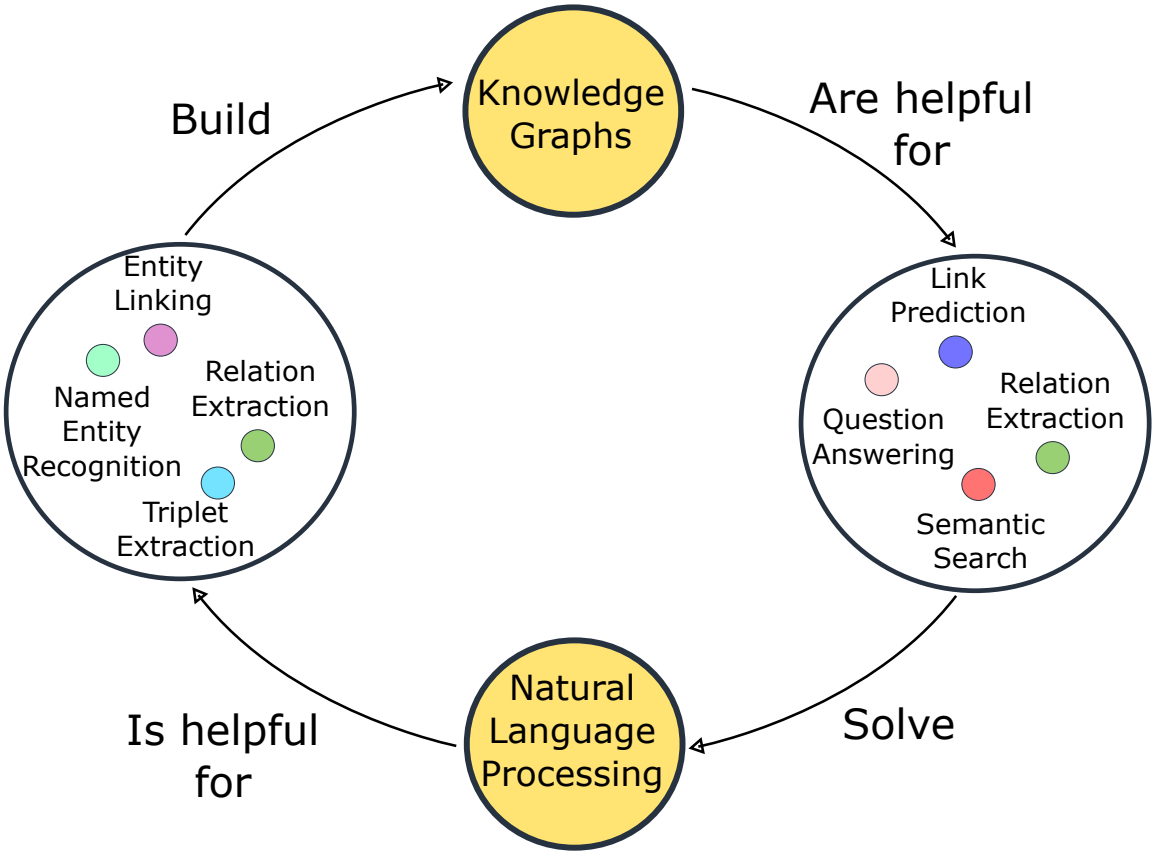


Figure 7.1: The relationship between Knowledge Graphs and Natural Language Processing seen as a positive feedback loop. Knowledge Graphs are helpful to solve several language-related reasoning tasks, such as Question Answering or Relation Extraction and, therefore, aid Natural Language Processing models that make use of them. At the same time, Natural Language Processing models are able to, specifically, extract entities from unstructured text (Named Entity Recognition), disambiguate them in a Knowledge Base (Entity Linking) and find the relationships in the sentences that involve them (Relation Extraction). In other words, they provide the means to build a Knowledge Graph representation of text, which helps in building richer and more complete Knowledge Bases. Hence, a better Knowledge Graph allows for improved language understanding and modeling capabilities, which, in turns, enable more accurate Knowledge Graph construction and completion.

