UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione



MASTER THESIS

Fine-tuning di un Large Language Model mediante l'utilizzo di comunità in Knowledge Graph

Fine-tuning a Large Language Model using Knowledge Graph communities

Supervisor

Prof. Domenico Ursino

Candidate

Christopher Buratti

Co-supervisors

Dott. Luca Virgili

Dott. Michele Marchetti

ACADEMIC YEAR 2023-2024

"The greatest enemy of knowledge is not ignorance,
it is the illusion of knowledge."

Stephen Hawking

Sommario

Negli ultimi anni, l'utilizzo dei Large Language Model e dei Knowledge Graph ha visto una rapida crescita in diversi settori, grazie alle loro capacità complementari nel trattamento e nella gestione delle informazioni. I Large Language Model, addestrati su enormi quantità di dati testuali, spiccano nei task inerenti al Natural Language Processing, ma tendono a soffrire di problemi di accuratezza e allucinazioni. I Knowledge Graph, d'altro canto, sono strutture dati che rappresentano conoscenze sotto forma di entità e relazioni, fornendo un contesto strutturato e solido. Una modalità di integrazione di tali tecnologie è quella di utilizzare la conoscenza del Knowledge Graph di un settore specifico per effettuare il fine-tuning di un Large Language Model, al fine di consentire una generazione di testo più accurata in merito all'ambito di cui il grafo fornisce conoscenza. È proprio in tale contesto che si colloca la presente tesi, che ha come obiettivo quello di efficientare l'intero processo di fine-tuning, utilizzando una sottoparte del grafo di partenza, cercando di garantire, comunque, la più ampia varietà di informazioni e nozioni, grazie alla suddivisone in community del grafo stesso, tramite un algoritmo di community detection.

Keyword: Deep Learning, Reti Neurali a Grafo, Knowledge Graph, Large Language Model, Natural Language Processing, Reti Neurali, Transformer

Abstract

In recent years, the use of Large Language Models and Knowledge Graphs has grown rapidly in several areas due to their complementary capabilities in information processing and management. Large Language Models, which are trained on large amounts of textual data, excel at Natural Language Processing tasks, but they tend to suffer from accuracy and hallucination problems. Knowledge Graphs, on the other hand, are data structures that represent knowledge in terms of entities and relationships, providing a structured and robust context. One way to integrate the two technologies is to use Knowledge Graph knowledge about a specific domain to fine-tune a Large Language Model to enable more accurate text generation on the domain about which the graph provides knowledge. It is in this context that this thesis is set, in fact, it aims to make the entire fine-tuning process efficient by using a subset of the Knowledge Graph. At the same time, it still tries to ensure the widest variety of information and terms through community partitioning of the graph itself using a community detection algorithm.

Keywords: Deep Learning, Graph Neural Networks, Knowledge Graphs, Large Language Models, Natural Language Processing, Neural Networks, Transformers

Contents

Introduction **1**

1 Introduction to Large Language Models **4**
1.1 Language . . . 4
1.1.1 Definitions of language . . . 5
1.1.2 Historical attitudes toward language . . . 6
1.1.3 Ways of characterizing language . . . 7
1.1.4 Types of meaning . . . 8
1.2 Natural Language Processing . . . 9
1.3 Deep Learning and Neural Networks . . . 12
1.4 Transformers . . . 18
1.4.1 Sequence to Sequence Learning and Attention . . . 18
1.4.2 Architecture . . . 19
1.4.3 Types of models . . . 20
1.4.4 Use cases . . . 21
1.5 Large Language Models . . . 21
1.5.1 Definition . . . 23
1.5.2 How they work . . . 24
1.5.3 Use cases . . . 25
1.5.4 Advantages and limitations . . . 26

2 Introduction to Knowledge Graph **28**
2.1 Graph theory . . . 28
2.1.1 Graph theory in Data Science . . . 30
2.2 Graph Neural Networks . . . 32
2.3 Knowledge Graph . . . 34
2.3.1 Definition . . . 35
2.3.2 Use cases . . . 39
2.3.3 Technical challenges . . . 42

3 Related works **46**
3.1 KG-enhanced LLMs . . . 46
3.1.1 KG-enhanced LLM Pre-training . . . 47
3.1.2 KG-enhanced LLM Inference . . . 48
3.1.3 KG-enhanced LLM Interpretability . . . 49

3.2 LLM-augmented KGs . . . 51
3.2.1 LLM-augmented KG Embedding . . . 51
3.2.2 LLM-augmented KG Completion . . . 52
3.2.3 LLM-augmented KG Construction . . . 52
3.2.4 LLM-augmented KG-to-text Generation . . . 52
3.2.5 LLM-augmented KG Question Answering . . . 53
3.3 Combination of synergized LLMs and KGs . . . 54
3.3.1 Synergized Knowledge Representation . . . 54
3.3.2 Synergized Reasoning . . . 55

4 Approach presentation **57**
4.1 Preliminaries . . . 57
4.1.1 Minerva . . . 57
4.1.2 PrimeKG . . . 59
4.2 Technical discussion of the proposed approach . . . 62

5 Approach validation **66**
5.1 Preliminaries . . . 66
5.1.1 Louvain Community Detection Algorithm . . . 66
5.1.2 Metrics . . . 67
5.2 A technical examination of the approach validation process . . . 69

6 Discussion **75**
6.1 Discussion of the proposed approach . . . 75

Conclusions **79**

Bibliography **81**

Sitography **89**

Acknowledgements **90**

List of Figures

1.1 Sentiment Analysis: given a text, it classifies its emotional quality 10
1.2 Example of named entity recognition tagging 11
1.3 Architecture of a Deep Neural Network 14
1.4 Architecture of a Transformer 19
1.5 Evolution process of the four generations of Language Models (LM) 23

2.1 Königsberg bridge problem 29
2.2 Basic categories of graphs 30
2.3 Example of graph-level task 32
2.4 Example of node-level task 33
2.5 Example of edge-level task 33
2.6 Example dataflows in three types of GNNs 34
2.7 GNN pipeline: a graph in input and predictions in output 35
2.8 Example of a Knowledge Graph representation 36
2.9 Difference between a graph, a Knowledge Graph and a Knowledge Graph with inference 37
2.10 RDF: Directed Labeled Cyclic Multigraph with Labels, Types, Logic and Semantics 40

3.1 Retrieving external knowledge to enhance the LLM generation 49
3.2 General framework of using Knowledge Graph for language model probing . 50
3.3 General framework of KG-to-text generation 53
3.4 General framework for the combination of synergized LLM and KG 55
3.5 Synergized knowledge representation by additional KG fusion modules . . . 56

4.1 (a) The figure depicts a schematic overview of the various types of nodes in PrimeKG and the relationships they have with other nodes in the graph. (b) The figure shows all disease nodes in PrimeKG in a circular layout together with disease-associated information. It also depicts all relationships between disease nodes and any other node type. Disease nodes are densely connected to four other node types in PrimeKG through seven types of relations (Chandak *et al.* [2023]). 61

4.2 This illustration depicts a representative example of the pathways in PrimeKG between the disease node "Autism" and the drug node "Risperidone." It also displays excerpts from the text features for both nodes, showcasing the multimodality of PrimeKG. Abbreviations: MF: molecular function, BP: biological process, CC: cellular component, APZ: Apiprazole, EPI: epilepsy, ABP: abdominal pain, +/- associations: positive and negative associations (Chandak *et al.* [2023]). 61
4.3 The workflow of the proposed approach 63

5.1 Example of the Sequence Matcher operation (Munk and Feitelson [2022]) . . . 69
5.2 Model's pipeline from the prompt to the generation, with illustrative examples 73

List of Tables

4.1 Minerva LLM family 58
4.2 Hyperparameters employed for training Minerva models 59
4.3 Information about nodes and node types in PrimeKG (Chandak *et al.* [2023]) . 62
4.4 Statistics on edges in PrimeKG (Chandak *et al.* [2023]) 62

5.1 Main characteristics of the communities selected as the initial Knowledge Graphs for the experiments 69
5.2 Number of data points included in the training, validation, and test sets for each experiment 71
5.3 Mean values of the metrics obtained from the Minerva 350m model on the test sets before fine-tuning 71
5.4 Mean values of the metrics obtained from the Minerva 350m model on the test sets after fine-tuning 72
5.5 Mean values of the metrics obtained from the Minerva 350m model on the test set after fine-tuning in the Experiment 1 73
5.6 Mean values of the metrics obtained from the Minerva 350m model on the test set after fine-tuning in the Experiment 2 73
5.7 Mean values of the metrics obtained from the Minerva 350m model on the test set after fine-tuning in the Experiment 3 74
5.8 Mean values of the metrics obtained from the Minerva 350m model on the test set after fine-tuning in the Experiment 3, with the resolution parameter undergoing variation 74

Introduction

The recent advancements in *Artificial Intelligence* (*AI*), particularly the development of *Large Language Models* (*LLMs*), such as GPT-4, have had a deep impact on the *Natural Language Processing* (*NLP*) area. These models exhibit remarkable capabilities in text generation, question answering, and language understanding, largely due to their extensive training on vast amounts of text data. Nevertheless, despite their overall utility, LLMs are not without critical limitations. Indeed, they frequently manifest hallucination, defined as the proclivity to generate statements that are factually inaccurate or nonsensical. Additionally, they exhibit difficulty in accessing domain-specific or recent knowledge. Moreover, they operate as opaque, or "black-box," models, lacking transparency and interpretability.

In contrast, *Knowledge Graphs* (*KGs*) offer structured, explicit representations of knowledge, storing information in the form of triples (subject, relation, object) and providing a means of linking facts together in a network of knowledge. Knowledge Graphs are of great value due to their precision, decisiveness, and interpretability, particularly in tasks that require clear factual knowledge or symbolic reasoning. However, they also present their own set of challenges, including difficulties in scalability, completeness, and adaptability to new knowledge. The construction and maintenance of a KG, particularly in dynamic, real-world environments, remains a labor-intensive and often incomplete process.

The integration of LLMs with KGs has emerged as a promising research challenge for overcoming the limitations of these models. This integration can be approached in several ways, providing mutual benefits that enhance both models' functionalities. These include approaches such as *KG-enhanced LLMs*, *LLM-augmented KGs*, and *synergized LLMs and KGs*.

In the first approach, i.e., KG-enhanced LLMs, KGs enhance the learning or inference processes of LLMs by embedding structured factual data. The introduction of KGs can occur either during the pre-training phase or at the inference stage, facilitating the access of LLMs to external, structured knowledge, thereby improving their performance in tasks that need precise and reliable information.

In the second approach, i.e., LLM-Augmented KGs, LLMs enhance KGs by improving various KG-related tasks, including embedding, completion, and construction. LLMs, with their capacity to process vast amounts of text and generalize from it, can enrich the representation of entities and relationships in KGs, particularly when dealing with incomplete graphs or previously unseen entities.

Finally, the third approach, i.e., synergized LLMs and KGs, entails the creation of synergized systems wherein LLMs and KGs operate in conjunction to increase each other's performance. In this scenario, LLMs contribute to the development of language understanding and generalization power, while KGs provide factual precision and logical structure. Such

systems enable bidirectional reasoning, whereby LLMs can interpret and generate text based on structured KG data, while KGs can be continuously updated and expanded by extracting information inferred or generated by LLMs.

This thesis aims to integrate LLMs and KGs by means of a KG-enhanced LLM approach. The time and computational resources required for training or fine-tuning an LLM are considerable. Therefore, the objective is to enhance the performance of an LLM via a KG constructed by a specific area, identifying a framework for reducing the time and computational resources required for the fine-tuning process.

The proposed framework facilitates the fine-tuning of an LLM via a KG comprising information about a specific industry in an entirely offline manner. Furthermore, it provides the generated dataset for fine-tuning of other models at the user's discretion, while simultaneously reducing the computational resources required for such a task.

It comprises two principal phases, i.e., dataset generation and model fine-tuning. Each phase is handled in such a way as to reduce the computational complexity while ensuring that the integrity of the knowledge embedded in the Knowledge Graph is maintained.

The initial phase of the process entails the construction of appropriate training and testing datasets derived from the Knowledge Graph. Rather than training the LLM on the whole graph, which is frequently impractical due to the considerable size and intricacy of the majority of KGs, we selectively extract pertinent sections of the graph that are most beneficial for fine-tuning. To accomplish this, we utilize the *Louvain algorithm*, a widely acknowledged community detection method, to partition the KG into smaller, more cohesive communities. Then, we partition the graph into communities and perform random walks within each community. Conducting these walks within the confines of each community allows us to extract sequences of nodes and edges that represent local patterns of interaction. These sequences are crucial for generating training data, as they capture the fine-grained structural properties of the graph. The length of each random walk is determined by the number of nodes and the diameter of the community, in such a way as to that the walks adequately explore the subgraph while remaining computationally efficient. The data obtained from these random walks are subsequently transformed into triples. Each triple comprises a pair of nodes and the relationship that defines the nature of their connection.

The second phase of our framework entails the fine-tuning of the LLM with the domain-specific data generated in the initial phase. The LLM fine-tuning typically needs substantial computational resources, particularly when extensive datasets are involved. To reduce the computational cost and evaluate the proposed approach on more than one Knowledge Graph, we extracted three graph samples from the initial Knowledge Graph.

This thesis was structured into six chapters, as follows:

- Chapter 1 explores the fundamental concepts of Natural Language Processing (NLP) and Neural Networks, with a particular emphasis on the role of Transformers and Large Language Models (LLMs). It illustrates the challenges of computational language comprehension, the evolution of NLP, and the impact of Neural Networks and Deep Learning on the field. Furthermore, it provides a comprehensive examination of the Transformer architecture, investigating its construction, training, and implications for advancing AI and enhancing human-machine communication.

- Chapter 2 presents a comprehensive examination of three interrelated concepts, i.e., graph theory, Graph Neural Networks (GNNs) and Knowledge Graphs (KGs). It begins with an examination of the fundamental principles of graph theory, encompassing pivotal concepts and historical developments. It then proceeds to investigate GNNs, which employ Deep Learning techniques to analyze graph-structured data. Finally, it examines Knowledge Graphs as effective instruments for organizing and querying

intricate datasets, emphasizing their incorporation with GNNs to increase performance and applicability across diverse domains.

- Chapter 3 presents a comprehensive analysis of the potential of integrating Knowledge Graphs with Large Language Models (LLMs) to address challenges like hallucination and interpretability. It begins with an examination of the benefits of utilizing Knowledge Graphs as structured repositories to increase the performance and reliability of LLM. It proceeds to delineate methodologies for integration at both the pre-training and inference stages. Finally, it examines particular techniques for aligning LLM training with knowledge-aware objectives, embedding KGs into input data, and fine-tuning KGs for task-specific guidance.

- Chapter 4 introduces our approach to enhancing Large Language Models through fine-tuning with Knowledge Graph communities. This chapter examines the potential of structured knowledge from Knowledge Graphs, particularly in specialized domains, to enhance an LLM's capacity to process and generate domain-specific information. It begins with an overview of foundational elements, including the Minerva model and PrimeKG, a biomedical Knowledge Graph, before detailing the methodology of integrating LLMs with KG communities to improve reasoning and overcome the generalization limitations of traditional models.

- Chapter 5 outlines experiments designed to evaluate the effectiveness of the proposed approach. The latter plays a pivotal role in identifying community structures, which is essential for organizing data for fine-tuning. The experiments assess the performance of the LLM using two key metrics, i.e., BLEU and Sequence Matcher. To optimize these metrics, we employ a heuristic search, whereby we adjust the resolution value of the Louvain algorithm.

- Chapter 6 presents a discussion of the experimental results, analyzing the effectiveness of fine-tuning Large Language Models with Knowledge Graph communities. It illustrates the strengths and limitations of the approach, compares the model's performance across different metrics, and highlights the key insights gained. Furthermore, it outlines potential improvements and future research directions to further enhance domain-specific LLMs.

CHAPTER 1

Introduction to Large Language Models

This chapter will provide an in-depth exploration of the foundational concepts and recent advancements in Natural Language Processing (NLP) and Neural Networks, with a particular focus on Transformers and Large Language Models (LLMs). The discussion will begin with a detailed examination of the intricacies of human language, elucidating the considerable difficulties it poses for computational comprehension. Subsequently, the chapter will examine the field of Natural Language Processing, tracing its historical development and identifying the fundamental techniques that enable machines to process and interpret human language. This will be followed by a detailed analysis of Neural Networks, including an examination of their architectures, learning mechanisms, and their pivotal role in advancing NLP capabilities. The discussion will then move on to Deep Learning, with particular emphasis on how Deep Neural Networks have transformed the landscape of NLP, achieving levels of language comprehension and generation that were previously unattainable. A substantial portion of the chapter will be dedicated to an in-depth examination of Transformers, encompassing their groundbreaking architectural design, attention mechanisms, and parallel processing capabilities that are poised to establish new benchmarks in NLP tasks. Finally, the chapter will examine Large Language Models, which exploit Transformer architecture to achieve remarkable proficiency in understanding and generating human-like text. The construction, the training paradigms, the advantages and the disadvantages of these models will be thoroughly analyzed. At the end of this chapter, it will be evident that LLMs play an important role in advancing Artificial Intelligence and have the potential to facilitate communication between humans and machines.

1.1 Language

Language is a system of conventional spoken or written symbols by means of which human beings, as members of a social group and participants in its culture, express themselves. The functions of language include communication, identity expression, play, imaginative expression, and emotional release.

So, language is a prominent ability in human beings to express and communicate, which develops in early childhood and evolves over a lifetime. Machines, however, cannot naturally grasp the abilities of understanding and communicating in the form of human language, unless equipped with powerful Artificial Intelligence (AI) algorithms. It has been a longstanding research challenge to achieve this goal, to enable machines to read, write, and communicate like humans.

1.1.1 Definitions of language

A multitude of definitions of language have been put forth over time. Henry Sweet, an English phonetician and language scholar, stated:

> "Language is the expression of ideas by means of speech-sounds combined into words. Words are combined into sentences, and this combination answering to that of ideas into thoughts."

The American linguists Bernard Bloch and George L. Trager formulated the following definition:

> "A language is a system of arbitrary vocal symbols by means of which a social group cooperates."

Any succinct definition of language makes a number of presuppositions and begs a number of questions. The first, for example, puts excessive weight on "thought," and the second uses "arbitrary" in a specialized, though legitimate, way.

Every physiologically and mentally typical person acquires in childhood the ability to make use, as both sender and receiver, of a system of communication that comprises a circumscribed set of symbols (sounds, gestures, written, or typed, characters). In spoken language, this symbol set consists of noises resulting from movements of certain organs within the throat and mouth. In signed languages, these symbols may be hand or body movements, gestures, or facial expressions. By means of these symbols, people are able to impart information, to express feelings and emotions, to influence the activities of others, and to comport themselves with varying degrees of friendliness or hostility toward persons who make use of substantially the same set of symbols.

Different systems of communication constitute different languages; the degree of difference needed to establish different languages cannot be stated exactly. Generally, systems of communication are recognized as different languages if they cannot be understood without specific learning by both parties, though the precise limits of mutual intelligibility are hard to draw and belong on a scale rather than on either side of a definite dividing line. Substantially different systems of communication that may impede but do not prevent mutual comprehension are called dialects of a language. In order to describe in detail the actual different language patterns of individuals, the term idiolect, meaning the habits of expression of a single person, has been coined.

Language, as described above, is specific to human beings. Other members of the animal kingdom have the ability to communicate, through vocal noises or by other means, but the most important single feature characterizing human language, against every known mode of animal communication, is its infinite productivity and creativity. Human beings are unrestricted in what they can communicate; no area of experience is accepted as necessarily incommunicable, though it may be necessary to adapt one's language in order to cope with new discoveries or new modes of thought.

In most accounts, the primary purpose of language is to facilitate communication, in the sense of transmission of information from one person to another. However, sociolinguistic and psycholinguistic studies have drawn attention to a range of other functions for language. Among these is the use of language to express a national or local identity (a common source of conflict in situations of multiethnicity around the world, such as in Belgium, India, and Quebec). Also important are the "ludic" (playful) function of language, encountered in such phenomena as puns, riddles, and crossword puzzles, and the range of functions seen in imaginative or symbolic contexts, such as poetry, drama, and religious expression.

In conclusion, language is a fundamental aspect of human life in society, and its full comprehension necessitates an examination of its role within the social context. The science of

language is known as linguistics. It includes what are generally distinguished as descriptive linguistics and historical linguistics. Linguistics is now a highly technical subject; it embraces, both descriptively and historically, such major divisions as phonetics, grammar (including syntax and morphology), semantics, and pragmatics, dealing in detail with these various aspects of language.

1.1.2 Historical attitudes toward language

As has been shown in the previous discussion, the existence of human life in its current form would be unfeasible and unimaginable without the utilization of language. People have long recognized the force and significance of language. The act of naming, defined as the application of a word to distinguish and identify a person, animal, object, or category of beings, constitutes a fundamental and prominent aspect of linguistic usage.

The origin of language has consistently provided a subject for speculation, and its inaccessibility contributes to its enduring fascination. Investigations into the probable conditions under which language might have originated and developed are exemplified by the late 18th-century essay of the German philosopher Johann Gottfried von Herder, entitled "Abhandlung über den Ursprung der Sprache" ("Essay on the Origin of Language"). However, there have been attempts to discover or to reconstruct the precise forms and structure of the original language. This is a goal that remains forever elusive to the scientific community, given that spoken language in some form is almost certainly coeval with Homo sapiens. The earliest known written language, which represents the only linguistic fossil that humanity can reasonably expect to discover, dates back no more than 4000 to 5000 years.

The relationship between language and thought is a central concern for those engaged in the academic study of language. Several cultures have independently viewed the main function of language as the expression of thought. Ancient Indian grammarians speak of the soul apprehending things with the intellect and inspiring the mind with a desire to speak; in the Greek intellectual tradition Aristotle declared, “Speech is the representation of the experiences of the mind”. Medieval grammarians identified three stages in the speaking process: first, that things in the world exhibit properties; second, that these properties are understood by the minds of humans; and third, that in the manner in which they have been understood, they are communicated to others by the resources of language. The rationalist writers on language in the 17th century offered a comparable account. They asserted that speaking is the act of expressing thoughts through the use of signs that have been specifically devised for this purpose. Additionally, they proposed that the various classes of words (the different parts of speech) have emerged to correspond with the diverse aspects of thinking.

The role of language in social cooperation in everyday life has been increasingly acknowledged, leading to a more comprehensive understanding of language functions across cultures. While the communication of thought remains a central focus for philosophers, the recognition of language’s multifaceted nature has led to a more nuanced approach to its analysis. The medieval and rationalist views implied that humans, as rational, thinking creatures, invented language to express their thoughts, fitting words to an already developed structure of intellectual competence. With the examination of the actual and the probable historical relations between thinking and communicating, it became more plausible to say that language emerged not as the means of expressing already formulated judgments, questions, and the like, but as the means of thought itself, and that humans’ rationality developed together with the development of their capacity for communicating.

The relationship between thought and communication remains an area of significant complexity. Defining thought as subvocal speech, as some behaviorists have done, is a notable oversimplification. However, it is evident that propositions and other purported logical structures are inextricably linked to the language structures that express them. Even

the symbolizations of modern formal logic are ultimately derived from statements made in some natural language and are interpreted in that context.

In conclusion, the intimate connection between language and thought, as opposed to the earlier assumption of a unilateral dependence of language on thought, provided the basis for the recognition of the possibility that different language structures might in part favor, or even determine, different ways of understanding and thinking about the world. All people inhabit a broadly similar world, or they would be unable to translate from one language to another. However, translation is not merely a matter of substituting different, but equivalent, words for the contents of the same inventory. These issues give rise to the well-known difficulties in translation, particularly when the systematizations of science, law, morals and social structure are involved. The extent of the interdependence of language and thought, that has been termed linguistic relativity, remains a matter of debate. However, the fact of such interdependence is beyond dispute.

1.1.3 Ways of characterizing language

Languages are complex structures, and thus can be approached from a variety of perspectives or levels of analysis. Each of these approaches offers insights that are both essential and distinctive, collectively contributing to a comprehensive understanding of the subject matter.

Phonetics and phonology

The most apparent aspect of language is speech. However, despite its prominence, speech is not a defining feature of an infinitely productive communication system, such as a language. Nevertheless, speech is the universal material of most human language, and the conditions of speaking and hearing throughout human history have shaped and determined its development. The study of the anatomy, physiology, neurology, and acoustics of speaking is called *phonetics*. Articulatory phonetics pertains to the physiology of speech, whereas acoustic phonetics concerns the physics of sound waves, including their transmission and reception.

Phonetics covers much of the ground loosely referred to in language study as pronunciation. But, from a rather different point of view, speech sounds are also studied in phonology. Spoken language makes use of a very wide range of the articulations and resultant sounds that are available within the human vocal and auditory resources. Each spoken language uses a somewhat different range, and this is partly responsible for the difficulty of learning to speak a foreign language and for speaking it "with an accent." But mere repertoires of sounds are not all that is involved. Far fewer general classes of sounds are distinctive in any language than the number of sounds that are actually phonetically different. What is distinctive in one language may not be distinctive in another or may be used in a different way; so this is an additional difficulty to be overcome in learning a foreign language. Additionally, languages differ in the ways in which consonant and vowel sounds can be grouped into syllables in words.

All these considerations relating to the use of speech sounds fall under the general heading of phonology, which may be defined as the sound system of a language; phonology is often regarded as one component of language structure.

Grammar

Another component of language structure is *grammar*. There is more to language than sounds, and words are not to be regarded as merely sequences of syllables. The concept of the word is a grammatical concept; in speech, words are not separated by pauses, but they are

recognized as recurrent units that make up sentences. Very generally, grammar is concerned with the relations between words in sentences. Classes of words, or parts of speech, as they are often called, are distinguished because they occupy different places in sentence structure, and in most languages some of them appear in different forms according to their function.

Traditionally, grammar has been divided into *syntax* and *morphology*; syntax deals with the relations between words in sentence structure, whereas morphology deals with the internal grammatical structure of words. It is also different from phonology and vocabulary (see below), though word grammar is often used comprehensively to cover all aspects of language structure. Categories, such as plural, past tense, and genitive case are not phonological categories. In spoken language, like everything else, they are expressed in speech sounds; however, within a language these may be very different for one and the same category.

Grammatical forms and structures are part of the communicative apparatus of languages, and along with vocabulary, or lexicon (which is the stock of individual words in a language), they serve to express all the meanings required. In addition, spoken language has resources such as emphatic stressing and intonation. This is not to say, however, that grammatical categories can be everywhere directly related to specific meanings.

Semantics

Language exists to be meaningful; the study of meaning, both in general theoretical terms and in reference to a specific language, is known as *semantics*. Semantics embraces the meaningful functions of phonological features, such as intonation, and of grammatical structures, as well as the meanings of individual words. It is this last domain, i.e., the lexicon, that forms much of the subject matter of semantics.

A number of areas of investigation fall within the field of semantics. These include the sense relations between words (such as synonymy and antonymy), the nature of the semantic features of word meaning (for example, woman = adult, female, human), and the ways in which words group themselves into domains or semantic fields. It should be noted that questions arising from the relations between semantics, grammar and phonology remain the subject of ongoing debate.

1.1.4 Types of meaning

The whole object and purpose of language is to be meaningful. Languages have developed, and are constituted in their present forms, in order to meet the needs of communication in all its aspects. It is because the needs of human communication are so various that the study of meaning is probably the most difficult and baffling part of the serious study of language. Traditionally, language has been defined as the expression of thought, but this involves far too narrow an interpretation of language or far too wide a view of thought to be serviceable.

Structural meaning

It is first necessary to recognize that the meaning of any given sentence comprises two distinct parts, i.e., the meanings of the words that it contains and the structural or grammatical meaning that is carried by the sentence itself.

The formal resources of any language are constrained by two factors, namely, the linear dimension of speaking and the limited capacity of the human brain. The act of writing replicates the temporal progression of speech through the linear arrangement of written symbols. Diagrams and images utilize two-dimensional representations, whereas models employ three-dimensional forms. However, the permanence of visual marks mitigates the constraints imposed by the limited memory span. Written texts are largely independent of

oral pronunciation, allowing sentence length and complexity to reach unprecedented levels. This is evident in certain legal and legislative documents that are virtually unintelligible when read aloud.

Within these linear restrictions, distinctions corresponding to the main uses of language can be made. All languages can employ different sentence structures to state facts (declarative), to ask questions (interrogative), and to enjoin or forbid some course, of action (imperative). Languages use their resources differently for these purposes, but, generally speaking, each seems to be equally flexible structurally. The principal resources are word order, word form, and, in speech, pitch and stress placement.

Lexical meaning

The other component of sentence meaning is word meaning, which refers to the individual meanings of words in a sentence as lexical items. The concept of word meaning is a familiar one. Dictionaries list words and, in one way or another, state their meanings.

Lexical resources enable languages to maintain the flexibility required by their open-ended commitments. The vocabulary of every language comprises thousands of words, though not all are in active use. Furthermore, some words are known only to relatively few speakers. Perhaps the most common mistake in considering vocabularies is the assumption that the words of different languages, or at least their nouns, verbs, and adjectives, label the same inventory of things, processes, and qualities in the world. However, they do so with different labels from language to language. Nevertheless, the fact that translation is possible indicates that people are talking about similar worlds of experience in their various languages.

The relations between sentence structure and structural meanings are also largely arbitrary and tacitly conventional. The use of loudness and stress for emphasis in spoken languages as well as certain linguistic indications of anger and excitement are akin to nonlinguistic expressions of emotion and are somewhat similar across language divisions. But actual intonations and features, such as word order, word inflection, and grammatical particles, used in maintaining distinctions in structural meaning, differ markedly in different languages.

1.2 Natural Language Processing

Natural Language Processing (*NLP*) is one of the most active and promising areas of Artificial Intelligence. Its applications include text generators that compose coherent essays, chatbots that simulate human-like intelligence, and text-to-image programs that create photo-realistic images of any subject described in text. In recent years, there has been a significant advancement in the capacity of computers to comprehend human languages, programming languages, and even biological and chemical sequences that bear resemblance to language, such as DNA and protein structures. The latest AI models have enabled the analysis of input text and the generation of meaningful, expressive output in these domains.

It is the discipline of building machines that can manipulate human language, or data that resembles human language, in the way that it is written, spoken, and organized. It evolved from computational linguistics, which uses computer science to understand the principles of language, but rather than developing theoretical frameworks, NLP is an engineering discipline that seeks to build technology to accomplish useful tasks. NLP can be divided into two overlapping subfields: *Natural Language Understanding* (*NLU*), which focuses on semantic analysis or determining the intended meaning of text, and *Natural Language Generation* (*NLG*), which focuses on text generation by a machine.

NLP is important because it is an integral part of everyday life, and is becoming more so as language technology is applied to fields as diverse as retail (for example, in customer service chatbots) and medicine (interpreting or summarising electronic health records). Conversational agents, such as Amazon's Alexa and Apple's Siri, use NLP to listen to user queries and find answers. The most sophisticated of these agents, such as GPT-3, which was recently opened for commercial applications, can generate sophisticated prose on a wide range of topics, as well as powering chatbots that can carry on coherent conversations. Google uses NLP to improve its search engine results, and social networks such as Facebook use it to detect and filter hate speech. Despite the growing sophistication of NLP, significant challenges remain. Current systems are prone to bias and incoherence, and occasionally exhibit erratic behavior. However, Machine Learning engineers have numerous opportunities to apply NLP in ways that are increasingly central to a functioning society.

NLP is used for a variety of language-related tasks, including answering questions, classifying text in different ways, and conversing with users. Here are some tasks that can be solved using NLP.

