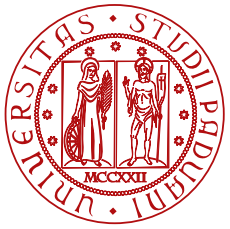

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Computer Engineering

Large Language Models as Knowledge Graph Accuracy Estimators: An Exploratory Analysis

Master Candidate

Andrea Segala

Student ID 2082154

Supervisor

Prof. Stefano Marchesin

University of Padova

Co-supervisor

Prof. Gianmaria Silvello

University of Padova

Academic Year: 2023/2024

Graduation Date: 05/12/2024

You'll never know what you're made of
until you set foot down the road

Abstract

Knowledge Graphs (KGs) are gaining popularity across many real-world applications where accuracy is crucial. The current standard for accuracy evaluation is human-based manual annotation, which is costly due to the large size of the KGs. Emerging studies investigated how KGs' accuracy can be estimated efficiently by employing sampling approaches, aiming to provide a reliable assessment with reduced costs. Yet, to a lesser extent, manual annotation is still required. In this thesis, to further reduce (or even eliminate) the manual costs involved, we explore the use of Large Language Models (LLMs) as automated annotators. To this end, we apply three prompting techniques over multiple KGs to investigate what are the advantages and limitations of LLMs when used to evaluate the accuracy of a KG. The techniques are: Baseline, featuring a simple and straightforward prompt, used as a reference point; Iterative, adding detailed instructions and examples to the prompt; and Stepwise, which approaches the annotation process from a different perspective, employing natural language statements instead of raw KG data. Experiments on popular KGs show that the Iterative prompting demonstrates enhanced reliability, especially in the few-shot scenario, while the Stepwise prompting improves only with respect to the Baseline. However, its high costs make this latter technique unsuitable for the task at hand.

In summary, this study highlights the potential of LLMs in automating KG Accuracy Evaluation processes, and lays out a path for further exploration in retrieval-augmented pipelines and fine-tuned models.

Sommario

I Knowledge Graph stanno acquisendo popolarità in numerose applicazioni reali, dove l'accuratezza è cruciale. Lo standard attuale per la valutazione dell'accuratezza è l'annotazione manuale basata su esseri umani, che risulta costosa a causa delle grandi dimensioni dei KG. Studi recenti hanno investigato come l'accuratezza dei KG possa essere stimata in modo efficiente usando approcci di campionamento, con l'obiettivo di fornire una stima affidabile a costi ridotti. Tuttavia, in misura minore, l'annotazione manuale rimane ancora necessaria. In questa tesi, per ridurre ulteriormente (o addirittura eliminare) i costi manuali coinvolti, esploriamo l'uso dei Large Language Models come annotatori automatizzati. A tal fine, applichiamo tre tecniche di prompting su diversi KG per investigare quali siano i vantaggi e le limitazioni dei LLM quando utilizzati per valutare l'accuratezza di un KG. Le tecniche sono: Baseline, caratterizzata da un prompt semplice e diretto, usata come punto di riferimento; Iterative, che aggiunge istruzioni dettagliate ed esempi al prompt; e Stepwise, che affronta il processo di annotazione da una prospettiva diversa, usando asserzioni in linguaggio naturale al posto dei dati grezzi del KG. Gli esperimenti condotti su KG popolari mostrano che la tecnica Iterative dimostra una maggiore affidabilità, soprattutto nello scenario few-shot, mentre la tecnica Stepwise migliora solo rispetto alla Baseline. Tuttavia, i suoi alti costi rendono quest'ultima tecnica inadatta per il task in questione.

In sintesi, questo studio evidenzia il potenziale dei LLM nell'automazione dei processi di valutazione dell'accuratezza dei KG, e traccia un percorso per ulteriori esplorazioni in pipeline arricchite tramite recupero delle informazioni e modelli fine-tuned.

Contents

List of Figures

List of Tables

List of Algorithms

List of Prompts

List of Acronyms

CI Confidence Interval

CoT Chain-of-Thought

CS Cluster Sampling

EOS End Of Sentence

IRS Information Retrieval System

JSON JavaScript Object Notation

KG Knowledge Graph

LLM Large Language Model

LM Language Model

LSTM Long-Short Term Memory

MLP Multi-Layer Perceptron

NELL Never Ending Language Learner

NER Named Entity Recognition

NLM Neural Language Model

NLP Natural Language Processing

NN Neural Network

RAG Retrieval-Augmented Generation

RCS Random Cluster Sampling

RDF Resource Description Framework

RNN Recurrent Neural Network

SRS Simple Random Sampling

TWCS Two-way Weighted Cluster Sampling

URI Uniform Resource Identifier

WCS Weighted Cluster Sampling

WWW World Wide Web

XML eXtensible Markup Language

YAGO Yet Another Great Ontology

1 Introduction

Knowledge Graphs are often considered perfect repositories, full of highly accurate and structured information, and frequently equated with gold mines, offering rich and reliable data. This perception assumes that any inaccuracies lie in the original unstructured sources, such as textual documents, while the KG is flawless. However, this assumption doesn't hold completely true. Despite their structured nature, KGs are not immune to errors.

Consider DBpedia[1] [2], one of the first and most well-known Knowledge Graph, which is derived from Wikipedia[2]. DBpedia is the knowledge graph version of Wikipedia, so any error occurring in Wikipedia will inevitably propagate into DBpedia as well. Moreover, the unstructured data (Wikipedia articles) is converted into structured data (DBpedia triples) by an automated process. This automation can potentially introduce several errors, including: (1) recognition errors, such as those deriving from Named Entity Recognition (NER) techniques; (2) mapping errors, arising from some incorrect association between textual elements and KG concepts; (3) encoding issues, producing misrepresented data; (4) predicate selection errors, where entities gets linked using inappropriate relationships.

These issues stress the necessity for evaluating the accuracy of Knowledge Graphs, particularly when they are used in critical applications across domains like search engines or healthcare data management.

[1] https://www.dbpedia.org/
[2] https://www.wikipedia.org/

Typically, KG evaluation involves sampling a subset of facts (often represented as RDF triples) and manually annotating them to estimate their accuracy. However, this process is time-consuming and requires substantial resources, such as human annotators and expertise. These challenges motivate the need for new, scalable approaches to assess the quality of KGs.

This thesis explores a novel question, merging the KG quality assessment with Natural Language Processing: **Can Large Language Models be employed to determine whether a fact in a Knowledge Graph is true or false?**

By leveraging the capabilities of pre-trained LLMs, this work aims to offer insight into how they can serve as Knowledge Graphs annotators. This exploration highlights the potential of LLMs in this research field, and contributes to the broader understanding of their capabilities and limitation in knowledge-intensive tasks.

The outline of this thesis is described next. Chapter 2 provides the background for the study, introducing foundational concepts such as the Resource Description Framework (RDF), Knowledge Graphs and their evolution over time, and the architecture, key features and limitations of Large Language Models. Chapter 3 reviews related works, focusing on areas like prompt engineering, the relationship between LLMs and KGs, how LLMs have been utilized for Fact Validation, and the efforts made to reduce annotation costs. Chapter 4 introduces the Accuracy Evaluation task, as well as the three pipelines we developed: Baseline, Iterative and Stepwise prompting. Chapter 5 details the experimental setup, including datasets (DBpedia, FactBench, NELL, YAGO), the models used (Gemma2, Llama3.1, Mistral, Qwen2.5), evaluation metrics, and hardware specifications. Chapter 6 discusses the results, presenting them for each pipelines in terms of standard metrics, tailored metrics, and time measurements. The Chapter includes also a comparative analysis, and an ablation study assessing predicate-level performance of the Baseline prompting. Chapter 7 describes the implementation of the pipeline, starting from the technologies used to carry out this project and proceeding with the pseudocode for each pipeline. Chapter 8 summarizes the main findings of this thesis and provides future directions to explore. Lastly, Appendix A contains all the prompts used for the thesis.

2 Background

In this chapter, the fundamental concepts of this thesis are described: the Resource Description Framework, which is the framework Knowledge Graphs are based on, Knowledge Graphs and Large Language Models.

2.1 Resource Description Framework

The foundations of knowledge representation in a graph-based manner dates back to the earlier Semantic Web initiatives. The Semantic Web [23] is an extension of the World Wide Web (WWW), that is both human- and machine-readable. This has been achieved by incorporating semantic aspects and context into web contents, transitioning from a Web of documents to a Web of (linked) data. One of the technologies supporting this transition is the Resource Description Framework[1], providing a standardized way to represent resources and the relationships between them.

RDF is a data modeling framework that relies on the Web infrastructure to manage distributed data. It consists in a graph-based approach where data is organized into (s, p, o) triples, also called *statements*. In such triples, s is the subject, p is the predicate and o is the object of the triple. Subject and object are nodes of the graph, while predicates are the edges of the graph. A collection of RDF triples is called a *RDF graph*.

[1] `http://www.w3.org/RDF/`

As an example, consider the triple "`Francis Ford Coppola directed The Godfather`": "`Francis Ford Coppola`" is the subject, "`directed`" is the predicate and "`The Godfather`" is the object.

The subject of a triple is either a resource identified by its Uniform Resource Identifier (URI), or a blank node, that is, an anonymous resource without a URI; the object of a triple is either a resource, a blank node, or a literal (that is, a string associated with an optional language tag or a datatype URI); the predicate of a triple is a directed arc starting from the subject and pointing to the object of the RDF triple, representing the relationship that is linking them.

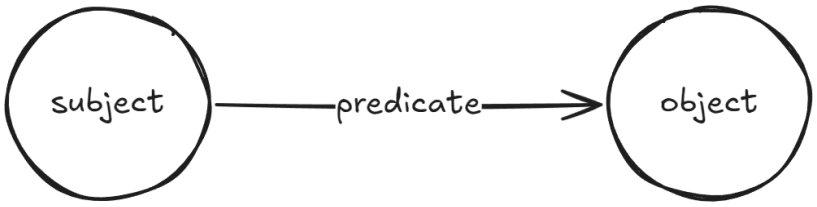

Figure 2.1: Graphical representation of a generic RDF triple.

2.1.1 Serialization

The process of serialization allows to convert a RDF graph into text, enabling its processing by computers. Possible serialization formats are:

- **RDF/XML**: it represents triples using the eXtensible Markup Language (XML). This format uses a tree representation, resulting in a verbose and redundancy prone but human-readable output;
- **N-Triples**: this format requires triples to be organized one per line, expressed in the $s - p - o$ order using the full URIs. Each line ends with a period;
- **Turtle/N3**: compact and human-friendly serialization, employing simpler syntax than both *RDF/XML* and *N-Triples*. Indeed, by using *Turtle* we can express predicate and object lists (respectively, triples sharing the same subject or the same subject and predicate);
- **JSON-LD**: mainly used for Web application development, it's a JSON-based (JavaScript Object Notation) serialization that supports the addition of semantics and context information, making it suitable for Linked Data.

2.2 Knowledge Graphs

The term "Knowledge Graph" gained significant popularity in 2012, with the introduction of the Google Knowledge Graph [44], from which the *Knowledge Panels* besides the search results are generated. Still, there is no common definition stating what is a Knowledge Graph and what is not.

Paulheim [10] defines a series of characteristics that a KG must comply with: (a) mainly describe real world entities and their interrelations, organized into a graph; (b) defines possible classes and relations of entities in a schema; (c) allows for potentially interrelating arbitrary entities with each other; (d) covers various topical domains.

The second and fourth criteria cover the concept of "ontology": a way to model the reality of one or more domains. An ontology comprises: (i) individuals, representing objects in the domain(s) of interest; (ii) attributes, describing the objects in the ontology; (iii) classes, collections (or sets) of objects, and (iv) properties, that are binary relations on individuals or classes. Classes in an ontology can also be organized in a hierarchy, called *taxonomy*.

KGs can be considered as partially structured models: despite adhering to an ontology, they also allow for flexibility in adding new classes and relationships.

While there is no universally accepted definition of a KG, we can provide a formal definition if we stick to the RDF standard. In particular, we take inspiration from the definition given by Bonifati [5]:

Definition 2.1 (Knowledge Graph)**.** A Knowledge Graph is a directed, edge-labeled multi-graph $G = (V, R, \eta)$:

- $V = E \cup A \cup B$ is the set of nodes in G, where E are entities (*i.e.*, resources), A attributes (literals) and B blank nodes;
- R is the set of relationships between nodes in G (*i.e.*, all possible predicates);
- $\eta : R \rightarrow (E \cup B) \times (E \cup A \cup B)$ is a function assigning an ordered pair of nodes to each relationship.

2.2.1 Knowledge Graphs Evolution

It's worth exploring the evolution of Knowledge Graphs over time. Following the classification provided in [12], we can outline three generations of KGs:

- **entity-based KGs**: these KGs present an ontology with clear semantics, leaving little room for overlaps and ambiguity. The entities that can be found in such KGs are *named* entities, *i.e.*, corresponding to real-world entities. Examples for this KG generation are Yago [46], DBpedia [2], Google KG [44], and WikiData [53].
- **text-rich KGs**: this generation shines in domains where defining a taxonomy is challenging due to the widespread overlapping of concepts, *e.g.*, Products, Health and Events. The construction of this kind of KGs relies on knowledge extraction techniques.
- **dual neural KGs**: this generation is not fully shaped yet, but the main idea behind dual neural KGs follows the reduction of structure happened from entity-based to text-rich KGs. The core question this generation addresses is whether structure can be completely removed with the use of embeddings, implicitly capturing semantics, or KGs and LLMs can be merged into a single, hybrid system.

This work will make use of entity-based KGs.

2.3 Large Language Models

Language Modeling is the task of predicting which words come next in a sequence of words, or, with a *generative* point of view, estimating how likely it is for a sequence of words to belong to the modeled language.

The development of Language Models (LMs) has been an essential advancement in Natural Language Processing and has evolved rapidly over time. Earlier approaches to Language Models involved statistical methods, such as N-Grams, which approximates probabilities using fixed-length word sequences. These models are simple and efficient, but fail at capturing long-range dependencies within words and sentences, and are unable to share statistical strength across similar words.

The advent of Neural Networks (NNs) brought a huge improvement on the scene of language modeling. N-Gram models have been supplanted by Neural Language Models (NLMs), which can handle the distance limitation of N-Grams while remaining computationally and statistically tractable, and to generalize better over context and similar words. As for the negatives, they are more complex, slower, and less interpretable than N-Grams. To illustrate, when using N-Gram models the observations of ”green apple” don't affect the observations of ”red apple”. On the other side, NLMs may understand that ”green” and ”red” are both colors and relate them to the noun ”apple”.

Recurrent Neural Networks (RNNs) [14] are one of the first NN architectures developed to tackle sequential data, as they could process one word at a time while maintaining memory of previously seen words. However, RNNs suffer from the vanishing gradient problem, making it difficult to retain information from earlier parts of long word sequences. To overcome this limitation, more powerful models have been developed such as Long-Short Term Memory (LSTM) models [19].

The major and most recent improvement in Language Modeling came with the Transformer architecture [52]. Transformers employ the Attention mechanism, which allows the model to focus on different parts of a text simultaneously rather than processing word sequentially. Attention revolutionized the way in which Language Models work, making it possible to capture complex relationships between words over long distances efficiently.

The Transformers architecture laid the foundation for modern Large Language Models, which leverage vast amounts of textual data and billions of parameters to perform tasks smaller models struggled in.

We can now outline what differentiates Large Language Models from Language Models.

Definition 2.2 (Large Language Model)**.** A Language Model is called *Large Language Model* if it has the following features:

- neural network architecture based on Transformers;
- more than 1 billion parameters;
- pre-trained on vast amount of textual data;
- can be adapted to a wide range of downstream tasks.

2.3.1 Transformers' Architectures

The original Transformer model uses an encoder-decoder architecture. The encoder processes the input text and returns contextual embeddings (a latent representation of the inputs), while the decoder uses this context to generate the outputs.
Nevertheless, different classes of LLMs utilize this architecture in distinct ways:

- **encoder-only models**, *e.g.*, BERT[2] [11], employ just the encoder to produce a word representation for each input token. These models learn how words work and are used as a preprocessing step in tasks such as Sentence Pair Classification and Named Entity Recognition;
- **decoder-only models**, *e.g.*, the GPT family [38, 6, 33], generate text autoregressively, predicting one token at a time based on previously generated tokens. They are optimal for text generation tasks;
- **encoder-decoder models**, *e.g.*, T5 [40], use both encoder and decoder. They create their own latent representation of the input tokens through the encoder, and then the decoder generates the output tokens based on this latent representation. In other words, they cast any problem into text-to-text problems. They are employed in tasks like Machine Translation, Text Summarization, and Paraphrase Generation.

2.3.2 Large Language Models Key Features

The key features of LLMs are the mechanisms and strategies which allow LLMs to understand the input text and give us control over the generation of the output.

Attention is the core of the transformer model, allowing it to focus on different parts of the input during the whole generation process. When a token is being generated, attention makes the model attend not only to the immediate context (*i.e.*, the previous token) but to any part of the input. This can solve some long-distance relationships between words, *e.g.*, pronouns. As an example, consider the following: "Jane bought the new AirPods and let me try them.

[2]BERT reaches 340M parameters in its larger variant, hence it is not considered a *Large* Language Model according to the given definition. Still, its size was significant at the time of its release in 2018, and the innovations it has introduced have made it a foundational model.

Now I want them too!". Attention helps the model understand that both `them` are referring to `AirPods`, even if `them` and `AirPods` are not consecutive words or in the same sentence.

Transformers also use self-attention in the encoder, which comes into play when the model is understanding the input sequence itself when it's being encoded.

To control the generation of the output, it's important to clarify two aspects of the output texts. Ideally, the text generated by a LLM should be *coherent*, in order for it to make sense, and *diverse*, meaning the model should be able to produce very different text samples when prompted with the same input text. We do have a trade-off between coherence and diversity.

A popular method to deal with this trade-off is the **(softmax) temperature**: it changes the model behavior by changing its generation distribution, *i.e.*, the distribution from which the model samples to create the output text. Its value ranges from 0 to 1, where 0 correspond to high coherence (peaky distribution) and 1 represents high diversity (flat distribution). High temperature may also cause hallucinations, which will be described in Section 2.3.3. Other methods to control the model's behavior are **top-k sampling** and **top-p sampling**, which are two ways to choose the pool of tokens from which the model will generate the output text. Top-k sampling reduces the sampling pool size by considering only the k most probable tokens. Instead, top-p sampling allows a dynamic size for the pool, considering all the most probable tokens whose cumulative probability sums up to a threshold p.