Therefore, the interaction between the two modalities, natural language and its graphical representation, appears to be bidirectional: language models benefit from the addition of Knowledge Graph information and, vice versa, graphical models profit from the integration of textual elements. Therefore, I see their interaction as a positive feedback loop like the one illustrated in Figure 7.1.

Knowledge Graphs provide the means to solve various problems that involve reasoning with natural language. Question Answering, for instance, can be modeled as a link prediction problem. Relation Extraction may be performed thanks to the distant supervision a Knowledge Base provides and the topological structure of the graph allows for more meaningful results in Semantic Search. At the same time, Natural Language Processing models and methods allow for extracting knowledge triplets from text through Named Entity Recognition, Entity Linking and Relation Extraction, thus enabling automatic Knowledge Graph construction and enrichment. A better Knowledge Graph leads to improved language understanding capabilities, which in turns allow for a richer and more accurate knowledge to be extracted from text and included in Knowledge Bases. The ability to jointly make use of the two modalities, therefore, represents in my view a cardinal ingredient to constantly improve the understanding capabilities and robustness of language models in the future.

Bibliography

Alam, M. et al. (2022). 'Neural Entity Linking: A survey of Models Based on Deep Learning'. *Semant. Web*, 13(3), pp. 527–570. DOI: 10.3233/SW-222986 (cited on p. 73).

Angeli, G., Johnson Premkumar, M. J. and Manning, C. D. (2015). 'Leveraging Linguistic Structure For Open Domain Information Extraction'. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 344–354. DOI: 10.3115/v1/P15-1034 (cited on pp. 22, 36).

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L. and Parikh, D. (2015). 'VQA: Visual Question Answering'. In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2425–2433. DOI: 10.1109/ICCV.2015.279 (cited on p. 31).

Araci, D. (2019). *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models* (cited on p. 38).

Bastos, A., Nadgeri, A., Singh, K., Mulang, I. O., Shekarpour, S., Hoffart, J. and Kaul, M. (2021). 'RECON: Relation Extraction Using Knowledge Graph Context in a Graph Neural Network'. In: *Proceedings of the Web Conference 2021*. WWW '21. Ljubljana, Slovenia: Association for Computing Machinery, pp. 1673–1685. ISBN: 9781450383127. DOI: 10.1145/3442381.3449917 (cited on pp. 52, 54).

Bastos, A., Nadgeri, A., Singh, K., Mulang', I. O., Shekarpour, S., Hoffart, J. and Kaul, M. (2021). *RECON: Relation Extraction using Knowledge Graph Context in a Graph Neural Network* (cited on pp. 30, 35, 36, 42, 44, 91).

Bengio, S. (2004). 'Multimodal speech processing using asynchronous Hidden Markov Models'. *Information Fusion*, 5(2). Robust Speech Processing, pp. 81–89. DOI: https://doi.org/10.1016/j.inffus.2003.04.001 (cited on p. 31).

Bodenreider, O. (2004). 'The Unified Medical Language System (UMLS): integrating biomedical terminology'. *Nucleic Acids Research*, 32(suppl_1), pp. D267–D270. DOI: 10.1093/nar/gkh061 (cited on p. 35).

Bollacker, K., Cook, R. and Tufts, P. (2007). 'Freebase: A Shared Database of Structured General Human Knowledge'. In: *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*. AAAI'07. Vancouver, British Columbia, Canada: AAAI Press, pp. 1962–1963. ISBN: 9781577353232 (cited on pp. 16, 35).

Bordes, A., Chopra, S. and Weston, J. (2014). *Question Answering with Subgraph Embeddings*. (cited on p. 28).

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J. and Yakhnenko, O. (2013). 'Translating Embeddings for Modeling Multi-relational Data'. In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Weinberger. Vol. 26. Curran Associates, Inc. (cited on pp. 5, 21, 22, 25, 36, 42, 57, 95).

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L. and Roossin, P. S. (1990). 'A Statistical Approach to Machine Translation'. *Computational Linguistics*, 16(2), pp. 79–85. (cited on p. 11).