- *Sentiment analysis*: It is the process of classifying the emotional intent of text. Generally, the input to a sentiment classification model is a piece of text, and the output is the probability that the sentiment expressed is positive, negative, or neutral, as illustrated in Figure 1.1. Typically, this probability is based on either hand-generated features, word n-grams, TF-IDF features, or using Deep Learning models to capture sequential long-term and short-term dependencies. Sentiment analysis is also used to classify customer reviews on various online platforms as well as for niche applications, like identifying signs of mental illness in online comments.

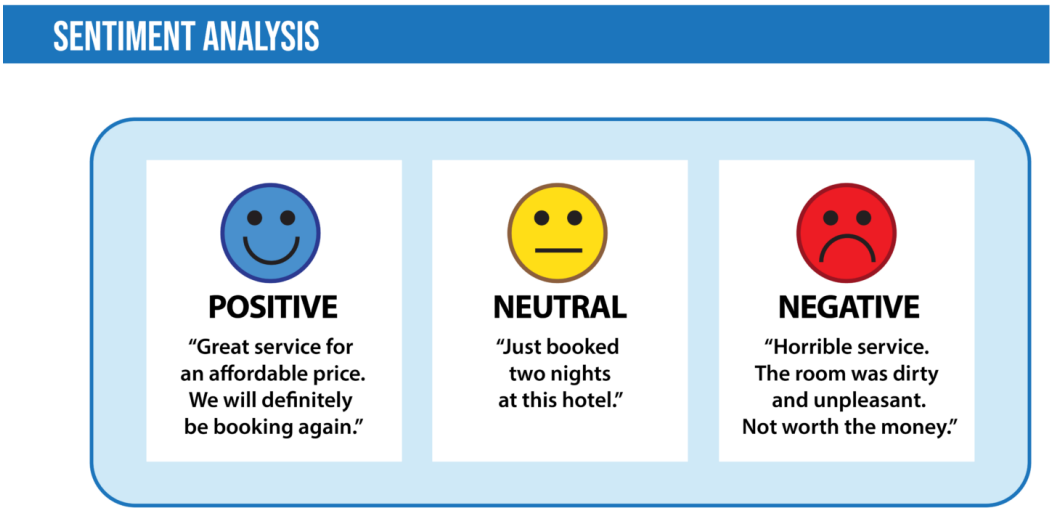


Figure 1.1: Sentiment Analysis: given a text, it classifies its emotional quality

- *Toxicity classification*: This branch of sentiment analysis aims not only to classify hostile intent, but also to categorize specific forms of toxicity, including threats, insults, obscenities, and hatred directed towards particular identities. The input to such a model is text, and the output is typically the probability of each class of toxicity. Toxicity classification models can be used to moderate and improve online conversations by silencing offensive comments, detecting hate speech, or scanning documents for defamation.
- *Machine translation*: It is a process that automates the translation of text between different languages. The input to such a model is text in a specified source language, and

the output is the text in a specified target language. Google Translate is perhaps the most well-known mainstream application of this technology. Such models are used to facilitate communication between people on social media platforms such as Facebook or Skype. Effective approaches to machine translation are able to distinguish between words with similar meanings. Some systems also perform language identification, which is the process of classifying text as being in one language or another.

- *Named entity recognition*: The objective is to extract entities in a given text into predefined categories, including personal names, organisations, locations, and quantities. The input to such a model is typically text, and the output is a list of the named entities along with their respective start and end positions, as illustrated in Figure 1.2. Named entity recognition is a valuable tool in applications such as summarizing news articles and combating disinformation.

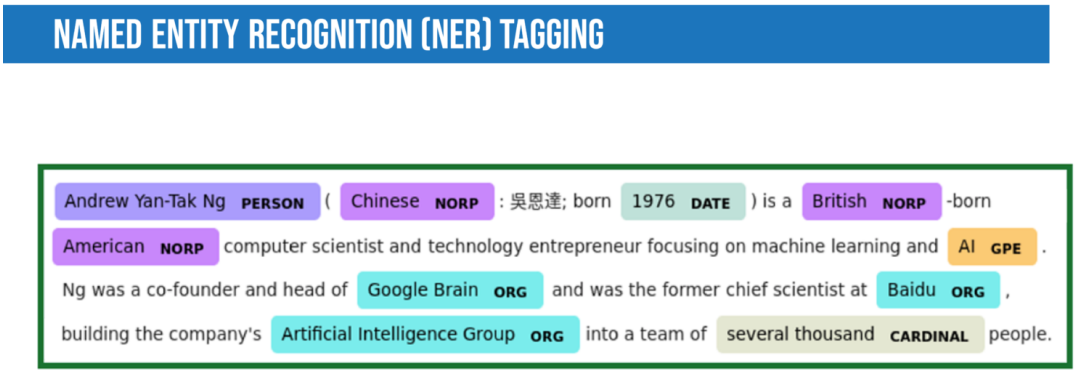


Figure 1.2: Example of named entity recognition tagging

- *Spam detection*: It represents a pervasive binary classification challenge in the field of Natural Language Processing (NLP), wherein the objective is to categorize electronic messages as either legitimate or unsolicited. Spam detectors accept an email text as input, along with supplementary contextual information such as the message's title and the sender's name. Their output is a probability estimate indicating the likelihood of the message being classified as spam. Major email providers, including Gmail, utilize such models to enhance the user experience by detecting and redirecting unsolicited and unwanted emails to a designated spam folder.

- *Text generation*: This process is more formally referred to as Natural Language Generation (NLG). It results in the creation of text that exhibits a high degree of similarity to text written by humans. Such models can be adapted to generate text in a variety of genres and formats, including social media posts, online articles, and even computer code. Text generation has been performed using a variety of approaches, including Markov processes, long short-term memory (LSTM) networks, BERT, GPT-2, LaMDA, and others. This is particularly the case in the context of autocomplete (that is the process of forecasting the next word in a sequence) and chatbots (they automate one side of the conversation, while a human usually provides the other).

- *Information retrieval*: Its objective is to identify the documents that are most pertinent to a given query. This is a challenge that is inherent to every search and recommendation system. The aim is not to provide a definitive answer to a specific query but rather to retrieve, from a vast corpus of documents that may number in the millions, a subset that is most relevant to the query.

NLP models operate by identifying relationships between the constituent elements of language, such as the letters, words, and sentences, which comprise a given text dataset. NLP architectures employ a variety of techniques for data preprocessing, feature extraction, and modelling. Some examples of these processes may be classified as follows:

- *Data preprocessing*: Prior to a model's processing of text for a specific task, the text frequently necessitates preprocessing to enhance model performance or to transform words and characters into a format that the model can comprehend. A variety of techniques may be employed in the preprocessing of data:
  - *Stemming and lemmatization*: Stemming is an informal process of converting words to their base forms using heuristic rules. For example, "university," "universities," and "university's" might all be mapped to the base "univers"; lemmatization is a more formal way to find roots by analyzing a word's morphology using vocabulary from a dictionary.
  - *Sentence segmentation*: It breaks a large piece of text into linguistically meaningful sentence units. This process becomes even more complex in languages, such as ancient Chinese, that do not have a delimiter that marks the end of a sentence.
  - *Stop word removal*: The objective is to eliminate the most frequently occurring words that provide minimal information and do not contribute significantly to the text. To illustrate, the definite article "the", the indefinite article "a", are examples of such words in the English language.
  - *Tokenization*: It splits text into individual words and word fragments. The result is typically a word index and tokenized text, in which words may be represented as numerical tokens for use in various Deep Learning methods. The incorporation of a method that instructs Language Models to ignore unimportant tokens has been demonstrated to enhance efficiency.
- *Feature extraction*: The majority of conventional Machine Learning techniques operate with features, which are typically numerical representations that describe a document in relation to the corpus in which it is embedded. These features are generated through the use of techniques such as Bag-of-Words, TF-IDF, or generic feature engineering, which may include aspects such as document length, word polarity, and metadata (for instance, if the text has associated tags or scores).
- *Modeling*: After data is preprocessed, it is then fed into an NLP architecture which is used to model the data in order to accomplish a variety of tasks.

1.3 Deep Learning and Neural Networks

Deep Learning is a subset of *Machine Learning* that uses multilayered *Neural Networks*, called *Deep Neural Networks*, to simulate the complex decision-making power of the human brain. Some form of Deep Learning powers most of the Artificial Intelligence (AI) applications in our lives today. The principal distinction between Deep Learning and Machine Learning is the structure of the underlying Neural Network architecture. In contrast to Deep Learning,

traditional Machine Learning models utilise relatively simple Neural Networks comprising one or two computational layers. In contrast, Deep Learning models employ three or more layers, with the number of layers typically reaching hundreds or thousands.

A *Neural Network* is a Machine Learning program, or model, that makes decisions in a manner similar to the human brain, by using processes that mimic the way biological neurons work together to identify phenomena, weigh options and arrive at conclusions. It is composed of a series of interconnected nodes, or artificial neurons, arranged in a hierarchical structure. This structure typically comprises an input layer, one or more hidden layers, and an output layer. Each node is connected to others and is associated with a specific weight and threshold. The output of any given node should exceed the pre-established threshold value, so the node is deemed activated and the data is transmitted to the subsequent layer of the network. In the absence of this condition, no data is transferred to the subsequent layer of the network. The efficacy of Neural Networks is contingent upon the availability of training data, which enables the networks to enhance their accuracy over time. Once they have been calibrated to achieve optimal accuracy, they become formidable tools in the fields of Computer Science and Artificial Intelligence, enabling the rapid classification and clustering of data. In comparison to manual identification by human experts, tasks in speech recognition or image recognition can be completed in a matter of minutes, rather than hours. The term "Neural Network", which is sometimes used interchangeably with the terms "Artificial Neural Network" (ANN), constitutes a subset of Machine Learning and are at the core of Deep Learning models.

The terms "Deep Learning" and "Neural Networks" are often used interchangeably in conversation, which can be confusing. It is therefore important to note that the term "deep" in the context of Deep Learning refers to the depth of layers in a Neural Network. A Neural Network that consists of more than three layers, which would include the inputs and the output, can be considered a Deep Learning algorithm. Conversely, a Neural Network that only has two or three layers is simply a basic Neural Network.

As illustrated in Figure 1.3, a Neural Network comprises interconnected artificial neurons organized into three distinct types of layer, which are:

- *Input layer*: Information from the outside world enters the Neural Network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.
- *Hidden layers*: They take their input from the input layer or other hidden layers. Neural Networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.
- *Output layer*: The output layer gives the final result of all the data processing by the Neural Network. It may be constituted by a single node or multiple nodes. To illustrate, in the case of a binary classification problem (i.e., a problem where the output is either yes or no), the output layer will have a single output node, which will provide the result as 1 or 0. Conversely, in the case of a multi-class classification problem, the output layer may consist of more than one output node.

Deep Neural Networks, or Deep Learning Networks, have several hidden layers with millions of artificial neurons linked together. A number, called weight, represents the connections between one node and another. The weight is a positive number if one node excites another, or negative if one node suppresses the other. Nodes with higher weight values have more influence on the other nodes. Theoretically, Deep Neural Networks can map any input type to any output type. However, they also need much more training as compared to other Machine Learning methods. They need millions of examples of training data rather than perhaps the hundreds or thousands that a simpler network might need.

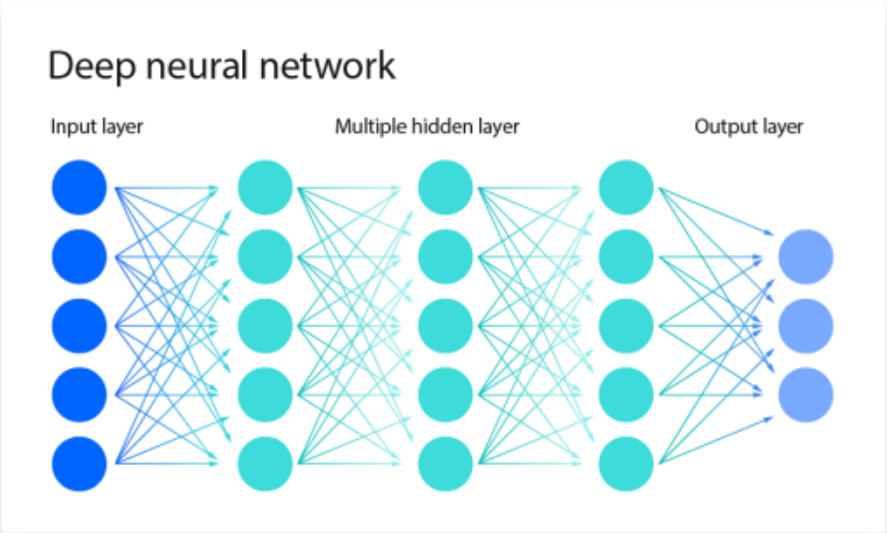


Figure 1.3: Architecture of a Deep Neural Network

Deep Learning algorithms are incredibly complex, and there are different types of Neural Networks to address specific problems or datasets. One potential weakness across them all is that Deep Learning models are often "black boxes," making it difficult to understand their inner workings and posing interpretability challenges. But this can be balanced against the overall benefits of high accuracy and scalability. The following list comprises the principal algorithms:

- *Convolutional Neural Networks* (*CNNs*): They are used primarily in computer vision and image classification applications. They can detect features and patterns within images and videos, enabling tasks such as object detection, image recognition, pattern recognition and face recognition. These networks utilize principles from linear algebra, particularly matrix multiplication, to identify patterns within an image. CNNs are a specific type of Neural Network, which is composed of node layers, containing an input layer, one or more hidden layers and an output layer. Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. CNNs are distinguished from other Neural Networks by their superior performance with image, speech or audio signal inputs. Before CNNs, manual and time-consuming feature extraction methods were used to identify objects in images. However, CNNs now provide a more scalable approach to image classification and object recognition tasks, and process high-dimensional data. They can also exchange data between layers, to deliver more efficient data processing. While information might be lost in the pooling layer, this might be outweighed by the benefits of CNNs, which can help to reduce complexity, improve efficiency and limit risk of overfitting. There are some disadvantages to CNNs, which are computationally demanding time and budget, requiring many Graphical Processing Units (GPUs). They also require highly trained experts with cross-domain knowledge, and careful testing of configurations, hyperparameters and configurations.

- *Recurrent Neural Networks* (*RNNs*): They are typically used in natural language and

speech recognition applications as they use sequential or time-series data. RNNs can be identified by their feedback loops. These learning algorithms are primarily used when using time-series data to make predictions about future outcomes. Use cases include stock market predictions or sales forecasting, or ordinal or temporal problems, such as language translation, Natural Language Processing (NLP), speech recognition and image captioning. These functions are often incorporated into popular applications such as Siri, voice search and Google Translate. RNNs use their "memory" as they take information from prior inputs to influence the current input and output. While traditional Deep Neural Networks assume that inputs and outputs are independent of each other, the output of RNNs depends on the prior elements within the sequence. While future events would also be helpful in determining the output of a given sequence, unidirectional recurrent Neural Networks cannot account for these events in their predictions. RNNs share parameters across each layer of the network and share the same weight parameter within each layer of the network, with the weights adjusted through the processes of backpropagation and gradient descent to facilitate reinforcement learning. An advantage over other neural network types is that RNNs use both binary data processing and memory. However, RNNs tend to run into two basic problems, known as:

  - *Exploding gradients*: It occurs when the gradient is too large, creating an unstable model; in this case, the model weights grow too large, and they will eventually be represented as NaN.
  - *Vanishing gradients*: It occurs when the gradient is vanishing and is too small, it continues to become smaller, updating the weight parameters until they become insignificant. These issues are defined by the size of the gradient, which is the slope of the loss function along the error curve.

  Furthermore, RNNs might also require long training time and be difficult to use on large datasets. Optimizing RNNs add complexity when they have many layers and parameters.

- *Autoencoders and variational autoencoders* (*VAEs*): Autoencoders work by encoding unlabeled data into a compressed representation, and then decoding the data back into its original form. Plain Autoencoders were used for a variety of purposes, including reconstructing corrupted, or blurry, images. Variational autoencoders added the critical ability not only to reconstruct data, but also to output variations of the original data. This ability to generate novel data sparked a rapid succession of new technologies, from Generative Adversarial Networks (GANs) to Diffusion Models, capable of producing increasingly realistic, but fake, images. Autoencoders are built out of blocks of encoders and decoders, an architecture that also underpins the Large Language Models. Encoders compress a dataset into a dense representation, arranging similar data points closer together in an abstract space. Decoders sample from this space to create something new while preserving the dataset's most important features. The main advantage of autoencoders is their ability to handle large volumes of data and to compress input data so that the most important aspects (like anomaly detection and classification tasks) can be highlighted. This also speeds up transmission and reduces storage requirements. Autoencoders can be trained on unlabelled data, so they can be used where labelled data is not available. Unsupervised training saves time, because Deep Learning algorithms learn automatically and gain accuracy without the need for manual feature engineering. In addition, VAE can generate new sample data for text or image generation. There are also disadvantages to Autoencoders. Training deep or

complex structures can be a drain on computing resources. And during unsupervised training, the model may overlook the required features and simply replicate the input data instead. Autoencoders may also overlook complex data links in structured data, so that they do not correctly identify complex relationships.

- *Transformer models*: They combine an encoder-decoder architecture with a text-processing mechanism and have revolutionized how Language Models are trained. An encoder converts raw, unannotated text into representations known as embeddings; the decoder takes these embeddings together with previous outputs of the model, and successively predicts each word in a sentence. Using fill-in-the-blank guessing, the encoder learns how words and sentences relate to each other, building up a powerful representation of language without having to label parts of speech and other grammatical features. Transformers, in fact, can be pretrained at the outset without a particular task in mind. After these powerful representations are learned, the models can later be specialized, with much less data, to perform a requested task. Several innovations make this possible. Transformers process words in a sentence simultaneously, enabling text processing in parallel, speeding up training. Natural Language Processing (NLP) Transformers provide remarkable power since they can run in parallel, processing multiple portions of a sequence simultaneously, which then greatly speeds training. Transformers also track long-term dependencies in text, which enables them to understand the overall context more clearly and create superior output. In addition, Transformers are more scalable and flexible in order to be customized by task. As to limitations, because of their complexity, Transformers require huge computational resources and a long training time. Also, the training data must be accurately on-target, unbiased and plentiful to produce accurate results.

- *Generative Adversarial Networks* (*GANs*): They are Neural Networks that are used both in and outside of Artificial Intelligence (AI) to create new data resembling the original training data. These can include images appearing to be human faces, but they are generated, not taken of real people. The "adversarial" part of the name comes from the back-and-forth between the two portions of the GAN, namely a generator and a discriminator.

  - The *generator* creates something (images, video or audio) and then produces an output with a twist. For example, a horse can be transformed into a zebra with some degree of accuracy. The result depends on the input and how well-trained the layers are in the generative model for this use case.
  - The *discriminator* is the adversary, where the generative result is compared against the real images in the dataset. The discriminator tries to distinguish between the real and fake images, videos or audios.

  GANs train themselves. The generator creates fakes while the discriminator learns to spot the differences between the generator's fakes and the true examples. When the discriminator is able to flag the fake, then the generator is penalized. The feedback loop continues until the generator succeeds in producing output that the discriminator cannot distinguish. The prime GAN benefit is creating realistic output that can be difficult to distinguish from the originals, which in turn may be used to further train Machine Learning models. Setting up a GAN to learn is straightforward, since GANs are trained by using unlabeled data or with minor labeling. However, the potential disadvantage is that the generator and discriminator might go back-and-forth in competition for a long time, creating a large system drain. One training limitation is that a huge amount of input data might be required to obtain a satisfactory output. Another potential problem

is "mode collapse", that occurs when the generator produces a limited set of outputs rather than a wider variety.

The number of applications for Deep Learning continues to expand on a daily basis. This section will conclude with an overview of the ways in which this technology is now assisting businesses in becoming more efficient and in providing a superior level of service to their customers:

- *Application modernization*: Generative AI can enhance the capabilities of developers and reduce the ever-widening skills gap in the domains of application modernization and IT automation. Generative AI for coding is possible because of recent breakthroughs in Large Language Model (LLM) technologies and Natural Language Processing (NLP). It uses Deep Learning algorithms and Large Neural Networks trained on vast datasets of existing source code.

- *Computer vision*: It is a field of Artificial Intelligence (AI) that includes image classification, object detection and semantic segmentation. It uses Machine Learning and Neural Networks to teach computers and learning systems to derive meaningful information from digital images, videos and other visual inputs. If AI enables computers to think, computer vision enables them to see, observe and understand. Computer vision enables systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, to take action. This ability to provide recommendations distinguishes it from simple image recognition tasks. Some common applications of computer vision today can be seen in many business sectors like Automotive, Healthcare, Marketing, Retail, etc.

- *Customer care*: AI is helping businesses to better understand and cater to increasing consumer demands. With the rise of highly personalized online shopping, direct-to-consumer models, and delivery services, generative AI can help further unlock a host of benefits that can improve customer care, talent transformation and the performance of applications. AI empowers businesses to adopt a customer-centric approach by harnessing valuable insights from customer feedback and buying habits. This data-driven approach can help improve product design and packaging and can help drive high customer satisfaction and increased sales. Generative AI can also serve as a cognitive assistant for customer care, providing contextual guidance based on conversation history, sentiment analysis and call center transcripts. In addition, generative AI can enable personalized shopping experiences, foster customer loyalty and provide a competitive advantage.

- *Digital labor*: Organizations can augment their workforce by building and deploying Robotic Process Automation (RPA) and digital labor to collaborate with humans to increase productivity, or assist whenever backup is needed. Digital labor uses foundation models to automate and improve the productivity of knowledge workers by enabling self-service automation in a fast and reliable way—without technical barriers. To automate task performance, or calling APIs, an enterprise-grade LLM-based slot filling model can identify information in a conversation and gather all the information required for completing an action or calling an API without much manual effort. In lieu of having technical experts record and encode repetitive action flows for knowledge workers, digital labor automations constructed with a foundation of model-powered conversational instructions and demonstrations can be utilized by the knowledge worker for self-service automation. To illustrate, to expedite the creation of applications, no-code digital apprentices can assist end-users, who lack expertise in programming, by effectively teaching, supervising, and validating code.

- *Generative AI*: It is a category of AI that autonomously creates text, images, video, data or other content in response to a user's prompt or request. Generative AI relies on Deep Learning models that can learn from patterns in existing content and generate new, similar content based on that training. It has applications in many fields, including customer service, marketing, software development and research, and offers enormous potential to streamline enterprise workflows through fast, automated content creation and augmentation.
- *Natural Language Processing and Speech Recognition*: NLP combines computational linguistics rule-based modeling, of human language, with statistical and Machine Learning models to enable computers and digital devices to recognize, understand and generate text and speech. NLP powers applications and devices that can translate text from one language to another, respond to typed or spoken commands, recognize or authenticate users based on voice. It helps summarize large volumes of text, assess the intent or sentiment of text or speech and generate text or graphics or other content on demand.

1.4 Transformers

The introduction of new Deep Learning models is occurring with increasing frequency, making it challenging to remain abreast of all the recent developments. However, one particular Neural Network model has demonstrated remarkable efficacy in addressing common Natural Language Processing tasks. This models, known as a *Transformers*, is a type of Neural Network architecture that transforms, or changes, an input sequence into an output one. It does this by learning context and tracking relationships between sequence components. To illustrate, one might consider the following input sequence: "What is the colour of the sky?". The Transformer model employs an internal mathematical representation that identifies the relevance and relationship between the words "color," "sky," and "blue." The aforementioned knowledge is then employed by the model to generate the output. The sentence "The sky is blue" is an example of a correct output generated by the model. Organizations use Transformer models for all types of sequence conversions, from speech recognition to machine translation and protein sequence analysis.

1.4.1 Sequence to Sequence Learning and Attention

Sequence-to-Sequence (or *Seq2Seq*) is a Neural Network that transforms a given sequence of elements, such as the sequence of words in a sentence, into another sequence. Seq2Seq models are particularly good at translation, where the sequence of words from one language is transformed into a sequence of different words in another language. A popular choice for this type of model is *Long-Short-Term-Memory* (*LSTM*) models. With sequence-dependent data, the LSTM modules can give meaning to the sequence while remembering (or forgetting) the parts it finds important (or unimportant). Sentences, for example, are sequence-dependent since the order of the words is crucial for understanding the sentence. LSTM are a natural choice for this type of data.

This type of models consist of an encoder and a decoder. The encoder takes the input sequence and maps it into a higher dimensional space (n-dimensional vector). That abstract vector is fed into the decoder which turns it into an output sequence. The output sequence can be in another language, symbols, a copy of the input, etc.

The primary mechanism enabling Transformers to comprehend is *attention*. This mechanism looks at an input sequence and decides at each step which other parts of the sequence are important.

So, for each input that the LSTM (encoder) reads, the attention-mechanism takes into account several other inputs at the same time and decides which ones are important by attributing different weights to those inputs. The decoder will then take as input the encoded sentence and the weights provided by the attention-mechanism.

1.4.2 Architecture

Transformer architecture has several software layers that work together to generate the final output. Figure 1.4 shows the components of transformation architecture.

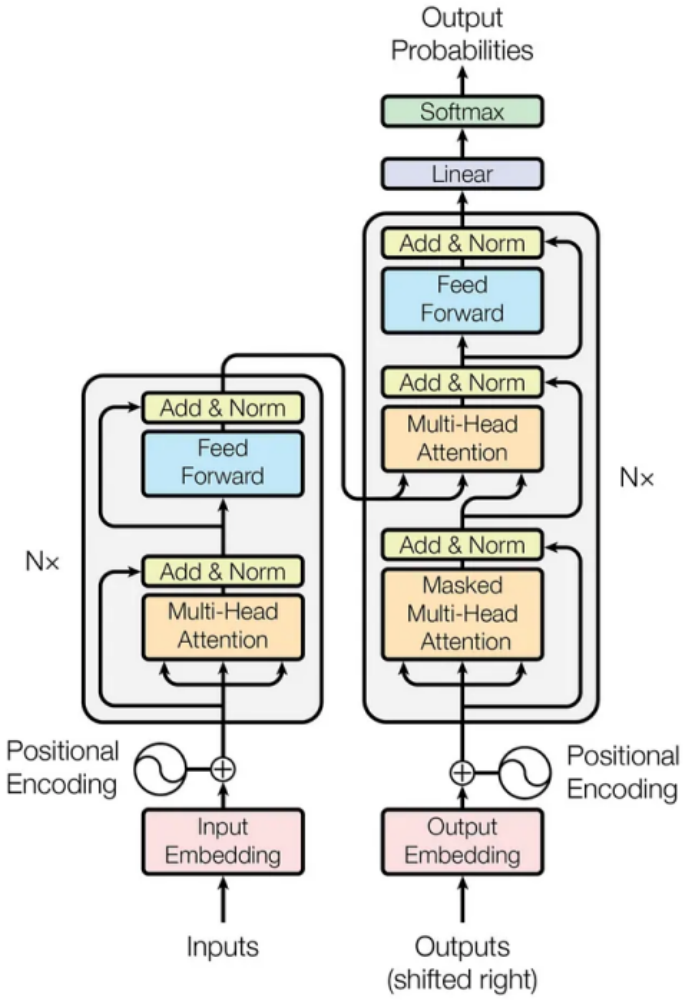


Figure 1.4: Architecture of a Transformer

The stage of *input embeddings* converts the input sequence into the mathematical domain that software algorithms understand. At first, the input sequence is broken down into a series of tokens or individual sequence components. For instance, if the input is a sentence, the tokens are words. Embedding then transforms the token sequence into a mathematical vector sequence. The vectors carry semantic and syntax information, represented as numbers, and their attributes are learned during the training process. Software can use the numbers to calculate the relationships between words in mathematical terms and understand the human language model. Embeddings provide a way to represent discrete tokens as continuous vectors that the model can process and learn from.

Positional encoding is a crucial component in the Transformer architecture because the

model itself does not inherently process sequential data in order. The Transformer needs a way to consider the order of the tokens in the input sequence. Positional encoding adds information to each token's embedding to indicate its position in the sequence. This is often done by using a set of functions that generate a unique positional signal that is added to the embedding of each token. With positional encoding, the model can preserve the order of the tokens and understand the sequence context.

A typical Transformer model has multiple *Transformer blocks* stacked together. Each Transformer block has two main components: a *multi-head self-attention mechanism* and a *position-wise feed-forward Neural Network*. The self-attention mechanism enables the model to weigh the importance of different tokens within the sequence. It focuses on relevant parts of the input when making predictions. The feed-forward layer has additional components that help the transformer model train and function more efficiently. Each transformer block includes:

- Connections around the two main components that act like shortcuts. They enable the flow of information from one part of the network to another, skipping certain operations in between.
- Layer normalization that keeps the numbers, specifically the outputs of different layers in the network, inside a certain range so that the model trains smoothly.
- Linear transformation functions so that the model adjusts values to better perform the task it is being trained on, like document summary as opposed to translation.

Ultimately, the model needs to make a concrete prediction, such as choosing the next word in a sequence. This is where the *linear block* comes in. It is another fully connected layer, also known as a dense layer, before the final stage. It performs a learned linear mapping from the vector space to the original input domain. This crucial layer is where the decision-making part of the model takes the complex internal representations and turns them back into specific predictions that can be interpreted and used. The output of this layer is a set of scores (often called logits) for each possible token.

The *softmax function* is the final stage that takes the logit scores and normalizes them into a probability distribution. Each element of the softmax output represents the model's confidence in a particular class or token.

1.4.3 Types of models

Transformers have evolved into a diverse family of architectures. The following are some types of Transformer models:

- *Bidirectional Transformers*: *Bidirectional Encoder Representations from Transformers* (*BERT*) models modify the base architecture to process words in relation to all the other words in a sentence rather than in isolation. Technically, they employ a mechanism called the *Bidirectional Masked Language Model* (*MLM*). During pretraining, BERT randomly masks some percentage of the input tokens and predicts these masked tokens based on their context. The bidirectional aspect comes from the fact that BERT takes into account both the left-to-right and right-to-left token sequences in both layers for greater comprehension.
- *Generative Pretrained Transformers* (*GPT*): These models use stacked Transformer decoders that are pretrained on a large corpus of text by using language modeling objectives. They are autoregressive, which means that they regress or predict the next value in a sequence based on all preceding values. By using more than 175 billion parameters, GPT models can generate text sequences that are adjusted for style and tone. They

have sparked the research in AI toward achieving Artificial General Intelligence. This means that organizations can reach new levels of productivity while reinventing their applications and customer experiences.

- *Bidirectional and AutoRegressive Transformers* (*BART*): It is a type of Transformer model that combines bidirectional and autoregressive properties. It is like a blend of BERT's bidirectional encoder and GPT's autoregressive decoder. It reads the entire input sequence at once and is bidirectional like BERT. However, it generates the output sequence one token at a time, conditioned on the previously generated tokens and the input provided by the encoder.

- *Vision Transformers* (*ViT*): They repurpose the Transformer architecture for image classification tasks. Instead of processing an image as a grid of pixels, they view image data as a sequence of fixed-size patches, similar to how words are treated in a sentence. Each patch is flattened, linearly embedded, and then processed sequentially by the standard Transformer encoder. Positional embeddings are added to maintain spatial information. This usage of global self-attention enables the model to capture relationships between any pair of patches, regardless of their position.

1.4.4 Use cases

Transformers have a multitude of applications; the following list provides an overview of the characteristics of the main ones:

- *Natural Language Processing*: Transformers enable machines to understand, interpret, and generate human language in a way that is more accurate than ever before. They can summarize large documents and generate coherent and contextually relevant text for all kinds of use cases. Virtual assistants, like Alexa, use Transformer technology to understand and respond to voice commands.

- *Machine translation*: Translation applications use Transformers to provide real-time, accurate translations between languages. Transformers have significantly improved the fluency and accuracy of translations as compared to previous technologies.

- *DNA sequence analysis*: By treating segments of DNA as a sequence similar to language, Transformers can predict the effects of genetic mutations, understand genetic patterns, and help identify regions of DNA that are responsible for certain diseases. This capability is crucial for personalized medicine, where understanding an individual's genetic makeup can lead to more effective treatments.