Another way to control the model's behavior is the **Beam Search**. Beam Search is a decoding strategy used to keep the best B output sequences, where B is a hyperparameter called *beam size*. Beam Search is an iterative algorithm used to reduce the complexity of exhaustive search, which is unfeasible. The algorithm builds a tree where the nodes are the generated tokens. It starts with the B initial best hypotheses (*i.e.*, most probable tokens). In each iteration, it expands those hypotheses, scores the results and prunes the tree in order to keep only the best B hypotheses. When a complete hypothesis is found, it gets removed from the frontier of the tree and B gets decreased by 1. This process is repeated until $B = 0$.

When $B = 1$ we can talk about **greedy search** (or greedy decoding), a scenario

where the LLM does not perform sampling and always takes the most probable token at each step, until the End Of Sentence (EOS) token is generated. This can also be achieved by setting the temperature to zero.

2.3.3 Large Language Models Limitations

Despite all the positive aspects of LLMs, they're far from perfect. Among the most critical limitations of these models we can find hallucinations and the scalability problem.

Hallucinations are a phenomenon where the model generates a text that is incorrect, nonsensical, or not real. Yet, this text is still convincing since these are models meant to mimic how humans write. Hallucinations happen because LLMs are not databases or search engines: they don't have a notion of what is true or false, and they don't provide sources for their responses. Hallucinations may also be caused by high temperature values: a flat distribution means words are equiprobable, leading to, as mentioned, incorrect, nonsensical or not real text. To avoid hallucinations, we can use the so-called *controlled generation*: provide enough details and constraints in the prompt to the model.

Scalability relates to the ability of any system to cope with growth, in terms of data, computational load or task complexity, without accusing a significant decrease in performance. In the case of LLMs, we refer to the ability to handle increasing amounts of data, such as Knowledge Graphs.

The size and the complexity of both LLMs and KGs make running these models on the entirety of such data infeasible. The main limitations are (a) computational resource demand, particularly during training, where resource requirements are significantly increased; (b) performance bottlenecks, as the increased scale leads to longer processing times and higher energy consumption; and (c) costs, because running these models at scale results in prohibitive costs for most organizations, except for those with massive computational resources *e.g.*, Google, Meta and OpenAI.

In order to deal with scalability, possible solutions are:

- **quantization**: approaches like [3] applied weight quantization to reduce the hardware requirements and improve the model efficiency, while maintaining an acceptable level of performance;

- **distillation**: the LLM serves as a "teacher" model to train a smaller (and simpler) model, called "student", designed to mimic the teacher's behavior [18]. An example is DistilBERT [43], 60% faster and 40% smaller than its teacher BERT [11];
- **caching**: in the case of repetitive queries, caching some frequently accessed results may be beneficial, reducing both computational load and latency;
- **parallelization**: models can be distributed over multiple devices, resulting again in lighter computational load due to the parallel computations.

3 Related Works

3.1 Prompt Engineering

Prompt engineering rose with GPT-3 [6], in 2020. OpenAI introduced the outstanding abilities of their model showing that a wide range of downstream tasks could be performed without fine-tuning, just by providing a detailed enough prompt comprising few examples. This technique is now known as **few-shot prompting**, in contrast to **zero-shot prompting** where the prompt contains no examples. This discovery pointed out that we can effectively control the models' behavior across multiple tasks just by formulating an appropriate prompt.

Subsequent research delved into prompt engineering, discovering new prompting techniques, *e.g.*, Chain-of-Thought (CoT) [55] and ReAct [59].

Chain-of-Thought demonstrates the reasoning capabilities of LLMs in arithmetic, commonsense and symbolic reasoning, making use of few-shot prompts in which the examples report all the steps of the reasoning process leading to the final (and correct) answer. ReAct instead investigates LLMs as agents, combining reasoning and acting. The reasoning is similar to CoT, thinking explicitly step by step through the problem. The reasoning process is made explicit in order to reduce hallucinations. Once the reasoning is done, the agent performs an action, such as querying a database or taking a decision, based on the reasoning output. ReAct then continues in this loop of reasoning and acting until the problem is solved.

Another fashion of research provided further evidence of LLMs capabilities, showing that larger models exhibit unexpected capabilities when prompted in a certain way. The study calls these **emergent abilities** and while it doesn't propose any new methods, it surveys the existing literature showing that the presence of these abilities spans a broad landscape of models and tasks. Among the tasks, we can find some included in BIG-Bench [45] (arithmetic, transliteration, word unscramble and PersianQA), TruthfulQA [25], grounded conceptual mappings, multi-task Natural Language Understanding (from Massive Multi-task Language Understanding benchmark [17]), Word in Context [36] (a semantic understanding benchmark), multi-step reasoning (CoT), and instruction following. As for the models, these abilities have been observed in GPT-3 [6], Gopher [39], Chinchilla [21], PaLM [9] and LaMDA [50]. The authors then acknowledge that risks may also emerge, briefly citing gender bias and toxicity. The common traits of emergent abilities are the sharpness with which they appear, in the sense that the transition from *not present* (in smaller models) to *present* (in larger models) occurs all of a sudden, and the unpredictability of the scale from which the transition happens. Indeed, the latter depends on a variety of factors and it is not an immutable property of the specific ability.

In this thesis we rely on zero- and few-shot prompting in its simpler forms as introduced by OpenAI, employing these techniques in the Iterative prompting, which will be outlined in Section 4.2.2.

3.2 Large Language Models as Knowledge Graphs

As LLMs continue to evolve, researchers have started to investigate their potential to serve similar functions to Knowledge Graphs.

This hypothesis was first explored by Petroni et al. [35] through LAMA. LAMA is a framework to inspect the knowledge encoded in LLMs using Masked Language Modeling, covering a variety of types of knowledge. Models are asked to fill in the blanks in a masked sentence about some factual information. They observed that LLMs can store vast amount of (linguistic) knowledge, that is accessible by either conditioning on a latent context representation produced by the model itself or by fine-tuning the original model weights through transfer learning, resulting in a new model tailored for the specific task. Furthermore,

their work highlights that LLMs appear to be a suitable replacement for the complex Natural Language Processing (NLP) pipelines needed for the extraction of relational information from text, as the former provide more flexibility by eliminating the requirements of schema and human annotations. Their main findings revolve around BERT [11]: it can capture accurate relational knowledge, recovering factual knowledge surprisingly better than its competitors and reaching a level that is competitive with non-neural alternatives.

Roberts et al. [42] proposed a similar work. The key differences are the usage of fine-tuned models and the task, as they preferred to evaluate models on an open-domain question answering task. The results report that state-of-the-art performance is reached only with the largest models (around 11B parameters) and that models arrange knowledge in their parameters in an explicable way, hence leaving no room for interpretability. In contrast, open-book models leave room for interpretability, given that they are able to cite the source of information that was accessed to provide the answer.

Further evidence of the amount of knowledge stored in LLMs is brought up by Wang et al. [54]. This study developed MaMa, an unsupervised approach that retrieves factual knowledge from both pre-trained LLMs and web corpora, to build a KG from scratch. The algorithm is divided in two stages: the Match stage generates a set of candidate facts matching the information on the textual corpora with the knowledge in the LLM, and then the Map stage produces what they call an "open KG", sorting the candidates out in a fixed schema or an open schema. The latter is employed for facts that don't exist in the fixed schema.

Mruthyunjaya et al. [31] instead focused on nuanced attributes of KGs, that are, topological and semantic contraints such as symmetric, asymmetric, hierarchic and inverse relationships between entities. They built a benchmark composed of 9 sets of statements, where each set is tailored for a specific constraint. Subsequently, statements were turned into prompts and submitted to the LLMs. However, results showed that LLMs struggle to capture such symbolic representations.

3.3 Large Language Models for Fact Validation

Earlier fact checking pipelines involved Information Retrieval Systems (IRSs) to retrieve the evidence needed to validate some fact. In general, IRSs have three major components: a document retriever which provides relevant documents, an evidence selector which scores the relevant documents, and a classifier which outputs the final judgment (*i.e.*, whether the fact is supported). With LLMs now employed successfully on a wide variety of tasks, researchers have begun to consider them as a possible replacement for traditional IRSs, exploring their capacity to condense retrieval, evaluation and classification within a single component.

Inspired by LAMA [35], the work of Lee et al. [24] represents one of the first attempts in building a fact checker system leveraging solely the implicit knowledge contained in LLMs. They first determined the feasibility of the approach with a human review study, where the participants were asked to validate claims from FEVER [51] using only BERT [11]. This preliminary study confirmed the capabilities of BERT as a fact checker, so they moved on designing an end-to-end neural approach. The proposed pipeline is as follows: for each claim from FEVER, an evidence sentence is generated by masking the last named entity appearing in the claim. BERT is then asked to fill in the blank in the masked claim. Lastly, a Multi-Layer Perceptron (MLP) classifier compares the original claim and the claim filled by BERT to decide whether the fact is supported, refuted, or if there is not enough information to confidently make a definitive judgment.

This pipeline achieved slightly better results than traditional methods, demonstrating the effectiveness of zero-shot prompting in a closed-book setting (*i.e.*, without accessing external sources of information), but still lags behind state-of-the-art methods that rely on KGs for retrieval.

However, more recently the focus shifted from closed-book scenarios to open-book scenarios, with the development of Retrieval-Augmented Generation (RAG) pipelines. As a brief outline, RAG is a two-step process (retrieval and generation) that allows to enhance prompts with additional context coming from external sources (including the Web). RAG advantages encompass: flexibility, by enabling the injection of knowledge that wasn't available during pre-training;

efficiency, since it can be set up with smaller models; and the updatability of the external information sources.

Yang et al. [58] built a factual question answering benchmark spanning different domains, question categories, popularity and temporal dynamism. They included mock KGs and mock APIs simulating web search, and tested LLMs with and without RAG. RAG produced better results, proving that the extra information does help, but can also introduce hallucinations if it's not relevant for the question. In turn, hallucinations are mitigated by KG knowledge, that is typically brief and precise. They conclude with an overview of state-of-the-art industry models, which achieve better performance but still fail to significantly reduce hallucinations, resulting in untrustworthy answers.

While RAG is out of scope for this thesis, the findings from the closed-book scenario are highly relevant, as the models we will use will be probed just with the KG facts.

3.4 Large Language Models Knowledge Boundary

Another research topic addresses how well LLMs perceive their knowledge and are able to evaluate themselves. Ren et al. [41] propose Judgmental Prompting, where the model first assesses its confidence before answering a prompt, and then evaluates its own response. A similar route is taken by Zhang et al. [60], where the pipeline reverts to a RAG approach that retrieves information from the Web when the model doesn't feel confident enough.

Building on both of these works, we developed the Stepwise prompting, described in Section 4.2.3.

Another facet of Knowledge Boundary is provided by Sun et al. [47], where the main result is that LLMs have better performance in head (*i.e.*, popular) entities. This was hypothesized by the authors since head entities appears more frequently during pre-training, and can be an important factor to consider during the evaluation.

3.5 Annotation Costs

Building and maintaining high-quality KGs require abundant manual effort in the annotation process, that is, human resources and expertise. As KGs grow in scale, both costs and time needed for the annotation process can become a bottleneck, slowing down KGs development and adoption. In particular, manually evaluating every triple in a KG is infeasible (due to their large size), and the common practice is to evaluate only a small but representative sample of triples. This highlights the importance of finding efficient ways to reduce the annotation costs.

Ohja and Talukdar [32] were the first to recognize and address the need for an efficient estimation of the accuracy of large-scale KGs. They proposed KGEval, an automatic evaluation framework for KGs based on Horn clauses and coupling constraints among triples. Triples are also called *beliefs* in this study. In a pre-processing step, beliefs and constraints are linked in a bipartite graph, to ease the inference process. Their algorithm consists of a control mechanism, which selects the constraint to be evaluated next, and an inference mechanism that derives the correctness of additional beliefs.

The control mechanism adopts greedy hill-climbing algorithms to choose the next constraint to be evaluated with crowdsourcing, accounting for the inferable set size it carries, while the inference mechanism applies Horn clauses (that are, first-order-logic conjunctions of coupling constraints) to automatically estimate the correctness of connected beliefs within the graph, *i.e.*, the aforementioned inferable set.

However, their algorithm presents some limitations: the inference process is probabilistic, so it can propagate errors, and it does not scale well [15].

Gao et al. [15] focused on sampling strategies that generate a representative set of a KG. The evaluation model is composed by two steps: entity identification and relationships validation. They argued that annotating a given amount of triples from the same entity is faster than annotating the same amount of triples having different entities, because the entity identification happens just once (while in the second scenario it needs to be performed *at least* once for each entity). In their work, the term "entity" refers to the subject of a (s, p, o) triple. This study provides an overview of many sampling strategies, ending with the

state-of-the-art Two-way Weighted Cluster Sampling (TWCS).

They start from Simple Random Sampling (SRS), which draws n triples from the KG with replacement. This is not ideal since by drawing independent triples we are likely to incur in many distinct entities, therefore not reducing costs at all.

They proceed with Cluster Sampling (CS) techniques, that are based on entity clusters. An entity cluster is a set of triples about the same entity e: $G[e] = \{t \mid t : (s, p, o) \wedge s = e\}$. For the analysis of these techniques, we are interested in an accuracy estimator with the lowest variance possible, as this minimizes the spread of estimates across samples, therefore improving the reliability and the consistency of the results. Here's an overview of all the CS techniques analyzed:

- **Random Cluster Sampling (RCS)** draws n entity clusters and evaluates all of them: this leads to an estimator with high variance, due to the wide variations in cluster size.

- **Weighted Cluster Sampling (WCS)** improves RCS by sampling entity clusters with probabilities proportional to their size $\pi_i = \frac{M_i}{M}$, where π_i is the weight of the i-th entity cluster, M_i the size of the i-th entity cluster and M the total number of triples in the KG. Here the variance is lower than in RCS, because the accuracies involved in its computation are computed on the entire clusters and not on the individual triples.

- **Two-way Weighted Cluster Sampling** is finally introduced to further reduce the costs of WCS. The accuracy is now computed on sampled clusters from sample of triples. The process is two-fold: we sample using WCS in the first stage, and then we sample $min\{M_{I_k}, m\}$ triples from the k-th entity cluster (using SRS without replacement) in the second stage. m is a hyperparameter for the maximum number of triples to sample from each entity cluster. Note that TWCS is scalable: according to sampling theory, for sufficiently large population, the sample size needed to achieve a certain level of precision is not affected by the population size itself.

Next, they present a stratification approach. Stratification is a statistical method where the population is divided into subpopulations (called *strata*), such that some feature of interest remains homogeneous in the strata if compared to the overall population. Stratified sampling can thus be applied to further boost efficiency of cluster sampling. In fact, we can stratify the KG and

then apply TWCS in each stratum obtaining even lower variance, which allows for an additional reduction of the sample size.

Qi et al. [37] outline two primary stages in the KG construction process: Information Extraction and Entity Linking. However, prior studies focused solely on the triple accuracy, therefore evaluating only Information Extraction quality and overlooking the Entity Linking stage. In their vision, machines and humans are complementary: the former are good at computing, while the latter are good at interpreting natural language. They built an interactive framework out of this view: in short, the machine performs the sampling and all the computations, while the human does the annotations.

This framework makes use of Inference Graphs (similarly to [32]) by integrating triples, entity linking results and dependencies rules together. The process is outlined next: the Inference Graph Constructor combines triples, linked entities and dependencies together; the Sampler draws a subset of the Inference Graph; the Annotation Helper pre-computes what the human will annotate next and performs inference based on the human feedback; and the Collector collects the annotated samples and decides when to stop (based on the Confidence Intervals). The Annotator Helper highlights when and how the interaction happens: the cost reduction provided by this framework arises from the overlap and interleaving of machine and human efforts.

Up to now, most of the efforts for reducing the costs of annotating KGs revolved around the sampling strategy, ignoring the aspect of reliability of the estimates. As a direct consequence, all of these methods rely on the simplest method to compute Confidence Intervals (CIs), namely, Wald interval [8]. Marchesin and Silvello [26] take a step in this direction, recognizing the limitations of Wald interval. Despite its simplicity, this method suffers from zero-width and overshooting intervals. A zero-width CI implies certainty, underestimating the error; overshoots happens when the CI exceeds the interval $[0, 1]$. Zero-width intervals also exacerbate the unpredictable coverage probability behavior of Wald method. The coverage probability is the probability that a CI contains the parameter of interest (in this case, the KG accuracy). Additionally, Wald interval can be inaccurate for small samples.

To overcome these problems, they encourage the adoption of Wilson interval [56], a binomial CI commonly used for binary labels. Compared to Wald, Wil-

son interval is asymmetric, improving the reliability of the estimates by shifting the center estimate towards the center of the accuracy range. The Agresti-Coull interval [1] is also briefly discussed, as it combines the Wald's simplicity with the Wilson's reliability.

Eventually, a thorough evaluation shows that Wilson method outperforms all the others when it comes to reliability, emerging also as the most efficient option when all methods prove reliable.

As a downstream application example, we can consider another work of Marchesin et al. [28], analyzing the impacts of veracity on entity-oriented search. To remain aligned with the current topic, we will concentrate only on their KG veracity framework. In this framework, the KGs are first partitioned with stratification according to an utility model based on the facts' popularity on the Web. Subsequently, partitions are annotated within a fixed budget. The presented algorithm estimates the veracity of a partition by sampling and annotating small batches of triples from the partition, one batch at a time. The algorithm is constrained to both an (user-specified) accuracy threshold and the budget, and it terminates when one of the following conditions is verified: the estimation results meet the accuracy threshold, preventing unnecessary manual annotations, or the budget is exhausted. In both cases, the costs are minimized.

4

Prompting LLMs for Accuracy Evaluation

This chapter provides an overview of the task of this thesis, and introduces the three prompting techniques developed to approach it: Baseline, Iterative, and Stepwise. Each technique is tailored to different aspects of KG evaluation, as detailed in the following sections.

4.1 Accuracy Evaluation Task

Evaluating the accuracy of a KG implies determining the truthfulness of each fact it contains. In the KGs we considered, facts are RDF (s, p, o) triples and the goal is to assess whether each triple reflects a real-world fact.