Brown, T. B., Mann, B. et al. (2020a). *Language Models are Few-Shot Learners* (cited on pp. 26, 74, 80, 81, 93).

Brown, T. B., Mann, B. et al. (2020b). *Language Models are Few-Shot Learners* (cited on p. 74).

Bunescu, R. and Paşca, M. (2006). 'Using Encyclopedic Knowledge for Named entity Disambiguation'. In: *11th Conference of the European Chapter of the Association for Computational Linguistics*. Ed. by D. McCarthy and S. Wintner. Trento, Italy: Association for Computational Linguistics, pp. 9–16. (cited on p. 20).

Castro Ferreira, T., Gardent, C., Ilinykh, N., Lee, C. van der, Mille, S., Moussallem, D. and Shimorina, A. (2020). 'The 2020 Bilingual, Bi-Directional WebNLG+ Shared Task: Overview and Evaluation Results (WebNLG+ 2020)'. In: *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*. Dublin, Ireland (Virtual): Association for Computational Linguistics, pp. 55–76. (cited on p. 81).

Chase, H. (2022). *LangChain*. (cited on p. 82).

Chen, D., Fisch, A., Weston, J. and Bordes, A. (2017). 'Reading Wikipedia to Answer Open-Domain Questions'. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by R. Barzilay and M.-Y. Kan. Vancouver, Canada: Association for Computational Linguistics, pp. 1870–1879. DOI: 10.18653/v1/P17-1171 (cited on p. 26).

Chia, Y. K., Bing, L., Poria, S. and Si, L. (2022). 'RelationPrompt: Leveraging Prompts to Generate Synthetic Data for Zero-Shot Relation Triplet Extraction'. In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, pp. 45–57. DOI: 10.18653/v1/2022.findings-acl.5 (cited on pp. 32, 73, 74).

Chomsky, N. (1957). *Syntactic Structures*. Mouton (cited on p. 10).

Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S. and Amodei, D. (2023). *Deep reinforcement learning from human preferences* (cited on p. 81).

Cui, H., Peng, T., Bao, T., Han, R., Han, J. and Liu, L. (2022). 'Stepwise relation prediction with dynamic reasoning network for multi-hop knowledge graph question answering'. *Applied Intelligence*, 53, pp. 1–15. DOI: 10.1007/s10489-022-04127-6 (cited on pp. x, 28).

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. and Harshman, R. (1990). 'Indexing by latent semantic analysis'. *Journal of the American Society for*

Information Science, 41(6), pp. 391–407. DOI: https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9 (cited on p. 10).

Detroja, K., Bhensdadia, C. and Bhatt, B. S. (2023). 'A survey on Relation Extraction'. *Intelligent Systems with Applications*, 19, p. 200244. DOI: https://doi.org/10.1016/j.iswa.2023.200244 (cited on p. 73).

Dettmers, T., Minervini, P., Stenetorp, P. and Riedel, S. (2017). *Convolutional 2D Knowledge Graph Embeddings*. DOI: 10.48550/ARXIV.1707.01476 (cited on pp. xii, xvi, 62–67).

Dettmers, T., Minervini, P., Stenetorp, P. and Riedel, S. (2018). 'Convolutional 2D Knowledge Graph Embeddings'. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press. ISBN: 978-1-57735-800-8 (cited on pp. 5, 57, 58, 64, 68, 92).

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. DOI: 10.48550/ARXIV.1810.04805 (cited on pp. 4, 15, 26, 31, 54, 95).

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (cited on pp. 38, 42).

Dozat, T. and Manning, C. D. (2017). *Deep Biaffine Attention for Neural Dependency Parsing* (cited on pp. x, 37, 39).

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books (cited on pp. 10, 58).

Firth, J. R. (2020). 'Papers in linguistics, 1934-1951' (cited on p. 10).

Frome, A., Corrado, G., Shlens, J., Bengio, S., Dean, J., Ranzato, M. and Mikolov, T. (2013). 'DeViSE: A Deep Visual-Semantic Embedding Model'. In: *Neural Information Processing Systems (NIPS)* (cited on p. 31).