- *Protein structure analysis*: Transformer models can process sequential data, which makes them well suited for modeling the long chains of amino acids that fold into complex protein structures. Understanding protein structures is vital for drug discovery and understanding biological processes. Additionally, Transformers can be employed in applications that utilize amino acid sequences to predict the three-dimensional structure of proteins.

1.5 Large Language Models

As previously stated, language is a prominent ability in human beings to express and communicate, which develops in early childhood and evolves over a lifetime. Machines, however, cannot naturally grasp the abilities of understanding and communicating in the

form of human language, unless equipped with powerful Artificial Intelligence algorithms. It has been a longstanding research challenge to achieve this goal, to enable machines to read, write, and communicate like humans. Technically, Language Modeling (LM) is one of the major approaches to advancing language intelligence of machines. In general, LM aims to model the generative likelihood of word sequences, so as to predict the probabilities of future (or missing) tokens. The research of LM has received extensive attention in the literature, which can be divided into four major development stages (Zhao *et al.* [2023]), as shown in Figure 1.5:

- *Statistical Language Models* (*SLM*). SLMs are developed based on statistical learning methods that rose in the 1990s. The fundamental premise is to construct a word prediction model that is based on the Markov assumption. This can be exemplified by predicting the next word based on the most recent context. The SLMs with a fixed context length "n" are also called n-gram language models. SLMs have been widely applied to enhance task performance in Information Retrieval (IR) and Natural Language Processing (NLP). However, they often suffer from the curse of dimensionality: it is difficult to accurately estimate high-order language models since an exponential number of transition probabilities need to be estimated. Thus, specially designed smoothing strategies, such as *backoff estimation* and *Good–Turing estimation*, have been introduced to alleviate the data sparsity problem.

- *Neural Language Models* (*NLM*). NLMs characterize the probability of word sequences by Neural Networks, like Multi-Layer Perceptron (MLP) and Recurrent Neural Networks (RNNs). Many works introduced the concept of distributed representation of words and built the word prediction function conditioned on the aggregated context features. By extending the idea of learning effective features for text data, a general Neural Network approach was developed to build a unified, end-to-end solution for various NLP tasks. Furthermore, *word2vec* was proposed to build a simplified shallow Neural Network for learning distributed word representations, which were demonstrated to be very effective across a variety of NLP tasks. These studies have initiated the use of language models for representation learning (beyond word sequence modeling), having an important impact on the field of NLP.

- *Pre-trained Language Models* (*PLM*). As an early attempt, "ELMo" was proposed to capture context-aware word representations by first pre-training a bidirectional LSTM (biLSTM) network (instead of learning fixed word representations) and then fine-tuning the biLSTM network according to specific downstream tasks. Furthermore, based on the highly parallelizable Transformer architecture with self-attention mechanisms, BERT was proposed by pre-training bidirectional language models with specially designed pre-training tasks on large-scale unlabeled corpora. These pre-trained context-aware word representations are very effective as general-purpose semantic features, which have largely raised the performance bar of NLP tasks. In this case, it often requires fine-tuning the PLM for adapting to different downstream tasks.

- *Large Language Models* (*LLM*). Researchers find that scaling PLM (scaling model size or data size) often leads to an improved model capacity on downstream tasks. A number of studies have explored the performance limit by training an ever larger PLM. Although scaling is mainly conducted in model size (with similar architectures and pre-training tasks), these large-sized PLMs display different behaviors from smaller PLMs and show surprising abilities (called "emergent abilities") in solving a series of complex tasks. Thus, the research community coins the term “Large Language Models” for these large-sized PLMs, which attract increasing research attention. A remarkable application

of LLMs is ChatGPT2 that adapts the LLMs from the GPT series for dialogue, which presents an amazing conversation ability with humans.

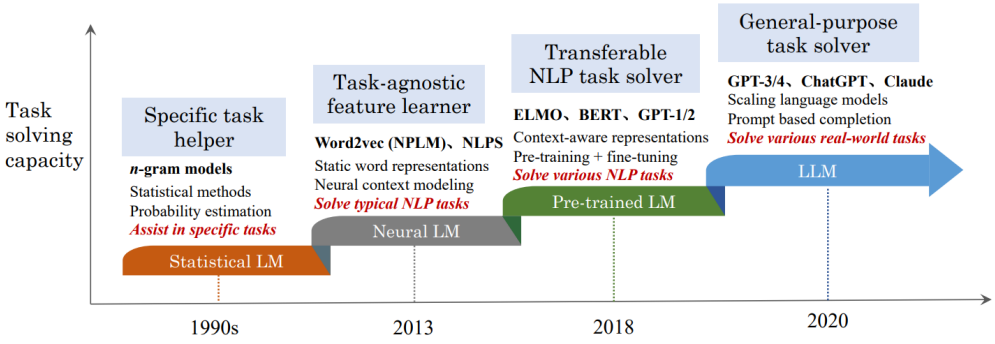


Figure 1.5: Evolution process of the four generations of Language Models (LM)

1.5.1 Definition

A *Large Language Model* (*LLM*) is a Deep Learning algorithm that is capable of performing a multitude of Natural Language Processing (NLP) tasks. LLMs employ Transformer models and are trained using extensive datasets, which is why they are so large. Such capabilities include the recognition, translation, prediction, and generation of text and other content. They are also known as a type of *Generative AI* algorithms that has been specifically architected to help generate text-based content.

In addition to teaching human languages to Artificial Intelligence (AI) applications, Large Language Models can also be trained to perform a variety of tasks, including the understanding of protein structures, the writing of software code, and more. Similar to the human brain, they must be pre-trained and subsequently fine-tuned in order to enable them to solve problems pertaining to text classification, question answering, document summarization, and text generation. The problem-solving capabilities of LLMs can be applied to fields such as healthcare, finance, and entertainment, where they serve a variety of Natural Language Processing applications, including translation, chatbots, and Artificial Intelligence assistants.

The first Artificial Intelligence Language Models trace their roots to the earliest days of AI. The *Eliza* Language Model debuted in 1966 at MIT and is one of the earliest examples of an AI language model. All language models are first trained on a set of data, then make use of various techniques to infer relationships before ultimately generating new content based on the trained data. Language models are commonly used in NLP applications where a user inputs a query in natural language to generate a result. In fact, an LLM is the evolution of the language model concept in AI that dramatically expands the data used for training and inference. In turn, it provides a massive increase in the capabilities of the AI model. While there is not a universally accepted figure for how large the data set for training needs to be, an LLM typically has at least one billion or more parameters. Parameters are a machine learning term for the variables present in the model on which it was trained that can be used to infer new content.

Modern LLMs emerged in 2017 and use Transformer models. So, with a large number of parameters and the Transformer model, LLMs are able to understand and generate accurate responses rapidly, which makes the AI technology broadly applicable across many different domains. Some LLMs are referred to as "foundation models", a term coined by the "Stanford

Institute for Human-Centered Artificial Intelligence" in 2021. A foundation model is so large and impactful that it serves as the foundation for further optimizations and specific use cases.

The following list provides examples of LLMs:

- *GPT-3*, *GPT-3.5*, *GPT-4*;
- *OPT*
- *Bert*;
- *Claude*;
- *Cohere*;
- *Ernie*;
- *Falcon 40B*;
- *Galactica*;
- *Lamda*.

1.5.2 How they work

A Large Language Model is based on a Transformer model and works by receiving an input, encoding it, and then decoding it to produce an output prediction. But, before an LLM can receive text input and generate an output prediction, it requires training, so that it can fulfill general functions, and fine-tuning, which enables it to perform specific tasks.

Large Language Models are pre-trained using large textual datasets from sites like Wikipedia, GitHub, or others. These datasets consist of trillions of words, and their quality will affect the language model's performance. At this stage, the LLM engages in unsupervised learning, meaning it processes the datasets fed to it without specific instructions. During the process of *training*, the LLM's AI algorithm can learn the meaning of words, and of the relationships between words. It also learns to distinguish words based on context. So, LLM learns to predict the next word in a sentence based on the context provided by the preceding words. The model performs this function by attributing a probability score to the recurrence of words that have been tokenized, which entails the breakdown of words into smaller sequences of characters. These tokens are then transformed into embeddings, which are numeric representations of this context.

The process of training LLMs involves several steps:

- *Data Collection*: It gathers a diverse dataset of text from various sources;
- *Preprocessing*: It cleans and standardizes the collected text data;
- *Tokenization*: It divides the preprocessed text into smaller units, called tokens;
- *Model selection*: It chooses an appropriate Large Language Model;
- *Training*: It trains process to get the model to learn the data;
- *Improving results*: It optimizes the model by making adjustments and fine-tuning;
- *Evaluation*: It evaluates the results and accuracy of the model;
- *Deployment*: It deploys the model to a live system for use.

Once trained on training data, LLMs can generate text by autonomously predicting the next word based on the input they receive, and drawing on the patterns and knowledge they have acquired. The result is coherent and contextually relevant language generation that can be harnessed for a wide range of NLU and content generation tasks.

Model performance can also be increased through *prompt engineering*, *prompt-tuning*, *fine-tuning* and other tactics, like *Reinforcement Learning with Human Feedback* (*RLHF*), to remove the biases, hateful speech and factually incorrect answers known as "hallucinations" that are often unwanted byproducts of training on so much unstructured data. This is one of the most important aspects of ensuring enterprise-grade LLMs are ready for use and do not expose organizations to unwanted liability, or cause damage to their reputation.

1.5.3 Use cases

LLMs are redefining an increasing number of business processes and have proven their versatility across a myriad of use cases and tasks in various industries. They augment conversational AI in chatbots and virtual assistants to enhance the interactions that underpin excellence in customer care, providing context-aware responses that mimic interactions with human agents. They also excel in content generation, automating content creation for blog articles, marketing or sales materials and other writing tasks. In research and academia, they aid in summarizing and extracting information from vast datasets, accelerating knowledge discovery. LLMs also play a vital role in language translation, breaking down language barriers by providing accurate and contextually relevant translations. They can even be used to write code, or "translate" between programming languages.

Moreover, they contribute to accessibility by assisting individuals with disabilities, including text-to-speech applications and generating content in accessible formats. From healthcare to finance, LLMs are transforming industries by streamlining processes, improving customer experiences and enabling more efficient and data-driven decision making. Most excitingly, all of these capabilities are easy to access, in some cases literally an API integration away. Here is a list of some of the most important areas where LLMs benefit organizations:

- *Text generation*: It comprises language generation abilities, such as writing emails, blog posts or other mid-to-long form content in response to prompts that can be refined and polished. An excellent example is Retrieval-Augmented Generation (RAG).
- *Content summarization*: It summarizes long articles, news stories, research reports, corporate documentation and even customer history into thorough texts tailored in length to the output format.
- *AI assistants*: It comprises chatbots that answer customer queries, perform backend tasks and provide detailed information in natural language as a part of an integrated, self-serve customer care solution.
- *Code generation*: It assists developers in building applications, finding errors in code and uncovering security issues in multiple programming languages, even "translating" between them.
- *Sentiment analysis*: It analyzes text to determine the customer's tone in order to understand customer feedback at scale and aid in brand reputation management.
- *Language translation*: It provides wider coverage to organizations across languages and geographies with fluent translations and multilingual capabilities.
- *Classification and categorization*: An LLM is able to classify and categorize content.

- *Conversational AI and chatbots*: LLMs can enable a conversation with a user in a way that is typically more natural than older generations of AI technologies.

LLMs stand to impact every industry, from finance to insurance, human resources to healthcare and beyond, by automating customer self-service, accelerating response times on an increasing number of tasks, as well as providing greater accuracy, enhanced routing and intelligent context gathering.

1.5.4 Advantages and limitations

With a broad range of applications, Large Language Models are exceptionally beneficial for problem-solving since they provide information in a clear, conversational style that is easy for users to understand. In summary, the primary advantages are as follows:

- *Extensibility and adaptability*: LLMs can serve as a foundation for customized use cases. Additional training on top of an LLM can create a fine-tuned model for an organization's specific needs.
- *Flexibility*: An LLM can be used for many different tasks and deployments across organizations, users and applications.
- *Performance*: Modern LLMs are typically high-performing, with the ability to generate rapid, low-latency responses.
- *Accuracy*: As the number of parameters and the volume of trained data grow in an LLM, the Transformer model is able to deliver increasing levels of accuracy.
- *Efficiency*: LLMs can save employees time by automating routine tasks.
- *Advanced NLP capabilities*: Natural Language Processing has revolutionized the ability of AI machines to comprehend and interpret human texts and spoken words. With the emergence of Large Language Models, AI-based tools have become more efficient in understanding human language. This has drastically improved their ability to converse and generate insights from data.
- *Improved generative capabilities*: The generative capability of LLMs has augmented the conversational ability of AI machines. The models analyze vast amounts of data to generate valuable insights, enabling them to interact effectively with humans and provide accurate feedback for given prompts. This potential of LLMs has attracted the attention of industry leaders.
- *Increased efficiency*: The capacity to understand human language has made LLMs ideal for handling tedious or time-consuming tasks. For example, finance professionals can automate financial transactions and data processing with the help of LLMs, thus reducing manual effort. This is one of the reasons why companies are now using LLMs for enterprise-level operations.

While there are numerous advantages to the utilization of LLMs, there are also a number of challenges and limitations. Indeed, they may appear to demonstrate an understanding of meaning and the capacity to respond to it accurately. However, they remain a technological instrument, and thus present a number of risks and concerns, including:

- *Bias Amplification*: LLMs can inadvertently amplify existing biases in the training data, potentially leading to unfair or discriminatory outputs.

- *Misinformation and manipulation*: LLMs are vulnerable to spreading misinformation, generating fake news, or manipulating public opinion.
- *Ethical implications*: LLM-generated content raises ethical concerns, including plagiarism, copyright infringement, and the creation of misleading or harmful information.
- *Energy Consumption*: Training and running LLMs consume significant computational resources, leading to high energy consumption and environmental impact.
- *Lack of accountability*: LLMs' decision-making process and the sources of generated content can be opaque, raising concerns about transparency and accountability.
- *Hallucinations*: They refer to the generation of content that is irrelevant, made-up, or inconsistent with the input data. This problem leads to incorrect information, challenging the trust placed in these models. Hallucinations are a critical obstacle in the development of LLMs, often arising from the training data's quality and the models' interpretative limits.
- *Scaling*: It can be difficult and resource-consuming to scale and maintain LLMs.
- *Deployment*: Deploying LLMs requires Deep Learning, a Transformer model, distributed software and hardware, and overall technical expertise.

CHAPTER 2

Introduction to Knowledge Graph

This chapter will offer a detailed exploration of three interrelated concepts that are central to modern computational and data-driven disciplines, namely graph theory, Graph Neural Networks (GNNs), and Knowledge Graphs (KGs). Each section of the chapter will be built upon the previous one, passing from fundamental mathematical principles to advanced applications in Artificial Intelligence and Data Science. The chapter will begin with an introduction to the fundamental principles of graph theory. This section will cover the basics of graph theory, explaining key concepts, various types of graphs, and significant properties of graphs. By tracing the historical development of graph theory, including Euler's solution to the Königsberg bridge problem, this section will establish a solid grounding in the mathematical structures that will form the backbone of more advanced topics discussed later in the chapter. Subsequently, the chapter will introduce the reader to the concept of Graph Neural Networks, which will represent an advanced computational framework applying Deep Learning techniques to graph-structured data. This section will commence with an explanation of the limitations of traditional Neural Networks when compared with non-Euclidean data, such as graphs. It will then proceed to delineate how GNNs will surmount these challenges by effectively capturing the relationships and dependencies between graph nodes. The section will examine various types of GNNs, including convolutional and recurrent variants, and will consider their applications in diverse fields. Through case studies and examples, the section will demonstrate how GNNs will capitalize on the insights of graph theory to facilitate sophisticated data analysis and pattern recognition in intricate and interrelated systems. Finally, the chapter will turn its attention to the construction and utilization of Knowledge Graphs, which will represent a powerful tool for the structured representation of knowledge in graph format. This section will define Knowledge Graphs and highlight their significance in organizing and querying large and complex datasets by representing entities and their interrelationships. It will address the fundamental elements of Knowledge Graphs and will examine the role of standards, such as RDF (Resource Description Framework), in their development. Furthermore, it will delve into the integration of Graph Neural Networks with Knowledge Graphs, emphasizing how GNNs will increase the performance and capabilities of Knowledge Graphs in diverse applications.

2.1 Graph theory

Graph theory is a branch of mathematics concerned with networks of points connected by lines. The subject of graph theory had its beginnings in recreational math problems, but it has grown into a significant area of mathematical research, with applications in chemistry, operations research, social sciences and computer science.

The history of graph theory may be specifically traced to 1735, when the Swiss mathematician Leonhard Euler solved the Königsberg bridge problem. This was an old puzzle, shown in Figure 2.1, concerning the possibility of finding a path over every one of seven bridges that span a forked river flowing past an island—but without crossing any bridge twice. Euler argued that no such path exists. His proof involved only references to the physical arrangement of the bridges, but essentially he proved the first theorem in graph theory.

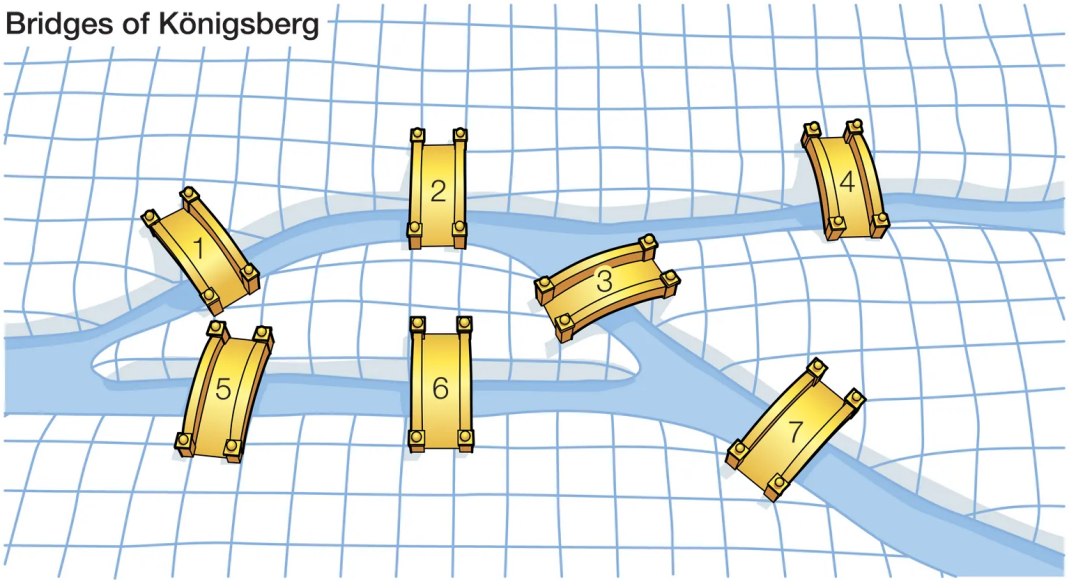


Figure 2.1: Königsberg bridge problem

As used in graph theory, the term *graph* does not refer to data charts, such as line graphs or bar graphs. Instead, it refers to a set of vertices, called *nodes*, and of lines, called *edges*, that connect the vertices. When any two vertices are joined by more than one edge, the graph is called *multigraph*. A graph without loops and with at most one edge between any two vertices is called *simple graph*. When each vertex is connected by an edge to every other vertex, the graph is called *complete graph*. When appropriate, a direction may be assigned to each edge to produce what is known as a *directed graph*, or *digraph*. Figure 2.2 shows these basic categories of graphs.

An important number associated with each vertex is its *degree*, which is defined as the number of edges that enter, or exit, from it. Knowing the number of vertices in a complete graph characterizes its essential nature. For this reason, complete graphs are commonly designated as K_n, where n refers to the number of vertices, and all vertices of K_n have degree n-1. In the terminology of modern graph theory, Euler's theorem regarding the Königsberg bridge problem can be restated as follows:

> In a multigraph, if there is a path along the edges that traverses each edge once and only once, then there are at most two vertices of odd degree. Furthermore, if the path begins and ends at the same vertex, then no vertices will have odd degree.

Another important concept in graph theory is the *path*, which is any route along the edges of a graph. A path may follow a single edge directly between two vertices, or it may follow multiple edges through multiple vertices. If there is a path linking any two vertices in a graph, that graph is said to be *connected*. A path that begins and ends at the same vertex without traversing any edge more than once is called a *circuit*, or a *closed path*. A circuit that follows each edge exactly once while visiting every vertex is known as an *Eulerian circuit*, and the

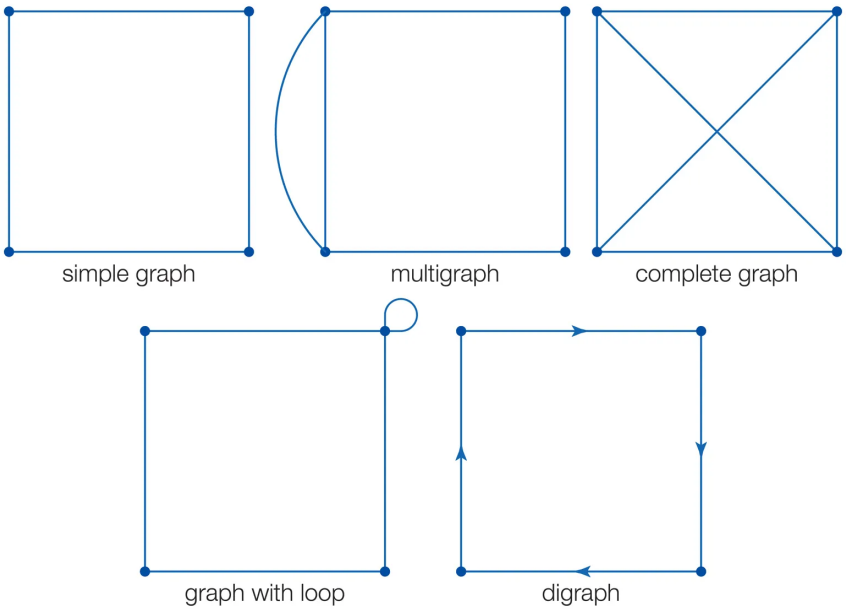


Figure 2.2: Basic categories of graphs

graph is called an *Eulerian graph*. An Eulerian graph is connected and, in addition, all its vertices have even degree.

2.1.1 Graph theory in Data Science

Graph theory offers a mathematical framework for modeling connections and relationships in data. It has become an indispensable tool in the field of Data Science, particularly for analyzing networked structures, such as social graphs, transportation networks, biological pathways, and more.

A graph is comprised of nodes (or vertices) that are connected by edges (or links). Nodes represent entities, which may be people, places, or things, while edges show relationships between those entities. The strength of a connection can be indicated by assigning weights to edges. The key components of a graph are as follows:

- *Nodes*: They are the entities that are represented in a graph.
- *Edges*: They are the connections between the nodes in a graph.
- *Weights*: They are values that quantify the strength of the links between the nodes.
- *Paths*: They are sequence of edges between nodes.
- *Cycles*: They are paths that start and end at the same node.

So, graph theory provides a valuable mathematical framework for the analysis of relationships and connections between data in networks. The following list outlines some of the main applications of graph theory in Data Science:

- *Social Network Analysis*: Its objective is to identify patterns and relationships within a social network. The mapping of social connections between individuals enables the identification of key influencers, communities, and information flows. The field of graph theory offers a set of algorithms that can be used to quantify the properties of a given network.

- *Recommendation systems*: Services such as Netflix employ graph-theoretic techniques to map users, content, and interactions. This serves as the foundation for collaborative filtering models, which are used to generate recommendations.
- *Logistics and Transport*: Airlines utilize graph-based models to represent their routes and schedules, enabling the optimization of traffic flow, the identification of efficient paths, and the prevention of cascading delays through the application of sophisticated algorithms.
- *Fraud detection*: Graphs capturing financial transactions and account relationships enable the identification of suspicious patterns, such as money laundering ones. Community detection allows for the detection of coordinated fraud.
- *Biology and Chemistry*: Modeling molecular interactions, disease networks, and protein structures as graphs facilitates the acquisition of biological insights through topological analysis.
- *Infrastructure network mapping*: It is the representation of connections between servers, users, packets, and other relevant entities, as graphs. This process allows for the monitoring of performance, usage patterns and potential vulnerabilities in IT systems.

The primary objective of these applications is to transform real-world data into graphs in order to leverage the mathematical rigor and algorithms inherent to graph theory. This process enables the identification of clusters, central nodes, optimal paths, and other key elements, facilitating the advancement of predictive analysis, intricate pattern recognition, and decision-making optimization within the domain of Data Science.

Finally, graph theory is a fundamental tool in the design and analysis of algorithms. The following are some key ways in which graph theory concepts are applied in algorithms:

- *Modeling problems as graphs*: A considerable number of real-world problems can be modeled as graphs. To illustrate, a road network can be represented as a graph comprising intersections as nodes and roads as edges. This graph representation enables the application of graph algorithms to solve problems on these networks.
- *Pathfinding algorithms*: Algorithms such as breadth-first search and Dijkstra's algorithm employ graph traversal to identify the shortest paths between nodes. These algorithms are employed in GPS navigation, network routing, and other pathfinding problems.
- *Clustering algorithms*: Algorithms such as minimum cut and Karger's algorithm employ graph partitioning to divide networks into clusters. These facilitate the detection of communities and the clustering of data.
- *Scheduling algorithms*: The mapping of scheduling problems to bipartite matching allows for the development of optimal task assignment and resource allocation algorithms.
- *Network analysis*: It is supported by a number of measures derived from graph theory, including centrality measures, connectivity metrics, and other graph statistics. These facilitate the analysis of networks and the prediction of links between them.

In conclusion, graph theory offers the mathematical models and properties that facilitate the development of efficient and optimal algorithms for a multitude of real-world problems. It plays a crucial role in the analysis of complex networks and the development of algorithms that provide effective solutions.

2.2 Graph Neural Networks

Graph Neural Networks (*GNNs*) are a class of Deep Learning methods designed to perform inference on data described by graphs (which have been described in Section 2.1). They are Neural Networks that can be directly applied to graphs, and provide an easy way to do graph-level, node-level, and edge-level prediction tasks. Particularly, in a graph-level task, the objective is to predict a single property for the entire graph. In a node-level task, the goal is to predict a specific property for each node within the graph. Finally, in an edge-level task, the aim is to predict the property or presence of edges within the graph. The following list will provide a detailed examination of the three classes of graph prediction problems, accompanied by illustrative examples (Sanchez-Lengeling *et al.* [2021]):

- *Graph-level task*: The objective is to predict the property of an entire graph. To illustrate, in the case of a molecule represented as a graph, one might seek to predict the molecule's olfactory properties or its capacity to bind with a receptor implicated in a disease (Figure 2.3). This bears resemblance to image classification problems with MNIST and CIFAR, where the objective is to associate a label with an entire image. In the context of text, a comparable problem is sentiment analysis, which involves identifying the mood or emotion of an entire sentence simultaneously.

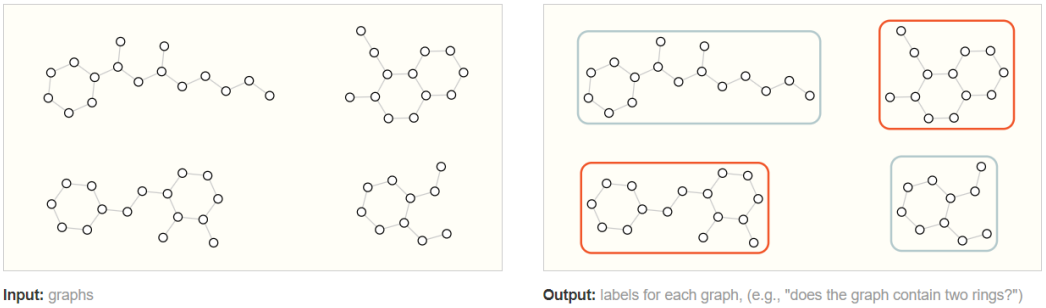


Figure 2.3: Example of graph-level task

- *Node-level task*: It is concerned with the prediction of the identity or role of each node within a graph. A prototypical example of a node-level prediction problem is Zach's karate club. The dataset is a single social network graph comprising individuals who have sworn allegiance to one of two karate clubs as a result of a political schism. As the narrative unfolds, a rift between Mr. Hi (Instructor) and John H (Administrator) gives rise to a schism within the karate club. The nodes represent individual karate practitioners, and the edges represent interactions between these members outside of karate. The prediction problem is to classify whether a given member becomes loyal to either Mr. Hi or John H after the feud. In this case, the distance between a node and either the Instructor or Administrator is highly correlated with this label (Figure 2.4). Node-level prediction problems are analogous to image segmentation, where we are trying to label the role of each pixel in an image. As far text, a similar task would be predicting the parts-of-speech of each word in a sentence (e.g. noun, verb, adverb, etc).

- *Edge-level task*: One example of edge-level inference is in the field of image scene understanding. In addition to identifying objects in an image, Deep Learning models can be employed to predict the relationship between them. This can be expressed as an edge-level classification, wherein nodes representing the objects in the image are used to predict which nodes are connected and the value of those edges (Figure 2.5).

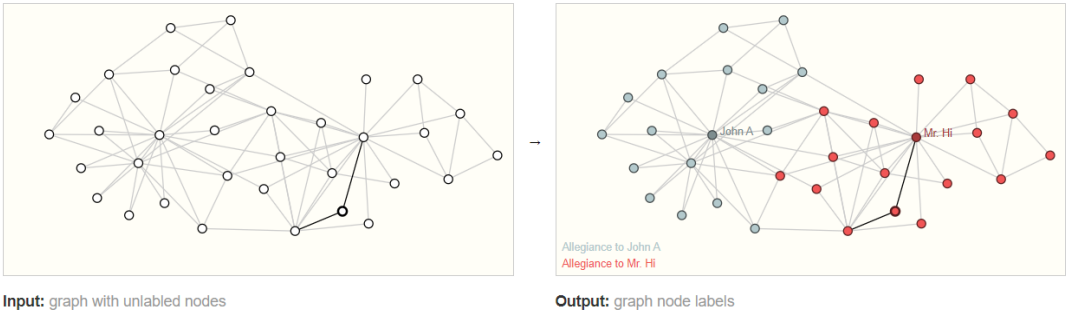


Figure 2.4: Example of node-level task

To discover connections between entities, a fully connected graph can be considered; based on the predicted value, edges can be pruned to arrive at a sparse graph.