The process consists in treating each KG fact as a claim, and verifying its validity by asking if the claim is true or false.

We utilize LLMs to assess the truthfulness of the KG facts by probing them to determine if a fact align with what the model "knows" or can infer from its pre-training data. We only rely on pre-trained LLMs, without accessing external information sources to gather more context.

With the additional objective of containing costs, we built an automatic evaluation framework where we ask LLMs to answer "T" if a fact is judged accurate (true) or else "F" is the fact is judged inaccurate (false). If the model fails to provide one of these expected answers, we will mark the answer as invalid, encoding this scenario with the value "NA" (Not an Answer).

4.2 Prompting Techniques

In this section, the three prompting techniques employed in this study will be outlined. As a brief summary, the Baseline technique utilizes a short prompt to set a baseline, the Iterative technique expands the Baseline technique by providing more detailed guidelines, and the Stepwise technique analyzes the KG facts by converting them into natural language statements.

4.2.1 Baseline Prompting

We establish a baseline for evaluating the KG accuracy by designing a basic and direct prompt. This prompt serves as a straightforward method to obtain responses from the LLM without providing significant guidance or complex instructions. The main goal is to understand the model's "default" ability to verify factual claims based solely on its pre-trained knowledge.

The prompt directly asks the model to assess the truthfulness of the KG fact. The workflow is described in Figure 4.1. Through this technique we can measure the solo performance of a model before introducing complexity, and the related results will serve as a reference point while evaluating the effectiveness of the other approaches.

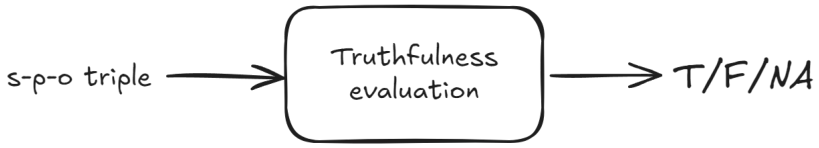

Figure 4.1: Workflow of the Baseline pipeline.

4.2.2 Iterative Prompting

Building on the Baseline prompt, the Iterative prompt introduces general-purpose guidelines with the aim to improve the accuracy and the consistency of the responses.

As with the Baseline approach, the model is given guidelines on the expected answers. In addition, instructions are included to provide further guidance over the following aspects:

- **language consistency**: to standardize the responses, the model is asked to answer exclusively in English. This requirement is added since some facts may contain non-Latin characters (for example, country names in their official language);
- **factual accuracy**: the model is asked not to spread incorrect information, to encourage the model to avoid generating information it cannot verify;
- **reliance on pre-trained knowledge**: as for the task definition, the model needs to rely on its internal knowledge only. This is made explicit in the Iterative prompt;
- **answer format**: the model is reminded to keep the response concise and limited to one of the expected judgments.

Beyond these aspects, this technique also employs a "retry mechanism" which was inspired by the work in [27]: the model is informed in advance that if the answer doesn't meet the guidelines, then it will be asked again for the same fact within a limited number of attempts, and that in the case all attempts are exhausted, the answer will be considered invalid (*i.e.*, "NA"). To this end, the retry prompt explicitly states that the previous answer was not compliant and includes a reminder of the expected answers and guidelines. Figure 4.2 illustrates the workflow of the Iterative pipeline.

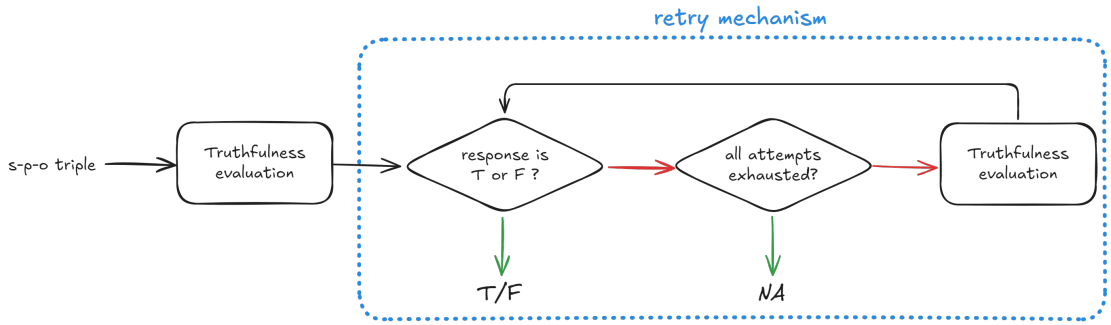

Figure 4.2: Workflow of the Iterative pipeline.

Prompt Variations

Now that the main characteristics of the prompt have been discussed, a brief explanation of the variations we tried follows:

- **Guidelines variants**: instructions are provided in a structured format (*structured*), presenting them in a clear and itemized manner, or in a prose format (*natural*), resembling natural language.
- **Fact presentation variants**: KG facts are presented in different formats to the model, either as a one-liner (*inline*), with subject, predicate and object in sequential order, or as unordered lists (*noprefix*), reporting just the values in separate lines. In both cases, the prompt contains a short description of the chosen fact format.
- **Zero-shot** and **few-shot** prompts: the prompt was applied in both a zero-shot setting, with no examples provided, and a few-shot setting, providing a few examples of correctly evaluated triples to enhance the model's understanding of the task. Examples are presented accordingly to the fact format.

We chose to evaluate the models only on the *natural-inline* and *structured-noprefix* guidelines and fact format pairs, considering both zero-shot and few-shot settings. The choice fell on these because they leave the prompt with a consistent style, with both guidelines and fact presented either in prose or through lists.

4.2.3 Stepwise Prompting

This technique keeps the simple and direct style of the Baseline prompting, but investigates on the model's ability to evaluate natural language statements. While both Baseline and Iterative prompting consisted of a single prompt for each fact, the process now unfolds in three steps:

1. **Statement production**: we ask the model to produce a statement in natural language based on the provided KG fact. The model is allowed to answer with "IDK" (I Don't Know) if it is unable to produce a statement.
2. **Confidence assessment**: taking inspiration from Judgmental Prompting [41], we ask the model to assess its confidence in evaluating the statement it produced earlier. In this step, the accepted answers are "H" for high confidence and "L" for low confidence.

3. **Truthfulness evaluation**: we ask the model to evaluate the truthfulness of the statement. Note that the prompt for this step differs from the Baseline prompt, as it is part of a different and more articulated process.

The workflow is as follows: the model generates a statement in the first step, and then assesses its confidence. If the model shows high confidence, the conversation history is preserved and the model proceeds with the truthfulness evaluation. Conversely, if the model lacks confidence, the prior context is cleared and the model evaluates the statement's truthfulness independently. The workflow is depicted in Figure 4.3.

For the sake of clarity, each step in the process is addressed by a dedicated prompt, which contains a header that explicitly identifies the step currently in progress.

This technique allows to inspect the model's knowledge boundary, by evaluating whether the model's confidence assessment aligns with its accuracy.

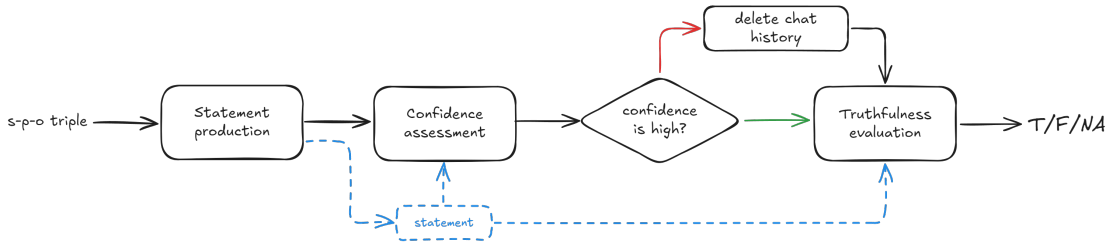

Figure 4.3: Workflow of the Stepwise pipeline.

5 Experimental Setup

5.1 Datasets

This section is dedicated to briefly describing the datasets used for the Accuracy Evaluation, namely, DBpedia, FactBench, NELL and YAGO. Table 5.1 reports the statistics of the datasets.

5.1.1 DBpedia

The sample of DBpedia [2] we used derives from the work of Marchesin et al. [27], which created a sample from the 2015-10 English version of DBpedia[1].

They first developed a utility model to assign triples (*i.e.*, facts) with a score, exploiting SPARQL[2] query logs. Utility is a task-specific metric, and in this scenario it reflects the popularity and the relevance of the facts.

They then stratified the KG according to the utility score, and applied TWCS [15] on the strata, obtaining a sample of 9344 triples.

5.1.2 FactBench

FactBench[3] [16] is a multilingual benchmark for the evaluation of fact validation algorithms, comprising 12060 facts spanning 10 predicates. As of December

[1] https://github.com/KGAccuracyEval/dbpedia-accuracy-estimation
[2] https://www.w3.org/TR/sparql11-query/
[3] https://github.com/DeFacto/FactBench

Dataset	Total # triples	# Correct triples	# Incorrect triples
DBpedia	9344	7949	1395
FactBench	2800	1500	1300
NELL	1860	1699	161
YAGO	1386	1375	11

Table 5.1: Statistics of the datasets.

2024, FactBench supports English, French and German. It contains both correct and wrong facts, alongside with their temporal validity. The temporal information can be either a timespan or a time in point, represented as an interval `start–end`. A time in point is considered as a timespan with the same start and end year, *e.g.*, `2008–2008`. The granularity of the temporal information is year.

The positive instances (*i.e.*, correct facts) are taken from DBpedia and Freebase [4] by issuing SPARQL and MQL[4] queries to their endpoint respectively, and selecting the top 150 results. Instead, the negative instances (*i.e.*, wrong facts) are generated synthetically by randomly replacing some items of the triples. Six negative sets are created, with the following criteria: (i) *domain* replaces the subject, (ii) *range* replaces the object, (iii) *domainrange* replaces both subject and object, (iv) *property* replaces the predicate, (v) *random* replaces all the items, and (vi) *date* replaces the time information. An additional negative set is built, called *mix*, that is a heterogeneous set consisting of $1/6$ of each of the above sets. It is ensured that the synthetic triples aren't already existing triples in the KG.

For our purposes, we considered only the English version of FactBench, ignoring the time information and focusing only on the facts themselves. To this end, we filtered the dataset to eliminate the "*wrong mix date*" subset of facts, obtaining a total of 2800 facts (1500 correct facts, 1300 wrong facts).

5.1.3 NELL

Never Ending Language Learner (NELL) derives from the "Never-ending learning" paradigm introduced by Mitchell et al. [30], which describes an agent capable to learn from the Web in a continuous and autonomous way. By their

[4]Now deprecated, covered briefly at `https://github.com/nchah/freebase-mql`

definition, the agent learns many types of knowledge from years of diverse and primarily self-supervised experience, using previously learned knowledge to improve subsequent learning and avoiding plateaus in performance as it learns.

Carlson et al. [7] built an early prototype of such a system, focusing on two main tasks: the extraction of information from the Web and "reading better each day than the day before". This system learns how to extract knowledge from both free-form text and semi-structured data, namely, sentences and tables or lists. Furthermore, it is able to capture morphological regularities of the extracted instances and to learn probabilistic Horn clauses to infer new knowledge.

NELL continuously crawls the Web, refining its knowledge day by day. Given an initial ontology, the authors let the prototype run for 67 consecutive days, populating the KG with 242453 facts. The knowledge is organized into categories (*e.g.*, *Athletes, Team, Writers*) and the system learns relationships between entities (*e.g.*, *teamPlaysSport, bookWriter*), adding new facts through category-based extraction methods.

We utilized the sample provided by Ojha and Talukdar [32], called *NELL-sports* (hereafter, NELL): a sub-graph of the NELL KG about athletes, teams, leagues, stadiums, etc. The sample includes 1860 facts across 18 predicates. The reported gold accuracy is 91.34%.

5.1.4 YAGO

Yet Another Great Ontology (YAGO), introduced in [46], is a lightweight and extensible ontology, aiming for high coverage and quality. It has been populated from Wikipedia and WordNet [29]. The entities it contains come from Wikipedia, and they have been extracted exploiting its inner structure: each Wikipedia page's title is an entity, as such titles are unique, and their type is inherited from the Wikipedia Category system.

However, Wikipedia categories follow the theme of the pages, creating a complex hierarchy that is not suitable for the construction of an ontology. This hierarchy can be represented as a directed acyclic graph, where only the leaves are the items of interest. To organize such categories, the authors used WordNet synsets as they provide an ontologically well-defined taxonomy.

Other relations are extracted with parsing methods (*e.g.*, to get per-

son names) or exploiting relational Wikipedia categories (*e.g.*, `bornInYear`, `establishedIn`, `hasWonPrize`). The results are then filtered to ensure all extracted facts are valid, meaning all the entities appearing in those facts must have a Wikipedia page.

For instance, `(Eugene_of_Savoy, politicianOf, Habsburg_Netherlands)` is excluded from YAGO because `Habsburg_Netherlands` does not exist as a page in Wikipedia.

YAGO2 [20] is a newer version of YAGO. It adds temporal, spatial and contextual dimensions to the facts, aspects that were largely missing in earlier KGs.

The temporal information is similar to what we presented when describing FactBench, indicating when a fact was valid. Differing from FactBench, the granularity of YAGO2 KG's time information is day. As a consequence, YAGO2 can now express facts like "Barack Obama was the President of United States from 2009-01-20 to 2017-01-20". The spatial information allows facts to be linked with geographical locations, enabling a fact to incorporate the geographical coordinates of the location it contains. Contextual information is added to improve querying on the KG, storing keywords and keyphrases that characterize entities.

These dimensions aid in extrinsic tasks, like Named Entity Disambiguation and Question Answering. Lastly, they are added with a conservative approach: the new information is added only when it's available or inferable, leaving some entities without a time (or location) scope and avoiding nonsensical deductions of their correct values. YAGO2 contains 124 millions of facts for 2.6 millions entities (excluding geographical entities) extracted from Wikipedia and WordNet, with a reported 95% precision (*i.e.*, absence of false positives).

As in NELL, we utilized the sample dataset created from YAGO2 KG by Ojha and Talukdar [32], called *YAGO2-sample* (hereafter, YAGO).

Unlike NELL, YAGO is not restricted to a specific domain. YAGO contains 1386 facts across 16 predicates, with a reported gold accuracy of 99.20%. Note that said sample does not include temporal information.

5.2 Models

We selected models in the 7–9 billion parameter range for our experiments, as they run faster and are less demanding in terms of computational resources than their larger counterparts. We evaluated four popular models: Gemma 2 9B[5] [48], Llama 3.1 8B[6] [13], Mistral 7B v0.3[7] [22], and Qwen2.5 7B[8] [49, 57]. For each of these, we considered the latest instruction-tuned version at the time of writing (December 2024).

5.3 Evaluation Measures

To evaluate the performance of the three pipelines (Baseline, Iterative and Stepwise), we considered both predictive accuracy and the peculiar aspects of the pipelines. To do so, we selected both standard and tailored metrics.

The standard metrics are applied across all pipelines, and include Accuracy (regular and balanced), Precision, Recall, F1 Score, the average processing time per fact, Informativeness and Truthfulness. We took inspiration from [25] for Informativeness and Truthfulness.

Before introducing their definition, it is useful to define some notation:

- **TP** are the True Positives: true facts classified as true;
- **TN** are the True Negatives: false facts classified as false;
- **FP** are the False Positives: false facts misclassified as true;
- **FN** are the False Negatives: true facts misclassified as false.

We can now provide a detailed explanation of each metric.

Definition 5.1 (Accuracy)**.** Accuracy represents the percentage of predictions that the model got right, computed by dividing the number of correct predic-

[5] `https://huggingface.co/google/gemma-2-9b-it`
[6] `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`
[7] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
[8] `https://huggingface.co/Qwen/Qwen2.5-7B-Instruct`

tions by the total number of predictions.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Accuracy is a straightforward metric, but it does have limitations, especially with imbalanced datasets, as in our case.

Definition 5.2 (Balanced Accuracy)**.** Balanced Accuracy extends the regular Accuracy by accounting for imbalanced classes in the dataset. This is done by averaging the accuracy of each class in the dataset.

$$\text{Balanced Accuracy} = \frac{\frac{\text{TP}}{\text{TP+FN}} + \frac{\text{TN}}{\text{TN+FP}}}{2}$$

Balanced Accuracy depicts the model's effectiveness, assigning each class the same weight.

Definition 5.3 (Precision)**.** Precision is the proportion of positive predictions that were actually correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision shows the reliability of the model's positive classification, and it is particularly useful when dealing with False Positives.

Definition 5.4 (Recall)**.** Recall is the proportion of actual positives that are successfully identified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Definition 5.5 (F1-Score)**.** F1-Score is the harmonic mean of Precision and Recall.

$$\text{F1-Score} = \frac{\text{PR}}{\text{P} + \text{R}}$$

Definition 5.6 (Informativeness)**.** Informativeness [25] measures the model's ability to provide an informative response, *i.e.*, everything but "NA".

Definition 5.7 (Truthfulness)**.** Truthfulness [25] measures the model's ability to provide "honest" answers, *i.e.*, the correct label or "NA".

Definition 5.8 (Average Processing Time per Fact)**.** Average Processing Time per Fact is the average time the model needed to process a single fact. It is measured in seconds.

The Baseline pipeline is evaluated only by means of the standard metrics.

5.3.1 Iterative pipeline

The evaluation of the Iterative pipeline adds Compliance [61] and average number of retries per fact.

Definition 5.9 (Compliance)**.** Following [27], Compliance measures the model's ability to provide a compliant answer on the first try, without taking extra attempts.

Definition 5.10 (Average Number of Retries per Fact)**.** Average Number of Retries per Fact measures the average number of retries the model needed to provide a compliant answer.

5.3.2 Stepwise pipeline

For the Stepwise pipeline, as the pipeline is based on the work of Ren et al. [41] so are the metrics, with the Statement Rate being the only addition we introduced.

Definition 5.11 (Statement Rate)**.** Statement Rate is the proportion of facts for which the model has been able to generate a statement, *i.e.*, it did not answer with "IDK".

Definition 5.12 (Give-up Rate)**.** Give-up rate is the proportion of answers where the model gave up, *i.e.*, it showed low confidence in evaluating the statement.