Fu, T.-J., Li, P.-H. and Ma, W.-Y. (2019). 'GraphRel: Modeling Text as Relational Graphs for Joint Entity and Relation Extraction'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 1409–1418. DOI: 10.18653/v1/P19-1136 (cited on pp. 32, 73, 74, 82, 90).

Gardent, C., Shimorina, A., Narayan, S. and Perez-Beltrachini, L. (2017). 'Creating Training Corpora for NLG Micro-Planners'. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 179–188. DOI: 10.18653/v1/P17-1017 (cited on pp. 5, 32, 73, 79, 93).

Giorgi, J., Wang, X., Sahar, N., Shin, W. Y., Bader, G. D. and Wang, B. (2019). *End-to-end Named Entity Recognition and Relation Extraction using Pre-trained Language Models* (cited on pp. x, 37–39).

Grishman, R. and Sundheim, B. (1996). 'Message Understanding Conference- 6: A Brief History'. In: *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*. (cited on pp. 18, 21).

Grover, A. and Leskovec, J. (2016). 'node2vec: Scalable Feature Learning for Networks'. *CoRR*, abs/1607.00653. (cited on p. 22).

Harris, Z. S. (1981). 'Distributional Structure'. In: *Papers on Syntax*. Ed. by H. Hiż. Dordrecht: Springer Netherlands, pp. 3–22. ISBN: 978-94-009-8467-7. DOI: 10.1007/978-94-009-8467-7_1 (cited on p. 10).

Ionescu, B. et al. (2022). 'Overview Of The ImageCLEF 2022: Multimedia Retrieval In Medical, Social Media And Nature Applications'. In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 13th International Conference of the CLEF Association, CLEF 2022, Bologna, Italy, September 5–8, 2022, Proceedings*. Bologna, Italy: Springer-Verlag, pp. 541–564. ISBN: 978-3-031-13642-9. DOI: 10.1007/978-3-031-13643-6_31 (cited on p. 52).

Johnson, M., Anderson, P., Dras, M. and Steedman, M. (2018). 'Predicting accuracy on large datasets from smaller pilot data'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Ed. by I. Gurevych and Y. Miyao. Melbourne, Australia: Association for Computational Linguistics, pp. 450–455. DOI: 10.18653/v1/P18-2072 (cited on p. 73).

Jordan, M. I. (1986). 'Serial order: a parallel distributed processing approach. Technical report, June 1985-March 1986'. (cited on p. 15).

Jurafsky, D. and Martin, J. H. (2024). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Online manuscript released August 20, 2024. (cited on p. 9).

Kim, B., Iso, H., Bhutani, N., Hruschka, E., Nakashole, N. and Mitchell, T. (2023). *Zero-shot Triplet Extraction by Template Infilling* (cited on pp. 32, 73, 74).

Kipf, T. N. and Welling, M. (2017). *Semi-Supervised Classification with Graph Convolutional Networks* (cited on pp. 25, 36).

Kolitsas, N., Ganea, O.-E. and Hofmann, T. (2018). 'End-to-End Neural Entity Linking'. In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Ed. by A. Korhonen and I. Titov. Brussels, Belgium: Association for Computational Linguistics, pp. 519–529. DOI: 10.18653/v1/K18-1050 (cited on p. 21).

Lerer, A., Wu, L., Shen, J., Lacroix, T., Wehrstedt, L., Bose, A. and Peysakhovich, A. (2019). *PyTorch-BigGraph: A Large-scale Graph Embedding System* (cited on pp. 36, 37, 39, 40, 42).

Liben-Nowell, D. and Kleinberg, J. (2003). 'The link prediction problem for social networks'. In: *Proceedings of the Twelfth International Conference on Information and Knowledge Management*. CIKM '03. New Orleans, LA, USA: Association for Computing Machinery, pp. 556–559. ISBN: 1581137230. DOI: 10.1145/956863.956972 (cited on p. 22).

Liu, J. (2022). *LlamaIndex*. DOI: 10.5281/zenodo.1234 (cited on pp. 79, 82).

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach* (cited on pp. 38, 42).