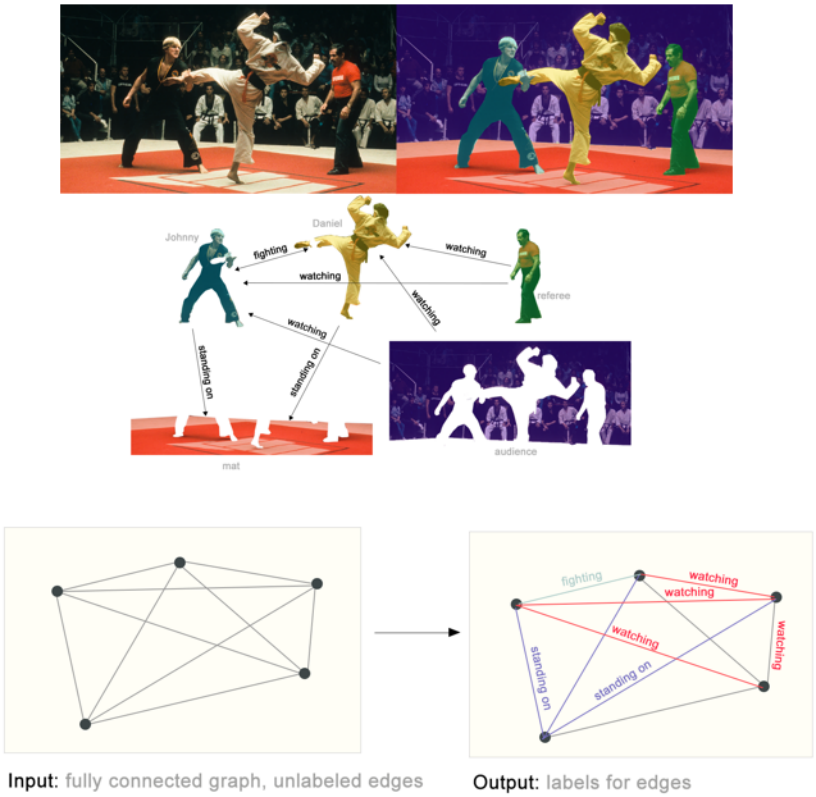


Figure 2.5: Example of edge-level task

The application of Neural Networks to the resolution of these different graph tasks presents a significant challenge. The initial step is to consider how graphs can be represented in a manner that is compatible with Neural Networks. Graphs may contain up to four types of information that could be employed to make predictions. These are nodes, edges, global context, and connectivity. While the first three are relatively straightforward, representing a graph's connectivity is a highly complex undertaking. The most obvious option would be to utilize an adjacency matrix, given its straightforward tensorization. However, this frequently results in highly sparse adjacency matrices, which are inefficient in terms of spatial utilization. Another issue is that numerous adjacency matrices can encode the same

connectivity. However, there is no assurance that these distinct matrices will yield identical outcomes in a Deep Neural Network. This is because they lack permutation invariance.

So, an elegant and memory-efficient method for representing sparse matrices is through the use of adjacency lists. These lists describe the connectivity of the edge e_k between nodes n_i and n_j as a tuple *(i, j)* in the *k-th* entry of an adjacency list. This approach avoids the need for computation and storage on disconnected parts of the graph.

Given that the graph's description is now in a matrix format that is permutation invariant, Graph Neural Networks can be employed to address graph prediction tasks. In fact, a GNN is an optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances).

GNNs exploit a process termed *message passing* to organize graphs in a manner conducive to their utilization by Machine Learning algorithms. This process embeds into each node information about its neighbors. AI models then employ this embedded information to identify patterns and generate predictions. Figure 2.6 shows example dataflows in three types of GNNs.

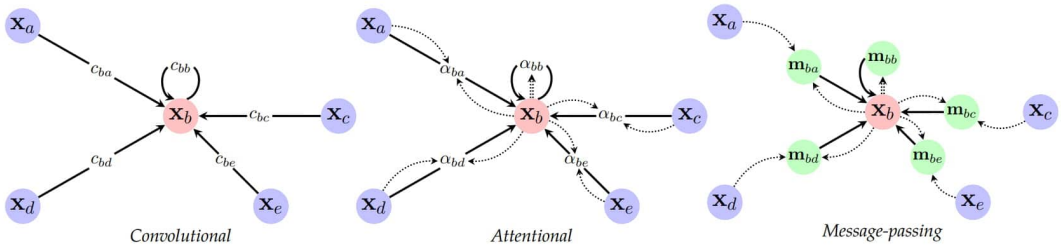


Figure 2.6: Example dataflows in three types of GNNs

For instance, recommendation systems employ a form of node embedding in GNNs to facilitate the matching of customers with products. In the context of fraud detection, edge embeddings are employed to identify anomalous transactions. Similarly, drug discovery models utilize graph-based comparisons of molecules to ascertain their interactions and potential reactions.

Graph Neural Networks can be distinguished from other Machine Learning models in two additional ways: they employ sparse mathematics, and the models typically possess only two or three layers. In contrast to GNNs, other AI models typically employ dense mathematics and comprise hundreds of layers. Figure 2.7 shows a typical GNN pipeline.

A growing number of companies are employing GNNs to enhance drug discovery, fraud detection, and recommendation systems. These and numerous other applications rely on identifying patterns in the relationships between data points. Researchers are investigating the potential applications of GNNs in a range of fields, including computer graphics, cybersecurity, genomics, and materials science. Many scientific and industrial disciplines have already adopted graph databases as a means of storing valuable data. By leveraging Deep Learning techniques, they can train predictive models that facilitate the discovery of novel insights from their graph-based data.

2.3 Knowledge Graph

The term *Knowledge graph* (*KG*) has been used in the literature since at least 1972. However, the modern incarnation of the term can be traced back to the 2012 announcement of the Google Knowledge Graph, which was followed by further announcements of Knowledge Graphs by Airbnb, Amazon, eBay, Facebook, IBM, LinkedIn, Microsoft, Uber, and others.

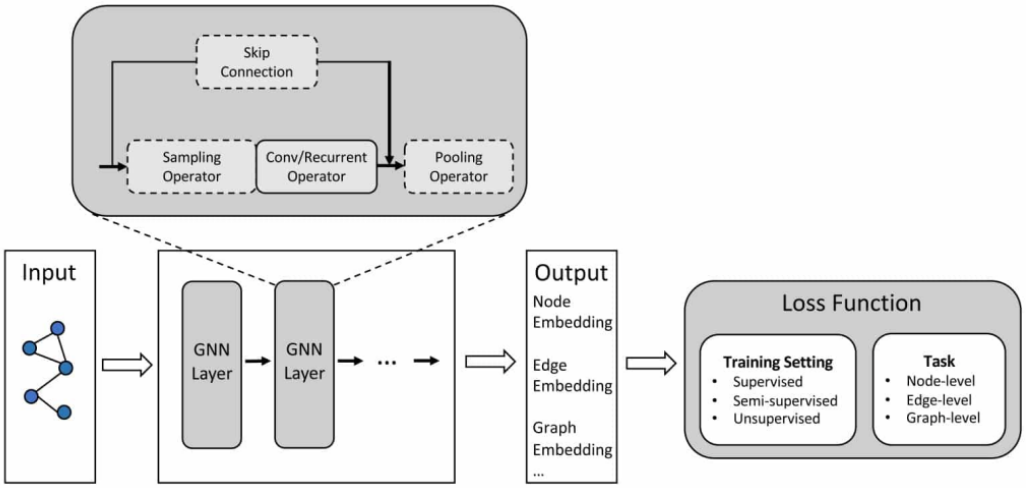


Figure 2.7: GNN pipeline: a graph in input and predictions in output

Knowledge graphs employ a graph-based data model to encapsulate knowledge within application scenarios that entail the integration, administration, and extraction of value from a multitude of data sources on a vast scale.

The use of a graph-based abstraction of knowledge offers a number of advantages when compared with alternative data modeling approaches, such as relational models or NoSQL alternatives. Graphs offer a concise and intuitive abstraction for a variety of domains, where edges and paths capture different, potentially complex relationships between the entities of a domain. The use of graphs permits the postponement of schema definition, thereby facilitating the evolution of data in a more flexible manner.

Graph query languages are capable of supporting not only standard relational operators (such as joins, unions, projections, and so forth), but also navigational operators for the purpose of locating entities that are connected through paths of arbitrary length.

Ontologies and rules can be employed to define and reason about the semantics of the terms utilized in the graph. Scalable frameworks for graph analytics can be utilized for computing centrality, clustering, summarization, and other operations to gain insights about the domain being described. Promising techniques are now emerging for applying Machine Learning over graphs (Hogan *et al.* [2021]).

2.3.1 Definition

A Knowledge Graph is defined as an organized representation of real-world entities and their relationships. Such data is typically stored in a graph database, which is designed to natively store the relationships between data entities. The entities that comprise a Knowledge Graph may represent a multitude of concepts, including, but not limited to, objects, events, situations, and other conceptual entities. The relationships between these entities serve to capture the context and meaning of how they are connected.

A Knowledge Graph stores data and relationships alongside frameworks known as organizing principles. These principles can be conceived of as rules or categories around the data that provide a flexible, conceptual structure to drive deeper data insights. The usefulness of a Knowledge Graph lies in the way it organizes the principles, data, and relationships to surface new knowledge for users or businesses.

The design is useful for many usage patterns, including real-time applications, search and discovery, and grounding generative AI for question-answering.

Individuals tend to overcomplicate the concept of a Knowledge Graph. One may encounter the term "enterprise-wide structure", which refers to a system designed to consolidate and connect information across disparate data silos and sources. Focusing exclusively on the task of bridging large datasets and multiple data sources can lead to the perception that the creation and implementation of Knowledge Graphs is a complex and time-consuming process. However, Knowledge Graphs do not need to be comprehensive or complex. It is possible to construct one with a much smaller scope, tailored to address a specific use case.

The heart of the Knowledge Graph is a knowledge model, which is a collection of interlinked descriptions of concepts, entities, relationships, and events. These descriptions are characterized by formal semantics that facilitate efficient and unambiguous processing by both humans and computers. Additionally, they contribute to one another, forming a network where each entity represents a portion of the description of the entities related to it. Diverse data is connected and described by semantic metadata in accordance with the knowledge model. Figure 2.8 shows an example of a Knowledge Graph representation.

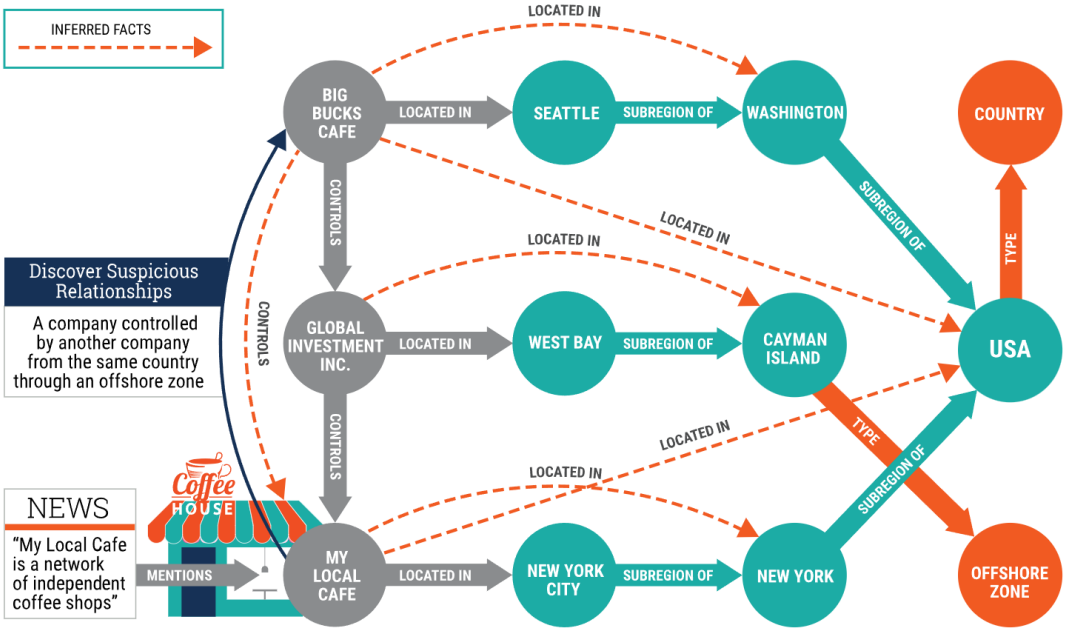


Figure 2.8: Example of a Knowledge Graph representation

Knowledge graphs combine characteristics of several data management paradigms, namely:

- *database*, because the data can be explored via structured queries;
- *graph*, because they can be analyzed as any other network data structure;
- *knowledge base*, because they bear formal semantics, which can be used to interpret the data and infer new facts.

The utilization of Knowledge Graphs, represented in *RDF* (which is explored in depth in Section 2.3.1), offers an optimal framework for data integration, unification, linking, and reuse. This is due to the fact that they combine the following properties:

- *Expressivity*: The standards that comprise the Semantic Web stack (RDF and OWL) facilitate the fluid representation of diverse data and content types, including data schemas, taxonomies, vocabularies, and an array of metadata, reference data, and

master data. The RDF extension simplifies the modeling of provenance and other structured metadata.

- *Performance*: The specifications have been meticulously devised and rigorously tested to ensure the efficient management of graphs comprising billions of facts and properties.
- *Interoperability*: A variety of specifications exist for data serialization, access (SPARQL Protocol for Endpoints), management (SPARQL Graph Store), and federation. The utilization of globally unique identifiers enables the integration and publication of data.
- *Standardization*: The aforementioned elements are standardized through the W3C community process to ensure that the requirements of all stakeholders are met, including those in the fields of logic, enterprise data management, and system operations.

Figure 2.9 illustrates the distinction between a graph, a Knowledge Graph, and a Knowledge Graph with inference. It is evident that Knowledge Graphs offer a superior representation compared to simple graphs.

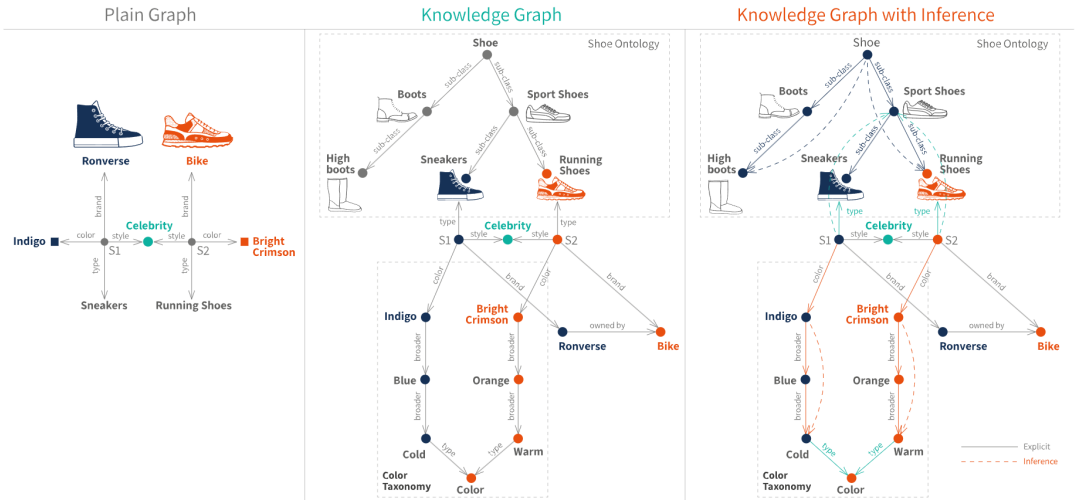


Figure 2.9: Difference between a graph, a Knowledge Graph and a Knowledge Graph with inference

Another significant concept is that of *ontologies*. In fact, these represent the backbone of the formal semantics of a Knowledge Graph. They can be seen as the data schema of the graph. They serve as a formal contract between the developers of the Knowledge Graph and its users regarding the meaning of the data in it. A user could be another human being or a software application that wants to interpret the data in a reliable and precise way. Ontologies ensure a shared understanding of the data and its meanings.

When formal semantics are used to express and interpret the data of a Knowledge Graph, there are a number of representation and modeling instruments. These are:

- *Classes*: In the majority of cases, an entity description comprises a classification of the entity itself with respect to a class hierarchy. To illustrate, when dealing with business information, there could be classes Person, Organization and Location. Persons and organizations can have a common superclass Agent. Location usually has numerous sub-classes, e.g., Country, Populated place, City, etc. The concept of class is also employed in object-oriented design, where each entity is typically associated with a single class.
- *Relationship types*: The relationships between entities are typically defined with specific types, which provide information about their nature. These types may include terms

such as "friend," "relative," "competitor," and so on. Additionally, relationship types may have formal definitions. For instance, the "parent-of" type may be defined as the inverse of the "child-of" type. Both the "parent-of" and "child-of" types are considered special cases of the "relative-of" type, which is a symmetric relationship. Similarly, the "sub-region" and "subsidiary" types may be defined as transitive relationships.

- *Categories*: An entity may be associated with categories that describe some aspect of its semantics. For example, "Big Four consultants" or "19th-century composers" are both examples of such categories. A book may belong simultaneously to all of these categories: "Books about Africa," "Bestseller," "Books by Italian authors," "Books for kids," and so on. These categories are described and ordered into a taxonomy.

- *Free text descriptions*: In many cases, a "human-friendly text" description is provided to facilitate a more comprehensive understanding of the design intentions for the entity and enhance search functionality.

It is crucial to recognize that not all RDF graphs are Knowledge Graphs. To illustrate, a set of statistical data, such as Gross Domestic Product (GDP) data for countries, represented in Resource Description Framework (RDF) is not a Knowledge Graph (KG). A graph representation of data is often useful; however, it may be unnecessary to capture the semantic knowledge of the data. For an application, it may be sufficient to have a string, such as "Italy," associated with a string, such as "GDP," and a number, such as "1.95 trillion," without defining what countries are or what the "Gross Domestic Product" of a country is. It is the connections and the graph that make up a Knowledge Graph, not the language used to represent the data. Additionally, it should be noted that not all knowledge bases are necessarily Knowledge Graphs. A fundamental characteristic of a Knowledge Graph is the interlinking of entity descriptions. The definition of one entity includes another entity. This linking process is how the graph is formed. For example, one might posit that A is B, that B is C, that C has D, and that A has D. It is evident that knowledge bases devoid of formal structure and semantics (such as a Q&A "knowledge base" about a software product) do not represent a KG. It is possible to have an expert system that has a collection of data organized in a format that is not a graph but uses automated deductive processes, such as a set of "if-then" rules, to facilitate analysis.

The following list provides representative examples of large-scale Knowledge Graphs:

- *Google Knowledge Graph*: Google made this term popular with the announcement of its Knowledge Graph in 2012. However, there are very few technical details about its organization, coverage and size. There are also very limited means for using this Knowledge Graph outside Google's own projects.

- *DBPedia*: This project exploits the structure inherent in the infoboxes of Wikipedia to create an extensive dataset comprising 6 million entities and an ontology that provides comprehensive coverage of a diverse range of subjects, including people, places, films, books, organizations, species, diseases, and more. This dataset is at the heart of the Open Linked Data movement. It has proven to be a valuable resource for organizations seeking to populate their internal Knowledge Graphs with millions of crowdsourced entities.

- *Geonames*: In accordance with the terms of the Creative Commons license, users of the Geonames dataset are granted access to a repository of 25 million geographical entities and features.

- *Wordnet*: One of the most well-known lexical databases for the English language, it provides definitions and synonyms. It is frequently employed to augment the performance of Natural Language Processing and search applications.
- *FactForge*: Following years of experience in the news publishing industry, Ontotext developed a Knowledge Graph of Linked Open Data and news articles about people, organizations, and locations. This graph incorporates data from the aforementioned Knowledge Graphs, as well as specialized ontologies, including the Financial Industry Business Ontology.

RDF

RDF stands for *Resource Description Framework* and it is a standard for describing web resources and data interchange, developed and standardized with the World Wide Web Consortium (W3C). While there are many conventional tools for dealing with data, and more specifically for dealing with the relationships between data, RDF is the easiest, most powerful and expressive standard designed by now. In fact, RDF is a general method of describing data by defining relationships between data objects. It enables effective data integration from multiple sources, detaching data from its schema. This allows multiple schemas to be applied, interlinked, queried as one and modified without changing the data instances. Additionally, it is built around the existing Web standards: XML and URL (URI).

The manner in which RDF links data elements is through the use of triples, which are three-part statements. This approach allows for the expression of facts, relationships, and data by linking resources of diverse types. With the aid of an RDF statement, a wide range of information can be conveyed through a consistent structure comprising three interconnected data elements. As a robust and expressive framework for representing data, RDF is employed in the construction of Knowledge Graphs, which are characterized by their richness, interlinkage, interoperability, and flexibility.

An RDF Knowledge Graph may comprise three types of nodes:

- *resources*, represented by a unique resource identifier (URI), such as a well-known URL;
- *literals*, which are analogous to those in XML;
- *auxiliary blank nodes*.

The types of edges are referred to as predicates, exemplified by "partOf" or "knows" (Figure 2.10). The use of named graphs or contexts (e.g., g1 and g2 in Figure 2.10) facilitates the management of components in the graph (e.g., by provenance). Each edge in the graph represents a fact and can be seen as a quadruple comprising the subject, predicate, object, and context. Classes, predicates, and named graphs are all defined as Uniform Resource Identifiers (URIs). This allows them to appear as nodes in the graph, thereby enabling the retrieval of their descriptions. This, in turn, facilitates the management and access to instance data and schema in a uniform model.

RDF is often criticized for its inability to accommodate descriptions or properties attached to edges in a graph. This is perceived as a disadvantage compared to property graphs. However, this concern has been addressed with RDF-Star (RDF*), which allows one to make statements about other statements, thereby enabling the attachment of metadata to edges in a graph.

2.3.2 Use cases

Knowledge graphs provide a robust tool for storing and organizing data, facilitating a more sophisticated understanding of that data. To comprehend the ways in which companies

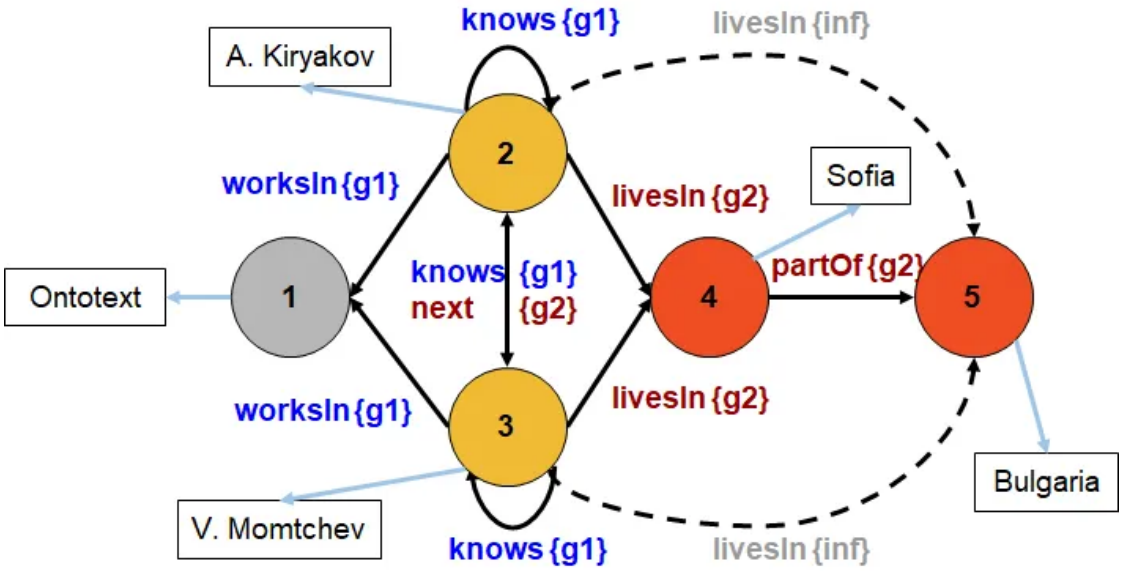


Figure 2.10: RDF: Directed Labeled Cyclic Multigraph with Labels, Types, Logic and Semantics

have employed Knowledge Graphs, it is instructive to examine instances where they have been utilized to address specific challenges. While not a comprehensive enumeration of use cases, this selection of concrete examples illustrates the practical applications of Knowledge Graphs in real-world scenarios:

- *Generative AI for Enterprise Search Applications*: In the context of generative AI applications, Knowledge Graphs serve to capture and organize key domain-specific or proprietary company information. Knowledge graphs are not constrained by the limitations of structured data; they are capable of accommodating less organized data as well. GraphRAG, a technique that grounds large language models with Knowledge Graphs, is emerging as the foundation of AI applications that exploit proprietary domain data (these are known as RAG applications). A Knowledge Graph grounding enhances the accuracy of responses and improves the explainability of results by providing context through data relationships. Industry leaders, such as Deloitte, emphasize the pivotal role of Knowledge Graphs in developing enterprise-grade GenAI.

- *Fraud Detection and Analytics in Financial Services, Banking, and Insurance*: In the field of fraud detection and analytics, a Knowledge Graph represents a network of transactions, their participants, and relevant information about them. Companies can utilize this Knowledge Graph to expeditiously identify suspicious activity, investigate suspected fraud, and evolve their Knowledge Graph to keep pace with evolving fraud patterns. Algorithms such as pathfinding and community detection provide crucial signals to Machine Learning algorithms that can uncover more sophisticated fraud networks.

- *Master Data Management*: In Master Data Management (e.g., for Customer 360 use cases), the Knowledge Graph model provides an organized, resolved (i.e., "de-duped"), and comprehensive database of a company's customers and the company's interactions with them. This organized view of customers is especially important for companies with multiple divisions or applications interacting with customers. Without a Knowledge Graph, it can be difficult or impossible to obtain an accurate view of the customer. A Knowledge Graph links customer behaviors across multiple applications through an organizing principle that identifies them as coming from the same customer.

- *Supply Chain Management*: In the field of supply chain management, a Knowledge Graph represents the complex network of suppliers, raw materials, products, and logistics that work together to supply a company's operations and customers. This end-to-end supply chain visibility allows managers to identify potential weaknesses and predict where disruptions may occur. Graph algorithms, such as shortest path, optimize the supply chain in real time by identifying the most direct route between any two points.

- *Investigative Journalism*: In the field of investigative journalism, Knowledge Graphs are utilized to capture and organize key entities (such as companies, individuals, and financial accounts) and activities under investigation. By organizing these entities in relation to one another, it becomes possible to identify hidden patterns, including distant relationships between entities that may not be immediately apparent. Investigators may employ techniques such as entity resolution to uncover entities that are masking their activities behind fake or shell identities. Additionally, algorithms like community detection and link prediction offer valuable insights and potential avenues for further investigation.

- *Drug Discovery in Healthcare Research*: A Knowledge Graph is a data structure that stores information about a research subject in medical and other research use cases. For example, it may contain protein and genome sequences, along with environmental and chemical data. This data can be used to identify intricate patterns and expand our understanding of proteins.

- *Retail*: Knowledge Graphs have been for up-sell and cross-sell strategies, recommending products based on individual purchase behavior and popular purchase trends across demographic groups.

- *Entertainment*: Knowledge Graphs are leveraged for Artificial Intelligence based recommendation engines for content platforms, like Netflix, SEO, or social media. Based on click and other online engagement behaviors, these providers recommend new content for users to read or watch.

- *Finance*: This technology has also been used for "know-your-customer" and "anti-money laundering" initiatives within the finance industry. They assist in financial crime prevention and investigation, allowing banking institutions to understand the flow of money across their clientele and identify noncompliant customers.

- *Healthcare*: Knowledge Graphs are also benefiting the healthcare industry by organizing and categorizing relationships within medical research. This information assists providers by validating diagnoses and identifying treatment plans based on individual needs.

- *Education*: Processing complex and unstructured educational data is challenging, making structured data like Knowledge Graphs valuable. Knowledge graph-based systems improve both offline school teaching and online learning by supporting tasks such as course allocation and filtering educational content. In the context of the ongoing Covid-19 pandemic, as well as in the subsequent period, online learning became crucial, prompting researchers to improve online learning environments using Knowledge Graphs. Overall, these technologies make education more efficient and accessible, benefiting both traditional and digital learning contexts.

- *Scientific research*: Knowledge Graphs help researchers explore research knowledge and identify relevant materials by interlinking documents, authors, entities, and contextual

information. Examples include the Microsoft Academic Graph (MAG), AMiner Graph, and Artificial Intelligence Knowledge Graph (AI-KG), which collectively contain vast amounts of academic data. Researchers also develop applications using these Knowledge Graphs, such as scientific publication management models and recommender systems for citations, collaborations, and reviewer assignments, enhancing the efficiency and effectiveness of scientific research.

- *Social networks*: The integration of Knowledge Graphs with social media enhances user experiences and addresses challenges like content recommendation, social relationship extraction, and fake news detection. Social Knowledge Graphs, derived from users' activities and posts on platforms like Facebook and Twitter, help analyze critical information for various applications. These graphs enable personalized content recommendations, as demonstrated by models like GraphRec, which use social and item interactions to improve accuracy. Additionally, social relationship extraction from media posts assists companies in tracking user behavior. To contrast fake news, models like DEAP-FAKED utilize Knowledge Graphs and Neural Networks to detect misinformation on social media.

2.3.3 Technical challenges

Although Knowledge Graphs offer relevant opportunities for various services and applications, many challenges are yet to be addressed. Specifically, the limitations of existing Knowledge Graph technologies are the key challenges for promoting the development of Knowledge Graphs (Peng *et al.* [2023]).

First of all, the objective of *Knowledge Graph embeddings* is to represent Knowledge Graphs in a low-dimensional vector space in a manner that effectively preserves the semantics. Firstly, the entities and relationships are embedded into a high-dimensional space in a given Knowledge Graph, and a scoring function is defined to measure the plausibility of each fact (triplet). Subsequently, the plausibility of the facts is optimized in order to obtain the entity and relationship embeddings. The representation of Knowledge Graphs offers a multitude of advantages with respect to downstream tasks. The three principal categories of triplet fact-based Knowledge Graph embedding approaches are tensor factorization-based, translation-based, and Neural Network-based methods. But these methods still suffer several severe limitations. Many established methods only consider surface facts (triplets) of Knowledge Graphs. However, additional information, such as entity types and relation paths, are ignored, which can further improve the embedding accuracy. The performance of most traditional methods that do not consider the additional information is unsatisfactory.

Some researchers have started to combine additional information with a Knowledge Graph to improve the efficiency of embedding models. Many of these approaches leverage additional entity type information, specifically the semantic category of each entity, to identify correlations between entities and address data sparsity issues. Consequently, Knowledge Graphs are represented with greater precision. In addition to entity types, researchers are also focusing on other forms of information, including relation paths, temporal data in dynamic graphs, and textual descriptions of entities.

Nevertheless, effectively leveraging this expanded data to enhance the precision of Knowledge Graph embeddings remains a challenging issue. It is evident that general additional information is insufficient for adequately representing the semantic meaning of triplets. For instance, the entity types are not related to the semantic information of triplets. Furthermore, the types of additional information that can be incorporated into the features of the triplets are now severely limited. In order to enhance the performance of existing Knowledge Graph embedding methods, it is imperative to incorporate multivariate information (such as the

hierarchical descriptions of relationships and the combination of entity types and textual descriptions) into the features of the triplets. The field of complex relationships path remains an open research problem. For example, the inherent relationships, which pertain to the indirect relationships between two unconnected entities, are not represented effectively. While the inherent relationships between the entities can be explored based on the chain of relationships in Knowledge Graphs, the inherent relationships are complex and multiple. As a result, it is not straightforward to represent these relationships effectively.

Secondly, also *knowledge acquisition* is a critical step for combining data from different sources and generating new Knowledge Graphs. The knowledge is extracted from both structured and unstructured data. Three main methods of knowledge acquisition are relationhip extraction, entity extraction, and attribute extraction. Nevertheless, existing methods for knowledge acquisition continue to be faced with the challenge of low accuracy, which could result in incomplete or noisy Knowledge Graphs and impede the completion of subsequent tasks. It is evident that the reliability of knowledge acquisition tools and their evaluation represents a critical issue. Furthermore, it is important to note that a domain-specific Knowledge Graph schema is knowledge-oriented, whereas a constructed Knowledge Graph schema is data-oriented and designed to encompass all data features. Consequently, it is inefficient to produce domain-specific Knowledge Graphs by extracting entities and properties from raw data. This highlights the necessity for the development of efficient methods for achieving knowledge acquisition tasks by generating domain-specific Knowledge Graphs. Besides, most existing knowledge acquisition methods focus on constructing Knowledge Graphs with one specific language.