Definition 5.13 (Right|G)**.** Right|G is the proportion of facts for which the model answered correctly when giving up, *i.e.*, when showing low confidence.

Definition 5.14 (Right|¬G)**.** Right|¬G is the proportion of facts for which the model answered correctly when not giving up, *i.e.*, when showing high confidence.

5.4 Hardware Specifications

The experiments were conducted on a machine with the following hardware specifications:

- **Model Name**: Mac Studio
- **Model Identifier**: Mac14,14
- **Model Number**: Z180000M3T/A
- **Chip**: Apple M2 Ultra
- **Total Number of Cores**: 24 (16 performance and 8 efficiency)
- **Memory**: 192 GB
- **System Firmware Version**: 11881.41.5
- **OS Loader Version**: 11881.41.5

Note that the hardware used significantly impacts processing times.

6

Evaluation and Analysis

This chapter presents and discusses the results obtained from the three pipelines. For each pipeline, we will start from the standard metrics, followed by the tailored metrics and finally the time measurements. We report results in table form, offering precise numerical values. Additionally, for the standard metrics, we provide plots for a more intuitive understanding.

As for the measurement units, standard and tailored metrics are measured in percentages up to the fourth decimal place; average times are measured in seconds up to the third decimal place; and total times are presented in the format `HH:MM:SS`, in order to maintain consistency across pipelines.

It is worth noting that Informativeness metric consistently exhibits very high values across all experiments and Truthfulness very frequently resembles Accuracy. To ensure clarity, these two metrics are excluded from the plots and are instead reported solely in the tables.

In the tables reporting the standard metrics, the highest scores for each metric and KG are highlighted in bold. We highlighted values only in Accuracy, Balanced Accuracy, Precision, Recall and F1 Score, leaving out Informativeness and Truthfulness. Similarly, in the tables reporting time measurements, we highlighted in bold the most efficient models.

6.1 Baseline Pipeline

This section discusses the performance of the models evaluated over the Baseline pipeline, across all the datasets. The analysis focuses only on the standard metrics, as there are no tailored metrics for the Baseline. Table 6.1 reports the results, and Figure 6.1 visualizes said tabular data.

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	**0.7522**	**0.6369**	0.8964	**0.8014**	**0.8462**	1.0000	0.7522
Llama3.1	0.6987	0.5839	0.8802	0.7476	0.8085	1.0000	0.6987
Mistral	0.6802	0.6315	0.9012	0.7010	0.7886	1.0000	0.6802
Qwen2.5	0.5216	0.6266	**0.9240**	0.4769	0.6291	1.0000	0.5216
	FactBench						
Gemma2	**0.7457**	**0.7491**	0.7989	**0.7020**	**0.7473**	1.0000	0.7457
Llama3.1	0.7368	0.7415	0.8016	0.6760	0.7335	1.0000	0.7368
Mistral	0.7086	0.7192	0.8327	0.5707	0.6772	1.0000	0.7086
Qwen2.5	0.6493	0.6684	**0.8776**	0.4013	0.5508	1.0000	0.6493
	NELL						
Gemma2	**0.7522**	**0.7097**	0.9592	**0.7610**	**0.8487**	1.0000	0.7522
Llama3.1	0.5855	0.6100	0.9444	0.5803	0.7189	1.0000	0.5855
Mistral	0.6075	0.6699	**0.9610**	0.5945	0.7345	1.0000	0.6075
Qwen2.5	0.3468	0.5778	0.9566	0.2984	0.4549	1.0000	0.3468
	YAGO						
Gemma2	**0.6977**	0.5320	0.9928	**0.7004**	**0.8213**	1.0000	0.6977
Llama3.1	0.5491	0.5473	0.9934	0.5491	0.7073	1.0000	0.5491
Mistral	0.4221	0.4382	0.9898	0.4218	0.5915	1.0000	0.4221
Qwen2.5	0.2734	**0.5887**	**0.9973**	0.2684	0.4229	1.0000	0.2734

Table 6.1: Standard metrics for the Baseline pipeline. For each metric and each KG, the highest scores are highlighted in bold.

As an overview, the performance of the models varies significantly across datasets, with Gemma2 consistently being the top performer, achieving the highest Accuracy score, and Qwen2.5 showing the lowest performance. However, all models maintain high Precision, demonstrating a high degree of reliability for the positives.

Informativeness always reaches 1.0, highlighting that all models on all datasets succeed in following the minimal instruction comprised in the Baseline prompt, hence always answering with one of the expected labels (”T”, ”F”).

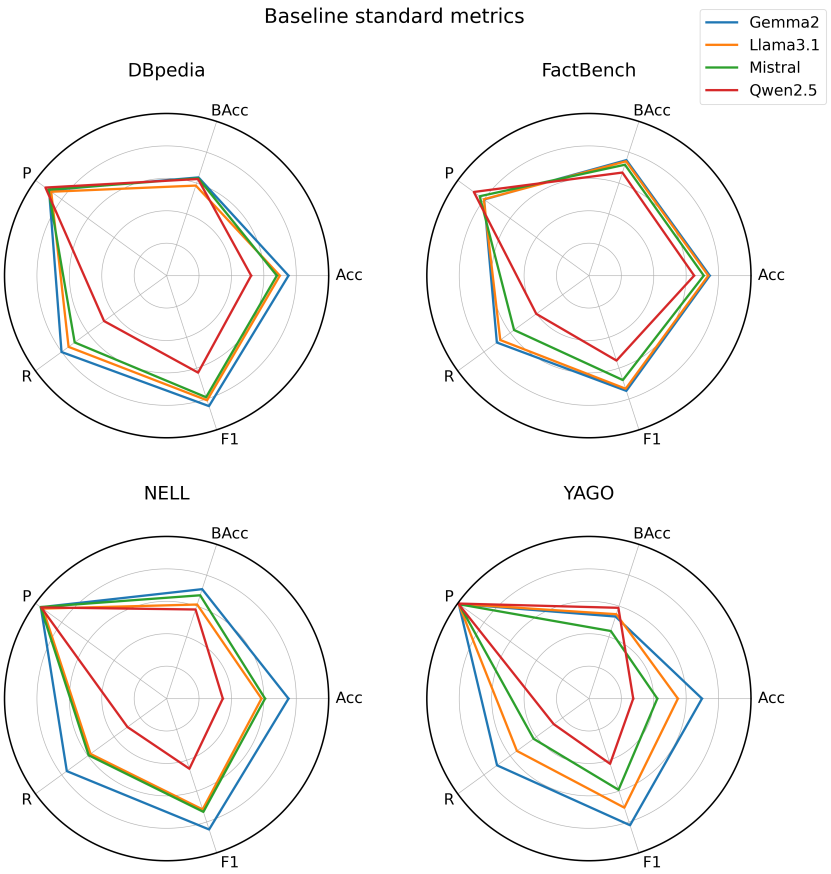

Figure 6.1: Visualization of the Standard metrics for the Baseline pipeline.

As a consequence, Truthfulness values are the same of Accuracy, as there aren't any "NA" values contributing to it.

6.1.1 Standard metrics

DBpedia

In DBpedia, Gemma2 achieves top Accuracy (75.22%), Balanced Accuracy and Recall. Performance-wise, Llama3.1 and Mistral follow, presenting very small gaps between them. Although Qwen2.5 exhibits the highest Precision (92.40%), its Accuracy and Recall are the lowest among all models, indicating that it struggles with imbalanced datasets. Furthermore, high precision and low recall show that Qwen2.5 is very cautious in predicting positives, responding with "T" only when it's highly confident about the fact in exam, and that it handles false positives fairly well. At the same time, it predicts many false negatives.

However, the Balanced Accuracy hovers around 60% for all models, showing that their predictions contain many more false negatives than they should, while the high Precision indicates that the models are generally trustworthy when judging a fact accurate.

FACTBENCH

For FactBench, most models improve in Accuracy and Balanced Accuracy with respect to DBpedia. Gemma2 is still the top performer, but the gap with the other models is now narrower.

Compared to DBpedia, the Balanced Accuracy shows consistent gains, while the other metrics observed a decrease. Despite the narrower gap, Qwen2.5 is still the worst of all models, while Llama3.1 gets closer to Gemma2 rather than Mistral.

The observed gains in Balanced Accuracy and drops in the other metrics can be attributed to the dataset's balance and the synthetic nature of the negative triples, which makes them easier to identify.

NELL

NELL sees Gemma2 as the best model accuracy-wise, again. Gemma2 achieves also the highest Balanced Accuracy and Recall, demonstrating the effectiveness and coverage of its positive predictions.

Here, the gap with the other models becomes even larger. Llama3.1 and Mistral are still performing comparably, achieving similar Accuracy, Precision and Recall, with only Balanced Accuracy in slight favor of Mistral.

Qwen2.5 faces significant challenges, with only 34.68% Accuracy and poor Recall (29.84%). Interestingly enough, Balanced Accuracy is not far away from the second lowest score (Llama3.1) indicating the model predicted lots of false negatives. Considering that NELL is heavily imbalanced, the amount of negative predictions helped in boosting the Balanced Accuracy score. Still, Qwen2.5 achieves competitive Precision.

YAGO

YAGO is another heavily imbalanced dataset, and the results reveal the same trends observed with NELL, but with an even more pronounced disparity in model performance. Once again, Gemma2 stands as the best model (69.77%

Accuracy) and Qwen2.5 as the worst (27.34% Accuracy). Balanced Accuracy is lower than that of the other datasets, presenting near-chance level scores. Precision is near-perfect for every model, confirming the reliability of their positive predictions, while we can observe significant trades in Recall, suggesting that models present limitations in correctly identifying positive instances.

Interestingly, Qwen2.5 achieves top Balanced Accuracy among all models, but the gap between its regular Accuracy and Balanced Accuracy reflects the trends seen with NELL, *i.e.*, given the very low number of negative instances and the high number of negative predictions, it is very likely that Qwen2.5 correctly classified the few negative instances, significantly increasing its Balanced Accuracy score.

6.1.2 Time measurements

Model	Fact Avg. Time	Total Time
	DBpedia	
Gemma2	0.364	00:56:41
Llama3.1	0.373	00:58:05
Mistral	**0.248**	**00:38:37**
Qwen2.5	0.281	00:43:45
	FactBench	
Gemma2	0.215	00:10:02
Llama3.1	0.297	00:13:51
Mistral	**0.172**	**00:08:01**
Qwen2.5	0.213	00:09:56
	NELL	
Gemma2	0.217	00:06:43
Llama3.1	0.302	00:09:21
Mistral	**0.184**	**00:05:42**
Qwen2.5	0.211	00:06:32
	YAGO	
Gemma2	0.224	00:05:10
Llama3.1	0.309	00:07:08
Mistral	**0.186**	**00:04:17**
Qwen2.5	0.213	00:04:55

Table 6.2: Time measurements for the Baseline pipeline.

Table 6.2 reports the average processing time per fact and the total time

needed to complete the evaluation task. The average time remains consistent across datasets, with a slight increase in DBpedia. This can be attributed to longer triple contents, for example `abstract`, `rdf-schema#comment` and `shortsummary` triples, which carry an entire paragraph in the object. As expected, the total time increases accordingly to the size of the datasets.

6.2 Iterative Pipeline

This section is dedicated to the results obtained from the Iterative pipeline. We will start from the standard metrics, proceeding with the tailored ones and the time measurements. Eventually, we will analyze the differences in performance between the Iterative and Baseline pipelines, and between the variants of the Iterative itself, with particular attention to the few-shot scenario.

Considering the amount of variants we have tested for the prompt of this pipeline, we introduce a shorthand notation to easily identify each of them. The notation is summarized in Table 6.3 below.

Variation	In-context learning	Guidelines format	Fact format
ZNI	zero-shot	natural	inline
FNI	few-shot	natural	inline
ZSN	zero-shot	structured	noprefix
FSN	few-shot	structured	noprefix

Table 6.3: Notation for the variants of the Iterative prompts.

It is noteworthy that the Standard metrics are computed on the last answer the model provided, regardless of how many attempts the model needed *i.e.*, we considered the first answer meeting the guidelines within the allowed attempts, or "NA" for the cases in which the model exhausted all attempts. The only metrics focusing the number of attempts are Compliance and Average Number of Retries per Fact.

Similarly to the Baseline, Informativeness consistently remains at or near 1.0 across all variants, with only occasional minor deviations. It follows that Truthfulness closely mirrors Accuracy.

6.2.1 Standard metrics

ZNI Variant

Models	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	0.7109	**0.6481**	0.9049	0.7377	0.8128	1.0000	0.7109
Llama3.1	0.4379	0.6008	0.9263	0.3686	0.5274	1.0000	0.4379
Mistral	**0.7823**	0.5502	0.8654	**0.8811**	**0.8732**	1.0000	0.7823
Qwen2.5	0.5264	0.6326	**0.9270**	0.4812	0.6335	0.9999	0.5264
	FactBench						
Gemma2	0.7318	0.7355	0.7879	0.6833	0.7319	1.0000	0.7318
Llama3.1	0.6332	0.6528	0.8578	0.3780	0.5248	1.0000	0.6332
Mistral	**0.7454**	**0.7422**	0.7505	**0.7860**	**0.7678**	1.0000	0.7454
Qwen2.5	0.6261	0.6467	**0.8647**	0.3580	0.5064	1.0000	0.6261
	NELL						
Gemma2	0.8710	**0.6342**	0.9371	0.9205	0.9287	1.0000	0.8710
Llama3.1	0.2452	0.5278	0.9377	0.1860	0.3104	1.0000	0.2452
Mistral	**0.8715**	0.5867	0.9284	**0.9311**	**0.9298**	1.0000	0.8715
Qwen2.5	0.5075	0.6011	**0.9474**	0.4879	0.6441	1.0000	0.5075
	YAGO						
Gemma2	**0.7908**	0.5789	0.9936	**0.7942**	**0.8828**	1.0000	0.7908
Llama3.1	0.3564	0.5855	0.9959	0.3527	0.5209	1.0000	0.3564
Mistral	0.6039	0.5298	0.9928	0.6051	0.7519	1.0000	0.6039
Qwen2.5	0.3680	**0.6364**	**0.9980**	0.3636	0.5330	1.0000	0.3680

Table 6.4: Standard metrics for the Iterative pipeline (ZNI variant). For each metric and each KG, the highest scores are highlighted in bold.

Results for Iterative ZNI are presented in Table 6.4 and Figure 6.2.

Figure 6.2 highlights two subclasses in performance: Gemma2 and Mistral on the higher end, and Qwen2.5 and Llama3.1 on the lower end. Mistral outperforms Gemma2 on DBpedia and FactBench, while Gemma2 performs better (or behaves as good as Mistral) on NELL and YAGO. Balanced Accuracy is the metric in which Gemma2 consistently does better than Mistral, although the gap is relatively small.

The plots make also clear that Llama3.1 achieves very poor results in Accuracy and Recall compared to the other models, being no better than Qwen2.5 (which was the worst model for every dataset in the Baseline). The most challenging dataset for Llama3.1 is NELL, with 24.52% Accuracy and 18.60%

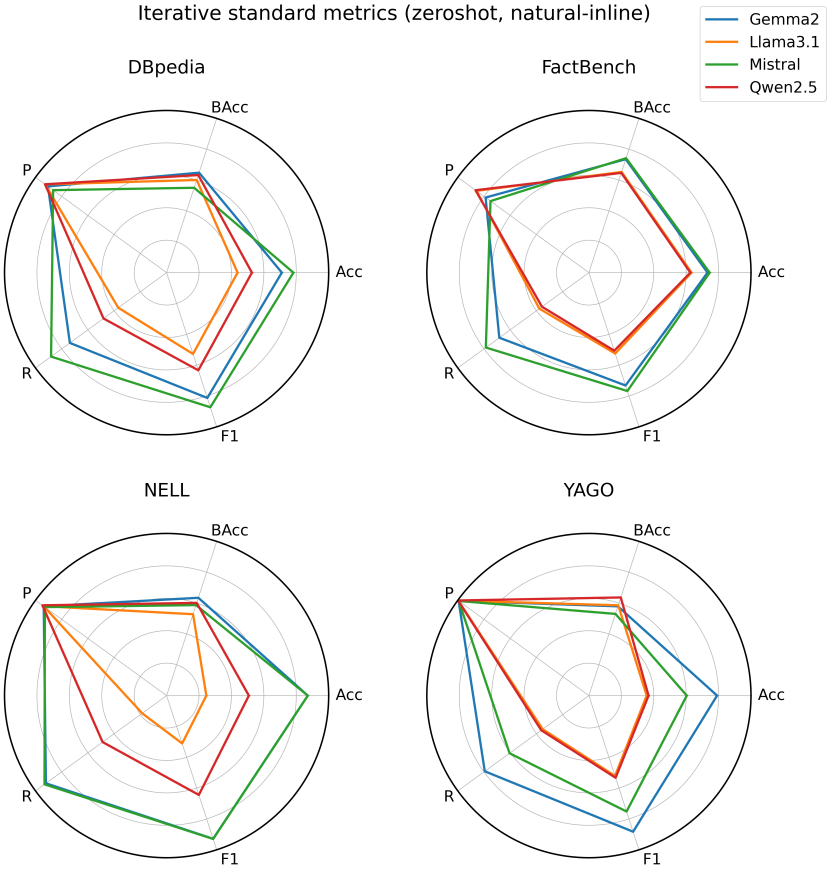

Figure 6.2: Visualization of the Standard metrics for the Iterative pipeline (ZNI variation).

Recall. However, Llama3.1 still achieves high Precision, revealing the same cautiousness Qwen2.5 showed in the Baseline. Notably, the Balanced Accuracy of Llama3.1 doesn't show major discrepancies from those of other models, even though it is on the lower end. This stands for a lot of false negative predictions. The same conclusion can be drawn also for the other models (although to a lesser extent for Gemma2 and Mistral). Despite the poor performance, the perfect Informativeness shows that Llama3.1 still follows instructions.

As observed in the Baseline, Precision always reaches very high scores, decreasing a bit only in FactBench. We argue this pattern using the same reasoning as for the Baseline, identifying the cause of the lower performance in the dataset's balance and synthetic nature of its negative instances.