Loshchilov, I. and Hutter, F. (2019). *Decoupled Weight Decay Regularization* (cited on p. 40).

Lukovnikov, D., Fischer, A., Lehmann, J. and Auer, S. (2017). 'Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level'. In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, pp. 1211–1220. ISBN: 9781450349130. DOI: 10.1145/3038912.3052675 (cited on p. 28).

Mahdisoltani, F., Biega, J. A. and Suchanek, F. M. (2015). 'YAGO3: A Knowledge Base from Multilingual Wikipedias'. In: *Conference on Innovative Data Systems Research* (cited on pp. 5, 57, 58, 61, 64, 67, 68, 92).

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space* (cited on pp. 10–12, 49).

Mintz, M., Bills, S., Snow, R. and Jurafsky, D. (2009). 'Distant supervision for relation extraction without labeled data'. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 1003–1011. (cited on pp. 30, 35).

Miwa, M. and Bansal, M. (2016). 'End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures'. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1105–1116. DOI: 10.18653/v1/P16-1105 (cited on p. 39).

Mokady, R., Hertz, A. and Bermano, A. H. (2021). *ClipCap: CLIP Prefix for Image Captioning*. DOI: 10.48550/ARXIV.2111.09734 (cited on pp. 31, 52, 53).

Nadgeri, A., Bastos, A., Singh, K., Mulang', I. O., Hoffart, J., Shekarpour, S. and Saraswat, V. (2021a). *KGPool: Dynamic Knowledge Graph Context Selection for Relation Extraction* (cited on pp. 31, 35, 36, 42, 44, 91).

Nadgeri, A., Bastos, A., Singh, K., Mulang', I. O., Hoffart, J., Shekarpour, S. and Saraswat, V. (2021b). 'KGPool: Dynamic Knowledge Graph Context Selection for Relation Extraction'. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, pp. 535–548. DOI: 10.18653/v1/2021.findings-acl.48 (cited on pp. 52, 54).

Nayak, T., Majumder, N., Goyal, P. and Poria, S. (2021). 'Deep Neural Approaches to Relation Triplets Extraction: a Comprehensive Survey'. *Cognitive Computation*, 13(5), pp. 1215–1232. DOI: 10.1007/s12559-021-09917-7 (cited on p. 73).

Nguyen, D. Q. and Verspoor, K. (2019). 'End-to-End Neural Relation Extraction Using Deep Biaffine Attention'. *Advances in Information Retrieval*, pp. 729–738. DOI: 10.1007/978-3-030-15712-8_47 (cited on pp. 37–39).

Nickel, M., Murphy, K., Tresp, V. and Gabrilovich, E. (2016). 'A Review of Relational Machine Learning for Knowledge Graphs'. *Proceedings of the IEEE*, 104(1), pp. 11–33. DOI: 10.1109/JPROC.2015.2483592 (cited on pp. 22, 25).

Nickel, M., Tresp, V. and Kriegel, H.-P. (2011). 'A Three-Way Model for Collective Learning on Multi-Relational Data'. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. Bellevue, Washington, USA: Omnipress, pp. 809–816. ISBN: 9781450306195 (cited on p. 42).

OpenAI (2023). *GPT-4 Technical Report* (cited on pp. 73, 74, 80, 81, 93).

Papaluca, A., Krefl, D., Rodríguez Méndez, S., Lensky, A. and Suominen, H. (2024). 'Zero- and Few-Shots Knowledge Graph Triplet Extraction with Large Language Models'. In: *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*. Ed. by R. Biswas, L.-A. Kaffee, O. Agarwal, P. Minervini, S. Singh and G. de Melo. Bangkok, Thailand: Association for Computational Linguistics, pp. 12–23. (cited on pp. i, 72).

Papaluca, A., Krefl, D., Suominen, H. and Lenskiy, A. (2022). 'Pretrained Knowledge Base Embeddings for improved Sentential Relation Extraction'. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Dublin, Ireland: Association for Computational Linguistics, pp. 373–382. DOI: 10.18653/v1/2022.acl-srw.29 (cited on pp. i, 34, 52, 54).