However, in order to make the information in Knowledge Graphs richer and more comprehensive, we need cross-lingual entity extraction. It is thus vitally important to give more attention to cross-lingual entity extraction and the generation of multilingual Knowledge Graphs. Nevertheless, the construction of Knowledge Graphs in multiple languages is still a challenging issue. This is due to the scarcity of non-English training data sets, the imperfections of language translation systems, and the necessity of retraining cross-lingual entity extraction models for each new language. The construction of multi-modal Knowledge Graphs is regarded as another challenging issue in the field of knowledge acquisition. The existing Knowledge Graphs are predominantly represented by pure symbols, which may result in the poor capability of machines to understand the nuances of the real world. Consequently, many researchers are focusing on the development of multi-modal Knowledge Graphs comprising a diverse range of entities, including texts and images. The construction of multi-modal Knowledge Graphs necessitates the exploration of entities with varying modalities, which makes the knowledge acquisition process complex and inefficient.

Additionally, it is not uncommon for Knowledge Graphs to be incomplete, with several relevant triplets and entities absent. Generally, semi-automated and human leveraging mechanisms, which can be applied to ensure the quality of Knowledge Graphs, are essential tools for the evaluation of Knowledge Graph completion. Specifically, human supervision is currently considered the gold standard evaluation in *Knowledge Graph completion*. Its objective is to augment existing Knowledge Graphs by incorporating new triplets through the application of techniques for link prediction and entity prediction. These methodologies typically entail the training of a Machine Learning model on a Knowledge Graph to evaluate the likelihood of new candidate triplets. Subsequently, the triplets with high plausibility are incorporated into the Knowledge Graph.

However, most current Knowledge Graph completion methods only focus on extracting triplets from a closed-world data source. That means the generated triplets are new, but the entities or relationships in the triplets need to already exist in the Knowledge Graph. As a result of this limitation, these methods are unable to incorporate new entities and relationships

into the Knowledge Graph. In order to address this issue, there has been a growing interest in the development of open-world techniques for Knowledge Graph completion, which aim to extract potential objects from sources external to existing knowledge bases.

Nevertheless, the current methodologies for open-world Knowledge Graph completion continue to have suboptimal accuracy. This is primarily due to the intrinsic complexity and noise inherent in the data source. Furthermore, the similarity between the predicted new entities and the existing ones can lead to erroneous results. In other words, two entities with high similarity are often regarded as connected entities, despite the absence of a direct relationship.

The assumption that Knowledge Graphs are static, which is inherent to Knowledge Graph completion methods, fails to account for their dynamic evolution. To obtain accurate facts over time, temporal Knowledge Graph completion, which considers temporal information reflecting the validity of knowledge, has emerged as a solution. In contrast to static Knowledge Graph completion, temporal Knowledge Graph completion methods integrate timestamps into the learning process, thereby enabling the exploration of time-sensitive facts and a notable improvement in link prediction accuracy. Despite the impressive performance demonstrated by temporal Knowledge Graph completion methods, they still have to address significant challenges. The incorporation of time information into learning models may prove less efficient, necessitating the development of effective strategies for the integration of timestamps and the capture of temporal dynamics in facts.

Once more, *knowledge fusion* is the process of combining and integrating knowledge from disparate data sources. It is often a necessary step for the generation of Knowledge Graphs. The primary method of knowledge fusion is entity alignment or ontology alignment, which aims to match the same entity from multiple Knowledge Graphs. Achieving efficient and accurate Knowledge Graph fusion is a challenging task because of the complexity, variety, and large volume of data available today. Although considerable research has been conducted in this area, there are still several promising strands for future investigation. One such strand concerns cross-language knowledge fusion, which enables the integration of information from disparate linguistic backgrounds. This approach is frequently employed to bolster cross-lingual recommender systems. To illustrate, some researchers have employed a graph-matching Neural Network to achieve cross-language entity alignment. Nevertheless, the outcomes of cross-language knowledge fusion remain suboptimal, as the accuracy of matching entities across languages remains relatively low. Consequently, the exploration of cross-language knowledge fusion represents a challenging issue.

A further significant challenge is that of entity disambiguation. As a consequence of the polysemy problem inherent in natural language, the same entity may be referred to in a variety of ways across different Knowledge Graphs. Consequently, entity disambiguation is a prerequisite for conducting entity alignment. The existing entity disambiguation methods primarily focus on distinguishing and matching ambiguous entities based on the extraction of knowledge from texts that contain a wealth of contextual information. However, these methods are unable to accurately assess the semantic similarity of entities when the texts are short and lack sufficient contextual detail. Moreover, numerous knowledge fusion methodologies prioritize the matching of entities with identical modalities, while disregarding the presence of multi-modal scenes wherein knowledge is conveyed in disparate forms. In particular, entity alignment within a single-modality Knowledge Graph context exhibits limited effectiveness, as it is unable to adequately represent the intricate relationships between entities in the real world. In response to this challenge, recent research has proposed the integration of multi-modal knowledge fusion, a technique that aligns entities with varying modalities and generates a multi-modal Knowledge Graph. The primary objective of multi-modal knowledge fusion is to identify equivalent entities by integrating their multi-modal features. However,

the challenge remains in efficiently incorporating features with multiple modalities, which is a significant hurdle for current methods.

Finally, the goal of *knowledge reasoning* is to infer new knowledge, such as the implicit relationships between two entities, based on existing data. For a given Knowledge Graph, wherein there are two unconnected entities h and t, denoted as $h, t \in G$ (here G means the Knowledge Graph), knowledge reasoning can find out the potential relationship r between these entities and form a new triplet (h, r, t). The knowledge reasoning methods are mainly categorized into logic rule-based, distributed representation-based, and Neural Network-based methods. Logic rule-based knowledge reasoning aims to discover knowledge according to the random walk and logic rules, while distributed representation-based knowledge reasoning embeds entities and relations into a vector space to obtain distributed representation. Neural Network-based knowledge reasoning method utilizes Neural Networks to infer new triplets given the body of knowledge in the graph.

Two distinct tasks are involved in knowledge reasoning, namely single-hop prediction and multi-hop reasoning. Single-hop prediction entails the prediction of a single element of a triplet, given two elements. In contrast, multi-hop reasoning involves the prediction of one or more elements in a multihop logical query. In other words, in the multi-hop reasoning scenario, the answer to a typical question and the formation of new triplets require the prediction and imputation of multiple edges and nodes. Multi-hop reasoning achieves a more precise formation of triplets when compared with single-hop prediction. Therefore, multi-hop reasoning has attracted more attention and become a critical need for the development of Knowledge Graphs in recent years. Multi-hop reasoning needs to traverse multiple relations and intermediate entities in the Knowledge Graph, which could lead to exponential computation cost. Therefore, it is still a daunting task to explore multi-hop knowledge reasoning. Furthermore, the verification of inferred new knowledge represents a significant challenge. The incorporation of knowledge reasoning into existing Knowledge Graphs offers the potential to enhance the performance of downstream tasks. However, the inferred new knowledge may be subject to uncertainty, necessitating the verification of the veracity of new triplets. Additionally, the detection of conflicts between new and existing knowledge is essential. To address these challenges, research has proposed multi-source knowledge reasoning as a means of identifying erroneous and conflicting knowledge. Overall, there is a need for greater focus on multi-source knowledge reasoning and the reduction of erroneous knowledge.

CHAPTER 3

Related works

This chapter will provide a comprehensive exploration of the integration of Knowledge Graphs (KGs) with Large Language Models (LLMs) to overcome challenges such as hallucination and lack of interpretability. It is organized into several sections, each examining different methodologies and strategies for enhancing LLMs with KGs. The initial section will present the concept of integrating Knowledge Graphs with Large Language Models (LLMs). It will examine how KGs, as structured repositories of knowledge, can enhance the performance and reliability of LLMs by providing access to domain-specific information, thereby improving their interpretability and reducing their propensity to generate inaccurate outputs. This section will delineate various methodologies for integrating KGs with LLMs at both the pre-training and inference stages. The next section will examine specific techniques for integrating Knowledge Graphs during the pre-training phase of Large Language Models. This section will be divided into three main subsections. The initial one will examine techniques for aligning the training processes of LLMs with knowledge-aware objectives, thereby enabling models to learn effectively from both token-level and entity-level data. The second subsection will examine methods for embedding KGs directly into the input data of LLMs, thereby enhancing their contextual understanding and improving the quality of their outputs. Finally, the third subsection will investigate how KGs can be fine-tuned to guide LLMs during specific tasks, thereby improving the models' interpretability and task-specific performance. The final section of the chapter will review recent advancements and ongoing research in the integration of KGs with LLMs, highlighting both the successes and challenges faced in this area. It will discuss the implications of these advancements for future research and practical applications in Natural Language Processing. The work presented in this chapter draws inspiration from the foundational research of Pan et al. *[2024], which provides a comprehensive analysis of the benefits and challenges of integrating Knowledge Graphs with Large Language Models. Building upon these insights, this chapter aims to contribute to the ongoing development of more interpretable and reliable Large Language Models, thereby setting the stage for future advancements in the field.*

3.1 KG-enhanced LLMs

Large Language Models (LLMs) are renowned for their ability to learn knowledge from large-scale corpora and achieve state-of-the-art performance in a range of Natural Language Processing (NLP) tasks. However, LLMs are frequently the subject of criticism regarding their propensity for hallucination and their lack of interpretability. In order to address these issues, researchers have put forth the proposal of enhancing LLMs with the use of Knowledge Graphs (KGs). Knowledge Graphs are repositories of vast amounts of explicit and structured knowledge, which can be leveraged to enhance the knowledge awareness

of LLMs. Some researchers have proposed incorporating Knowledge Graphs into Large Language Models during the pre-training stage, which can facilitate LLM learning from KGs. Other researchers have proposed incorporating KGs into LLMs during the inference stage. Retrieving knowledge from KGs, can markedly enhance the performance of LLMs in accessing domain-specific knowledge. Finally, in order to enhance the interpretability of LLMs, researchers also utilize KGs to interpret the facts and reasoning processes of LLMs.

3.1.1 KG-enhanced LLM Pre-training

Existing Large Language Models mostly rely on unsupervised training on the large-scale corpus. While these models may exhibit impressive performance on downstream tasks, they often lack practical knowledge relevant to the real world. The majority of works that integrate Knowledge Graphs (KGs) into Large Language Models (LLMs) can be categorized into three main approaches, namely:

- *integration of Knowledge Graphs into the training objective*;
- *integration of Knowledge Graphs into Large Language Model inputs*;
- *tuning of Knowledge Graphs for instructional purposes*.

In the initial approach, namely "integration of Knowledge Graphs into the training objective", research efforts are concentrated on the design of novel knowledge-aware training objectives. An intuitive idea is to incorporate a greater number of knowledge entities into the pre-training objective. GLM (Shen *et al.* [2020]) employs the Knowledge Graph structure to assign a masking probability. In particular, entities that can be reached within a specified number of hops are deemed the most crucial for learning and are thus assigned a higher masking probability during pre-training. In addition, E-BERT (Zhang *et al.* [2020a]) regulates the equilibrium between the token-level and entity-level training losses. The training loss values are employed as indicators of the learning process for token and entity, which enables the dynamic determination of their ratio for the subsequent training epochs. It can be observed that other types of models also adhere to a comparable fusion to inject sentiment knowledge during LLMs pre-training, as exemplified by SKEP (Tian *et al.* [2020]). This type of model initially determines words with positive and negative sentiment through the utilization of PMI in conjunction with a predefined set of seed sentiment words. Subsequently, it assigns a higher masking probability to those identified sentiment words in the word masking objective. The other line of work explicitly leverages the connections with knowledge and input text. In this instance, models, such as ERNIE (Zhang *et al.* [2019]), put forward a novel word-entity alignment training objective as a pre-training objective. In particular, both sentences and corresponding entities mentioned in the text are fed into LLMs, which are then trained to predict alignment links between textual tokens and entities in Knowledge Graphs.

The second approach, namely "the integration of Knowledge Graphs into Large Language Model inputs", is concerned with the introduction of relevant knowledge sub-graphs into the inputs of LLMs. With ERNIE 3.0 (Sun *et al.* [2021]), a Knowledge Graph triple and the corresponding sentences are represented as a sequence of tokens, which are then directly concatenated with one another. Furthermore, the relation token in the triple or tokens in the sentences are randomly masked in order to more effectively combine knowledge with textual representations. However, this direct knowledge triple concatenation method allows the tokens in the sentence to interact intensively with the tokens in the knowledge sub-graph, which could result in the introduction of noise into the knowledge base. To address this challenge, alternative models, such as K-BERT (Liu *et al.* [2020]), have been developed that integrate the knowledge triple into the sentence via a visible matrix. In these models,

only the knowledge entities have access to the knowledge triple information, while the tokens in the sentences can only interact with each other through the self-attention module. The aforementioned methods are indeed capable of imparting a substantial amount of knowledge to LLMs. However, the majority of these methods prioritize entities with high popularity, while neglecting to consider those with low frequency and limited representation. Consequently, Zhang *et al.* [2022a] propose DkLLM, which is a novel measure to identify long-tail entities, and then replace these selected entities in the text with pseudo-token embedding as new input to the Large Language Models. Moreover, recent methodologies, such as Dict-BERT (Yu *et al.* [2021]), suggest leveraging external dictionaries to address this challenge. In particular, the approach entails enhancing the representation quality of rare words by appending their definitions from the dictionary at the conclusion of the input text and training the Large Language Model to locally align rare word representations in input sentences and dictionary definitions. Additionally, the model is trained to discern whether the input text and definition are correctly mapped.

The third approach, namely "the tuning of Knowledge Graphs for instructional purposes", differs from the previous two in that it does not inject factual knowledge into Large Language Models. Instead, it aims to fine-tune LLMs to better comprehend the structure of KGs and effectively follow user instructions to conduct complex tasks. The process of KG instruction tuning employs both factual data and the structural characteristics of KGs to generate instruction-tuning datasets. LLMs that have been fine-tuned on these datasets are able to extract both factual and structural knowledge from KGs, thereby enhancing the reasoning ability of LLMs. KPPLM (Wang *et al.* [2022]) initially creates a series of prompt templates with the objective of transforming structural graphs into natural language text. Subsequently, two self-supervised tasks are proposed for the fine-tuning of LLMs, with the objective of further leveraging the knowledge derived from the aforementioned prompts. In addition, alternative models, such as OntoPrompt (Ye *et al.* [2022]), suggest an ontology-enhanced prompt tuning approach that can situate entity knowledge within the context of LLMs, which are subsequently fine-tuned on a range of downstream tasks. In order to facilitate more effective reasoning about graphs, Luo *et al.* [2023b] have proposed a planning-retrieval-reasoning framework, named RoG. This type of model is fine-tuned on KG structure to generate relation paths that are grounded by KGs as faithful plans. These plans are then used to retrieve valid reasoning paths from the KGs for LLMs to conduct faithful reasoning and generate interpretable results.

3.1.2 KG-enhanced LLM Inference

In the context of KG-enhanced LLM inference, a considerable body of research has been dedicated to maintaining the separation between the knowledge space and the text space, and to the injection of knowledge during the inference process. These types of methods have primarily been developed for use in Question-Answering (QA) tasks, given the necessity for the model to capture both the textual semantic meanings and the up-to-date real-world knowledge inherent to such tasks.

Retrieval-Augmented Knowledge Fusion is a prevalent methodology for incorporating knowledge into Large Language Models during the inference phase. The fundamental concept is to retrieve pertinent knowledge from a vast corpus and subsequently integrate it into LLM systems. As illustrated in Figure 3.1, *Retrieval-Augmented Generation* (*RAG*), as proposed by Lewis *et al.* [2020], suggests a methodology whereby non-parametric and parametric modules are integrated to facilitate the incorporation of external knowledge. In order to process the input text, RAG initially searches for a relevant Knowledge Graph within the non-parametric module via the *Maximum Inner-Product Search* (*MIPS*) search engine, thereby retrieving a number of relevant documents. Subsequently, these documents are treated as

hidden variables and fed into the output generator, which is empowered by Seq2Seq LLMs, as additional context information. The findings suggest that utilizing a range of retrieved documents as conditions at various stages of the generation process yields superior outcomes compared to relying on a single document to direct the entirety of the generation procedure. The experimental results demonstrate that RAG outperforms other parametric-only and non-parametric-only baseline models in open-domain QA. Additionally, it is capable of generating text that is more specific, diverse, and factual than other parameter-only baselines. Story-fragments further enhance the architecture by incorporating an additional module to identify salient knowledge entities and integrate them into the generator, thereby improving the quality of generated long stories. Moreover, alternative methodologies, such as EMAT (Wu *et al.* [2022]), enhance the efficiency of this system by encoding external knowledge into a key-value memory and leveraging the rapid MIPS for memory querying.

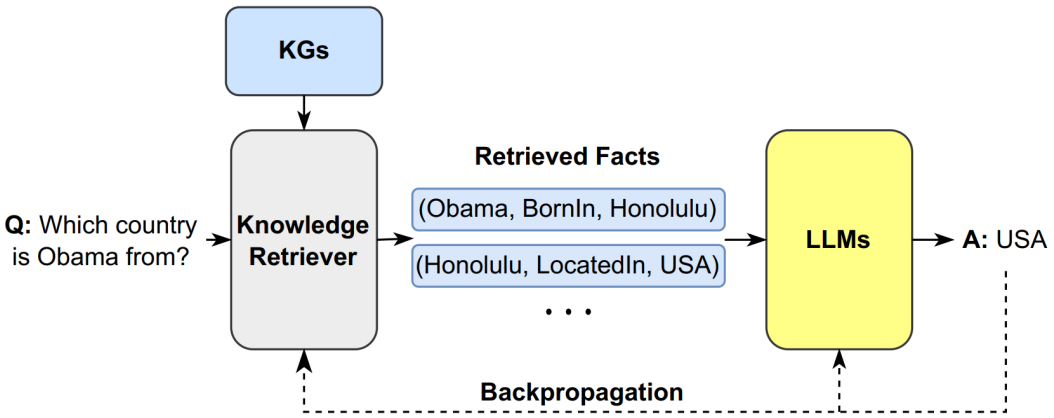


Figure 3.1: Retrieving external knowledge to enhance the LLM generation

Furthermore, the objective of *KG prompting* is to develop a bespoke prompt that can transform structured KGs into text sequences, which can then be incorporated as context into LLMs during the inference process. In this way, LLMs are able to more effectively leverage the structural characteristics of KGs to facilitate reasoning processes. Li *et al.* [2023] utilize a pre-defined template to transform each triple into a brief sentence that can be comprehended by LLMs for reasoning purposes. Some methods, such as MindMap (Wen *et al.* [2023]), have been developed which enable KG prompts to be converted into mind maps, thus providing LLMs with the capacity to perform reasoning by consolidating the facts present in KGs with the implicit knowledge held by LLMs. Other methods, such as ChatRule (Luo *et al.* [2023a]), employ a sampling approach, whereby relation paths from Knowledge Graphs are verbalized and fed into Large Language Models. Subsequently, LLMs are prompted to generate meaningful logical rules that can be employed for reasoning purposes.

In conclusion, the use of prompts represents a straightforward method for integrating LLM and KG technologies. By employing prompts, it is possible to effectively leverage the capabilities of LLM for reasoning operations based on KGs, obviating the necessity for model retraining. However, the prompt is typically designed manually, a process that requires considerable human effort.

3.1.3 KG-enhanced LLM Interpretability

Despite the considerable achievements of LLM in numerous NLP tasks, they remain subject to criticism regarding their lack of interpretability. The interpretability of a Large Language Model pertains to the comprehension and elucidation of the internal mechanisms

and decision-making processes of such a model (Danilevsky *et al.* [2020]). This can enhance the reliability of LLM and facilitate their deployment in high-stakes contexts, such as medical diagnosis and legal judgment. Knowledge Graphs represent knowledge in a structured manner and can provide effective interpretability for the reasoning results. Consequently, researchers endeavor to leverage KGs to enhance the interpretability of LLM, which can be broadly classified into two categories:

- *KGs for Large Language Model probing*;
- *KGs for Large Language Model analysis*.

The objective of probing with a Large Language Model is to gain insight into the knowledge stored in LLMs. Large Language Models, trained on large-scale corpus, are often described as containing vast amounts of knowledge. However, the manner in which LLMs store knowledge makes it challenging to ascertain the nature of the stored knowledge. Furthermore, LLMs are prone to the hallucination problem (Ji *et al.* [2023]), which manifests as the generation of statements that contradict established facts. This issue has a significant impact on the reliability of LLMs. It is therefore essential to probe and verify the knowledge stored in LLMs. LAMA (Petroni *et al.* [2019]) represents the inaugural work to probe the knowledge in LLMs by utilizing KGs. As illustrated in Figure 3.2, LAMA initially converts the facts in KGs into cloze statements using a predefined prompt template, and then employs LLMs to predict the missing entity. The prediction outcomes are employed to evaluate the knowledge stored in LLMs. In addition, Mallen *et al.* [2022] investigate the capacity of LLMs to retain less popular factual knowledge. They select unpopular facts from Wikidata Knowledge Graphs that have low-frequency clicked entities. These facts are then used for evaluation, the results of which indicate that LLMs encounter difficulties with such knowledge and that scaling fails to appreciably improve the memorization of factual knowledge in the tail.

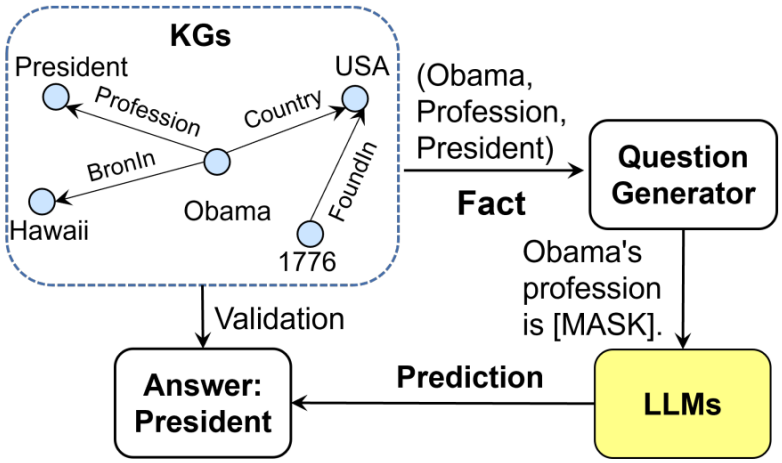


Figure 3.2: General framework of using Knowledge Graph for language model probing

The objective of analyzing Knowledge Graphs for pre-trained Large Language Models is to address two fundamental questions: "How do LLMs generate the results?" and "How do the functions and structures of LLMs operate?". To examine the inference process of LLMs, some methodologies, such as KagNet (Lin *et al.* [2019]) and QA-GNN (Yasunaga *et al.* [2021]), utilize Knowledge Graphs to ground the results generated by LLMs at each reasoning step. This approach enables the reasoning process of LLMs to be elucidated by extracting the graph structure from KGs. Li *et al.* [2022] investigate the way in which LLMs generate

results. They adopt a causal-inspired analysis of facts extracted from Knowledge Graphs. This analysis quantitatively measures the word patterns that LLMs rely on to generate results. The findings indicate that LLMs generate missing factual information with greater frequency using positionally closed words rather than those dependent on knowledge. Consequently, they assert that LLMs are unable to accurately memorize factual knowledge due to their imperfect dependence. Moreover, Swamy *et al.* [2021] employ the Large Language Model during pre-training to generate Knowledge Graphs, thereby facilitating the interpretation of LLM training. The knowledge acquired by LLM during training can be elucidated by the facts explicitly present in KGs.

3.2 LLM-augmented KGs

The structure of knowledge stored in Knowledge Graphs (KGs) plays an indispensable role in a multitude of practical applications. The current methodologies employed in Knowledge Graphs are inadequate for addressing the challenges posed by incomplete KGs and the processing of text corpora for the construction of KGs. Due to the generalizability of LLMs, numerous researchers are endeavoring to leverage the capabilities of LLMs for the resolution of KG-related tasks. Researchers have sought to leverage LLMs for processing the textual corpora within KGs, with the objective of enhancing the representation of KGs through the utilization of text-derived representations. Additionally, some studies employ LLMs to process the original corpus and extract relationships and entities for KG construction. Recently, studies have sought to develop a KG prompt that can effectively convert structural KGs into a format comprehensible to LLMs. This enables direct application of LLMs to KG-related tasks, such as KG completion and KG reasoning.

3.2.1 LLM-augmented KG Embedding

The objective of *Knowledge Graph Embedding* (*KGE*) is to map each entity and relationship into a low-dimensional vector (embedding) space. These embeddings contain both semantic and structural information of KGs, which can be exploited for a variety of tasks, including question answering (Huang *et al.* [2019]), reasoning (Lin *et al.* [2019]), and recommendation (Wang *et al.* [2018]).

Conventional Knowledge Graph embedding methods, such as TransE (Bordes *et al.* [2013]) and DisMult (Yang *et al.* [2014]), rely on the structural information of Knowledge Graphs to optimize a scoring function defined on embeddings. However, these approaches often fail to adequately represent unseen entities and long-tailed relationships due to their limited structural connectivity (Xiong *et al.* [2018], Wang *et al.* [2019a]). To address this issue, recent researches have employed Large Language Models to enrich the representations of Knowledge Graphs by encoding the textual descriptions of entities and relationships (Zhang *et al.* [2020b], Wang *et al.* [2021]).

Instead of employing the KGE model to discover graph structure, an alternative methodology has emerged that directly employs LLMs to integrate both graph structure and textual information into the embedding space concurrently (Xie *et al.* [2022], Wang *et al.* [2023]). This approach treats the entities and relationships as distinct tokens within the LLM. During the training phase, each triple and its corresponding text description are transformed into a sentence, wherein the tailed entities are replaced by a mask token. This sentence is then fed into an LLM, which subsequently refines the model to predict the masked entity.

3.2.2 LLM-augmented KG Completion

Knowledge Graph Completion (*KGC*) refers to the task of inferring missing facts in a given Knowledge Graph. Similar to KGE, traditional KGC methods have mainly focused on the structure of the graph without considering the rich textual information. However, the recent integration of LLMs enables KGC methods to encode text or generate facts for better KGC performance. These methods fall into two different categories based on their usage styles:

- *LLM as encoders*;
- *LLM as generators*.

The first category, named "LLM as encoders", employs encoder-only LLMs to encode textual information and Knowledge Graph facts. Subsequently, it predicts the plausibility of triples or masked entities by feeding the encoded representation into a prediction head, which may be a simple Multi-Layer Perceptron (MLP) or a conventional KG score function (e.g., TransE, proposed by Bordes *et al.* [2013], and TransR, proposed by Lin *et al.* [2015]).

The second category, named "LLM as generators", encompasses encoder-decoder or decoder-only LLM models. The LLMs receive a sequence text input of the query triple *(h, r, ?)*, and generate the text of the tail entity *t* directly. In the case of closed-source LLMs (e.g., ChatGPT and GPT-4), AutoKG employs prompt engineering to design customized prompts (Zhu *et al.* [2024]). These prompts contain the task description, few-shot examples, and the test input, which instruct the LLMs to predict the tail entity for KG completion.

3.2.3 LLM-augmented KG Construction

The construction of a Knowledge Graph entails the creation of a structured representation of knowledge within a particular domain. This includes the identification of entities and the relationships between them. The process of Knowledge Graph construction typically involves multiple stages, including *entity discovery*, *coreference resolution*, and *relation extraction*.

Recent research has investigated the construction of comprehensive Knowledge Graphs in a single step, a process known as *end-to-end Knowledge Graph construction*. Kumar *et al.* [2020] propose a unified approach to build Knowledge Graphs from raw text, which contains two Large Language Models powered components. First, the LLM is fine-tuned on named entity recognition tasks to enhance its ability to identify entities within raw text. Subsequently, a second model is proposed for addressing the relation extraction task, comprising two BERT-based classifiers. The initial classifier is tasked with learning the relationship class, while the second binary classifier is responsible for discerning the direction of relationships between entities. The predicted triples and relationships are subsequently utilized to construct the KG.

An alternative methodology is to directly *distill Knowledge Graphs from LLMs*. As it was shown, LLM systems are capable of implicit knowledge encoding on a vast scale (Petroni *et al.* [2019]). In light of this, Bosselut *et al.* [2019] have proposed a commonsense transformer model, named COMET, that constructs commonsense KGs by using existing tuples as a seed set of knowledge on which to train. By leveraging this seed set, an LLM is able to adapt its learned representations for knowledge generation, thereby producing novel tuples of high quality. The experimental results demonstrate that implicit knowledge from LLMs is transferred to generate explicit knowledge in commonsense KGs.

3.2.4 LLM-augmented KG-to-text Generation

The objective of *Knowledge Graph-to-text generation* (*KG-to-text generation*) is to produce high-quality texts that accurately and consistently describe the input Knowledge Graph

information (Gardent *et al.* [2017]). The process of KG-to-text generation links Knowledge Graphs and texts, thereby enhancing the applicability of Knowledge Graphs in a broader range of realistic Natural Language Generation scenarios, including storytelling (Guan *et al.* [2019]) and knowledge-grounded dialogue (Zhou *et al.* [2018]). However, the collection of large amounts of graph-text parallel data is a challenging and costly undertaking, which has resulted in insufficient training and a lack of quality in the generation process. Consequently, many research efforts have sought to address this challenge by either *leveraging knowledge from LLMs* or *constructing large-scale weakly-supervised KG-text corpus*.

In the first case, pioneering research efforts in utilizing LLMs for KG-to-Text generation are exemplified by the work of Ribeiro *et al.* [2020] and Kale and Rastogi [2020]. These studies directly fine-tune a range of LLMs, including BART and T5, with the objective of transferring the knowledge base of LLMs for this specific task. As illustrated in Figure 3.3, they represent the input graph as a linear traversal and demonstrate that this straightforward approach outperforms numerous existing state-of-the-art KG-to-text generation systems.

In the second case, although LLM systems have achieved remarkable empirical success, their unsupervised pre-training objectives are not necessarily aligned well with the task of KG-to-text generation. This has motivated researchers to develop large-scale KG-text aligned corpora. In fact, Jin *et al.* [2020] put forth a proposal for a 1.3M unsupervised KG-to-graph training data set derived from Wikipedia. Similarly, Chen *et al.* [2020] have also put forth a proposal for a KG-grounded text corpus collected from the English Wikidump.

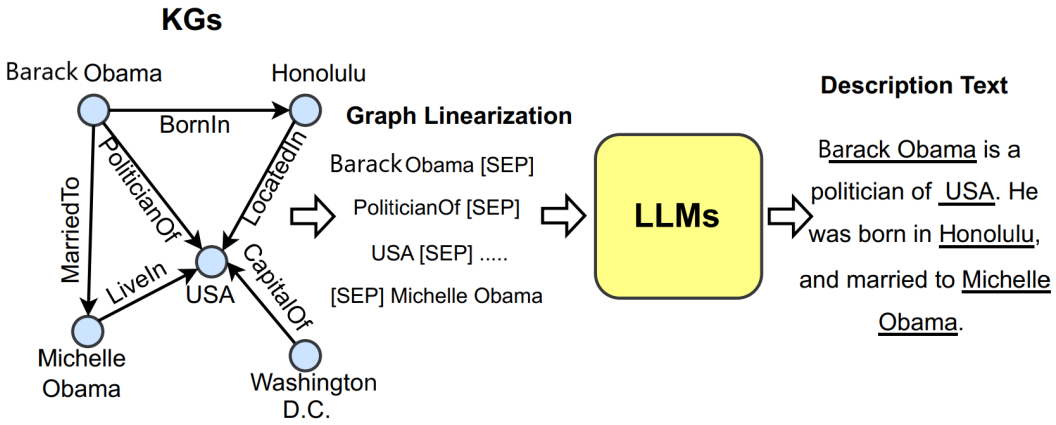


Figure 3.3: General framework of KG-to-text generation

3.2.5 LLM-augmented KG Question Answering

The objective of *Knowledge Graph Question Answering* (*KGQA*) is to identify responses to natural language queries based on the structured data stored in Knowledge Graphs (Saxena *et al.* [2020], Feng *et al.* [2020]). The inherent challenge in KGQA is to retrieve pertinent facts and leverage the reasoning capabilities of Knowledge Graphs in answering questions. Consequently, recent studies have employed LLMs to bridge the gap between natural language questions and structured Knowledge Graphs. In this approach, LLMs can be utilized as *entity/relationship extractors* and *answer reasoners*.