FNI Variant

Models	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	0.7534	0.6281	0.8931	0.8068	0.8477	1.0000	0.7534
Llama3.1	0.5791	0.6179	0.9075	0.5626	0.6946	1.0000	0.5791
Mistral	**0.8070**	0.5420	0.8625	**0.9199**	**0.8902**	1.0000	0.8070
Qwen2.5	0.6702	**0.6519**	**0.9117**	0.6779	0.7776	1.0000	0.6702
	FactBench						
Gemma2	**0.7746**	**0.7745**	**0.7974**	0.7767	0.7869	1.0000	0.7746
Llama3.1	0.7325	0.7327	0.7609	0.7300	0.7452	1.0000	0.7325
Mistral	**0.7746**	0.7666	0.7459	**0.8787**	**0.8069**	1.0000	0.7746
Qwen2.5	0.7339	0.7373	0.7871	0.6900	0.7353	1.0000	0.7339
	NELL						
Gemma2	0.8349	**0.7466**	**0.9615**	0.8534	0.9043	1.0000	0.8349
Llama3.1	0.8812	0.6679	0.9430	0.9258	0.9344	1.0000	0.8812
Mistral	**0.9022**	0.6372	0.9367	**0.9576**	**0.9470**	1.0000	0.9022
Qwen2.5	0.8317	0.6999	0.9518	0.8593	0.9032	1.0000	0.8317
	YAGO						
Gemma2	**0.8593**	0.5233	0.9925	**0.8647**	**0.9242**	1.0000	0.8593
Llama3.1	0.7071	0.5818	0.9939	0.7091	0.8277	1.0000	0.7071
Mistral	0.8240	0.4604	0.9913	0.8298	0.9034	1.0000	0.8240
Qwen2.5	0.5635	**0.6447**	**0.9961**	0.5622	0.7187	1.0000	0.5635

Table 6.5: Standard metrics for the Iterative pipeline (FNI variant). For each metric and each KG, the highest scores are highlighted in bold.

Using the same arrangement of the other sections, we provide exact values in Table 6.5. Figure 6.3 points out a more homogeneous performance distribution in the few-shot scenario, especially on FactBench and NELL datasets, with the largest gaps happening on DBpedia and YAGO. All models achieve high Precision, with the usual decrease observed on FactBench (which has already been discussed).

FactBench and NELL see all models perform equivalently, with Mistral surpassing every other model in Recall. The Balanced Accuracy fluctuates around 75% for FactBench and gets a little lower for NELL, showing the modest effectiveness of the models. Recall is another metric with the highest and most consistent results, ranging from 70% to 80% on FactBench, and from 85% to

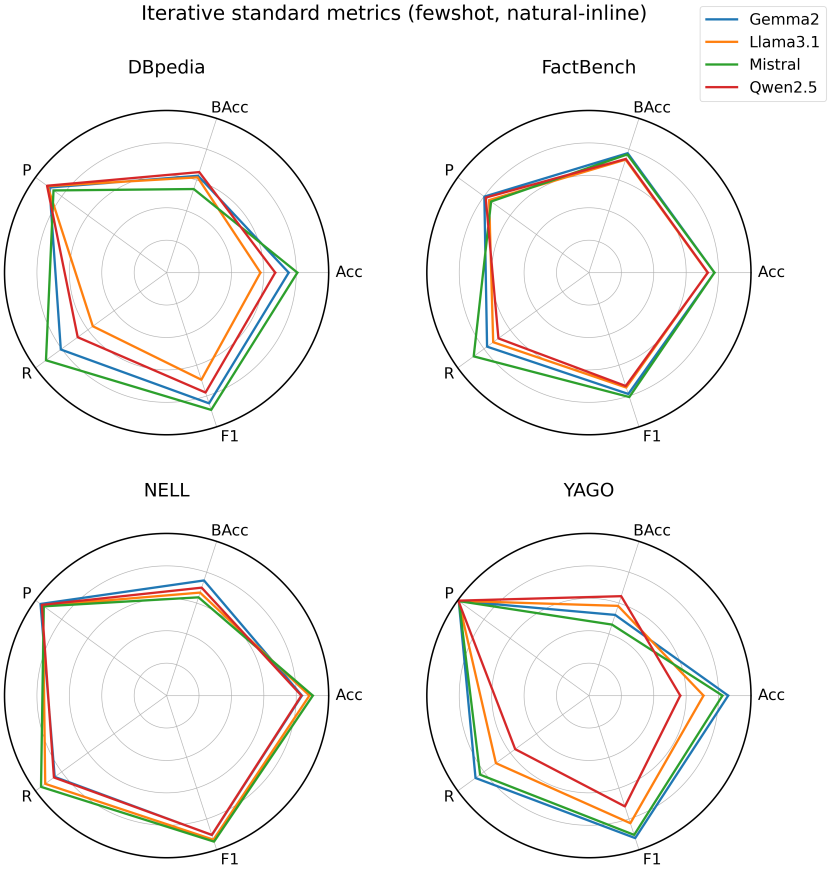

Figure 6.3: Visualization of the Standard metrics for the Iterative pipeline (FNI variation).

95% on NELL.

In DBpedia and YAGO we still observe the same two classes of performance as in the zero-shot setting. Here, Qwen2.5 achieves the best Balanced Accuracy, performing as the second best model on DBpedia. For what concerns YAGO, Qwen2.5 is weaker in Recall, indicating the presence of false negatives.

Llama3.1 received a huge boost in every metric in the few-shot scenario, gaining ground against the other models and proving the effectiveness of few-shot prompting. Considering that its Informativeness has not changed, we assume the examples we provided helped the model a lot in determining the real veracity of the facts that underwent the evaluation process.

ZSN Variant

Models	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	0.8099	0.6220	0.8870	0.8899	0.8885	1.0000	0.8099
Llama3.1	0.3799	0.5835	**0.9298**	0.2932	0.4459	1.0000	0.3799
Mistral	**0.8130**	0.5384	0.8613	**0.9299**	**0.8943**	1.0000	0.8130
Qwen2.5	0.6231	**0.6455**	0.9155	0.6135	0.7347	0.9997	0.6232
	FactBench						
Gemma2	**0.7807**	**0.7766**	0.7741	0.8340	0.8030	1.0000	0.7807
Llama3.1	0.6146	0.6373	**0.8905**	0.3200	0.4708	1.0000	0.6146
Mistral	0.7462	0.7279	0.6882	**0.9667**	**0.8040**	0.9950	0.7475
Qwen2.5	0.7179	0.7266	0.8216	0.6047	0.6966	1.0000	0.7179
	NELL						
Gemma2	0.8817	0.5389	0.9199	0.9535	0.9364	1.0000	0.8817
Llama3.1	0.1737	0.5196	**0.9451**	0.1012	0.1829	1.0000	0.1737
Mistral	**0.9016**	0.5244	0.9174	**0.9806**	**0.9479**	1.0000	0.9016
Qwen2.5	0.8188	**0.6084**	0.9338	0.8629	0.8969	1.0000	0.8188
	YAGO						
Gemma2	0.8319	0.5095	0.9922	0.8371	0.9081	1.0000	0.8319
Llama3.1	0.2648	**0.6295**	**1.0000**	0.2589	0.4113	1.0000	0.2648
Mistral	**0.9192**	0.4633	0.9914	**0.9265**	**0.9579**	1.0000	0.9192
Qwen2.5	0.4769	0.6011	0.9954	0.4749	0.6430	1.0000	0.4769

Table 6.6: Standard metrics for the Iterative pipeline (ZSN variant). For each metric and each KG, the highest scores are highlighted in bold.

Even with the change in the prompt style, Figure 6.4 illustrates the performance disparity distancing Gemma2 and Mistral from Llama3.1 and Qwen2.5 in most datasets, with NELL as the only exception. It also outlines the unsatisfactory results of Llama3.1, reflecting the ones seen in the ZNI variant. As usual, the precise numerical scores are described in Table 6.6.

Gemma2 and Mistral achieve very similar performance in every dataset, with Mistral shining over Gemma2 in all metrics but Balanced Accuracy. Conversely to the other cases, this decrease is caused by the abundance of false positives, confirmed by the high Recall values.

Llama3.1 shows again generalized underperformance, achieving even worse

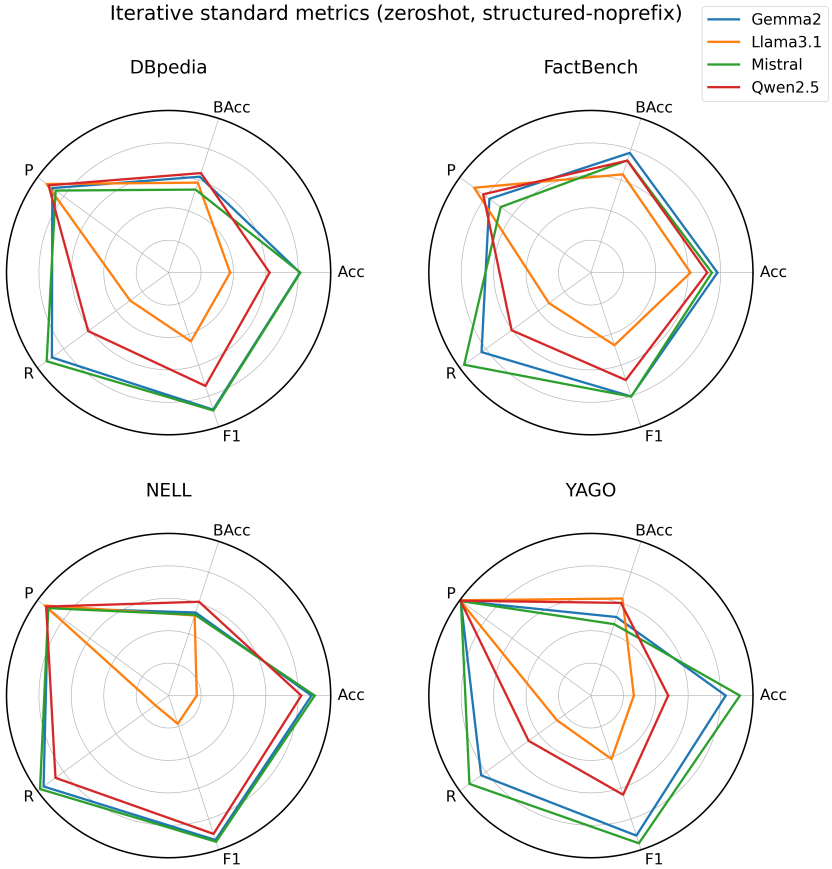

Figure 6.4: Visualization of the Standard metrics for the Iterative pipeline (ZSN variation).

Accuracy results than before. As in the ZNI variant, the Balanced Accuracy didn't suffer any significant degradation, proving again that the model answers with "T" very carefully. This is further justified by the fact that it achieved the highest Precision among all models, though its Recall is subpar.

Balanced Accuracy, with the exception of FactBench, still shows considerable room for improvement. Indeed, it remains close to 60% on DBpedia, hence performing better than random guessing, while NELL and YAGO exhibit values closer to 50%.

FSN Variant

Models	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	0.7845	**0.6514**	0.8990	0.8411	0.8691	1.0000	0.7845
Llama3.1	0.4249	0.5905	**0.9209**	0.3544	0.5118	1.0000	0.4249
Mistral	**0.8230**	0.5520	0.8650	**0.9384**	**0.9002**	1.0000	0.8230
Qwen2.5	0.6362	0.6497	0.9156	0.6305	0.7468	1.0000	0.6362
	FactBench						
Gemma2	**0.8054**	**0.8039**	**0.8148**	0.8240	**0.8194**	1.0000	0.8054
Llama3.1	0.7289	0.7344	0.8005	0.6580	0.7223	1.0000	0.7289
Mistral	0.7582	0.7426	0.6997	**0.9613**	0.8099	1.0000	0.7582
Qwen2.5	0.7336	0.7367	0.7847	0.6927	0.7358	1.0000	0.7336
	NELL						
Gemma2	0.8575	0.7055	0.9515	0.8893	0.9194	1.0000	0.8575
Llama3.1	0.8629	0.6719	0.9446	0.9029	0.9233	1.0000	0.8629
Mistral	**0.8989**	0.5848	0.9276	**0.9647**	**0.9458**	1.0000	0.8989
Qwen2.5	0.8145	**0.7157**	**0.9562**	0.8352	0.8916	1.0000	0.8145
	YAGO						
Gemma2	0.8766	0.5320	0.9926	0.8822	0.9342	1.0000	0.8766
Llama3.1	0.6797	**0.5680**	**0.9936**	0.6815	0.8085	1.0000	0.6797
Mistral	**0.9654**	0.4865	0.9918	**0.9731**	**0.9824**	1.0000	0.9654
Qwen2.5	0.6573	0.5567	0.9934	0.6589	0.7923	1.0000	0.6573

Table 6.7: Standard metrics for the Iterative pipeline (FSN variant). For each metric and each KG, the highest scores are highlighted in bold.

Results are reported in Table 6.7 and Figure 6.5. As in the FNI variant, the performance is more homogeneous in some datasets, namely FactBench and NELL. Again, Llama3.1 shows a notable improvement, despite remaining on the lower side, confirming again the effectiveness of the few-shot prompting.

In DBpedia and YAGO, the performance of Gemma2 and Mistral continues to diverge from that of Llama3.1 and Qwen2.5, with Llama3.1 still significantly lacking behind other models on DBpedia. Its low Recall shows that it struggles to identify positives, resulting in many false negative predictions, while its Precision presents widely high values.

It's worth to notice that Qwen2.5 in DBpedia performs just as good as in

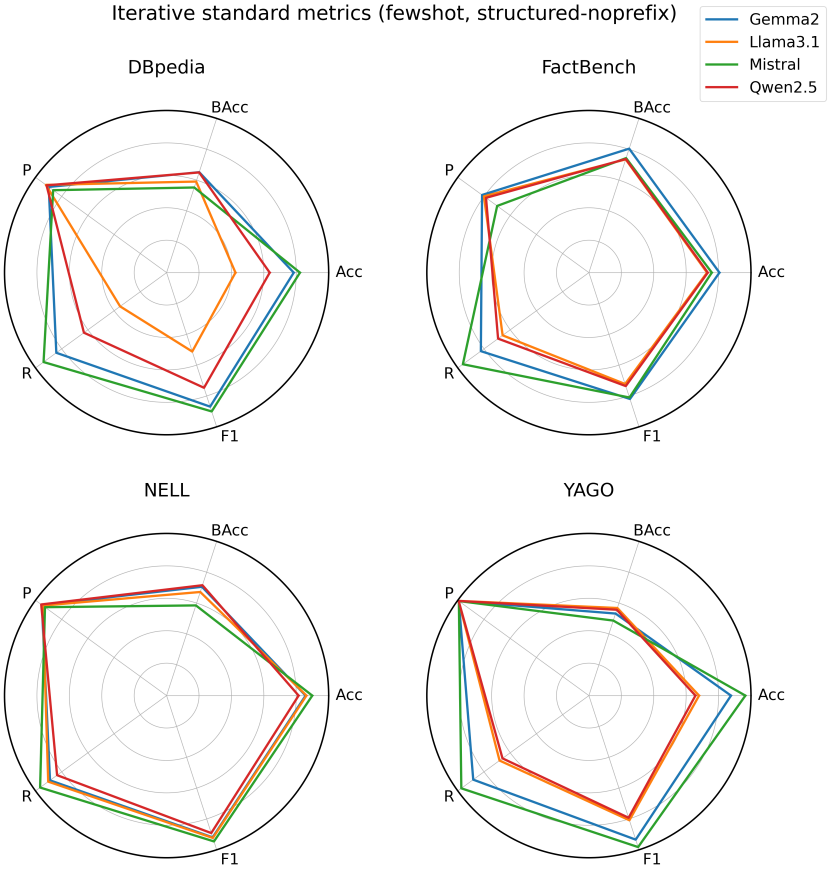

Figure 6.5: Visualization of the Standard metrics for the Iterative pipeline (FSN variation).

the ZSN setting, while it is improving when considering YAGO. In general, it achieves the lowest Recall, while its Balanced Accuracy is quite good: considering the imbalance of the dataset, this signals the presence of false negatives.

6.2.2 Tailored metrics

For the Iterative pipeline we have two tailored metrics, that are, Compliance and Average Number of Retries per Fact. Both of these address the instruction-following capabilities of the models, which are pushed even further by the retry mechanism of the Iterative prompt. We tested this pipeline allowing for 3 additional attempts.

The two metrics are inherently connected: achieving a perfect Compliance score would require the average number of retries to approach zero. On the other side, a low average number of retries indicates strong compliance with

the instructions provided.

Unlike the standard metrics, we present tailored results organized by prompt style, *i.e.*, NI and SN.

NI Variants

Models	Compliance	Avg. retries
	DBpedia	
Gemma2	0.9939	0.0067
Llama3.1	0.9997	0.0003
Mistral	0.9998	0.0002
Qwen2.5	0.9549	0.0619
	FactBench	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	0.9989	0.0011
	NELL	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	1.0000	0.0000
	YAGO	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	1.0000	0.0000

(a) Tailored metrics for the Iterative pipeline (ZNI variant).

Models	Compliance	Avg. retries
	DBpedia	
Gemma2	0.9979	0.0046
Llama3.1	0.9922	0.0081
Mistral	1.0000	0.0000
Qwen2.5	0.9964	0.0060
	FactBench	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	1.0000	0.0000
	NELL	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	1.0000	0.0000
	YAGO	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	1.0000	0.0000

(b) Tailored metrics for the Iterative pipeline (FNI variant).

Table 6.8: Tailored metrics for the Iterative pipeline (NI variants).

Although the dominance of high values, some flaws are present on DBpedia in the zero-shot setting. Qwen2.5 reveals itself as the most critical model in this regard. Its DBpedia Informativeness (reported in Tables 6.4 and 6.5) indicates that the model failed to evaluate some of the facts within the total number of attempts. For what concerns FactBench, Qwen2.5 achieved an Informativeness of 1.0, indicating that the model just needed some more attempts to provide

one of the expected labels. The latter observation holds true also for the other models in DBpedia, considering their perfect Informativeness scores.

Moving onto the few-shot setting, we can observe that while some flaws persist, they are substantially reduced. Pairing that with the perfect Informativeness achieved by every model on every dataset, we draw the same conclusion as before, acknowledging some extra attempt may be required.

For both scenarios, the most challenging predicates for all models are `thumbnail`, `point`, `depiction`, `page`, `homepage` and `name`. Qwen2.5 also adds the `subject` predicate to its list.