Paszke, A. et al. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. DOI: 10.48550/ARXIV.1912.01703 (cited on pp. 40, 58).

Patterson, E., Gurbuz, S., Tufekci, Z. and Gowdy, J. (2002). 'CUAVE: A new audio-visual database for multimodal human-computer interface research'. In: *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2, pp. II-2017-II–2020. DOI: 10.1109/ICASSP.2002.5745028 (cited on p. 31).

Pellissier Tanon, T., Vrandečić, D., Schaffert, S., Steiner, T. and Pintscher, L. (2016). 'From Freebase to Wikidata: The Great Migration'. In: *Proceedings of the 25th International Conference on World Wide Web*. WWW '16. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, pp. 1419–1428. ISBN: 9781450341431. DOI: 10.1145/2872427.2874809 (cited on p. 35).

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E. and Launay, J. (2023). *The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only* (cited on pp. 5, 73, 80, 81, 93).

Pennington, J., Socher, R. and Manning, C. D. (2014). 'GloVe: Global Vectors for Word Representation'. In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. (cited on pp. 12, 49).

Powers, D. (2008). 'Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation'. *Mach. Learn. Technol.*, 2 (cited on p. 40).

Radford, A., Kim, J. W. et al. (2021). 'Learning Transferable Visual Models From Natural Language Supervision'. *CoRR*, abs/2103.00020. (cited on pp. 6, 31, 52–54, 69, 71).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019a). 'Language Models are Unsupervised Multitask Learners'. In: (cited on pp. 4, 15, 58, 95).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019b). 'Language Models are Unsupervised Multitask Learners'. In: (cited on pp. 26, 80, 81, 93).

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P. J. (2023). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer* (cited on p. 74).

Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P. (2016). 'SQuAD: 100,000+ Questions for Machine Comprehension of Text'. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 2383–2392. DOI: 10.18653/v1/D16-1264 (cited on pp. xvi, 26, 67, 68).

Riedel, S., Yao, L. and McCallum, A. (2010a). 'Modeling Relations and Their Mentions without Labeled Text'. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by J. L. Balcázar, F. Bonchi, A. Gionis and M. Sebag. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 148–163. ISBN: 978-3-642-15939-8 (cited on pp. 5, 30, 73, 79, 91, 93).

Riedel, S., Yao, L. and McCallum, A. (2010b). 'Modeling Relations and Their Mentions without Labeled Text'. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by J. L. Balcázar, F. Bonchi, A. Gionis and M. Sebag. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 148–163. ISBN: 978-3-642-15939-8 (cited on pp. 30, 40, 44).

Rijsbergen, C. J. V. (1979). *Information Retrieval*. 2nd. USA: Butterworth-Heinemann. ISBN: 0408709294 (cited on p. 20).

Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B. (2021). *High-Resolution Image Synthesis with Latent Diffusion Models*. DOI: 10.48550/ARXIV.2112.10752 (cited on pp. 31, 52, 53, 71).

Rumelhart, D. E. and McClelland, J. L. (1987). 'Learning Internal Representations by Error Propagation'. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pp. 318–362 (cited on p. 15).

Salton, G. and Lesk, M. E. (1965). 'The SMART automatic document retrieval systems—an illustration'. *Commun. ACM*, 8(6), pp. 391–398. DOI: 10.1145/364955.364990 (cited on p. 10).

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019). 'DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter'. *ArXiv*, abs/1910.01108 (cited on pp. xvi, 58, 68).

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I. and Welling, M. (2017). *Modeling Relational Data with Graph Convolutional Networks*. DOI: 10.48550/ARXIV.1703.06103 (cited on pp. xii, xv, xvi, 5, 58, 61–66, 92, 95).

Schuster, M. and Paliwal, K. (1997). 'Bidirectional recurrent neural networks'. *IEEE Transactions on Signal Processing*, 45(11), pp. 2673–2681. DOI: 10.1109/78.650093 (cited on p. 36).

Shannon, C. E. (1948). 'A mathematical theory of communication'. *The Bell System Technical Journal*, 27(3), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x (cited on pp. 9, 11).