Entity/relationship extractors are designed to identify entities and relationships mentioned in natural language questions and retrieve related facts in Knowledge Graphs. LLMs' proficiency in language comprehension makes them an effective tool for this purpose. Lukovnikov *et al.* [2019] employ LLMs as classifiers for relation prediction, achieving notable performance improvements compared to shallow Neural Networks. Hu *et al.* [2023] introduce two LLM-

based KGQA frameworks that adopt LLMs to detect mentioned entities and relations. Then, they query the answer in KGs using the extracted entity-relation pairs. QA-GNN (Yasunaga *et al.* [2021]) employs LLM technology to encode question and candidate answer pairs, which are then utilized to assess the significance of relative KG entities. The identified entities are retrieved to construct a subgraph, where an answer reasoning process is conducted by a Graph Neural Network.

In contrast, *answer reasoners* are designed to reason over retrieved facts and generate answers. LLM models can be used as answer reasoners to generate answers directly. DEKCOR (Xu *et al.* [2020]) combines the retrieved facts with questions and potential responses, forming sentences, and then inputs them into LLMs to predict the scores for the responses. GreaseLM (Zhang *et al.* [2022b]) effectively combines representations from LLMs and Graph Neural Networks, enabling the reasoning over Knowledge Graph facts and language context. StructGPT (Jiang *et al.* [2023]) adopts a customized interface to allow Large Language Models (e.g., ChatGPT) directly reasoning on KGs to perform multi-step question answering.

3.3 Combination of synergized LLMs and KGs

The potential of combining LLM and KG technologies has garnered significant interest from the research community in recent years. LLMs and KGs are two inherently complementary techniques that should be unified into a general framework to mutually enhance each other. To further examine the potential for unification, a unified framework for the combination of synergized LLM and KG is illustrated in Figure 3.4. This comprises the following layers:

- *Data*;
- *Synergized Model*;
- *Technique*;
- *Application*.

In the data layer, LLMs and KGs are employed for the processing of textual and structural data, respectively. As multi-modal LLMs and KGs become more sophisticated, this framework can be extended to process multi-modal data, such as video, audio, and images. In the Synergized Model layer, LLMs and KGs can be combined to enhance their respective capabilities. In the Technique layer, techniques that have been utilized in LLMs and KGs can be incorporated into this framework to further improve performance. In the Application layer, LLMs and KGs can be integrated to address a range of real-world applications, including search engines, recommender systems, and AI assistants.

3.3.1 Synergized Knowledge Representation

Both text corpora and Knowledge Graphs are repositories of vast amounts of knowledge. However, the knowledge contained within text corpora is typically implicit and unstructured, whereas the knowledge present in Knowledge Graphs is explicit and structured. The objective of *Synergized Knowledge Representation* is to develop a synergized model that can effectively represent knowledge from both LLMs and KGs.

The synergized model offers a superior comprehension of the knowledge derived from both sources, making it invaluable for a multitude of subsequent tasks. To achieve joint representation of the knowledge, researchers have proposed the introduction of additional Knowledge Graph fusion modules, which are trained concurrently with LLMs. As illustrated

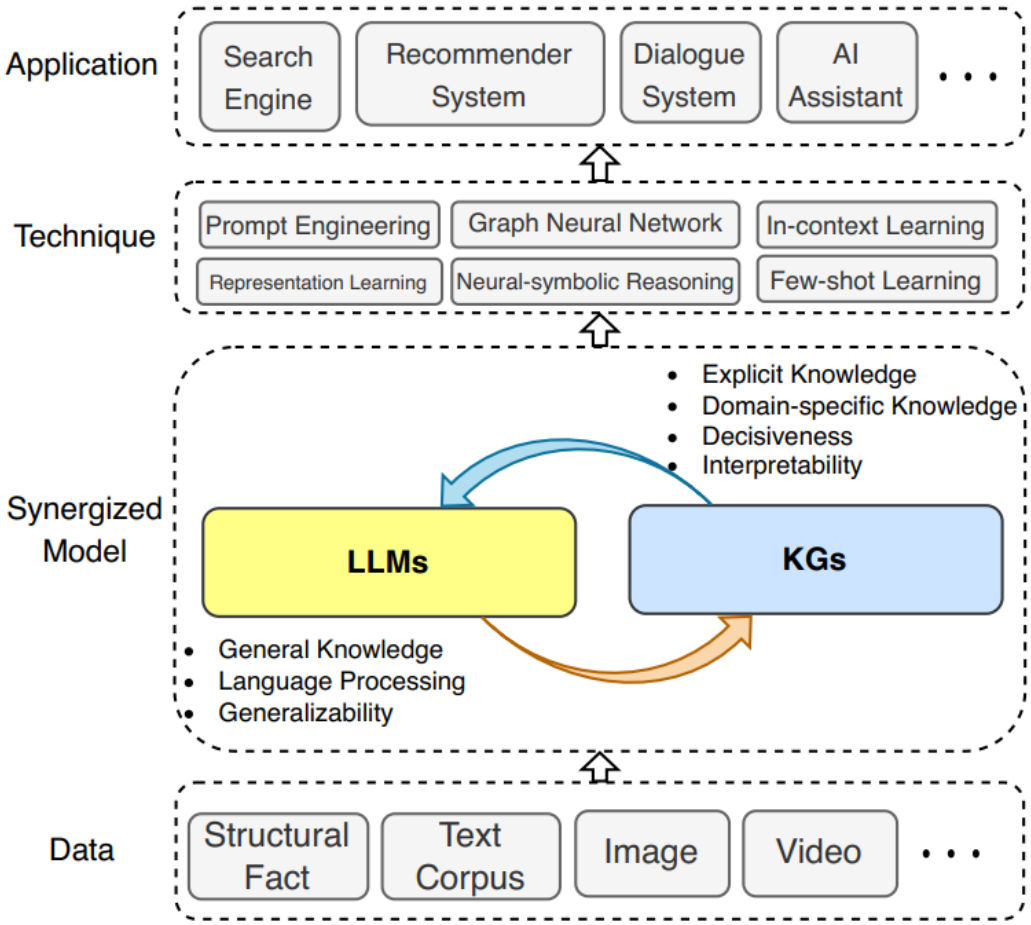


Figure 3.4: General framework for the combination of synergized LLM and KG

in Figure 3.5, ERNIE (Zhang *et al.* [2019]) proposes a textual-knowledge dual encoder architecture. In this approach, a T-encoder is first used to encode the input sentences, and then a K-encoder processes Knowledge Graphs, which are subsequently fused with the textual representation from the T-encoder. Furthermore, alternative methodologies, such as BERT-MK (He *et al.* [2019]), employ a comparable dual-encoder configuration, whereby supplementary data regarding proximate entities within the knowledge encoder module is incorporated during the pre-training of LLMs. Nevertheless, the relevance of some of these proximate entities within KGs may be questionable, potentially leading to superfluous redundancy and noise.

3.3.2 Synergized Reasoning

The objective of *synergized reasoning* is to develop a synergized model that can effectively conduct reasoning with both LLMs and KGs, thereby optimizing the utilization of knowledge from text corpus and Knowledge Graph reasoning. Two distinct approaches have been identified, which have been designated as *LLM-KG Fusion Reasoning* and *LLMs as Agents Reasoning*, respectively.

The first approach, designated as "LLM-KG Fusion Reasoning", employs two distinct LLM and KG encoders to facilitate the processing of textual data and pertinent KG inputs (Wang *et al.* [2019b]). The two encoders are of equal importance and jointly fuse the knowledge from the two sources for reasoning. To enhance the interaction between text and knowledge,

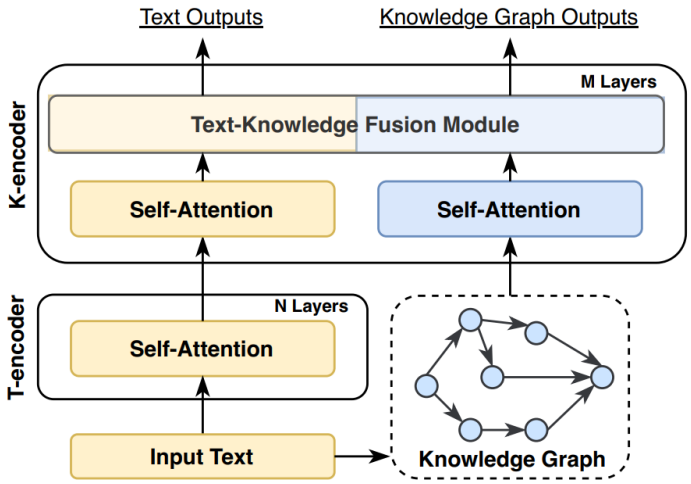


Figure 3.5: Synergized knowledge representation by additional KG fusion modules

Lin *et al.* [2019] propose first encoding the input KG and then augmenting the input textual representation. In contrast, Feng *et al.* [2020] employ the final LLM outputs of the input text to guide the reasoning process on the KGs. However, both approaches only facilitate a unidirectional interaction between the text and KGs. To address this challenge, a novel approach, named QA-GNN (Yasunaga *et al.* [2021]), is proposed, wherein a GNN-based model is employed to concurrently reason over input context and KG information via message passing. In particular, the input textual information is represented as a specialized node through a pooling operation and linked to other entities in the KG. However, the textual inputs are merely pooled into a single dense vector, which constrains the information fusion performance.

The second approach, designated as "LLMs as Agents Reasoning", employs two encoders to fuse the knowledge. Additionally, LLMs can be treated as agents to interact with the KGs to conduct reasoning (Liu *et al.* [2023]). A model, named KD-CoT (Wang *et al.* [2024]), that is proposed on an iterative basis is capable of retrieving facts from Knowledge Graphs and producing faithful reasoning traces, which then guide LLMs to generate answers. Another type of model, such as KSL (Feng *et al.* [2023]), is designed to teach LLMs to search on Knowledge Graphs to retrieve relevant facts and then generate answers. Finally, StructGPT (Jiang *et al.* [2023]) is designed to create several API interfaces that allow LLMs to access the structural data and perform reasoning by traversing KGs.

CHAPTER 4

Approach presentation

This chapter will introduce a novel approach designed to enhance an LLM's understanding of domain-specific knowledge by fine-tuning it with Knowledge Graph (KG) communities. The motivation behind this approach is based on the idea that KGs provide a structured and interconnected representation of knowledge, particularly within specialized fields. By leveraging these structured communities, our approach aims to increase the LLM's ability to process and generate information within specific domains. The chapter will start with a comprehensive examination of the foundational elements of the approach, encompassing the Minerva model and PrimeKG, a well-established biomedical Knowledge Graph. Then, the proposed methodology will be presented, which entails integrating the strengths of LLM with the rich, domain-specific knowledge embedded in KG communities. By fine-tuning the LLM on this structured data, our approach aims to markedly enhance the model's capability of reasoning and its performance within specialized domains, transcending the generalization limitations encountered by traditional LLM.

4.1 Preliminaries

This section introduces the two primary components of our research, i.e. *Minerva*, a Large Language Model, and *PrimeKG*, a Knowledge Graph based on medical data. These elements form the foundation of the study, enabling the generation of language-based insights and the organization of domain-specific medical knowledge.

Minerva is a state-of-the-art Language Model that employs sophisticated Natural Language Processing (NLP) techniques to interpret, generate, and manipulate human language on a large scale. Its ability to handle context-driven text generation enables advanced reasoning and language-based task execution, making it a crucial tool for achieving research objectives. In contrast, PrimeKG represents a Knowledge Graph that is firmly rooted in medical data. It captures intricate relationships between diseases, treatments, genes, and other biomedical entities, thereby providing a structured framework for the representation and reasoning about medical knowledge. By organizing this information into a connected graph, PrimeKG enhances the ability to retrieve pertinent medical facts and relationships in a precise and interpretable manner.

4.1.1 Minerva

In a groundbreaking advancement in the domain of Artificial Intelligence, a team of Italian researchers has unveiled *Minerva*, a series of Large Language Models (LLMs) that

have been entirely trained on the Italian language. The name "Minerva", derived from the Roman goddess of wisdom and loyalty, reflects the significant advancement in AI technology specifically designed for the Italian language.

The Minerva model, developed by the *Sapienza Natural Language Processing* research group led by Professor Roberto Navigli of the Department of Computer, Control, and Management Engineering "Antonio Ruberti" at Sapienza University of Rome, represents a significant advancement in AI technology. It is the first set of LLMs to be trained from scratch on Italian language content, a notable distinction from existing models that rely on adaptation from other languages.

The training process for Minerva was meticulous and comprehensive. The models were trained on a wide range of Italian and English sources, comprising a total of over 500 billion words, which is equivalent to more than 5 million novels. This extensive corpus was developed with the objective of providing a robust foundation for AI systems that can be applied in a variety of fields, including Natural Language Understanding, text generation and automated customer support. In this context, Professor Navigli stated:

> "We have worked tirelessly to create Minerva, the first set of Large Language Models entirely in Italian. This milestone represents a crucial step towards the technological autonomy of our country and the affirmation of Italian excellence in Artificial Intelligence."

The Minerva model is distinguished by a number of innovative features. Firstly, its modular architecture allows for the development of models of varying sizes and computational capabilities, thereby enabling the fulfillment of a broad spectrum of applications. This flexibility is essential to ensure the accessibility and adaptability of the technology to different contexts and use cases. Furthermore, Minerva is based on a "from scratch" training approach, without relying on pre-existing models in other languages. This approach allows for a more faithful capture of the peculiarities and nuances of the Italian language, thereby providing a solid foundation for the development of AI systems capable of understanding and interacting naturally with Italian users. Table 4.1 offers a comprehensive overview of the Minerva LLM family, highlighting the principal attributes of each model.

Model Name	Tokens	Layers	Hidden Size	Attention Heads	KV Heads	Sliding Window	Max Context Length
Minerva-350M-base-v1.0	70B (35B it + 35B en)	16	1152	16	4	2048	16384
Minerva-1B-base-v1.0	200B (100B it + 100B en)	16	2048	16	4	2048	16384
Minerva-3B-base-v1.0	660B (330B it + 330B en)	32	2560	32	8	2048	16384

Table 4.1: Minerva LLM family

The specific hyperparameters employed for training Minerva models are outlined in Table 4.2.

Minerva is notable for its transparency and interpretability, in addition to linguistic fidelity. Rather than functioning as "black boxes", Minerva models have been constructed to be as explanatory and understandable as possible, enabling experts to understand the decision-making process behind their predictions and outputs. This attribute is pivotal for fostering trust and acceptance of Artificial Intelligence, particularly in domains where transparency and accountability are paramount, such as healthcare, finance, and education.

Model Name	Optimizer	lr	betas	eps	Weight Decay	Scheduler	Warmup Steps	Batch size (Tokens)	Total Steps
Minerva-350M-base-v1.0	Decoupled AdamW	2e-4	(0.9, 0.95)	1e-8	0.0	Cosine	2%	4M	16690
Minerva-1B-base-v1.0	Decoupled AdamW	2e-4	(0.9, 0.95)	1e-8	0.0	Cosine	2%	4M	47684
Minerva-3B-base-v1.0	Decoupled AdamW	2e-4	(0.9, 0.95)	1e-8	0.0	Cosine	2%	4M	157357

Table 4.2: Hyperparameters employed for training Minerva models

Impact and applications

The influence of Minerva is pervasive, extending well beyond the domain of academic inquiry. This family of Large Language Models has the potential to transform numerous sectors, including Natural Language Understanding, text processing, content generation and automatic translation.

In the domain of customer service, for instance, Minerva could facilitate the development of chatbots and virtual assistants capable of interacting in a natural and contextual manner with Italian users, thereby enhancing the quality of support and the user experience.

In the field of education, Minerva could be utilized to develop adaptive learning tools that are capable of understanding students' needs and providing personalized content and feedback. Furthermore, Minerva models could be integrated into sentiment analysis and emotion detection systems to enhance user experience and communication effectiveness.

In the healthcare sector, Minerva has the potential to facilitate the development of medical language processing systems, thereby enhancing the comprehension of medical reports, the generation of diagnoses, and communication between healthcare professionals and patients. Similarly, in the financial sector, Minerva could be used for automatic document analysis, report generation and personalized consultancy.

The Sapienza NLP research team has engaged in active collaboration with academic institutions, research centers, and technology companies at the national and international levels. This has been done in order to ensure a holistic and multidisciplinary approach to research.

4.1.2 PrimeKG

The *Precision Medicine Knowledge Graph* (*PrimeKG*) is a Knowledge Graph developed by Chandak *et al.* [2023] that provides a comprehensive and multimodal perspective on diseases. The Knowledge Graph is built using 20 high-quality resources, biorepositories, and ontologies, integrated to construct the Knowledge Graph. They are illustrated in the list below:

- *Bgee gene expression knowledge base in animals* (Bastian *et al.* [2021]): It contains gene expression patterns across multiple animal species.
- *Comparative Toxicogenomics Database* (Davis *et al.* [2021]): It is focused on the impact of environmental exposures on human health.
- *DisGeNET knowledge base of gene-disease associations* (Piñero *et al.* [2020]): It is a resource about the relationships between genes and human disease that has been curated by experts.

- *Disease ontology* (Schriml *et al.* [2019]): It groups diseases in many meaningful clusters using clinically relevant characteristics. For instance, diseases are grouped by the anatomical entity.
- *DrugBank* (Wishart *et al.* [2018]): It is a resource that contains pharmaceutical knowledge.
- *Drug Central* (Avram *et al.* [2021]): It is a resource that curates information about drug-disease interactions.
- *Entrez Gene* (Maglott *et al.* [2010]): It is a resource maintained by the NCBI that contains vast amounts of gene-specific information.
- *Gene Ontology* (Consortium [2021]): It describes molecular functions, cellular components, and biological processes.
- *Human Phenotype Ontology* (Köhler *et al.* [2017]): It provides information on phenotypic abnormalities found in diseases.
- *Mayo Clinic*: It is a nonprofit academic medical center and biomedical research institution care. It maintains a knowledge base with information about symptoms, causes, risk factors, complications, and prevention of 2,227 diseases and conditions.
- *MONDO Disease Ontology* (Shefchek *et al.* [2020]): It harmonizes diseases from a wide range of ontologies, including the Online Mendelian Inheritance in Man (OMIM), SNOMED Clinical Terms (CT), International Classification of Diseases (ICD) and Medical Dictionary for Regulatory Activities (MedDRA).
- *Orphanet* (Weinreich *et al.* [2008]): It is a database that focuses on gathering knowledge about rare diseases; it has curated information about definitions, prevalence, management and treatment, epidemiology, and clinical description for 9,348 rare diseases.
- *Four sources of physical protein-protein interactions*: Protein-Protein Interactions (PPIs) consist of experimentally verified interactions between proteins. The interactions considered are diverse in nature and include signaling, regulatory, pathway, kinase-substrate, and protein complex interactions. The human PPI network compiled by Menche *et al.* [2015] is used as a starting resource.
- *Reactome pathway database* (Jassal *et al.* [2020]): It is an open-source, curated database for pathway-protein interactions.
- *Side Effect Resource* (Kuhn *et al.* [2016]): It contains data about adverse drug reactions.
- *Uberon multi-species anatomy ontology* (Mungall *et al.* [2012]): It is an ontology that contains information about the human anatomy.
- *Unified Medical Language System Knowledge Source* (Bodenreider [2004]): It contains information about biomedical and health-related concepts.

The data obtained by PrimeKG is displayed across 10 principal biological scales, including disease-related alterations in the proteome, biological processes, molecular pathways, anatomical and phenotypic scales, environmental exposures, and the scope of approved and experimental pharmaceuticals, together with their therapeutic effects (Figure 4.1).

PrimeKG enhances the coverage of diseases, both rare and common, by a factor of one to two compared to existing Knowledge Graphs. Furthermore, disease nodes in PrimeKG are densely connected to numerous other node types, including phenotypes, exposures, and

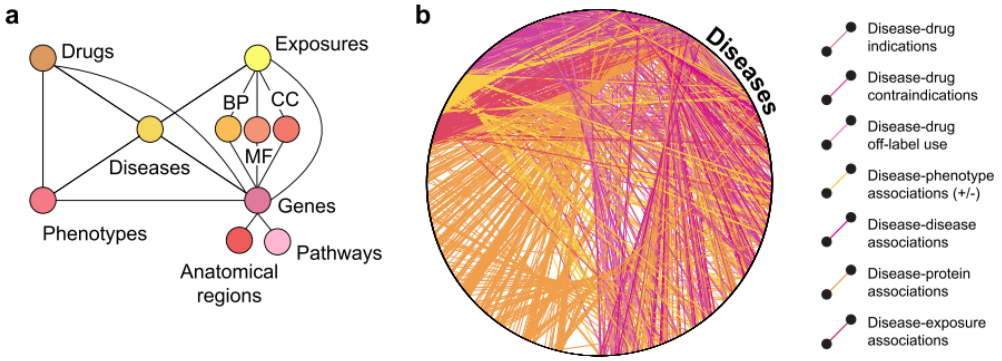


Figure 4.1: (a) The figure depicts a schematic overview of the various types of nodes in PrimeKG and the relationships they have with other nodes in the graph. (b) The figure shows all disease nodes in PrimeKG in a circular layout together with disease-associated information. It also depicts all relationships between disease nodes and any other node type. Disease nodes are densely connected to four other node types in PrimeKG through seven types of relations (Chandak *et al.* [2023]).

drugs. PrimeKG has been designed with the specific aim of supporting Artificial Intelligence analyses aimed at understanding how drugs target disease-associated molecular perturbations. This is achieved by including a wealth of drug-disease edges pertaining to indications, contradictions, and off-label use, which are typically absent or sparse in other Knowledge Graphs. The graph structure of PrimeKG is augmented by the addition of textual descriptions of clinical guidelines for drug and disease nodes. This enables the undertaking of multimodal analyses (Figure 4.2). Furthermore, the disease entity resolution challenge is addressed by improving the correspondence between diseases in PrimeKG and disease subtypes found in the clinic. This allows for PrimeKG-powered analyses in precision medicine.

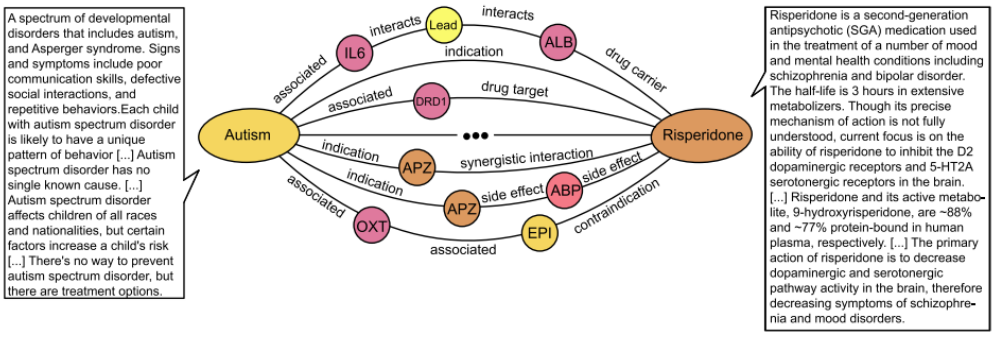


Figure 4.2: This illustration depicts a representative example of the pathways in PrimeKG between the disease node "Autism" and the drug node "Risperidone." It also displays excerpts from the text features for both nodes, showcasing the multimodality of PrimeKG. Abbreviations: MF: molecular function, BP: biological process, CC: cellular component, APZ: Apiprazole, EPI: epilepsy, ABP: abdominal pain, +/- associations: positive and negative associations (Chandak *et al.* [2023]).

Finally, PrimeKG is a multimodal Knowledge Graph comprising 10 types of nodes, 30 types of undirected edges, and natural language descriptions for disease and drug nodes. PrimeKG contains 129,375 nodes and 4,050,249 edges. Tables 4.3 and 4.4 offer a breakdown of the number of nodes by node type and the number of edges by edge type, respectively.

Node Type	Count	Percent (%)	Data Sources
Biological process	28,642	22.1	CTD, Entrez Gene, Gene Ontology
Protein	27,671	21.4	Bgee, CTD, DisGeNET, DrugBank, Entrez Gene, Human Phenotype Ontology, Human PPI Network, Reactome, UMLS
Disease	17,080	13.2	CTD, DisGeNET, Disease Ontology, Drug Central, Human Phenotype Ontology, Mayo Clinic, MONDO Disease Ontology, Orphanet
Phenotype	15,311	11.8	DisGeNET, Human Phenotype Ontology, SIDER
Anatomy	14,035	10.8	Bgee, UBERON
Molecular function	11,169	8.6	CTD, Entrez Gene, Gene Ontology
Drug	7,957	6.2	DrugBank, Drug Central, SIDER
Cellular component	4,176	3.2	CTD, Entrez Gene, Gene Ontology
Pathway	2,516	1.9	Reactome
Exposure	818	0.6	CTD
Total	129,375	100.0	20 primary data sources

Table 4.3: Information about nodes and node types in PrimeKG (Chandak *et al.* [2023])

Relation type	Count	Percent (%)
Anatomy - Protein (present)	3,036,406	37.5
Drug - Drug	2,672,628	33.0
Protein - Protein	642,150	7.9
Disease - Phenotype (positive)	300,634	3.7
Biological process - Protein	289,610	3.6
Cellular component - Protein	166,804	2.1
Disease - Protein	160,822	2.0
Molecular function - Protein	139,060	1.7
Drug - Phenotype	129,568	1.6
Biological process - Biological process	105,772	1.3
Pathway - Protein	85,292	1.1
Disease - Disease	64,388	0.8
Drug - Disease (contraindication)	61,350	0.8
Drug - Protein	51,306	0.6
Anatomy - Protein (absent)	39,774	0.5
Phenotype - Phenotype	37,472	0.5
Anatomy - Anatomy	28,064	0.3
Molecular function - Molecular function	27,148	0.3
Drug - Disease (indication)	18,776	0.2
Cellular component - Cellular component	9,690	0.1
Phenotype - Protein	6,660	0.1
Drug - Disease (off-label use)	5,136	0.1
Pathway - Pathway	5,070	0.1
Exposure - Disease	4,608	0.1
Exposure - Exposure	4,140	0.1
Exposure - Biological process	3,250	<0.1
Exposure - Protein	2,424	<0.1
Disease - Phenotype (negative)	2,386	<0.1
Exposure - Molecular function	90	<0.1
Exposure - Cellular component	20	<0.1
Total	8,100,498	100.0

Table 4.4: Statistics on edges in PrimeKG (Chandak *et al.* [2023])

4.2 Technical discussion of the proposed approach

The goal of our approach is to accelerate the fine-tuning of a Large Language Model (LLM) on a Knowledge Graph (KG) concerning a specific domain by using Complex Network

Analysis techniques. In particular, during the experimental campaign we adopted *Minerva 350m* as LLM and *PrimeKG* as Knowledge Graph. Our framework consists of two phases: the first generates the train and test datasets, whereas the second uses these datasets to fine-tune the LLM. The first phase is represented in Figure 4.3 and consists of the following steps: *(i)* extraction of the communities from the KG through the *Louvain algorithm* (Blondel *et al.* [2008]), *(ii)* computation of the random walks on the communities, *(iii)* generation of the training tuples.

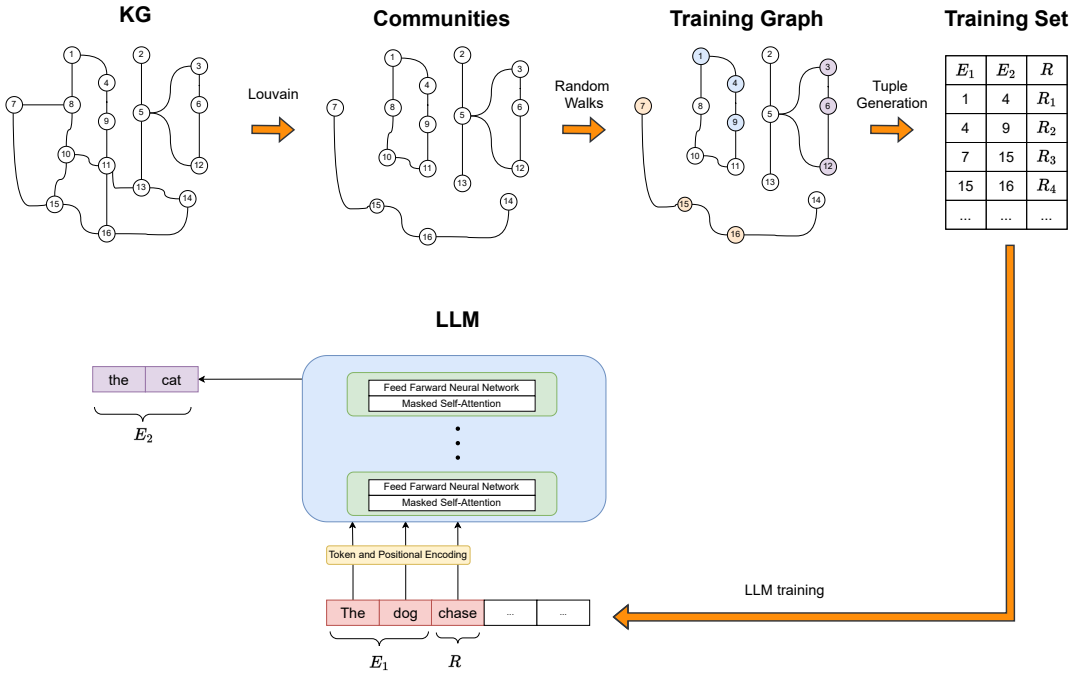


Figure 4.3: The workflow of the proposed approach

The first step of our framework is dedicated to constructing the training and testing datasets from the Knowledge Graph $\mathcal{G}$, which serve as the basis for fine-tuning the Large Language Model (LLM). This process is carried out in a series of methodical steps, each designed to extract and organize the relational structure of the KG for effective use in Machine Learning tasks.

Formally the Knowledge Graph is defined as a triple:

$$\mathcal{G} = \langle \mathcal{N}, \mathcal{E}, \mathcal{R} \rangle$$

where:

- $\mathcal{N}$ is the set of nodes in the graph. Each node $n \in \mathcal{N}$ represents an entity within the domain of the Knowledge Graph.
- $\mathcal{E}$ is the set of edges between nodes in the Knowledge Graph. Each edge $e_{i,j} \in \mathcal{E}$ represents a relationship from a node n_i to another node n_j. Formally, an edge $e_{i,j}$ is a pair (n_i, n_j), where $n_i, n_j \in \mathcal{N}$.
- $\mathcal{R}$ is the set of possible relationship types that define the semantic meaning of the connections between two nodes in $\mathcal{N}$. Each edge $e = (n_i, n_j) \in \mathcal{E}$ is labeled with a relationship $r \in \mathcal{R}$, indicating the type of relationship between n_i and n_j.

Given the Knowledge Graph $\mathcal{G}$, the goal of our framework is to transfer the knowledge contained in $\mathcal{G}$ into a Large Language Model (LLM). However, since $\mathcal{G}$ is very large, it is not feasible to fine-tune the LLM on the entire graph. Therefore, we train it on a subset of $\mathcal{G}$. Therefore, a key challenge is to identify an efficient method to select which portion of $\mathcal{G}$ should be used for training.

To achieve this, we divide $\mathcal{G}$ into communities, effectively partitioning its knowledge into smaller, more manageable components. By doing so, we can focus the training on specific portions of the graph, allowing for a more targeted and efficient fine-tuning of the LLM without needing to process the entire graph. Nodes within a community are typically entities that relate to the same topic or domain, which means the knowledge within each community is more cohesive.