SN Variants

The results obtained from the tailored metrics are presented in Table 6.9. As for the NI variants, Compliance presents only very high values. In the zero-shot setting, Qwen2.5 shows some minor flaws on DBpedia, FactBench and YAGO. Following the same analysis as before, its Informativeness (reported in Table 6.6) reveals the presence of some "NA" values on DBpedia, as well as the need for extra attempts in FactBench and YAGO. Gemma2 exhibits the need for extra attempts on DBpedia too, while Mistral generated some invalid responses on FactBench.

The few-shot setting mostly brought improvements for Mistral and Qwen2.5. Gemma2 on DBpedia doesn't improve, and actually get slightly worse results on YAGO. Gemma2 achieves the same score as in the zero-shot setting, taking more than one attempt on 5 triples. Interestingly, a brief inspection on the results shows that only 2 of these triples were shared between the two scenarios.

Given the perfect Informativeness score in the FSN setting (see Table 6.7), we can infer the absence of invalid responses.

An inspection of the results reveals an interesting pattern across all variants: when a model required more than one attempt to provide a valid answer, it most frequently classified the fact as false.

Models	Compliance	Avg. retries
	DBpedia	
Gemma2	0.9995	0.0005
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	0.9837	0.0238
	FactBench	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	0.9950	0.0150
Qwen2.5	0.9996	0.0004
	NELL	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	1.0000	0.0000
	YAGO	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	0.9993	0.0007

(a) Tailored metrics for the Iterative pipeline (ZSN variant).

Models	Compliance	Avg. retries
	DBpedia	
Gemma2	0.9995	0.0005
Llama3.1	0.9999	0.0001
Mistral	1.0000	0.0000
Qwen2.5	0.9990	0.0010
	FactBench	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	1.0000	0.0000
	NELL	
Gemma2	1.0000	0.0000
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	1.0000	0.0000
	YAGO	
Gemma2	0.9993	0.0007
Llama3.1	1.0000	0.0000
Mistral	1.0000	0.0000
Qwen2.5	1.0000	0.0000

(b) Tailored metrics for the Iterative pipeline (FSN variant).

Table 6.9: Tailored metrics for the Iterative pipeline (SN variants).

6.2.3 Time measurements

As for the tailored metrics, we organize the discussion of the time measurements by prompt style. Following the approach adopted for the Baseline, we report average and total times.

NI Variants

Models	Fact Avg. Time	Total Time
	DBpedia	
Gemma2	0.789	02:02:52
Llama3.1	0.630	01:38:06
Mistral	0.537	01:23:37
Qwen2.5	**0.488**	**01:15:59**
	FactBench	
Gemma2	0.636	00:29:40
Llama3.1	0.576	00:26:52
Mistral	0.457	00:21:19
Qwen2.5	**0.405**	**00:18:54**
	NELL	
Gemma2	0.775	00:24:01
Llama3.1	0.559	00:17:19
Mistral	**0.463**	**00:14:21**
Qwen2.5	0.514	00:15:56
	YAGO	
Gemma2	0.621	00:14:20
Llama3.1	0.452	00:10:26
Mistral	0.471	00:10:52
Qwen2.5	**0.415**	**00:09:35**

(a) Time measurements for the Iterative pipeline (ZNI variation).

Models	Fact Avg. Time	Total Time
	DBpedia	
Gemma2	0.902	02:20:28
Llama3.1	0.756	01:57:44
Mistral	0.866	02:14:51
Qwen2.5	**0.573**	**01:29:14**
	FactBench	
Gemma2	0.771	00:35:58
Llama3.1	0.741	00:34:34
Mistral	0.654	00:30:31
Qwen2.5	**0.516**	**00:24:04**
	NELL	
Gemma2	0.794	00:24:36
Llama3.1	0.734	00:22:45
Mistral	0.665	00:20:36
Qwen2.5	**0.663**	**00:20:33**
	YAGO	
Gemma2	0.782	00:18:03
Llama3.1	0.750	00:17:19
Mistral	0.666	00:15:23
Qwen2.5	**0.541**	**00:12:29**

(b) Time measurements for the Iterative pipeline (FNI variant).

Table 6.10: Time measurements for the Iterative pipeline (both NI variants).

Time measurements for the Iterative pipeline in the NI variants are reported in Table 6.10. While times remain consistent across datasets, indicating that models needs the same time to process the Iterative prompt regardless of the triple, they are longer than in the Baseline. This is due to the increased prompt length, that now comprises instructions.

The effects of the prompt length are even more noticeable in the few-shot settings: the prompt is enriched with a few examples, increasing the number of tokens models need to process. This results in slightly higher processing times.

SN Variants

Models	Fact Avg. Time	Total Time
	DBpedia	
Gemma2	0.816	02:07:04
Llama3.1	0.646	01:40:36
Mistral	0.635	01:38:53
Qwen2.5	**0.497**	**01:17:23**
	FactBench	
Gemma2	0.638	00:29:46
Llama3.1	0.640	00:29:52
Mistral	0.720	00:33:36
Qwen2.5	**0.415**	**00:19:22**
	NELL	
Gemma2	0.840	00:26:02
Llama3.1	0.633	00:19:37
Mistral	0.638	00:19:46
Qwen2.5	**0.513**	**00:15:54**
	YAGO	
Gemma2	0.646	00:14:55
Llama3.1	0.671	00:15:30
Mistral	0.662	00:15:17
Qwen2.5	**0.420**	**00:09:42**

(a) Time measurements for the Iterative pipeline (ZSN variant).

Models	Fact Avg. Time	Total Time
	DBpedia	
Gemma2	0.970	02:31:03
Llama3.1	0.876	02:16:25
Mistral	0.820	02:07:42
Qwen2.5	**0.617**	**01:36:05**
	FactBench	
Gemma2	0.839	00:39:09
Llama3.1	0.752	00:35:05
Mistral	0.849	00:39:37
Qwen2.5	**0.605**	**00:28:14**
	NELL	
Gemma2	0.866	00:26:50
Llama3.1	0.793	00:24:34
Mistral	0.902	00:27:57
Qwen2.5	**0.707**	**00:21:55**
	YAGO	
Gemma2	0.872	00:20:08
Llama3.1	0.813	00:18:46
Mistral	0.869	00:20:04
Qwen2.5	**0.570**	**00:13:10**

(b) Time measurements for the Iterative pipeline (FSN variant).

Table 6.11: Time measurements for the Iterative pipeline (SN variants).

Time measurements for the SN variants are presented in Tables 6.11a and 6.11b. Times are consistent with the observations made for the NI variants. We only need to point out that the marginal increases observable in the SN variants are once again caused by the additional prompt length introduced with the SN style: triples are now provided in three lines instead of one, and the lists contained in those prompts require a few more characters.

6.2.4 Comparative Analysis

This section addresses the comparisons between the Iterative pipeline and the Baseline, and between the variants of Iterative pipeline itself. We will consider only standard metrics and time measurements. As an outline, we will first compare both the zero- and few-shot settings against the Baseline, then moving on the comparison of the Iterative few-shot settings against the zero-shot settings, and finally analyzing the performance of the Iterative SN variants against the Iterative NI variants.

The tables below emphasize the performance of the first item in each comparison, *i.e.*, one variant of the Iterative pipeline. The second item in the comparison can be either the Baseline pipeline or another variant of the Iterative pipeline. A positive sign "+" indicates the variant in exam improved over the second item in the comparison; a negative sign "−" indicates a decrease in performance; and a dash "–" marks an equivalent performance.

Iterative zero-shot variants against Baseline comparison

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	−0.0413	+0.0112	+0.0085	−0.0637	−0.0334	–	−0.0413
Llama3.1	−0.2608	+0.0169	+0.0461	−0.3790	−0.2811	–	−0.2608
Mistral	+0.1021	−0.0813	−0.0358	+0.1801	+0.0846	–	+0.1021
Qwen2.5	+0.0048	+0.0060	+0.0030	+0.0043	+0.0044	−0.0001	+0.0048
	FactBench						
Gemma2	−0.0139	−0.0136	−0.0110	−0.0187	−0.0154	–	−0.0139
Llama3.1	−0.1036	−0.0887	+0.0562	−0.2980	−0.2087	–	−0.1036
Mistral	+0.0368	+0.0230	−0.0822	+0.2153	+0.0906	–	+0.0368
Qwen2.5	−0.0232	−0.0217	−0.0129	−0.0433	−0.0444	–	−0.0232
	NELL						
Gemma2	+0.1188	−0.0755	−0.0221	+0.1595	+0.0800	–	+0.1188
Llama3.1	−0.3403	−0.0822	−0.0067	−0.3943	−0.4085	–	−0.3403
Mistral	+0.2640	−0.0832	−0.0326	+0.3366	+0.1953	–	+0.2640
Qwen2.5	+0.1607	+0.0233	−0.0092	+0.1895	+0.1892	–	+0.1607
	YAGO						
Gemma2	+0.0931	+0.0469	+0.0008	+0.0938	+0.0615	–	+0.0931
Llama3.1	−0.1927	+0.0382	+0.0025	−0.1964	−0.1864	–	−0.1927
Mistral	+0.1818	+0.0916	+0.0030	+0.1833	+0.1604	–	+0.1818
Qwen2.5	+0.0946	+0.0477	+0.0007	+0.0952	+0.1101	–	+0.0946

Table 6.12: Comparison of results: Iterative (ZNI) vs. Baseline. + means improvement of Iterative over Baseline, zeros are replaced by ”–”.

In the comparison of the zero-shot setting of the Iterative pipeline against the Baseline pipeline, we will first concentrate on the ZNI variant against the Baseline, and then examine the ZSN variant against the Baseline. Table 6.12 reports the difference in performance between the ZNI variant and the Baseline, focusing on the ZNI variant.

Accuracy shows major degradation for Llama3.1, as discussed in Section 6.2.1. For every other model and dataset, Accuracy improved or presented minor deterioration. Indeed, Gemma2 suffers the major decrease in performance (on DBpedia, −4.13%). Balanced Accuracy sees an improvement in YAGO, while it decreased for all the other datasets.

Precision is on par with the Baseline, although when improvements happen, they are minimal. Recall follows the same trend as Accuracy, decreasing for Llama3.1 and usually improving for the other models.

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	+0.0577	−0.0149	−0.0094	+0.0885	+0.0423	–	+0.0577
Llama3.1	−0.3188	−0.0004	+0.0496	−0.4544	−0.3626	–	−0.3188
Mistral	+0.1328	−0.0931	−0.0399	+0.2289	+0.1057	–	+0.1328
Qwen2.5	+0.1015	+0.0189	−0.0085	+0.1366	+0.1056	−0.0003	+0.1016
	FactBench						
Gemma2	+0.0350	+0.0275	−0.0248	+0.1320	+0.0557	–	+0.0350
Llama3.1	−0.1222	−0.1042	+0.0889	−0.3560	−0.2627	–	−0.1222
Mistral	+0.0376	+0.0087	−0.1445	+0.3960	+0.1268	−0.0050	+0.0389
Qwen2.5	+0.0686	+0.0582	−0.0560	+0.2034	+0.1458	–	+0.0686
	NELL						
Gemma2	+0.1295	−0.1708	−0.0393	+0.1925	+0.0877	–	+0.1295
Llama3.1	−0.4118	−0.0904	+0.0007	−0.4791	−0.5360	–	−0.4118
Mistral	+0.2941	−0.1455	−0.0436	+0.3861	+0.2134	–	+0.2941
Qwen2.5	+0.4720	+0.0306	−0.0228	+0.5645	+0.4420	–	+0.4720
	YAGO						
Gemma2	+0.1342	−0.0225	−0.0006	+0.1367	+0.0868	–	+0.1342
Llama3.1	−0.2843	+0.0822	+0.0066	−0.2902	−0.2960	–	−0.2843
Mistral	+0.4971	+0.0251	+0.0016	+0.5047	+0.3664	–	+0.4971
Qwen2.5	+0.2035	+0.0124	−0.0019	+0.2065	+0.2201	–	+0.2035

Table 6.13: Comparison of results: Iterative (ZSN) vs. Baseline. + means improvement of Iterative over Baseline, zeros are replaced by ”–”.

Table 6.13 instead shows the disparities between the Iterative ZSN variant and the Baseline pipeline. We can observe the same trends as in the first comparison: the performance of Llama3.1 is underwhelming. Furthermore, Accuracy and Recall improve notably, and Precision remains quite equivalent.

NELL and YAGO see more pronounced gains in Accuracy and Recall than in the first comparison, with the biggest gains earned by Mistral on YAGO (+49.71%) and by Qwen2.5 on NELL (+47.20%).

Overall, the ZSN variant emerges as the winner in this comparison against the Baseline, as it delivers broader and more substantial improvements than those of the ZNI variant.

Iterative few-shot variants against Baseline comparison

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	+0.0012	−0.0088	−0.0033	+0.0054	+0.0015	–	+0.0012
Llama3.1	−0.1196	+0.0340	+0.0273	−0.1850	−0.1139	–	−0.1196
Mistral	+0.1268	−0.0895	−0.0387	+0.2189	+0.1016	–	+0.1268
Qwen2.5	+0.1486	+0.0253	−0.0123	+0.2010	+0.1485	–	+0.1486
	FactBench						
Gemma2	+0.0289	+0.0254	−0.0015	+0.0747	+0.0396	–	+0.0289
Llama3.1	−0.0043	−0.0088	−0.0407	+0.0540	+0.0117	–	−0.0043
Mistral	+0.0660	+0.0474	−0.0868	+0.3080	+0.1297	–	+0.0660
Qwen2.5	+0.0846	+0.0689	−0.0905	+0.2887	+0.1845	–	+0.0846
	NELL						
Gemma2	+0.0827	+0.0369	+0.0023	+0.0924	+0.0556	–	+0.0827
Llama3.1	+0.2957	+0.0579	−0.0014	+0.3455	+0.2155	–	+0.2957
Mistral	+0.2947	−0.0327	−0.0243	+0.3631	+0.2125	–	+0.2947
Qwen2.5	+0.4849	+0.1221	−0.0048	+0.5609	+0.4483	–	+0.4849
	YAGO						
Gemma2	+0.1616	−0.0087	−0.0003	+0.1643	+0.1029	–	+0.1616
Llama3.1	+0.1580	+0.0345	+0.0005	+0.1600	+0.1204	–	+0.1580
Mistral	+0.4019	+0.0222	+0.0015	+0.4080	+0.3119	–	+0.4019
Qwen2.5	+0.2901	+0.0560	−0.0012	+0.2938	+0.2958	–	+0.2901

Table 6.14: Comparison of results: Iterative (FNI) vs. Baseline. + means improvement of Iterative over Baseline, zeros are replaced by "–".

Table 6.14 highlights the difference of performance between the Iterative FNI variant and the Baseline. Starting from the Accuracy, we can observe substantial generalized gains, concentrated more on NELL and YAGO datasets. Balanced Accuracy is also increased, though the extent is smaller.

Precision remained relatively stable, showing small deviations in both directions. Its drops are concentrated on FactBench, with the most important one occurring in Qwen2.5 (−9.05%), while the improvements are usually negligible. To conclude, Recall showed notable enhancements from the few-shot settings, with all models gaining ground on most datasets, particularly Mistral (+21.89%, +30.80%, +36.31%, +40.80% on DBpedia, FactBench, NELL and YAGO, respectively).

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	+0.0323	+0.0145	+0.0026	+0.0397	+0.0229	–	+0.0323
Llama3.1	−0.2738	+0.0066	+0.0407	−0.3932	−0.2967	–	−0.2738
Mistral	+0.1428	−0.0795	−0.0362	+0.2374	+0.1116	–	+0.1428
Qwen2.5	+0.1146	+0.0231	−0.0084	+0.1536	+0.1177	–	+0.1146
	FactBench						
Gemma2	+0.0597	+0.0548	+0.0159	+0.1220	+0.0721	–	+0.0597
Llama3.1	−0.0079	−0.0071	−0.0011	−0.0180	−0.0112	–	−0.0079
Mistral	+0.0496	+0.0234	−0.1330	+0.3906	+0.1327	–	+0.0496
Qwen2.5	+0.0843	+0.0683	−0.0929	+0.2914	+0.1850	–	+0.0843
	NELL						
Gemma2	+0.1053	−0.0042	−0.0077	+0.1283	+0.0707	–	+0.1053
Llama3.1	+0.2774	+0.0619	+0.0002	+0.3226	+0.2044	–	+0.2774
Mistral	+0.2914	−0.0851	−0.0334	+0.3702	+0.2113	–	+0.2914
Qwen2.5	+0.4677	+0.1379	−0.0004	+0.5368	+0.4367	–	+0.4677
	YAGO						
Gemma2	+0.1789	–	−0.0002	+0.1818	+0.1129	–	+0.1789
Llama3.1	+0.1306	+0.0207	+0.0002	+0.1324	+0.1012	–	+0.1306
Mistral	+0.5433	+0.0483	+0.0020	+0.5513	+0.3909	–	+0.5433
Qwen2.5	+0.3839	−0.0320	−0.0039	+0.3905	+0.3694	–	+0.3839

Table 6.15: Comparison of results: Iterative (FSN) vs. Baseline. + means improvement of Iterative over Baseline, zeros are replaced by ”–”.

Table 6.15 points out the performance of the FSN variant over the Baseline. Similarly to the analysis of the ZSN variant, we can observe generalized and robust enhancements in all metrics. Mirroring the FNI variant, the Accuracy gains are more prominent on NELL and YAGO. While the gains in Balanced Accuracy remain small, they show improvement compared to the FNI variant.

Again, Precision behaves as with the FNI variant comparison, and Recall shows a consistent boost, indicating a way better ability in identifying true positives.

Overall, the few-shot setting carries remarkable improvements in both variants. Our choice once again favors the SN variant (FSN in this case). The rationale is that improvements in Accuracy, Balanced Accuracy and Precision are

comparable, with Recall showing notably higher results in the FSN variant.

Few-shot variants against Zero-shot variants comparison

In this section, we compare the differences between the few-shot prompts and the zero-shot prompts, keeping the prompt style fixed across the comparisons. We start from the NI variants, whose results are reported in Table 6.16.