Shen, J., Wang, C., Gong, L. and Song, D. (2022). 'Joint Language Semantic and Structure Embedding for Knowledge Graph Completion'. In: *Proceedings of the 29th International Conference on Computational Linguistics*. Ed. by N. Calzolari et al. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, pp. 1965–1978. (cited on pp. xvi, 31, 54, 65, 66).

Sorokin, D. and Gurevych, I. (2017). 'Context-Aware Representations for Knowledge Base Relation Extraction'. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1784–1789. DOI: 10.18653/v1/D17-1188 (cited on pp. 5, 40, 42, 91).

Suchanek, F., Kasneci, G. M. and Weikum, G. M. (2007). 'Yago: A Core of Semantic KnowledgeUnifying WordNet and Wikipedia'. In: *16th international conference on World Wide Web*. Proceedings of the 16th international conference on World Wide Web. Banff, Canada, pp. 697–697. DOI: 10.1145/1242572.1242667 (cited on p. 35).

Tang, W., Xu, B., Zhao, Y., Mao, Z., Liu, Y., Liao, Y. and Xie, H. (2022). 'UniRel: Unified Representation and Interaction for Joint Relational Triple Extraction'. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 7087–7099. DOI: 10.18653/v1/2022.emnlp-main.477 (cited on pp. 32, 73, 74, 82, 84).

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P. and Hashimoto, T. B. (2023). *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca (cited on p. 81).

Tjong Kim Sang, E. F. and De Meulder, F. (2003). 'Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition'. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147. (cited on p. 19).

Tjong Kim Sang, E. F. and Veenstra, J. (1999). 'Representing Text Chunks'. In: *Ninth Conference of the European Chapter of the Association for Computational Linguistics*. Ed. by H. S. Thompson and A. Lascarides. Bergen, Norway: Association for Computational Linguistics, pp. 173–179. (cited on p. 19).

Toutanova, K. and Chen, D. (2015). 'Observed versus latent features for knowledge base and text inference'. In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Beijing, China: Association for Computational Linguistics, pp. 57–66. DOI: 10.18653/v1/W15-4007 (cited on pp. 5, 57, 58, 64, 67, 68, 92).

Touvron, H. et al. (2023). *LLaMA: Open and Efficient Foundation Language Models* (cited on pp. 5, 73, 80, 81, 93).

Vashishth, S., Joshi, R., Prayaga, S. S., Bhattacharyya, C. and Talukdar, P. (2018). 'RESIDE: Improving Distantly-Supervised Neural Relation Extraction using Side

Information'. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 1257–1266. DOI: 10.18653/v1/D18-1157 (cited on pp. 30, 35, 36, 44, 52, 54, 91).

Vashishth, S., Sanyal, S., Nitin, V. and Talukdar, P. (2019). *Composition-based Multi-Relational Graph Convolutional Networks*. DOI: 10.48550/ARXIV.1911.03082 (cited on pp. xi, xii, xv, xvi, 5, 58, 60–68, 95).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017). *Attention Is All You Need* (cited on pp. 12, 13, 38).

Vinyals, O., Toshev, A., Bengio, S. and Erhan, D. (2014). 'Show and tell: A neural image caption generator'. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164. (cited on p. 31).

Vrandečić, D. (2012). 'Wikidata: A New Platform for Collaborative Data Collection'. In: *Proceedings of the 21st International Conference on World Wide Web*. WWW '12 Companion. Lyon, France: Association for Computing Machinery, pp. 1063–1064. ISBN: 9781450312301. DOI: 10.1145/2187980.2188242 (cited on pp. 16, 35, 42, 58).

Wadhwa, S., Amir, S. and Wallace, B. C. (2023). 'Revisiting Relation Extraction in the era of Large Language Models'. *Proceedings of the conference. Association for Computational Linguistics. Meeting*, 2023, pp. 15566–15589. (cited on pp. 32, 73–75, 89, 93).

Wang, C., Liu, X. and Song, D. (2020). *Language Models are Open Knowledge Graphs*. (cited on p. 93).