To extract the communities from the Knowledge Graph we use the Louvain algorithm (Blondel *et al.* [2008]). It is a community detection technique that operates by maximizing the modularity of the graph; for this purpose, it measures the density of edges within communities as compared to edges between different communities. It begins by assigning each node in the graph to its own community. Then, iteratively, nodes are moved between communities in a way that increases the modularity. This local optimization step is followed by an aggregation one, where each community is treated as a single node, and the process is repeated until no further increase in modularity is possible. The result is a set of distinct communities, each representing a densely connected subgraph where nodes have strong internal links. This step reduces the overall complexity of $\mathcal{G}$ while preserving its modular structure, allowing us to focus on a meaningful substructure of the graph.

The next step is to perform random walks within each community. A random walk is a stochastic process where, starting from a given node, a sequence of nodes is visited based on the probabilities determined by the graph's structure. In our case, for each node, every neighboring node has an equal probability of being visited. By focusing the random walks on individual communities, we ensure that the model captures the local context and interaction patterns unique to each subgraph, which is crucial for generating training data that reflects the fine-grained structural properties of $\mathcal{G}$. The extraction of random walks from each community allows us to generate a dataset that contains a representative portion of the knowledge in $\mathcal{G}$, spanning multiple domains or sectors. With this approach we ensure that the LLM is fed with different but focused areas of knowledge of $\mathcal{G}$, optimizing the training process while preserving the essential structure and relationships of the original graph.

To ensure a good exploration of the communities, the corresponding random walks need to consider the communities' characteristics. For this reason, given the community i, the random walks length δ_i is equal to the diameter of the community, i.e. the longest shortest path between any two vertices in a graph. The number γ_i of random walks can be computed as:

$$\gamma_i = \frac{\delta_i}{\eta_i} \tag{4.2.1}$$

where η_i is the number of nodes in the community i.

With the sequences from the random walks in hand, the next task is to generate training tuples. A training tuple is represented as $T = (n_i, n_j, e_{i,j}, r_k)$ where $n_i, n_j \in \mathcal{N}$ is a couple of nodes, $e_{i,j} \in \mathcal{E}$ is the connecting edge and $r_k \in \mathcal{R}$ is the edge relation. Following our algorithm, at this stage these tuples encapsulate the structural and relational information within the communities, as they represent how a node is situated in relation to others within its local neighborhood. Let W_i be a random walk of the community i, we divide it into a set of tuples $\{T_1, T_2, \dots\}$. The training set $\mathcal{TS}$ can be defined as the union of all the sets of tuples obtained from the random walks for every community. Formally speaking:

$$\mathcal{TS} = \bigcup_{i=1}^{c} \bigcup_{j=1}^{\gamma_i} W_j \tag{4.2.2}$$

where c is the number of communities.

Since different random walks can share tuples, the cardinality of the training set is influenced by the number of duplicates, as shown in the following formula:

$$|\mathcal{TS}| \leq \sum_{i=1}^{c} \sum_{j=1}^{\gamma_i} |T_j| \tag{4.2.3}$$

After the knowledge extraction phase from the Knowledge Graph $\mathcal{G}$ has been terminated, we proceed to fine-tune a pre-trained Large Language Model (LLM) using the training set generated from the random walks. The fine-tuning process allows the model to adapt to the domain-specific relationships and patterns found within the subgraphs of $\mathcal{G}$.

CHAPTER 5

Approach validation

This chapter presents a series of experiments designed to evaluate the effectiveness of the proposed approach. The Large Language Model (LLM) will be fine-tuned using a Knowledge Graph (KG) that has been subdivided into communities by means of the Louvain Algorithm. This algorithm will play a pivotal role in identifying community structures within the graph, offering an effective approach to organize the data into coherent clusters for fine-tuning. The experiments aim to assess the performance of the fine-tuned LLM using metrics such as BLEU and Sequence Matcher. The efficacy of the proposed approach will be validated through the performance of these experiments on three distinct graphs. Furthermore, a heuristic search is conducted to identify the optimal values for these metrics, modifying the resolution value of the Louvain algorithm.

5.1 Preliminaries

This section introduces the Louvain algorithm, which will be utilized in the upcoming experimental analysis. This algorithm is central to the community detection process, providing an efficient and scalable method for identifying community structures in networks.

Furthermore, the metrics adopted for the assessment of the LLM are presented and subjected to analysis. These metrics are BLEU and Sequence Matcher.

5.1.1 Louvain Community Detection Algorithm

Louvain Community Detection Algorithm is a simple method to extract the community structure of a network. This is a heuristic method based on modularity optimization (Blondel *et al.* [2008]) and it is available from the Python library *NetworkX*.

The algorithm consists of two distinct steps. Initially, it assigns each node to a community of its own. Subsequently, for each node, it tries to identify the greatest positive modularity gain by relocating it to all of its neighboring communities. If no positive gain is attained, the node remains in its original community. The modularity gain obtained by moving an isolated node i into a community C can easily be calculated by the following formula (combining Blondel *et al.* [2008], Traag *et al.* [2019] and some algebra):

$$\Delta Q = \frac{k_{i,in}}{2m} - \gamma \frac{\Sigma_{tot} \cdot k_i}{2m^2} \tag{5.1.1}$$

Here, m is the size of the graph, $k_{i,in}$ is the sum of the weights of the links from i to nodes in C, k_i is the sum of the weights of the links incident to node i, Σ_{tot} is the sum of the weights

of the links incident to nodes in C and γ is the resolution parameter.

For the directed case the modularity gain can be computed using the following formula (according to Dugué and Perez [2015]):

$$\Delta Q = \frac{k_{i,in}}{m} - \gamma \frac{k_i^{out} \cdot \Sigma_{tot}^{in} + k_i^{in} \cdot \Sigma_{tot}^{out}}{m^2} \tag{5.1.2}$$

Here, k_i^{out}, k_i^{in} are the outer and inner weighted degrees of node i and Σ_{tot}^{in}, Σ_{tot}^{out} are the sums of in-going and out-going links incident to nodes in C.

The first phase persists until the modularity of the system is unable to be enhanced by any further individual moves.

The second phase entails the construction of a new network, whose nodes are the communities identified in the initial phase. To this end, the weights of the links between the new nodes are determined by the sum of the weights of the links between nodes in the corresponding two communities. Upon the completion of this phase, it becomes feasible to reapply the first phase, thereby forming larger communities with incremented modularity.

The aforementioned two phases are executed until no further modularity gain is achieved, or the *threshold* is reached, or *max_levels* is reached.

The following list enumerates the parameters of this algorithm:

- *G*: A NetworkX graph.
- *weight*: String or None, optional (default=”weight”); it represents the name of an edge attribute that holds the numerical value used as a weight. If its value is designated as "None", the weight assigned to each edge is 1.
- *resolution*: Float, optional (default=1); if it is less than 1, the algorithm favors larger communities. Conversely, if it is greater than 1, the algorithm favors smaller communities.
- *threshold*: Float, optional (default=0.0000001); it represents the modularity gain threshold for each level. If the modularity gain between 2 levels of the algorithm is less than the given threshold then the algorithm stops and returns the resulting communities.
- *max_level*: Int or None, optional (default=None); it represents the maximum number of levels (steps of the algorithm) to compute. It must be a positive integer or "None". If its value is designated as "None", then there is no maximum level. Rather, the threshold parameter serves to determine the stopping condition.
- *seed*: Int, random_state, or None (default); it represents the indicator of random number generation state.

The following list provides a comprehensive enumeration of the data yielded by the algorithm:

- *list*: It is a list of sets, which form a partition of *G*. Each set represents a single community and contains all of the nodes comprising that community.

5.1.2 Metrics

In this section, we present the metrics used to evaluate the performance of our approach. Specifically, we focus on two key metrics, namely BLEU,and Sequence Matcher. Each of these metrics offers a distinct perspective on the model’s effectiveness, enabling a comprehensive assessment of its language generation capabilities. BLEU measures the accuracy of generated text by comparing it to reference texts and Sequence Matcher quantifies the similarity between sequences.

BLEU

BLEU (*BiLingual Evaluation Understudy*) is a metric employed to assess the quality of machine-generated translations in comparison to one or more reference translations. The BLEU score is a metric used to evaluate the quality of machine-generated translations. It measures the similarity between the machine-generated translation and the reference translations based on the *n-grams* (contiguous sequences of n words) present in both. The BLEU score ranges from 0 to 1, with a higher score indicating a better match between the generated translation and the references. A score of 1 indicates a perfect match, while a score of 0 indicates no overlap between the generated and reference translations.

To determine the BLEU calculation formula, Papineni *et al.* [2002] first take the geometric mean of the test corpus' modified precision scores and then multiply the result by an exponential brevity penalty factor. Currently, case folding is the only text normalization procedure conducted prior to the calculation of precision. The geometric mean of the modified *n-gram* precision, p_n, is initially determined using *n-grams* up to length N and positive weights w_n, which collectively sum to one. Subsequently, let c denote the length of the candidate translation and r represent the effective reference corpus length.

The *brevity penalty* (*BP*) is calculated according to the following formula:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{5.1.3}$$

Accordingly, the BLEU metric is calculated as follows:

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{5.1.4}$$

The ranking behavior is more readily discernible within the log domain, as follows:

$$\log \mathrm{BLEU} = \min\left(1 - \frac{r}{c}, 0\right) + \sum_{n=1}^{N} w_n \log p_n \tag{5.1.5}$$

In Papineni *et al.* [2002] baseline, the value of N is set to 4 and *uniform weights* (w_n) are used with a value of $1/N$.

In the context of our project, the BLEU metric was adapted for comparing two sequences of characters. These sequences are formed by the dataset's triple i and the related generated triple produced by the model.

Sequence Matcher

The Python *Sequence Matcher* algorithm (Munk and Feitelson [2022]), which is part of the *difflib* library, has been designed with the intention of more closely aligning with human intuitive matching processes. Its effectiveness has been demonstrated in the context of naming research by Etgar *et al.* [2022]. It has also been asserted that this approach is advantageous for comparing sketch-based passwords (Amarnadh *et al.* [2021]). The Ratcliff-Obershelp algorithm, which was designed in the context of educational software, is based on the premise that it should be "forgiving and understanding of simple typing mistakes, and allow intelligent responses to erroneous input" (Ratcliff *et al.* [1988]). The underlying concept is to prioritize matching contiguous substrings over matching subsequences of disjoint letters. The algorithmic approach entails initially identifying the longest contiguous substring within the two input strings and then proceeding recursively on both sides of this substring. For instance, in the example presented in Figure 5.1, the longest common substring is "myster", and the

recursive calls yield no results on the left but a space and "ch" on the right, resulting in a total of nine matching letters. The final score is the relative length of the resulting subsequence of identical letters.

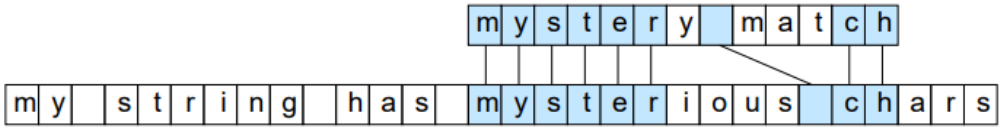


Figure 5.1: Example of the Sequence Matcher operation (Munk and Feitelson [2022])

5.2 A technical examination of the approach validation process

The objective of our approach is to fine-tune an LLM with a subpart of a Knowledge Graph. As discussed in Chapter 4, we adopted Minerva 350m as the LLM and PrimeKG as the Knowledge Graph.

Firstly, due to the considerable size of PrimeKG and the cost of high-performance computing resources, we decided to extract a subset of the initial graph. This approach allows us to conduct a greater number of experiments with diverse initial graphs and to evaluate the proposed methodology in a range of scenarios.

In this study, we selected three discrete samples from the initial Knowledge Graph for conducting experiments. The three subgraphs chosen have no node or edge overlap. Their main characteristics are presented in Table 5.1.

#KG	#Nodes	#Edges
1	5745	1357680
2	26122	177905
3	35597	94476

Table 5.1: Main characteristics of the communities selected as the initial Knowledge Graphs for the experiments

The initial step for each Knowledge Graph is to divide it into communities using the *Louvain Algorithm*. In order to ensure the reproducibility of the experiments, the parameter *seed* was set to 42.

Once the Louvain algorithm has been applied and the Knowledge Graph partitioned into communities, it is essential to construct the train and test datasets. To achieve this, we sample N random paths of length M from each community, where N represents the diameter of the community, and M is the ratio of the number of community nodes to community diameter. Consequently, each random path consists of entities connected by relationships. For each relationship, a triple is formed, comprising the labels of the starting entity, the relationship itself and the ending entity. All the triples obtained are used for the training and validation datasets, while the test set is obtained with the triples extracted from other random paths, conducted in the starting Knowledge Graph, without its subdivision into communities.

It is evident that the training set, the validation set, and the test set are mutually exclusive. In general, the percentage distribution of the three sets is approximately 74%, 8%, and 18%, respectively.

Below, we present some examples of the triples that are present within the dataset:

```
1) 'KRTAP4-4', 'ppi', 'LCE3B'
```

```
2) 'regulation of adenylate cyclase activity', 'interacts
   with', 'CAP2'
3) 'neuron development involved in amphid sensory organ
   development', 'parent-child', 'neuron development'
4) 'CALM2', 'interacts with', 'calcium channel inhibitor
   activity'
5) 'intestine', 'expression present', 'PRKCA-AS1'
```

From an analysis of the dataset's triples previously presented, it is evident that the straightforward concatenation of the three terms does not result in a syntactically well-formed phrase. Accordingly, we have chosen to create templates with the objective of constructing the phrase in a more optimal manner. Below, we present some of these templates, one for each type of relationship represented in the Knowledge Graph:

```
1) 'associated with': "{entity1} is associated with {entity2}"
2) 'carrier': "{entity1} acts as a carrier for {entity2}"
3) 'contraindication': "{entity1} is contraindicated in
   {entity2}"
4) 'enzyme': "{entity1} acts as an enzyme for {entity2}"
5) 'expression absent': "{entity1} does not express
   {entity2}"
6) 'expression present': "{entity1} expresses {entity2}"
7) 'indication': "{entity1} is indicated for treating
   {entity2}"
8) 'interacts with': "{entity1} interacts with {entity2}"
9) 'linked to': "{entity1} is linked to {entity2}"
10) 'off-label use': "{entity1} is used off-label for
    {entity2}"
11) 'parent-child': "{entity1} is a subtype of {entity2}"
12) 'phenotype absent': "{entity1} does not exhibit the
    phenotype {entity2}"
13) 'phenotype present': "{entity1} exhibits the phenotype
    {entity2}"
14) 'ppi': "{entity1} has a protein-protein interaction with
    {entity2}"
15) 'side effect': "{entity1} may cause the side effect
    {entity2}"
16) 'synergistic interaction': "{entity1} has a synergistic
    interaction with {entity2}"
17) 'target': "{entity1} targets {entity2}"
18) 'transporter': "{entity1} transports {entity2}"
```

Moreover, for each triple in the dataset, we provide a *prompt* and a *completion*, which will be employed in the training process.

The *prompt* is constructed in the following manner: *"Complete the relationship: {r}"*. Here, *{r}* is the phrase derived from the templates, with *entity1* represented by the first entity of the triple and *entity2* masked.

Instead, the *completion* is constructed in the following manner: *"{r}"*. Here, *{r}* is the phrase derived from the templates, with *entity1* represented by the first entity of the triple and *entity2* represented by the second entity of the triple.

For the sake of completeness, given that the initial Knowledge Graph is undirected, we decided to replicate the prompts and the completions, reversing each triple within the dataset.

This strategy enables us to mask the first entity in each triple of the dataset, instead of the second one, effectively reversing their order in the relationship.

Ultimately, each triple of the dataset is replaced with the prompt and the completion associated with it. The number of data points included in the training, validation, and test sets for each experiment is presented in Table 5.2.

#Experiment	#Train set	#Validation set	#Test set
1	44000	6562	11006
2	38000	3890	9004
3	24000	2720	6002

Table 5.2: Number of data points included in the training, validation, and test sets for each experiment

Before fine-tuning the LLM, it was essential to evaluate the models themselves. To this end, BLEU and Sequence Matcher have been exploited as metrics. So, for each triple of the test datasets, the corresponding prompt is provided as input to the model, and the model's output, representing the predicted completion, is obtained. In order to obtain the predicted completion, the original one was tokenized and the model was instructed to generate an equivalent number of tokens. Afterwords, the BLEU and Sequence Matcher metrics were calculated as differences between the original completion and the predicted one.

The mean values of the metrics obtained from the models on the test sets prior to fine-tuning are presented in Table 5.3.

#Experiment	Sequence Matcher (%)	BLEU (%)
1	54.0	42.5
2	42.8	31.6
3	47.2	29.8

Table 5.3: Mean values of the metrics obtained from the Minerva 350m model on the test sets before fine-tuning

Following the calculation of the metrics on the LLM, we proceeded to fine-tune it using the tokenized prompts and completions present in the training datasets obtained from the three Knowledge Graphs. We used the validation datasets to verify the training performance and prevent overfitting. Upon completion of this process, three distinct, fine-tuned models were generated, one for each training set, and consequently for each Knowledge Graph or experiment.

The main *TrainingArguments* employed in the *Transformers Trainer* are described in the following list:

- *per_device_train_batch_size*: It represents the batch size per GPU, XPU, TPU, MPS, NPU core or CPU exploited for training purposes. It was set to 8.
- *eval_strategy*: It represents the evaluation strategy to adopt during training. Possible values are "no" (no evaluation is done during training), "steps" (evaluation is done and logged every *eval_steps*) and "epoch" (evaluation is done at the end of each epoch). It was set to "steps".
- *do_eval*: It allows the user to determine whether the evaluation should be conducted on the validation set. It was set to True.
- *load_best_model_at_end*: It indicates whether the best model identified during training should be loaded at the conclusion of the training process. It was set to True.

- *metric_for_best_model*: It must be used in conjunction with *load_best_model_at_end* to specify the metric to be utilized for the comparison of two distinct models. It was set to "eval_loss".
- *greater_is_better*: It should be used in conjunction with *load_best_model_at_end* and *metric_for_best_model* in order to specify whether or not better models should have a greater metric. It was set to False.
- *prediction_loss_only*: It is used to indicate whether the evaluation and generation of predictions should be performed with the sole return of the loss. It was set to True.
- *num_train_epochs*: It represents the total number of training epochs to be performed. It was set to 10.
- *warmup_steps*: It represents the number of steps used for a linear warmup from 0 to the learning rate. It was set to 50.
- *save_steps*: It specifies the number of updates to be performed before two checkpoint saves are initiated when the *save_strategy* is set to "steps". It was set to 500.
- *save_total_limit*: It will limit the total amount of checkpoints. It was set to 3.
- *fp16*: It indicates whether the 16-bit (mixed) precision training should be employed, instead of the 32-bit training. It was set to True.
- *logging_steps*: It represents the number of update steps between two logs. It was set to 250.

Furthermore, the early stopping technique was employed with a patience setting of 3 steps, as this enables a further reduction in the computational resources required and prevents overfitting.

Following the fine-tuning of the models, the same two metrics (BLEU and Sequence Matcher) were recalculated for each experiment in the same manner as before, using the same test set employed before the fine-tuning process.

The results obtained performing the fine-tuning of the models using the identical test sets referenced in Table 5.3 are presented in Table 5.4.

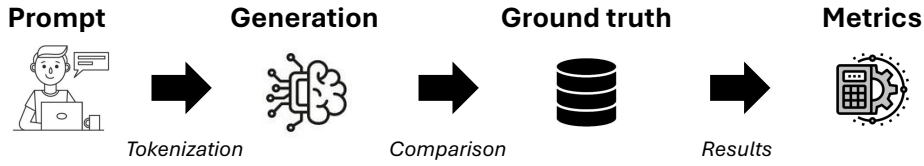
#Experiment	Sequence Matcher (%)	BLEU (%)
1	77.4	72.8
2	69.0	62.2
3	81.1	76.2

Table 5.4: Mean values of the metrics obtained from the Minerva 350m model on the test sets after fine-tuning

As evidenced by Tables 5.3 and 5.4, the employed metrics have increased in the post-fine-tuning scenario. These enhancements have been observed in the Sequence Matcher, with a 23.4%, 26.2%, and 33.9% increase in the first, second, and third experiments, respectively. Similarly, improvements in BLEU have been noted, with a 30.3%, 30.6%, and 46.4% increase in the first, second, and third experiments, respectively.

Figure 5.2 depicts the model's pipeline from the prompt to the generation. Furthermore, it illustrates a multitude of instances of the fine-tuned model's responses to prompts and the resulting outcomes in terms of Sequence Matcher and BLEU metrics. In light of the observation that the fine-tuning process yields advantageous outcomes in terms of model

performance, as evidenced by the metric values before and after it, the aim of this study was to illustrate that the proposed approach returns superior performance compared to the baseline scenario. This baseline scenario is represented by fine-tuning the model using the Knowledge Graph's random paths taken without the communities subdivision of the Knowledge Graph itself. Table 5.5, 5.6 and 5.7 illustrate the post-fine-tuning results obtained by applying the proposed approach and not applying it to the three Knowledge Graphs under consideration. It should be noted that, for each of the aforementioned tables, the experiments presented are conducted on a dataset comprising an identical number of data points for the training, validation, and test sets. Moreover, for each table, the test set employed is the same, ensuring that the experiments themselves are comparable.



"Complete the relationship: 'Edg-7 lysophosphatidica acid receptor binding is a subtype of '"	"Edg-7 lysophosphatidic acid receptor binding is a subtype of endothelial differentiation G protein-coupled receptor binding"	"Edg-7 lysophosphatidic acid receptor binding is a subtype of endothelial differentiation G protein-coupled receptor binding"	Sequence Matcher = 1.0 BLEU = 1.0
"Complete the relationship: 'regulation of testosterone biosynthetic process is a subtype of '"	"regulation of testosterone biosynthetic process is a subtype of regulation of steroid biosynthetic process"	"regulation of testosterone biosynthetic process is a subtype of regulation of ketone biosynthetic process"	Sequence Matcher = 0.94 BLEU = 0.91
"Complete the relationship: 'sporulation is a subtype of '"	"sporulation is a subtype of sporangiospore"	"sporulation is a subtype of developmental process"	Sequence Matcher = 0.60 BLEU = 0.56
"Complete the relationship: 'OR6N1 interacts with '"	"OR6N1 interacts with olfactory receptor activity, coupled via G protein-c"	"OR6N1 interacts with detection of chemical stimulus involved in sensory perception of smell"	Sequence Matcher = 0.43 BLEU = 0.33

Figure 5.2: Model's pipeline from the prompt to the generation, with illustrative examples

Experiment	Sequence Matcher (%)	BLEU (%)
Proposed approach	77.4	72.8
No communities subdivision	68.4	63.3

Table 5.5: Mean values of the metrics obtained from the Minerva 350m model on the test set after fine-tuning in the Experiment 1

Table 5.5 shows that the experiment conducted on the first Knowledge Graph reveals a 9% increase in Sequence Matcher and a 9.5% increase in BLEU when the proposed approach is employed, in comparison to the scenario where no community subdivision is present.

Experiment	Sequence Matcher (%)	BLEU (%)
Proposed approach	69.0	62.2
No communities subdivision	67.7	60.6

Table 5.6: Mean values of the metrics obtained from the Minerva 350m model on the test set after fine-tuning in the Experiment 2

Table 5.6 shows that the experiment conducted on the second Knowledge Graph reveals a

1.3% increase in Sequence Matcher and a 1.6% increase in BLEU when the proposed approach is employed, in comparison to the scenario where no community subdivision is present.

Experiment	Sequence Matcher (%)	BLEU (%)
Proposed approach	81.1	76.2
No communities subdivision	78.0	72.6

Table 5.7: Mean values of the metrics obtained from the Minerva 350m model on the test set after fine-tuning in the Experiment 3

Finally, Table 5.7 shows that the experiment conducted on the third Knowledge Graph reveals a 3.1% increase in Sequence Matcher and a 3.6% increase in BLEU when the proposed approach is employed, in comparison to the scenario where no community subdivision is present.

Therefore, after showing that the LLM's performance after fine-tuning is superior in terms of both Sequence Matcher and BLEU, it can be concluded that the proposed approach is effective and robust when compared to the baseline scenario, which involved random paths taken in a random way without subdividing the Knowledge Graph into communities. This is evidenced by the fact that the proposed approach was evaluated using three distinct Knowledge Graphs.

After extensive deliberation, we decided to modify the resolution parameter of the Louvain algorithm with the objective of achieving a more optimal subdivision of the Knowledge Graph into communities. It is noteworthy that the resolution parameter is, by default, set to 1, but it can be modified to influence the way in which communities are identified. This specific type of experiment was conducted on the largest of the three Knowledge Graphs employed. The results are presented in Table 5.8. Here, r represents the resolution parameter of the Louvain algorithm.

Experiment	Sequence Matcher (%)	BLEU (%)
No communities subdivision	68.4	63.3
Proposed approach (r=0.5)	76.4	71.4
Proposed approach (r=1)	77.4	72.8
Proposed approach (r=5)	71.2	65.6
Proposed approach (r=10)	79.4	74.9
Proposed approach (r=15)	73.4	68.3
Proposed approach (r=50)	73.3	68.4
Proposed approach (r=100)	74.5	69.4

Table 5.8: Mean values of the metrics obtained from the Minerva 350m model on the test set after fine-tuning in the Experiment 3, with the resolution parameter undergoing variation

As evidenced in Table 5.8, there is no discernible trend in the results when the resolution parameter is varied. Nevertheless, an improved subdivision into communities can be identified, which is reflected in higher results and consequently superior performance of the fine-tuned model. In our case, the optimal values are obtained by setting the resolution parameter to 10, resulting in a Sequence Matcher value of 79.4% and a BLEU value of 74.9%, which represents an increase of 2% and 2.1%, respectively, from the default case values.

In conclusion, the results presented in this chapter demonstrate the effectiveness of the proposed approach. The latter has been tested on three Knowledge Graphs, and the results have been positive in each case. These outcomes include an increase in the values of the adopted metrics and an enhancement in performance when the proposed approach is used, as opposed to the baseline case, which does not include the Knowledge Graph communities' subdivision.

CHAPTER 6

Discussion

This chapter presents the implications of our research, which introduces a novel methodology for integrating Knowledge Graphs (KGs) with Large Language Models (LLMs) in a computationally efficient manner. By employing community detection algorithms, particularly the Louvain algorithm, the proposed approach enables selective fine-tuning of LLMs using subsets of a KG. This stratified approach reduces the computational load by limiting the amount of data processed, while still ensuring that the model gains meaningful knowledge from the KG. The study demonstrates that the extraction of random paths from communities allows for the creation of a high-quality, domain-specific dataset for fine-tuning, while circumventing the necessity to process the entire KG. This innovation addresses the resource-intensive nature of LLM fine-tuning and provides a scalable solution for integrating large, structured datasets into Deep Learning models. Furthermore, this research has practical, theoretical, and societal implications. These include improvements in training efficiency, advancements in Knowledge Graph usage, the role of subgraph sampling in future AI applications, and considerations for fair and ethical use of knowledge in AI systems. The results presented here open new pathways for future research and innovation in the field of Deep Learning and knowledge representation.

6.1 Discussion of the proposed approach

The proposed approach aimed to integrate Large Language Models and Knowledge Graphs by means of a KG-enhanced LLM approach. The time and computational resources required for training or fine-tuning an LLM are considerable. Therefore, the objective was to enhance the performance of an LLM via a KG constructed by a specific area, identifying a framework for reducing the time and computational resources required for the fine-tuning process.

The proposed framework facilitates the fine-tuning of an LLM via a KG comprising information about a specific industry in an entirely offline manner, whereby the optimal subpart of the KG is selected for the fine-tuning process. Furthermore, it provides the generated dataset for fine-tuning other models at the user's discretion, while simultaneously reducing the computational resources required for such a task.

The optimal subpart of the KG for the fine-tuning process is obtained through the use of a community detection algorithm, specifically the Louvain algorithm. This approach ensures that the KG is divided into communities that share arguments, concepts, or insights. It is possible to extract knowledge from each of these communities in a stratified manner, ensuring

that no main concepts are discarded. This has been made possible by taking many random paths for each community, as previously described in detail in the previous chapters.

For each random path, the triples comprising it are extracted. Each triple is formed by a pair of entities and the relationship between them.

For each triple, a prompt and a completion are written. The completion is the triple itself, rewritten in a syntactically correct manner. The prompt is the input provided to the LLM, which requires the relative completion as the LLM's answer.

Next, the resulting dataset, comprising prompts and completions, was divided into three distinct sets, i.e., training set, validation set and test set. The LLM was then trained using the data from the training set. The validation set was used to calculate the validation loss throughout the training process and to implement the early stopping technique in order to prevent the wastage of computational resources and the onset of overfitting. Ultimately, the test dataset was used to evaluate the fine-tuned model. For this purpose, two metrics were employed, i.e., Sequence Matcher and BLEU.

To more accurately assess the proposed methodology, we conducted tests on three Knowledge Graphs. To this end, we extracted three graph samples from the original Knowledge Graph and used them for our experiments.

The results of all experiments were deemed to be satisfactory. Notable enhancement were observed with regard to the Sequence Matcher and BLEU. Initially, a notable increase in these metrics was evident following fine-tuning, when compared to the previous period. Moreover, a substantial enhancement was discerned when comparing the proposed methodology with the baseline scenario, in which no Knowledge Graph community subdivision was provided.

The favorable outcomes were achieved through the exploitation of all three Knowledge Graphs, which serves to illustrate the effectiveness of the methodology.

Ultimately, we conducted a heuristic investigation to identify the optimal value of the metrics utilized, varying the resolution parameter of the Louvain algorithm. Despite our efforts, we were unable to discern a trend. However, our results indicate that this parameter can be adjusted to facilitate more effective community subdivision, thereby enhancing the overall results.

As with any other methodology, the proposed approach has both strengths and weaknesses.

The effectiveness of the process represents a key strength of this approach. The objective is to consider a subpart of the Knowledge Graph for fine-tuning the Large Language Model. Furthermore, the process can be conducted entirely offline, which is advantageous given that the knowledge included in the KG is injected into the LLM to increase it. This operation must be performed only once, and then the LLM is available for repeated use, with no limitations. This strength is related to another, which concerns RAG (Retrieval-Augmented Generation). Indeed, with our framework, the LLM is fine-tuned with the KG's triples and thus has all the requisite knowledge to answer the user's question about this specific topic. This is a significant advantage, as it avoids the need for the LLM to retrieve knowledge and answers from the KG for each query, a process that is inherently time-consuming. By injecting the necessary knowledge into the LLM, this process is markedly accelerated. Furthermore, the reuse of the generated dataset for any other model training represents an additional advantage.

In terms of the model's limitations, the most significant challenge lies in the conventional approach of fine-tuning a model with a subset of the training data. This method inevitably results in a model that lacks comprehensive knowledge. Furthermore, a Knowledge Graph that comprehensively represents the domain of interest is necessary. However, this can be challenging to find in some cases. Ultimately, if the Knowledge Graph expands in terms of entities, relationships, knowledge or concepts, the LLM must be fine-tuned again, as it is

unable to automatically process this new type of information.

The findings of this research have significant implications for both the practical application of LLMs and the theoretical understanding of KG integration within Natural Language Processing frameworks. By employing a novel approach that fine-tunes an LLM with knowledge from a KG using community-based stratification, this study demonstrates a pathway towards more efficient, targeted and scalable model training.

The approach has the potential to enhance the contextual relevance of model outputs. By stratifying the data into communities, the model is exposed to more semantically cohesive subsets of the Knowledge Graph, which may facilitate the model's ability to learn domain-specific patterns and relationships. This presents significant implications for tasks such as domain-specific language generation, where knowledge integration is essential for the model's success. Consequently, the fine-tuned model benefits from a more focused and relevant training set, thereby enhancing its overall performance on tasks that need precise, context-aware responses.