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	+0.0425	−0.0200	−0.0118	+0.0691	+0.0349	–	+0.0425
Llama3.1	+0.1412	+0.0171	−0.0188	+0.1940	+0.1672	–	+0.1412
Mistral	+0.0247	−0.0082	−0.0029	+0.0388	+0.0170	–	+0.0247
Qwen2.5	+0.1438	+0.0193	−0.0153	+0.1967	+0.1441	+0.0001	+0.1438
	FactBench						
Gemma2	+0.0428	+0.0390	+0.0095	+0.0934	+0.0550	–	+0.0428
Llama3.1	+0.0993	+0.0799	−0.0969	+0.3520	+0.2204	–	+0.0993
Mistral	+0.0292	+0.0244	−0.0046	+0.0927	+0.0391	–	+0.0292
Qwen2.5	+0.1078	+0.0906	−0.0776	+0.3320	+0.2289	–	+0.1078
	NELL						
Gemma2	−0.0361	+0.1124	+0.0244	−0.0671	−0.0244	–	−0.0361
Llama3.1	+0.6360	+0.1401	+0.0053	+0.7398	+0.6240	–	+0.6360
Mistral	+0.0307	+0.0505	+0.0083	+0.0265	+0.0172	–	+0.0307
Qwen2.5	+0.3242	+0.0988	+0.0044	+0.3714	+0.2591	–	+0.3242
	YAGO						
Gemma2	+0.0685	−0.0556	−0.0011	+0.0705	+0.0414	–	+0.0685
Llama3.1	+0.3507	−0.0037	−0.0020	+0.3564	+0.3068	–	+0.3507
Mistral	+0.2201	−0.0694	−0.0015	+0.2247	+0.1515	–	+0.2201
Qwen2.5	+0.1955	+0.0083	−0.0019	+0.1986	+0.1857	–	+0.1955

Table 6.16: Comparison of results: Iterative (FNI) vs. Iterative (ZNI). + means improvement of the few-shot approach over the zero-shot approach (NI prompts), zeros are replaced by ”–”.

Accuracy wise, we can observe the completely different behavior of Llama3.1, with huge improvements in NELL and YAGO (+63.60% and +35.07%). In general, all models in all datasets shows improvements, even though they are more modest. Balanced Accuracy and Precision are the ones where the performance fluctuates a lot, with many small deviations, while Recall mirrors Accuracy and shows huge positive gaps for Llama3.1 and Qwen2.5.

We now move on the SN variants, with results reported in Table 6.17.

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	−0.0254	+0.0294	+0.0120	−0.0488	−0.0194	–	−0.0254
Llama3.1	+0.0450	+0.0070	−0.0089	+0.0612	+0.0659	–	+0.0450
Mistral	+0.0100	+0.0136	+0.0037	+0.0085	+0.0059	–	+0.0100
Qwen2.5	+0.0131	+0.0042	+0.0001	+0.0170	+0.0121	+0.0003	+0.0130
	FactBench						
Gemma2	+0.0247	+0.0273	+0.0407	−0.0100	+0.0164	–	+0.0247
Llama3.1	+0.1143	+0.0971	−0.0900	+0.3380	+0.2515	–	+0.1143
Mistral	+0.0120	+0.0147	+0.0115	−0.0054	+0.0059	+0.0050	+0.0107
Qwen2.5	+0.0157	+0.0101	−0.0369	+0.0880	+0.0392	–	+0.0157
	NELL						
Gemma2	−0.0242	+0.1666	+0.0316	−0.0642	−0.0170	–	−0.0242
Llama3.1	+0.6892	+0.1523	−0.0005	+0.8017	+0.7404	–	+0.6892
Mistral	−0.0027	+0.0604	+0.0102	−0.0159	−0.0021	–	−0.0027
Qwen2.5	−0.0043	+0.1073	+0.0224	−0.0277	−0.0053	–	−0.0043
	YAGO						
Gemma2	+0.0447	+0.0225	+0.0004	+0.0451	+0.0261	–	+0.0447
Llama3.1	+0.4149	−0.0615	−0.0064	+0.4226	+0.3972	–	+0.4149
Mistral	+0.0462	+0.0232	+0.0004	+0.0466	+0.0245	–	+0.0462
Qwen2.5	+0.1804	−0.0444	−0.0020	+0.1840	+0.1493	–	+0.1804

Table 6.17: Comparison of results: Iterative (FSN) vs. Iterative (ZSN). + means improvement of the few-shot approach over the zero-shot approach (SN prompts), zeros are replaced by ”–”.

We can make the same observation we did for the NI prompt style: Accuracy sees mostly gains, with Llama3.1 benefiting the most (+68.92% on NELL), and negligible drops. Balanced Accuracy increases too, but it sees small gains. Precision presents again small fluctuations (but remember Precision has always achieved high values), and Recall closely follows the same trend of Accuracy, with small generalized gains and big gaps covered when analyzing Llama3.1 (+80.17% on NELL).

To wrap up, these tables show that the few-shot setting is outperforming the zero-shot setting: despite small improvements in Accuracy, Llama3.1 (which was the most critical model in the zero-shot setting) completely recovers from its initial lackluster performance. This emphasizes the importance of the examples we injected in the prompt, as they provide clearer task comprehension and

serve as a prior context helping models in discerning correct and incorrect facts, thereby leveraging their internal knowledge to make more accurate judgments.

SN variants against NI variants comparison

This section delves into the differences in performance across the variants of the Iterative pipeline. We will first discuss the differences between ZSN and ZNI, then proceeding with the analysis of the differences between FSN and FNI. To this end, Table 6.18 reports the differences in the zero-shot setting.

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	+0.0990	−0.0261	−0.0179	+0.1522	+0.0757	–	+0.0990
Llama3.1	−0.0580	−0.0173	+0.0035	−0.0754	−0.0815	–	−0.0580
Mistral	+0.0307	−0.0118	−0.0041	+0.0488	+0.0211	–	+0.0307
Qwen2.5	+0.0967	+0.0129	−0.0115	+0.1323	+0.1012	−0.0002	+0.0968
	FactBench						
Gemma2	+0.0489	+0.0411	−0.0138	+0.1507	+0.0711	–	+0.0489
Llama3.1	−0.0186	−0.0155	+0.0327	−0.0580	−0.0540	–	−0.0186
Mistral	+0.0008	−0.0143	−0.0623	+0.1807	+0.0362	−0.0050	+0.0021
Qwen2.5	+0.0918	+0.0799	−0.0431	+0.2467	+0.1902	–	+0.0918
	NELL						
Gemma2	+0.0107	−0.0953	−0.0172	+0.0330	+0.0077	–	+0.0107
Llama3.1	−0.0715	−0.0082	+0.0074	−0.0848	−0.1275	–	−0.0715
Mistral	+0.0301	−0.0623	−0.0110	+0.0495	+0.0181	–	+0.0301
Qwen2.5	+0.3113	+0.0073	−0.0136	+0.3750	+0.2528	–	+0.3113
	YAGO						
Gemma2	+0.0411	−0.0694	−0.0014	+0.0429	+0.0253	–	+0.0411
Llama3.1	−0.0916	+0.0440	+0.0041	−0.0938	−0.1096	–	−0.0916
Mistral	+0.3153	−0.0665	−0.0014	+0.3214	+0.2060	–	+0.3153
Qwen2.5	+0.1089	−0.0353	−0.0026	+0.1113	+0.1100	–	+0.1089

Table 6.18: Comparison of results: Iterative (ZSN) vs. Iterative (ZNI). + means improvement of the SN approach over the NI approach, zeros are replaced by "–".

The zero-shot settings brings the most substantial improvements over Accuracy and Recall. Qwen2.5 is the model that most benefits from this prompt style, as it usually the one who sees the biggest increases in Accuracy compared to the other models. The gains in Accuracy don't reflect the ones in Balanced Accuracy,

where models experience small deteriorations in their performance, as well as in Precision.

Recall is the metric under which the models exhibit the advantages of the structured style, boosting their ability in recognizing positive instances in all the datasets. Despite some drops, Gemma2 improves on DBpedia and FactBench, Mistral shines on FactBench and YAGO, and Qwen2.5 achieves a better score on FactBench and NELL.

Llama3.1 seems to be the only model suffering from the change in style, as its improvements are minimal in Precision, while it gets worse scores in any other metric. At the same time, Llama3.1 performed very poorly in the zero-shot setting and showed solid improvements in the few-shot scenario, hence we believe further examination is unnecessary.

We now switch to the few-shot setting, illustrated in Table 6.19.

Accuracy is stable across the few-shot variants, as most of the deviations are small. The most noticeable ones are Llama3.1 in DBpedia, which seems to prefer the prose prompt style, and Mistral in YAGO, which favors the structured prompt. The same stability can be observed in the Balanced Accuracy (with the major drops occurring in NELL and YAGO for Mistral and Qwen2.5, respectively), in Precision, and even in Recall, where the fluctuations resemble the ones in Accuracy.

The stability observed across all metric, regardless of the prompt style, suggests that the instructions provide solid guidance, maintaining their effectiveness and clarity regardless of how they are integrated into the prompt.

Time comparison

The Iterative pipeline presents times that are just over double the ones required by the Baseline pipeline, specifically considering the SN variations. As we've already discussed, these increased costs come from the nature of the Iterative prompt, which contains detailed (and reliable) instructions and is therefore longer than the Baseline prompt.

At the same time, the Iterative pipeline brings notable improvements in performance. These enhancements highlight the value of the additional costs intro-

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	+0.0311	+0.0233	+0.0059	+0.0343	+0.0214	–	+0.0311
Llama3.1	−0.1542	−0.0274	+0.0134	−0.2082	−0.1828	–	−0.1542
Mistral	+0.0160	+0.0100	+0.0025	+0.0185	+0.0100	–	+0.0160
Qwen2.5	−0.0340	−0.0022	+0.0039	−0.0474	−0.0308	–	−0.0340
	FactBench						
Gemma2	+0.0308	+0.0294	+0.0174	+0.0473	+0.0325	–	+0.0308
Llama3.1	−0.0036	+0.0017	+0.0396	−0.0720	−0.0229	–	−0.0036
Mistral	−0.0164	−0.0240	−0.0462	+0.0826	+0.0030	–	−0.0164
Qwen2.5	−0.0003	−0.0006	−0.0024	+0.0027	+0.0005	–	−0.0003
	NELL						
Gemma2	+0.0226	−0.0411	−0.0100	+0.0359	+0.0151	–	+0.0226
Llama3.1	−0.0183	+0.0040	+0.0016	−0.0229	−0.0111	–	−0.0183
Mistral	−0.0033	−0.0524	−0.0091	+0.0071	−0.0012	–	−0.0033
Qwen2.5	−0.0172	+0.0158	+0.0044	−0.0241	−0.0116	–	−0.0172
	YAGO						
Gemma2	+0.0173	+0.0087	+0.0001	+0.0175	+0.0100	–	+0.0173
Llama3.1	−0.0274	−0.0138	−0.0003	−0.0276	−0.0192	–	−0.0274
Mistral	+0.1414	+0.0261	+0.0005	+0.1433	+0.0790	–	+0.1414
Qwen2.5	+0.0938	−0.0880	−0.0027	+0.0967	+0.0736	–	+0.0938

Table 6.19: Comparison of results: Iterative (FSN) vs. Iterative (FNI). + means improvement of the SN approach over the NI approach, zeros are replaced by ”–”.

duced by this pipeline, which we believe are justified by the significant gains it achieves.

6.3 Stepwise Pipeline

This section discusses the results of the Stepwise pipeline. Following the established order, we begin with standard metrics, and then continue with those tailored for this pipeline, as well as time measurements. We then conclude by comparing the performance of the Stepwise pipeline with both the other pipelines.

As a general observation, Informativeness remains very high although it presents some minor flaws, with Gemma2 in DBpedia being the lowest score. As has been the case, Truthfulness closely resembles Accuracy.

6.3.1 Standard Metrics

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	0.7506	**0.6382**	0.9005	0.7964	0.8452	0.9475	0.7637
Llama3.1	0.6301	0.6123	0.8981	0.6377	0.7458	0.9924	0.6329
Mistral	**0.8239**	0.5630	0.8684	**0.9347**	**0.9003**	0.9991	0.8241
Qwen2.5	0.5480	0.5928	**0.9011**	0.5298	0.6673	0.9854	0.5546
	FactBench						
Gemma2	**0.7597**	**0.7564**	0.7613	0.8025	**0.7814**	0.9971	0.7604
Llama3.1	0.7098	0.7080	0.7273	0.7332	0.7302	0.9993	0.7100
Mistral	0.6236	0.5950	0.5878	**0.9953**	0.7391	1.0000	0.6236
Qwen2.5	0.6761	0.6912	**0.8133**	0.5287	0.6408	0.9804	0.6825
	NELL						
Gemma2	0.6442	0.6591	**0.9543**	0.6411	0.7669	0.9973	0.6452
Llama3.1	0.7500	**0.6776**	0.9517	0.7652	0.8483	1.0000	0.7500
Mistral	**0.9167**	0.5439	0.9205	**0.9947**	**0.9562**	1.0000	0.9167
Qwen2.5	0.5382	0.6200	0.9516	0.5212	0.6735	0.9989	0.5387
	YAGO						
Gemma2	0.7358	**0.5512**	**0.9931**	0.7388	0.8473	0.9942	0.7374
Llama3.1	0.6472	0.4164	0.9900	0.6509	0.7854	1.0000	0.6472
Mistral	**0.9805**	0.4942	0.9920	**0.9884**	**0.9902**	1.0000	0.9805
Qwen2.5	0.3860	0.5102	0.9925	0.3840	0.5537	1.0000	0.3860

Table 6.20: Standard metrics for the Stepwise pipeline. For each metric and each KG, the highest scores are highlighted in bold.

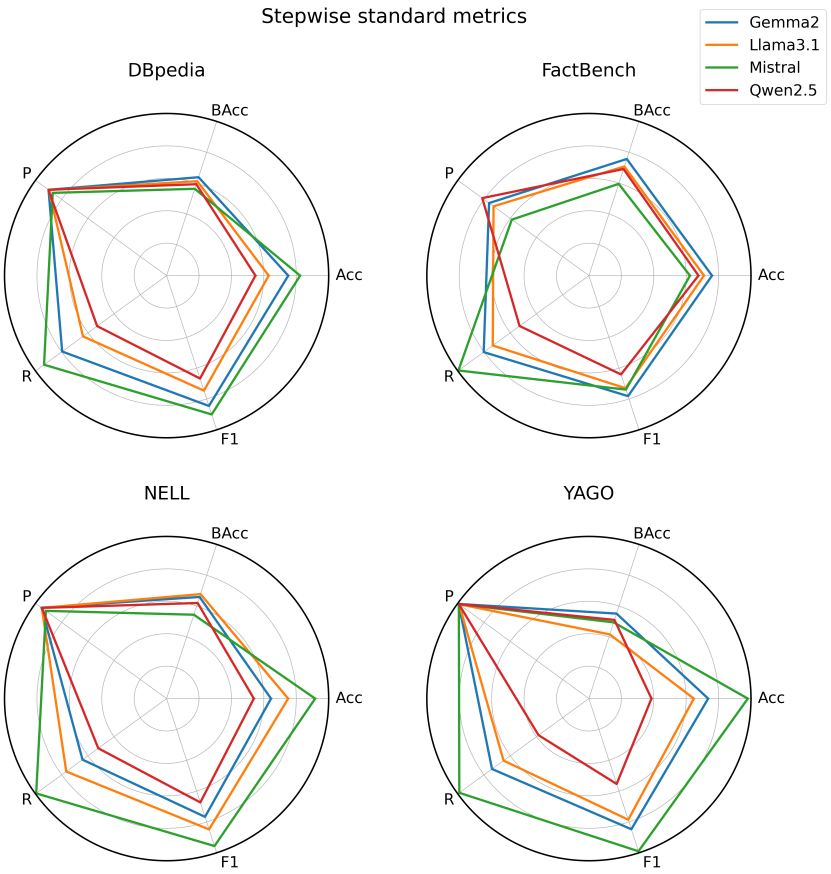

Figure 6.6: Visualization of the Standard metrics for the Stepwise pipeline.

Table 6.20 and Figure 6.6 report and visualize the results for the standard metrics obtained from the Stepwise pipeline.

DBpedia

Mistral stands out as the top performer in Accuracy, despite the lowest performance in Balanced Accuracy. Its fairly high Precision and Recall indicate that it correctly predicts many of the positive instances, but is lacking in the negatives. Gemma2 follows, with a more modest Accuracy, but better Balanced Accuracy and Precision. We then find Llama3.1, decreasing in the Accuracy but performing comparably to Gemma2 in the Balanced Accuracy, still maintaining high Precision but trading in Recall. Qwen2.5 remains the lowest performing model among all metrics, though it achieves high Precision.

FactBench

For FactBench, the leading model is Gemma2, with the highest Accuracy and Balanced Accuracy, and high Precision and Recall. Llama3.1 performs closely to Gemma2, achieving slightly lower scores. Interestingly enough, Qwen2.5 outperforms Mistral, achieving better Accuracy, Balanced Accuracy, and top Precision. However, as in DBpedia, it shows low Recall. On the other hand, Mistral achieve perfect Recall (99.53%) and ranges around 60% for Accuracy, Balanced Accuracy and Precision.

NELL

In NELL, we see Mistral as the best performing model in all metrics except Balanced Accuracy. This is due to the model predicting mostly true answers, therefore generating many false positives. The second best model is Llama3.1, showing the best Balanced Accuracy (67.76%) paired with high Precision and good Recall. We then find Gemma2, performing similar to Llama3.1 in the Balanced Accuracy and Precision, but exhibiting a decrease in Accuracy and Recall, and Qwen2.5 as the trailing model accuracy-wise, but passing Mistral in Balanced Accuracy and maintaining high Precision.

YAGO

YAGO sees Mistral as a near-perfect model, then Gemma2, Llama3.1 and Qwen2.5. Here, the Balanced Accuracy is noticeably lower with respect to the other datasets for every model, highlighting they all predict many false negatives. The boost in all the other metrics can therefore be attributed to the dataset imbalance. Another critical measure is Recall, where we can see a wide discrepancy among models, with Mistral achieving 98.84% while Qwen2.5 only reaches 38.40%.

6.3.2 Tailored Metrics

The tailored metrics are showed in Table 6.21.