Wang, L., Zhao, W., Wei, Z. and Liu, J. (2022). 'SimKGC: Simple Contrastive Knowledge Graph Completion with Pre-trained Language Models'. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, pp. 4281–4294. (cited on pp. xvi, 31, 54, 65, 66).

Wang, M., Zheng, D. et al. (2019). 'Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks'. *arXiv preprint arXiv:1909.01315* (cited on p. 58).

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N. and Zhou, M. (2020). *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers* (cited on p. 79).

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. and Zhou, D. (2023). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models* (cited on p. 78).

Wei, X., Cui, X. et al. (2023). *Zero-Shot Information Extraction via Chatting with ChatGPT* (cited on pp. 32, 73–75, 89, 93).

Wolf, T. et al. (2020a). *HuggingFace's Transformers: State-of-the-art Natural Language Processing* (cited on p. 38).

Wolf, T. et al. (2020b). 'Transformers: State-of-the-Art Natural Language Processing'. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, pp. 38–45. (cited on pp. 58, 81, 82).

Xie, R., Liu, Z., Jia, J., Luan, H. and Sun, M. (2016). 'Representation Learning of Knowledge Graphs with Entity Descriptions'. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1). DOI: 10.1609/aaai.v30i1.10329 (cited on pp. 31, 52, 54, 62).

Yadav, V. and Bethard, S. (2018). 'A Survey on Recent Advances in Named Entity Recognition from Deep Learning models'. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 2145–2158. (cited on p. 73).

Yang, B., Yih, W.-t., He, X., Gao, J. and Deng, L. (2014). *Embedding Entities and Relations for Learning and Inference in Knowledge Bases*. DOI: 10.48550/ARXIV.1412.6575 (cited on pp. xii, xvi, 57, 62–65).

Yang, B., Yih, W.-t., He, X., Gao, J. and Deng, L. (2015). *Embedding Entities and Relations for Learning and Inference in Knowledge Bases* (cited on p. 42).

Yao, L., Mao, C. and Luo, Y. (2019). *KG-BERT: BERT for Knowledge Graph Completion*. DOI: 10.48550/ARXIV.1909.03193 (cited on pp. xvi, 31, 52, 54, 62, 65, 66).

Yasunaga, M., Bosselut, A., Ren, H., Zhang, X., Manning, C. D., Liang, P. and Leskovec, J. (2022). *Deep Bidirectional Language-Knowledge Graph Pretraining*. DOI: 10.48550/ARXIV.2210.09338 (cited on pp. 31, 52, 54).

Zeng, X., He, S., Zeng, D., Liu, K., Liu, S. and Zhao, J. (2019). 'Learning the Extraction Order of Multiple Relational Facts in a Sentence with Reinforcement Learning'. 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)*. Hong Kong, China: Association for Computational Linguistics, pp. 367–377. DOI: 10.18653/v1/D19-1035 (cited on pp. 32, 73, 74, 82).

Zeng, X., Zeng, D., He, S., Liu, K. and Zhao, J. (2018). 'Extracting Relational Facts by an End-to-End Neural Model with Copy Mechanism'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 506–514. DOI: 10.18653/v1/P18-1047 (cited on pp. 32, 73, 74, 82, 90).

Zheng, S., Wang, F., Bao, H., Hao, Y., Zhou, P. and Xu, B. (2017). 'Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme'. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1227–1236. DOI: 10.18653/v1/P17-1113 (cited on pp. 32, 73, 74, 81–83, 90).

Zhu, H., Lin, Y., Liu, Z., Fu, J., Chua, T.-S. and Sun, M. (2019). 'Graph Neural Networks with Generated Parameters for Relation Extraction'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 1331–1339. DOI: 10.18653/v1/P19-1128 (cited on p. 36).

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A. and Fidler, S. (2015). *Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books* (cited on p. 81).

Zhu, Y., Wang, X., Chen, J., Qiao, S., Ou, Y., Yao, Y., Deng, S., Chen, H. and Zhang, N. (2023). *LLMs for Knowledge Graph Construction and Reasoning: Recent Capabilities and Future Opportunities* (cited on pp. 32, 73–75, 89, 93).