The concept of community detection within Knowledge Graphs, when applied to LLM fine-tuning, has broader implications for graph theory and its application to Deep Learning. The identification of meaningful clusters within the graph and their usage to inform the model's training data serves to emphasize the pivotal role of community structures in guiding learning processes. This has the potential to have ramifications for other domains where knowledge is structured in networks, including social network analysis, recommender systems and biological network analysis. In all these fields, the application of efficient learning based on subgraph selection could prove beneficial in improving model performance while simultaneously reducing computational overhead.

Furthermore, this study emphasizes the evolving role of Knowledge Graphs in the fields of Natural Language Processing and Artificial Intelligence. Knowledge Graphs have traditionally been regarded as vast, static repositories of structured data, frequently employed to enhance model learning through comprehensive integration. However, the approach proposed in this study illustrates that Knowledge Graphs can be used in a strategic and selective manner. By incorporating only a subset of the graph's information, specifically the elements most pertinent to specific tasks or domains, the efficiency and effectiveness of the integration process are markedly enhanced.

The introduction of community-based paths indicates that KGs are not necessarily required to be monolithic. Instead, they can be regarded as dynamic resources, offering different knowledge slices depending on the task at hand. This could result in a transformation of the manner in which KGs are constructed, maintained and employed across diverse applications. Furthermore, this approach encourages further research into the use of adaptive Knowledge Graphs, whereby the graph itself evolves in response to the addition of new information, and where the LLM can be fine-tuned with evolving knowledge structures over time.

In addition, this work highlights significant ethical considerations regarding to the selection of knowledge and potential biases associated with it. The process of selecting random paths from communities for training introduces the potential for biases in knowledge representation, as certain communities or portions of the graph may be prioritized over others. If the selection process consistently overlooks certain types of knowledge or communities within the graph, this could result in a skewed understanding by the model, which would consequently affect its outputs. It would be beneficial for future research to address these concerns by exploring fair and representative sampling techniques that ensure diverse and equitable knowledge integration.

Finally, the development of more efficient fine-tuning processes has the potential to facilitate broader access to advanced AI technologies. Reducing the computational and resource requirements for training sophisticated LLMs, smaller organizations, research groups, and

developers in resource-constrained environments can benefit from high-performing models without requiring access to extensive computational resources. This accessibility could further accelerate innovation and the application of AI in various fields, contributing to the democratization of AI technologies on a global scale.

Conclusions

The objective of this study was to examine the potential of integrating Large Language Model and Knowledge Graph technologies within a KG-enhanced LLM approach. Recognizing that the time and computational resources required for training or fine-tuning an LLM are considerable, we aim to enhance the performance of an LLM using a KG constructed for a specific domain, with the goal of identifying a framework that reduces the time and computational demands of the fine-tuning process.

The proposed framework successfully facilitated the fine-tuning of an LLM via a knowledge graph comprising domain-specific information in an entirely offline manner. Additionally, it provided a generated dataset that could be used for fine-tuning other models at the user's discretion, while simultaneously reducing the computational resources required for fine-tuning.

The proposed approach consisted of two main phases, i.e., dataset generation and model fine-tuning. Both phases were designed to lower computational complexity while maintaining the integrity of the knowledge embedded within the KG.

The first phase involved constructing training and testing datasets derived from the KG. Instead of training the LLM on the entire graph, which is often impractical due to the size and complexity of most KGs, relevant sections of the graph were selectively extracted for fine-tuning. The Louvain algorithm was used to partition the KG into smaller, cohesive communities. Once the KG was partitioned, random walks were conducted within each community, allowing sequences of nodes and edges representing local interaction patterns to be captured. These sequences played a crucial role in generating the training data by reflecting the fine-grained structural properties of the graph. The length of each random walk was determined by the number of nodes and the diameter of the community in order to ensure efficient exploration of the subgraph. These random walks were then transformed into triples, comprising two nodes and the relationship between them.

The second phase involved fine-tuning the LLM with the domain-specific data generated during the first phase. To reduce the computational cost and evaluate the proposed approach on more than one Knowledge Graph, we extracted three graph samples from the initial Knowledge Graph.

The outcome was the generation of models that were both highly specialized in the target domain and efficient in terms of training time and computational requirements. Each of these models was evaluated employing two different metrics, thereby demonstrating the validity of the proposed approach.

Potential future developments of this work may include modifications to the Louvain algorithm for the community detection task, with the aim of improving the partitioning of the

Knowledge Graph into subcommunities. Additionally, the use of a different or larger model, as well as the incorporation of weights in the relationships of the Knowledge Graph, may be considered. The last proposal can be achieved by embedding the relationships themselves. Furthermore, the training dataset can be enhanced by generating longer texts from the triples using a robust model. Moreover, it would be beneficial to consider the potential for enhancing and implementing the proposed approach with the aim of reducing the occurrence of hallucinations in the Large Language Model, while simultaneously increasing its knowledge base and capacity to generate scope-based texts.

Bibliography

AMARNADH, S., REDDY, P. P. and MURTHY, N. (2021), «WITHDRAWN: FreeHand sketch-based authenticated security system–A comparative study on convolutional neural network, Levenshtein distance and sequence matcher procedure», . (Cited at page 68)

AVRAM, S., BOLOGA, C. G., HOLMES, J., BOCCI, G., WILSON, T. B., NGUYEN, D.-T., CURPAN, R., HALIP, L., BORA, A., YANG, J. J. and OTHERS (2021), «DrugCentral 2021 supports drug discovery and repositioning», *Nucleic acids research*, vol. 49 (D1), pp. D1160–D1169. (Cited at page 60)

BASTIAN, F. B., ROUX, J., NIKNEJAD, A., COMTE, A., FONSECA COSTA, S. S., DE FARIAS, T. M., MORETTI, S., PARMENTIER, G., DE LAVAL, V. R., ROSIKIEWICZ, M. and OTHERS (2021), «The Bgee suite: integrated curated expression atlas and comparative transcriptomics in animals», *Nucleic acids research*, vol. 49 (D1), pp. D831–D847. (Cited at page 59)

BLONDEL, V. D., GUILLAUME, J.-L., LAMBIOTTE, R. and LEFEBVRE, E. (2008), «Fast unfolding of communities in large networks», *Journal of statistical mechanics: theory and experiment*, vol. 2008 (10), p. P10 008. (Cited at pages 63, 64 and 66)

BODENREIDER, O. (2004), «The unified medical language system (UMLS): integrating biomedical terminology», *Nucleic acids research*, vol. 32 (suppl_1), pp. D267–D270. (Cited at page 60)

BORDES, A., USUNIER, N., GARCIA-DURAN, A., WESTON, J. and YAKHNENKO, O. (2013), «Translating embeddings for modeling multi-relational data», *Advances in neural information processing systems*, vol. 26. (Cited at pages 51 and 52)

BOSSELUT, A., RASHKIN, H., SAP, M., MALAVIYA, C., CELIKYILMAZ, A. and CHOI, Y. (2019), «COMET: Commonsense transformers for automatic knowledge graph construction», *arXiv preprint arXiv:1906.05317*. (Cited at page 52)

CHANDAK, P., HUANG, K. and ZITNIK, M. (2023), «Building a knowledge graph to enable precision medicine», *Scientific Data*, vol. 10 (1), p. 67. (Cited at pages iv, v, vi, 59, 61 and 62)

CHEN, W., SU, Y., YAN, X. and WANG, W. Y. (2020), «KGPT: Knowledge-grounded pre-training for data-to-text generation», *arXiv preprint arXiv:2010.02307*. (Cited at page 53)

CONSORTIUM, T. G. O. (2021), «The Gene Ontology resource: enriching a GOld mine», *Nucleic acids research*, vol. 49 (D1), pp. D325–D334. (Cited at page 60)

DANILEVSKY, M., QIAN, K., AHARONOV, R., KATSIS, Y., KAWAS, B. and SEN, P. (2020), «A survey of the state of explainable AI for natural language processing», *arXiv preprint arXiv:2010.00711*. (Cited at page 50)

DAVIS, A. P., GRONDIN, C. J., JOHNSON, R. J., SCIAKY, D., WIEGERS, J., WIEGERS, T. C. and MATTINGLY, C. J. (2021), «Comparative toxicogenomics database (CTD): update 2021», *Nucleic acids research*, vol. 49 (D1), pp. D1138–D1143. (Cited at page 59)

DUGUÉ, N. and PEREZ, A. (2015), *Directed Louvain: maximizing modularity in directed networks*, Tesi di dottorato, Université d'Orléans. (Cited at page 67)

ETGAR, A., FRIEDMAN, R., HAIMAN, S., PEREZ, D. and FEITELSON, D. G. (2022), «The effect of information content and length on name recollection», in «Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension», pp. 141–151. (Cited at page 68)

FENG, C., ZHANG, X. and FEI, Z. (2023), «Knowledge solver: Teaching llms to search for domain knowledge from knowledge graphs», *arXiv preprint arXiv:2309.03118*. (Cited at page 56)

FENG, Y., CHEN, X., LIN, B. Y., WANG, P., YAN, J. and REN, X. (2020), «Scalable multi-hop relational reasoning for knowledge-aware question answering», *arXiv preprint arXiv:2005.00646*. (Cited at pages 53 and 56)

GAO, L., BIDERMAN, S., BLACK, S., GOLDING, L., HOPPE, T., FOSTER, C., PHANG, J., HE, H., THITE, A., NABESHIMA, N. and OTHERS (2020), «The pile: An 800gb dataset of diverse text for language modeling», *arXiv preprint arXiv:2101.00027*.

GARDENT, C., SHIMORINA, A., NARAYAN, S. and PEREZ-BELTRACHINI, L. (2017), «The WebNLG challenge: Generating text from RDF data», in «10th International Conference on Natural Language Generation», pp. 124–133, ACL Anthology. (Cited at page 53)

GUAN, J., WANG, Y. and HUANG, M. (2019), «Story ending generation with incremental encoding and commonsense knowledge», in «Proceedings of the AAAI Conference on Artificial Intelligence», vol. 33, pp. 6473–6480. (Cited at page 53)

HE, B., ZHOU, D., XIAO, J., LIU, Q., YUAN, N. J., XU, T. and OTHERS (2019), «Integrating graph contextualized knowledge into pre-trained language models», *arXiv preprint arXiv:1912.00147*. (Cited at page 55)

HOGAN, A., BLOMQVIST, E., COCHEZ, M., D'AMATO, C., MELO, G. D., GUTIERREZ, C., KIRRANE, S., GAYO, J. E. L., NAVIGLI, R., NEUMAIER, S., NGOMO, A.-C. N., POLLERES, A., RASHID, S. M., RULA, A., SCHMELZEISEN, L., SEQUEDA, J., STAAB, S. and ZIMMERMANN, A. (2021), «Knowledge Graphs», *ACM Comput. Surv.*, vol. 54 (4), URL `https://doi.org/10.1145/3447772`. (Cited at page 35)

HU, N., WU, Y., QI, G., MIN, D., CHEN, J., PAN, J. Z. and ALI, Z. (2023), «An empirical study of pre-trained language models in simple knowledge graph question answering», *World Wide Web*, vol. 26 (5), pp. 2855–2886. (Cited at page 53)

HUANG, X., ZHANG, J., LI, D. and LI, P. (2019), «Knowledge graph embedding based question answering», in «Proceedings of the twelfth ACM international conference on web search and data mining», pp. 105–113. (Cited at page 51)

JASSAL, B., MATTHEWS, L., VITERI, G., GONG, C., LORENTE, P., FABREGAT, A., SIDIROPOULOS, K., COOK, J., GILLESPIE, M., HAW, R. and OTHERS (2020), «The reactome pathway knowledgebase», *Nucleic acids research*, vol. 48 (D1), pp. D498–D503. (Cited at page 60)

JI, Z., LEE, N., FRIESKE, R., YU, T., SU, D., XU, Y., ISHII, E., BANG, Y. J., MADOTTO, A. and FUNG, P. (2023), «Survey of hallucination in natural language generation», *ACM Computing Surveys*, vol. 55 (12), pp. 1–38. (Cited at page 50)

JIANG, J., ZHOU, K., DONG, Z., YE, K., ZHAO, W. X. and WEN, J.-R. (2023), «Structgpt: A general framework for large language model to reason over structured data», *arXiv preprint arXiv:2305.09645*. (Cited at pages 54 and 56)

JIN, Z., GUO, Q., QIU, X. and ZHANG, Z. (2020), «GenWiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation», in «Proceedings of the 28th International Conference on Computational Linguistics», pp. 2398–2409. (Cited at page 53)

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 ed.

KALE, M. and RASTOGI, A. (2020), «Text-to-text pre-training for data-to-text tasks», *arXiv preprint arXiv:2005.10433*. (Cited at page 53)

KÖHLER, S., VASILEVSKY, N. A., ENGELSTAD, M., FOSTER, E., MCMURRY, J., AYMÉ, S., BAYNAM, G., BELLO, S. M., BOERKOEL, C. F., BOYCOTT, K. M. and OTHERS (2017), «The human phenotype ontology in 2017», *Nucleic acids research*, vol. 45 (D1), pp. D865–D876. (Cited at page 60)

KUHN, M., LETUNIC, I., JENSEN, L. J. and BORK, P. (2016), «The SIDER database of drugs and side effects», *Nucleic acids research*, vol. 44 (D1), pp. D1075–D1079. (Cited at page 60)

KUMAR, A., PANDEY, A., GADIA, R. and MISHRA, M. (2020), «Building knowledge graph using pre-trained language model for learning entity-aware relationships», in «2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)», pp. 310–315, IEEE. (Cited at page 52)

LEWIS, P., PEREZ, E., PIKTUS, A., PETRONI, F., KARPUKHIN, V., GOYAL, N., KÜTTLER, H., LEWIS, M., YIH, W.-T., ROCKTÄSCHEL, T. and OTHERS (2020), «Retrieval-augmented generation for knowledge-intensive nlp tasks», *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474. (Cited at page 48)

LI, S., LI, X., SHANG, L., DONG, Z., SUN, C., LIU, B., JI, Z., JIANG, X. and LIU, Q. (2022), «How pre-trained language models capture factual knowledge? a causal-inspired analysis», *arXiv preprint arXiv:2203.16747*. (Cited at page 50)

LI, S., GAO, Y., JIANG, H., YIN, Q., LI, Z., YAN, X., ZHANG, C. and YIN, B. (2023), «Graph reasoning for question answering with triplet retrieval», *arXiv preprint arXiv:2305.18742*. (Cited at page 49)

LIN, B. Y., CHEN, X., CHEN, J. and REN, X. (2019), «Kagnet: Knowledge-aware graph networks for commonsense reasoning», *arXiv preprint arXiv:1909.02151*. (Cited at pages 50, 51 and 56)

LIN, Y., LIU, Z., SUN, M., LIU, Y. and ZHU, X. (2015), «Learning entity and relation embeddings for knowledge graph completion», in «Proceedings of the AAAI conference on artificial intelligence», vol. 29. (Cited at page 52)

LIU, W., ZHOU, P., ZHAO, Z., WANG, Z., JU, Q., DENG, H. and WANG, P. (2020), «K-bert: Enabling language representation with knowledge graph», in «Proceedings of the AAAI Conference on Artificial Intelligence», vol. 34, pp. 2901–2908. (Cited at page 47)

LIU, X., YU, H., ZHANG, H., XU, Y., LEI, X., LAI, H., GU, Y., DING, H., MEN, K., YANG, K. and OTHERS (2023), «Agentbench: Evaluating llms as agents», *arXiv preprint arXiv:2308.03688*. (Cited at page 56)

LIU, Y. (2019), «Roberta: A robustly optimized bert pretraining approach», *arXiv preprint arXiv:1907.11692*.

LUKOVNIKOV, D., FISCHER, A. and LEHMANN, J. (2019), «Pretrained transformers for simple question answering over knowledge graphs», in «The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I 18», pp. 470–486, Springer. (Cited at page 53)

LUO, L., JU, J., XIONG, B., LI, Y.-F., HAFFARI, G. and PAN, S. (2023a), «Chatrule: Mining logical rules with large language models for knowledge graph reasoning», *arXiv preprint arXiv:2309.01538*. (Cited at page 49)

LUO, L., LI, Y.-F., HAFFARI, G. and PAN, S. (2023b), «Reasoning on graphs: Faithful and interpretable large language model reasoning», *arXiv preprint arXiv:2310.01061*. (Cited at page 48)

MAGLOTT, D., OSTELL, J., PRUITT, K. D. and TATUSOVA, T. (2010), «Entrez Gene: gene-centered information at NCBI», *Nucleic acids research*, vol. 39 (suppl_1), pp. D52–D57. (Cited at page 60)

MALLEN, A., ASAI, A., ZHONG, V., DAS, R., KHASHABI, D. and HAJISHIRZI, H. (2022), «When not to trust language models: Investigating effectiveness of parametric and non-parametric memories», *arXiv preprint arXiv:2212.10511*. (Cited at page 50)

MENCHE, J., SHARMA, A., KITSAK, M., GHIASSIAN, S. D., VIDAL, M., LOSCALZO, J. and BARABÁSI, A.-L. (2015), «Uncovering disease-disease relationships through the incomplete interactome», *Science*, vol. 347 (6224), p. 1257 601. (Cited at page 60)

MUNGALL, C. J., TORNIAI, C., GKOUTOS, G. V., LEWIS, S. E. and HAENDEL, M. A. (2012), «Uberon, an integrative multi-species anatomy ontology», *Genome biology*, vol. 13, pp. 1–20. (Cited at page 60)

MUNK, M. and FEITELSON, D. G. (2022), «When Are Names Similar Or the Same? Introducing the Code Names Matcher Library», *arXiv preprint arXiv:2209.03198*. (Cited at pages v, 68 and 69)

PAN, S., LUO, L., WANG, Y., CHEN, C., WANG, J. and WU, X. (2024), «Unifying Large Language Models and Knowledge Graphs: A Roadmap», *IEEE Transactions on Knowledge and Data Engineering*, vol. 36 (7), pp. 3580–3599. (Cited at page 46)

PAPINENI, K., ROUKOS, S., WARD, T. and ZHU, W.-J. (2002), «Bleu: a method for automatic evaluation of machine translation», in «Proceedings of the 40th annual meeting of the Association for Computational Linguistics», pp. 311–318. (Cited at page 68)

PENG, C., XIA, F., NASERIPARSA, M. and OSBORNE, F. (2023), «Knowledge graphs: Opportunities and challenges», *Artificial Intelligence Review*, vol. 56 (11), pp. 13 071–13 102. (Cited at page 42)

PETRONI, F., ROCKTÄSCHEL, T., LEWIS, P., BAKHTIN, A., WU, Y., MILLER, A. H. and RIEDEL, S. (2019), «Language models as knowledge bases?», *arXiv preprint arXiv:1909.01066*. (Cited at pages 50 and 52)

PIÑERO, J., RAMÍREZ-ANGUITA, J. M., SAÜCH-PITARCH, J., RONZANO, F., CENTENO, E., SANZ, F. and FURLONG, L. I. (2020), «The DisGeNET knowledge platform for disease genomics: 2019 update», *Nucleic acids research*, vol. 48 (D1), pp. D845–D855. (Cited at page 59)

RATCLIFF, J. W., METZENER, D. E. and OTHERS (1988), «Pattern matching: The gestalt approach», *Dr. Dobb's Journal*, vol. 13 (7), p. 46. (Cited at page 68)

RIBEIRO, L. F., SCHMITT, M., SCHÜTZE, H. and GUREVYCH, I. (2020), «Investigating pretrained language models for graph-to-text generation», *arXiv preprint arXiv:2007.08426*. (Cited at page 53)

SANCHEZ-LENGELING, B., REIF, E., PEARCE, A. and WILTSCHKO, A. B. (2021), «A Gentle Introduction to Graph Neural Networks», *Distill*, https://distill.pub/2021/gnn-intro. (Cited at page 32)

SAXENA, A., TRIPATHI, A. and TALUKDAR, P. (2020), «Improving multi-hop question answering over knowledge graphs using knowledge base embeddings», in «Proceedings of the 58th annual meeting of the association for computational linguistics», pp. 4498–4507. (Cited at page 53)

SCHRIML, L. M., MITRAKA, E., MUNRO, J., TAUBER, B., SCHOR, M., NICKLE, L., FELIX, V., JENG, L., BEARER, C., LICHENSTEIN, R. and OTHERS (2019), «Human Disease Ontology 2018 update: classification, content and workflow expansion», *Nucleic acids research*, vol. 47 (D1), pp. D955–D962. (Cited at page 60)

SHARIR, O., PELEG, B. and SHOHAM, Y. (2020), «The cost of training nlp models: A concise overview», *arXiv preprint arXiv:2004.08900*.

SHEFCHEK, K. A., HARRIS, N. L., GARGANO, M., MATENTZOGLU, N., UNNI, D., BRUSH, M., KEITH, D., CONLIN, T., VASILEVSKY, N., ZHANG, X. A. and OTHERS (2020), «The Monarch Initiative in 2019: an integrative data and analytic platform connecting phenotypes to genotypes across species», *Nucleic acids research*, vol. 48 (D1), pp. D704–D715. (Cited at page 60)

SHEN, T., MAO, Y., HE, P., LONG, G., TRISCHLER, A. and CHEN, W. (2020), «Exploiting structured knowledge in text via graph-guided representation learning», *arXiv preprint arXiv:2004.14224*. (Cited at page 47)

SMITH, S., PATWARY, M., NORICK, B., LEGRESLEY, P., RAJBHANDARI, S., CASPER, J., LIU, Z., PRABHUMOYE, S., ZERVEAS, G., KORTHIKANTI, V. and OTHERS (2022), «Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model», *arXiv preprint arXiv:2201.11990*.

SUN, Y., WANG, S., FENG, S., DING, S., PANG, C., SHANG, J., LIU, J., CHEN, X., ZHAO, Y., LU, Y. and OTHERS (2021), «Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation», *arXiv preprint arXiv:2107.02137*. (Cited at page 47)

SWAMY, V., ROMANOU, A. and JAGGI, M. (2021), «Interpreting language models through knowledge graph extraction», *arXiv preprint arXiv:2111.08546*. (Cited at page 51)

TIAN, H., GAO, C., XIAO, X., LIU, H., HE, B., WU, H., WANG, H. and WU, F. (2020), «SKEP: Sentiment knowledge enhanced pre-training for sentiment analysis», *arXiv preprint arXiv:2005.05635*. (Cited at page 47)

TRAAG, V. A., WALTMAN, L. and VAN ECK, N. J. (2019), «From Louvain to Leiden: guaranteeing well-connected communities», *Scientific reports*, vol. 9 (1), pp. 1–12. (Cited at page 66)

WANG, H., ZHANG, F., XIE, X. and GUO, M. (2018), «DKN: Deep knowledge-aware network for news recommendation», in «Proceedings of the 2018 world wide web conference», pp. 1835–1844. (Cited at page 51)

WANG, J., HUANG, W., SHI, Q., WANG, H., QIU, M., LI, X. and GAO, M. (2022), «Knowledge prompting in pre-trained language model for natural language understanding», *arXiv preprint arXiv:2210.08536*. (Cited at page 48)

WANG, P., HAN, J., LI, C. and PAN, R. (2019a), «Logic attention based neighborhood aggregation for inductive knowledge graph embedding», in «Proceedings of the AAAI Conference on Artificial Intelligence», vol. 33, pp. 7152–7159. (Cited at page 51)

WANG, P., XIE, X., WANG, X. and ZHANG, N. (2023), «Reasoning through memorization: Nearest neighbor knowledge graph embeddings», in «CCF International Conference on Natural Language Processing and Chinese Computing», pp. 111–122, Springer. (Cited at page 51)

WANG, X., KAPANIPATHI, P., MUSA, R., YU, M., TALAMADUPULA, K., ABDELAZIZ, I., CHANG, M., FOKOUE, A., MAKNI, B., MATTEI, N. and OTHERS (2019b), «Improving natural language inference using external knowledge in the science questions domain», in «Proceedings of the AAAI conference on artificial intelligence», vol. 33, pp. 7208–7215. (Cited at page 55)

WANG, X., GAO, T., ZHU, Z., ZHANG, Z., LIU, Z., LI, J. and TANG, J. (2021), «KEPLER: A unified model for knowledge embedding and pre-trained language representation», *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 176–194. (Cited at page 51)

WANG, Y., LIPKA, N., ROSSI, R. A., SIU, A., ZHANG, R. and DERR, T. (2024), «Knowledge graph prompting for multi-document question answering», in «Proceedings of the AAAI Conference on Artificial Intelligence», vol. 38, pp. 19 206–19 214. (Cited at page 56)

WEINREICH, S. S., MANGON, R., SIKKENS, J., TEEUW, M. E. and CORNEL, M. (2008), «Orphanet: a European database for rare diseases», *Nederlands tijdschrift voor geneeskunde*, vol. 152 (9), pp. 518–519. (Cited at page 60)

WEN, Y., WANG, Z. and SUN, J. (2023), «Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models», *arXiv preprint arXiv:2308.09729*. (Cited at page 49)

WISHART, D. S., FEUNANG, Y. D., GUO, A. C., LO, E. J., MARCU, A., GRANT, J. R., SAJED, T., JOHNSON, D., LI, C., SAYEEDA, Z. and OTHERS (2018), «DrugBank 5.0: a major update to the DrugBank database for 2018», *Nucleic acids research*, vol. 46 (D1), pp. D1074–D1082. (Cited at page 60)

WU, Y., ZHAO, Y., HU, B., MINERVINI, P., STENETORP, P. and RIEDEL, S. (2022), «An efficient memory-augmented transformer for knowledge-intensive nlp tasks», *arXiv preprint arXiv:2210.16773*. (Cited at page 49)

XIE, X., LI, Z., WANG, X., XI, Z. and ZHANG, N. (2022), «Lambdakg: A library for pre-trained language model-based knowledge graph embeddings», *arXiv preprint arXiv:2210.00305*. (Cited at page 51)

XIONG, W., YU, M., CHANG, S., GUO, X. and WANG, W. Y. (2018), «One-shot relational learning for knowledge graphs», *arXiv preprint arXiv:1808.09040*. (Cited at page 51)

XU, Y., ZHU, C., XU, R., LIU, Y., ZENG, M. and HUANG, X. (2020), «Fusing context into knowledge graph for commonsense question answering», *arXiv preprint arXiv:2012.04808*. (Cited at page 54)

YANG, B., YIH, W.-T., HE, X., GAO, J. and DENG, L. (2014), «Embedding entities and relations for learning and inference in knowledge bases», *arXiv preprint arXiv:1412.6575*. (Cited at page 51)

YASUNAGA, M., REN, H., BOSSELUT, A., LIANG, P. and LESKOVEC, J. (2021), «QA-GNN: Reasoning with language models and knowledge graphs for question answering», *arXiv preprint arXiv:2104.06378*. (Cited at pages 50, 54 and 56)

YE, H., ZHANG, N., DENG, S., CHEN, X., CHEN, H., XIONG, F., CHEN, X. and CHEN, H. (2022), «Ontology-enhanced Prompt-tuning for Few-shot Learning», in «Proceedings of the ACM Web Conference 2022», pp. 778–787. (Cited at page 48)

YU, W., ZHU, C., FANG, Y., YU, D., WANG, S., XU, Y., ZENG, M. and JIANG, M. (2021), «Dict-bert: Enhancing language model pre-training with dictionary», *arXiv preprint arXiv:2110.06490*. (Cited at page 48)

ZHANG, D., YUAN, Z., LIU, Y., ZHUANG, F., CHEN, H. and XIONG, H. (2020a), «E-BERT: A phrase and product knowledge enhanced language model for e-commerce», *arXiv preprint arXiv:2009.02835*. (Cited at page 47)

ZHANG, T., WANG, C., HU, N., QIU, M., TANG, C., HE, X. and HUANG, J. (2022a), «DKPLM: decomposable knowledge-enhanced pre-trained language model for natural language understanding», in «Proceedings of the AAAI Conference on Artificial Intelligence», vol. 36, pp. 11 703–11 711. (Cited at page 48)

ZHANG, X., BOSSELUT, A., YASUNAGA, M., REN, H., LIANG, P., MANNING, C. D. and LESKOVEC, J. (2022b), «Greaselm: Graph reasoning enhanced language models for question answering», *arXiv preprint arXiv:2201.08860*. (Cited at page 54)

ZHANG, Z., HAN, X., LIU, Z., JIANG, X., SUN, M. and LIU, Q. (2019), «ERNIE: Enhanced language representation with informative entities», *arXiv preprint arXiv:1905.07129*. (Cited at pages 47 and 55)

ZHANG, Z., LIU, X., ZHANG, Y., SU, Q., SUN, X. and HE, B. (2020b), «Pretrain-KGE: learning knowledge representation from pretrained language models», in «Findings of the Association for Computational Linguistics: EMNLP 2020», pp. 259–266. (Cited at page 51)

ZHAO, W. X., ZHOU, K., LI, J., TANG, T., WANG, X., HOU, Y., MIN, Y., ZHANG, B., ZHANG, J., DONG, Z. and OTHERS (2023), «A survey of large language models», *arXiv preprint arXiv:2303.18223*. (Cited at page 22)

ZHOU, H., YOUNG, T., HUANG, M., ZHAO, H., XU, J. and ZHU, X. (2018), «Commonsense knowledge aware conversation generation with graph attention.», in «IJCAI», pp. 4623–4629. (Cited at page 53)

ZHU, Y., WANG, X., CHEN, J., QIAO, S., OU, Y., YAO, Y., DENG, S., CHEN, H. and ZHANG, N. (2024), «Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities», *World Wide Web*, vol. 27 (5), p. 58. (Cited at page 52)

Sitography

- AI for Medicine and Science – https://zitniklab.hms.harvard.edu/
- Amazon Web Services (AWS) – https://aws.amazon.com/
- arXiv, e-print archive – https://arxiv.org/
- Cloudflare – https://www.cloudflare.com/
- DataHeadHunters – https://dataheadhunters.com/
- DeepLearning.AI – https://www.deeplearning.ai/
- Distill, latest articles about machine learning – https://distill.pub/
- Elastic, The Search AI Company – https://www.elastic.co/
- Encyclopedia Britannica – https://www.britannica.com/
- Hewlett Packard Enterprise (HPE) – https://www.hpe.com/
- IBM – https://www.ibm.com/
- Medium – https://medium.com/
- Neo4j Graph Database & Analytics – https://neo4j.com/
- neptune.ai – https://neptune.ai/
- NVIDIA – https://www.nvidia.com/
- NVIDIA Blog – https://blogs.nvidia.com/
- Ontotext – https://www.ontotext.com/
- Rivista AI – https://www.rivista.ai/
- Stanford University – https://www.stanford.edu/
- TechTarget – https://www.techtarget.com/
- Towards Data Science – https://towardsdatascience.com/

Acknowledgements

I would like to express my gratitude to Professor Ursino, who provided unwavering guidance and support throughout my thesis journey, showing exceptional patience and helpfulness. I am also grateful to Luca, Michele and Davide, who offered consistent assistance and encouragement during the development of the thesis project, helping me to address challenges along the way.

I am deeply grateful to my family for their unwavering support, motivation, and encouragement throughout this journey, both in the most joyous and most challenging moments. Their active listening, constructive feedback, and unconditional belief in me have been instrumental in my growth and success.

I would like to express my deep appreciation to Alice, my partner, for her unwavering support, trust and understanding throughout this journey. Her constant presence and encouragement have been invaluable. I am grateful for her willingness to share every aspect of this experience with me.

I would like to express my gratefulness to Luca, Vito and Massimiliano, my fellow students, for their invaluable contributions throughout the course. We collaborated closely, supporting and relying on each other to successfully navigate the coursework, projects and examinations.

Finally, I would like to express my thanks to all my friends for providing me with moments of recreation and entertainment, as well as for offering me occasions for mirth and delight.