Statement Rate is always very high, confirming the ability of the models to understand the data in the triples even considering the various formats of the triples across datasets. The most "critical" model is Qwen2.5, which fails to

Model	Statement rate	Give-up rate	Right|G	Right|¬G
	DBpedia			
Gemma2	0.9987	0.2312	0.2736	0.8427
Llama3.1	1.0000	0.0136	0.2913	0.6299
Mistral	1.0000	0.0098	0.4348	0.8271
Qwen2.5	0.9856	0.0940	0.2551	0.5696
	FactBench			
Gemma2	1.0000	0.3543	0.7994	0.7345
Llama3.1	1.0000	0.0814	0.9649	0.6866
Mistral	1.0000	0.0264	0.9459	0.6148
Qwen2.5	0.9807	0.2250	0.8063	0.6212
	NELL			
Gemma2	1.0000	0.0973	0.2983	0.6796
Llama3.1	1.0000	0.0048	0.6667	0.7504
Mistral	1.0000	0.0048	0.5556	0.9184
Qwen2.5	0.9989	0.0694	0.2558	0.5586
	YAGO			
Gemma2	1.0000	0.1595	0.0407	0.8627
Llama3.1	1.0000	0.0058	0.0000	0.6509
Mistral	1.0000	0.0014	0.0000	0.9819
Qwen2.5	1.0000	0.0830	0.0174	0.4194

Table 6.21: Tailored metrics for the Stepwise pipeline.

generate some of the statements in all dataset but YAGO.

Moving on the Give-up rate, we can observe models are pretty confident, with only Gemma2 presenting high ratios of give-ups in all datasets. FactBench reveals itself as the most challenging dataset, with Gemma2 and Qwen2.5 showcasing the greatest lacks in confidence (with 35.43% and 22.50% respectively), while NELL and YAGO stand out as the easiest ones. DBpedia falls in between.

Strictly tied to the Give-up rate we have the Right|G and Right|¬G metrics, decomposing the accuracy of the models according to the model's level of confidence (low and high, respectively).

YAGO and DBpedia are the datasets where the models are more "honest", as the low scores on Right|G mean that they are misjudging often when they exhibit low confidence. Conversely, FactBench is the one where the confidence assessment is the least aligned with the model's accuracy, with Gemma2 and

Qwen2.5 correctly evaluating the 80% of the facts they don't feel confident about. The high values reported for Llama3.1 and Mistral might appear concerning; however, the related Give-up rates and the lower scores on other datasets suggest that the issue may stem from the dataset itself rather than the models.

On the other side, the Right|¬G metric highlights that models are generally correct when they exhibit high confidence. All the values are above 50% except for Qwen2.5 in YAGO, and they mirror the distribution observable in the Accuracy (reported in Table 6.20). This is also supported by the dominance of low Give-up rates.

6.3.3 Time Measurements

Model	Fact Avg. Time	Stat. Avg. Time	Eval. Avg. Time	Total Time
	DBpedia			
Gemma2	4.713	2.778	1.935	12:13:58
Llama3.1	2.584	1.454	1.131	06:42:24
Mistral	2.729	1.593	1.136	07:04:59
Qwen2.5	**2.055**	**1.097**	**0.959**	**05:20:01**
	FactBench			
Gemma2	2.835	1.613	1.222	02:12:18
Llama3.1	2.536	1.197	1.339	01:58:20
Mistral	2.789	1.569	1.220	02:10:09
Qwen2.5	**2.026**	**1.038**	**0.988**	**01:34:32**
	NELL			
Gemma2	2.900	1.590	1.309	01:29:54
Llama3.1	2.524	1.274	1.250	01:18:14
Mistral	2.285	1.269	**1.016**	01:10:50
Qwen2.5	**2.069**	**1.000**	1.070	**01:04:08**
	YAGO			
Gemma2	3.002	1.700	1.302	01:09:20
Llama3.1	2.328	1.163	1.165	00:53:46
Mistral	2.302	1.274	**1.029**	00:53:10
Qwen2.5	**2.174**	**1.105**	1.069	**00:50:13**

Table 6.22: Time measurements for the Stepwise pipeline.

Table 6.22 shows the time measurements for the Stepwise pipeline. In contrast to both Baseline and Iterative, which are one-step processes, here we can provide a more detailed overview by dividing the time needed for the

statement generation (”Stat. Avg. Time” column) from the time needed for the confidence assessment and truthfulness evaluation (”Eval. Avg. Time” column). We also report the aggregated average time for a fact (”Fact Avg. Time” column) and the total time needed to fully evaluate the datasets through this pipeline.

We can observe that the average time needed for a single fact is much higher compared to the other pipelines. This is due to the inherent structure of the Stepwise pipeline, as the model needs to process the whole chat history up to three times (recall that it increases in length with each step) and generate more tokens than in the other pipelines.

Specifically, in the first step (Statement production) the model needs to produce an entire statement (that is, one or more sentences) and not a single character. Indeed, roughly half of the total average time is consumed by the statement production step. The other half comprises both the confidence assessment and the truthfulness evaluation of the statement. The average evaluation time, as expected, is double the time of the Iterative pipeline in the zero-shot setting. This is consistent, as those times reflect two steps of the pipeline, namely two prompts submitted to the model, and we have similar prompts lengths.

Taking a closer look, it is noticeable that Gemma2 takes longer than the other models to generate the statements. While for FactBench, NELL and YAGO the difference is negligible, DBpedia shows a wider gap. As briefly mentioned in Section 6.1.2, we identified the cause of this gap being triples with longer content in the object, such as `abstract`, `rdf-schema#comment` and `shortsummary` triples. These observations, paired with the fact that Gemma2 was the slowest model also for every variation of the Iterative pipeline, but not in the Baseline (where Llama3.1 is slower), indicate that Gemma2 may struggle in handling long texts.

6.3.4 Comparative Analysis

In this section, we compare the Stepwise pipeline with both the other pipelines, considering standard metrics and time measurements. We will first delve into the comparison against the Baseline pipeline, then proceed with the comparison against the Iterative pipeline, and concluding with time measure-

ments.

In the following tables, the focus is on the Stepwise pipeline: a positive sign "+" marks the improvements of Stepwise over the other pipeline, a negative sign "−" marks its degradations, while a dash "–" represents equivalent performance.

Stepwise against Baseline comparison

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	−0.0016	+0.0013	+0.0041	−0.0050	−0.0010	−0.0525	+0.0115
Llama3.1	−0.0686	+0.0284	+0.0179	−0.1099	−0.0627	−0.0076	−0.0658
Mistral	+0.1437	−0.0685	−0.0328	+0.2337	+0.1117	−0.0009	+0.1439
Qwen2.5	+0.0264	−0.0338	−0.0229	+0.0529	+0.0382	−0.0146	+0.0330
	FactBench						
Gemma2	+0.0140	+0.0073	−0.0376	+0.1005	+0.0341	−0.0029	+0.0147
Llama3.1	−0.0270	−0.0335	−0.0743	+0.0572	−0.0033	−0.0007	−0.0268
Mistral	−0.0850	−0.1242	−0.2449	+0.4246	+0.0619	–	−0.0850
Qwen2.5	+0.0268	+0.0228	−0.0643	+0.1274	+0.0900	−0.0196	+0.0332
	NELL						
Gemma2	−0.1080	−0.0506	−0.0049	−0.1199	−0.0818	−0.0027	−0.1070
Llama3.1	+0.1645	+0.0676	+0.0073	+0.1849	+0.1294	–	+0.1645
Mistral	+0.3092	−0.1260	−0.0405	+0.4002	+0.2217	–	+0.3092
Qwen2.5	+0.1914	+0.0422	−0.0050	+0.2228	+0.2186	−0.0011	+0.1919
	YAGO						
Gemma2	+0.0381	+0.0192	+0.0003	+0.0384	+0.0260	−0.0058	+0.0397
Llama3.1	+0.0981	−0.1309	−0.0034	+0.1018	+0.0781	–	+0.0981
Mistral	+0.5584	+0.0560	+0.0022	+0.5666	+0.3987	–	+0.5584
Qwen2.5	+0.1126	−0.0785	−0.0048	+0.1156	+0.1308	–	+0.1126

Table 6.23: Comparison of results: Stepwise vs. Baseline. + means improvement of Stepwise over Baseline, zeros are replaced by "–".

Table 6.23 shows the difference in performance on the standard metrics between Stepwise and Baseline.

As a brief overview, the Stepwise pipeline improves the Accuracy and Recall, with Mistral gaining the most in NELL and YAGO. This comes at the expense of a moderately lower Balanced Accuracy, as the gains in that metric are very marginal. Precision presents mostly small variations in both directions. Informativeness remains almost the same, and Truthfulness closely follows Accuracy.

While some of the deterioration in Informativeness comes from the "IDK" answers when models couldn't generate a statement, there are still some "NA" responses, as the deterioration doesn't align with the Statement rate (Table 6.21). We interpret these cases as a lack of comprehension in the statements generated by the models themselves.

The drops in Balanced Accuracy, paired with an on-par Precision, can be attributed to the higher Recall. While the improved Recall shouts for a better ability to identify positives, it may comes at the cost of increased false positives.

Stepwise against Iterative comparison

We now proceed to compare the differences in standard metrics between the Stepwise pipeline and the Iterative pipeline. In the previous sections, we identified the few-shot setting as the one where models achieve the best results, hence the comparisons will focus on those variants. Given the similarities between the two variants, we expect the same behavior when compared to the Stepwise pipeline, and we choose to discuss them together. The results ot both comparisons are reported in Tables 6.24 and 6.25.

These two tables show that the Stepwise pipeline is mostly underperforming when compared to the Iterative pipeline, independently of the variant considered. Most of the columns contain way more negative signs than positive ones, as to remark the drops in performance. While in some cases the drops are small, in others they become important. For example, the Accuracy metric (which is our primary interest) presents a handful of deteriorations in Gemma2, Mistral and Qwen2.5, all ranging from −10% to −30%. Balanced Accuracy also exhibits weaker results, with the most notable drops around −15%, but fewer in number with respect to Accuracy. Precision remains very high, denoting very small fluctuations, with only Mistral on FactBench experiencing more pronounced degradations. In general, Recall resembles Accuracy.

The only models which seem to take advantage of the Stepwise pipeline are Llama3.1 on DBpedia and Mistral on YAGO, achieving better Accuracy and Recall.

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	−0.0028	+0.0101	+0.0074	−0.0104	−0.0025	−0.0525	+0.0103
Llama3.1	+0.0510	−0.0056	−0.0094	+0.0751	+0.0512	−0.0076	+0.0538
Mistral	+0.0169	+0.0210	+0.0059	+0.0148	+0.0101	−0.0009	+0.0171
Qwen2.5	−0.1222	−0.0591	−0.0106	−0.1481	−0.1103	−0.0146	−0.1156
	FactBench						
Gemma2	−0.0149	−0.0181	−0.0361	+0.0258	−0.0055	−0.0029	−0.0142
Llama3.1	−0.0227	−0.0247	−0.0336	+0.0032	−0.0150	−0.0007	−0.0225
Mistral	−0.1510	−0.1716	−0.1581	+0.1166	−0.0678	–	−0.1510
Qwen2.5	−0.0578	−0.0461	+0.0262	−0.1613	−0.0945	−0.0196	−0.0514
	NELL						
Gemma2	−0.1907	−0.0875	−0.0072	−0.2123	−0.1374	−0.0027	−0.1897
Llama3.1	−0.1312	+0.0097	+0.0087	−0.1606	−0.0861	–	−0.1312
Mistral	+0.0145	−0.0933	−0.0162	+0.0371	+0.0092	–	+0.0145
Qwen2.5	−0.2935	−0.0799	−0.0002	−0.3381	−0.2297	−0.0011	−0.2930
	YAGO						
Gemma2	−0.1235	+0.0279	+0.0006	−0.1259	−0.0769	−0.0058	−0.1219
Llama3.1	−0.0599	−0.1654	−0.0039	−0.0582	−0.0423	–	−0.0599
Mistral	+0.1565	+0.0338	+0.0007	+0.1586	+0.0868	–	+0.1565
Qwen2.5	−0.1775	−0.1345	−0.0036	−0.1782	−0.1650	–	−0.1775

Table 6.24: Comparison of results: Stepwise vs. Iterative (FNI). + means improvement of Stepwise over Iterative, zeros are replaced by ”–”.

In light of these considerations, we believe the benefits of the Stepwise pipeline are relevant against the Baseline, but negligible against the Iterative pipeline. This comparison further reinforces the conclusions we drawn for the Iterative pipeline in the few-shot setting: the examples provide models with enough context to leverage their internal knowledge effectively, as well as showcasing the expected output format.

Time comparison

The Stepwise pipeline inherently introduces higher costs compared to Baseline and Iterative pipelines, as a direct consequence of the methodology it employs (recall Stepwise is a three-step process, while Baseline and Iterative are

Model	Acc	BAcc	P	R	F1	Inf	Tru
	DBpedia						
Gemma2	−0.0339	−0.0132	+0.0015	−0.0447	−0.0239	−0.0525	−0.0208
Llama3.1	+0.2052	+0.0218	−0.0228	+0.2833	+0.2340	−0.0076	+0.2080
Mistral	+0.0009	+0.0110	+0.0034	−0.0037	+0.0001	−0.0009	+0.0011
Qwen2.5	−0.0882	−0.0569	−0.0145	−0.1007	−0.0795	−0.0146	−0.0816
	FactBench						
Gemma2	−0.0457	−0.0475	−0.0535	−0.0215	−0.0380	−0.0029	−0.0450
Llama3.1	−0.0191	−0.0264	−0.0732	+0.0752	+0.0079	−0.0007	−0.0189
Mistral	−0.1346	−0.1476	−0.1119	+0.0340	−0.0708	–	−0.1346
Qwen2.5	−0.0575	−0.0455	+0.0286	−0.1640	−0.0950	−0.0196	−0.0511
	NELL						
Gemma2	−0.2133	−0.0464	+0.0028	−0.2482	−0.1525	−0.0027	−0.2123
Llama3.1	−0.1129	+0.0057	+0.0071	−0.1377	−0.0750	–	−0.1129
Mistral	+0.0178	−0.0409	−0.0071	+0.0300	+0.0104	–	+0.0178
Qwen2.5	−0.2763	−0.0957	−0.0046	−0.3140	−0.2181	−0.0011	−0.2758
	YAGO						
Gemma2	−0.1408	+0.0192	+0.0005	−0.1434	−0.0869	−0.0058	−0.1392
Llama3.1	−0.0325	−0.1516	−0.0036	−0.0306	−0.0231	–	−0.0325
Mistral	+0.0151	+0.0077	+0.0002	+0.0153	+0.0078	–	+0.0151
Qwen2.5	−0.2713	−0.0465	−0.0009	−0.2749	−0.2386	–	−0.2713

Table 6.25: Comparison of results: Stepwise vs. Iterative (FSN). + means improvement of Stepwise over Iterative, zeros are replaced by ”–”.

both one-step processes[1]). Its processing times are approximately 10 to 12 times the total duration of the Baseline, and 4 to 5 that of the Iterative pipeline, based on the shortest and longest time observed for each. However, the improvements it brings, while relevant, are not enough to justify the protracted execution time, especially when compared to the improvements achieved by the Iterative pipeline. For this reasons, we believe the Stepwise pipeline may not be well-suited for the Accuracy Evaluation Task, as its suboptimal efficiency shadows its gains.

[1]On paper, Iterative could require more than one step, but we observed no extra attempts are required most of the times, as discussed in Section 6.2.2

6.4 Ablation Study

This section discusses the ablation study we conducted over the Accuracy of the Baseline pipeline. For each dataset, we divided the Accuracy by predicate, considering also the Accuracy for the positive and negative labeled facts (according to the ground truth). This study allows to inspect the models' strengths and weaknesses in their understanding of the datasets. As in the previous sections, we provide both tables and plots.

6.4.1 DBpedia

DBpedia is the largest dataset, comprising 951 predicates. Given the high number of predicates, we decided to focus on the 20 most frequent ones. Table 6.26 presents such predicates, alongside with their frequency.

Predicate	**Frequency (occ.)**	**Frequency (%)**
subject	1401	0.1499
name	572	0.0612
birthPlace	493	0.0528
rdf-schema#label	342	0.0366
hometown	222	0.0238
abstract	198	0.0212
capital	168	0.0180
hypernym	160	0.0171
rdf-schema#comment	155	0.0166
genre	140	0.0150
starring	131	0.0140
isPrimaryTopicOf	123	0.0132
country	109	0.0117
deathPlace	100	0.0107
currency	95	0.0102
location	92	0.0098
language	85	0.0091
hasPhotoCollection	84	0.0090
birthDate	67	0.0072
producer	62	0.0066
Total	4799	0.5136

Table 6.26: The 20 most frequent predicates in DBpedia, ranked by occurrence.

Moving on the ablation study itself, Table 6.27 and Figure 6.7 show the Accuracy decomposition for DBpedia.

Predicate	**Gemma2**	**Llama3.1**	**Mistral**	**Qwen2.5**
	All labels			
subject	0.7637	**0.8101**	0.7066	0.5917
name	**0.8287**	0.6538	0.8129	0.6591
birthPlace	**0.7850**	0.6876	0.7485	0.4341
rdf-schema#label	**0.9649**	0.6813	0.9298	0.1316
hometown	0.5450	0.6486	**0.7658**	0.2973
abstract	0.8788	**0.8990**	**0.8990**	0.8939
capital	**0.7500**	0.7202	0.7083	0.5774
hypernym	**0.7875**	0.6000	0.7312	0.5188
rdf-schema#comment	**0.9161**	0.9097	**0.9161**	0.8516
genre	**0.7286**	0.7071	0.5357	0.5714
starring	**0.7481**	0.6412	0.6489	0.6870
isPrimaryTopicOf	**0.9675**	0.3252	0.5041	0.4472
country	0.5046	**0.5688**	0.4771	0.4037
deathPlace	0.5800	**0.6900**	0.6000	0.3300
currency	**0.8211**	0.7579	0.5789	0.6000
location	**0.8478**	0.7391	0.8043	0.6522
language	**0.6588**	**0.6588**	0.6000	0.5529
hasPhotoCollection	**0.4881**	0.4286	0.4286	0.4167
birthDate	**0.9403**	0.7761	0.6269	0.8657
producer	**0.8226**	**0.8226**	0.7258	0.4194
Total	**0.7812**	0.7189	0.7304	0.5428
	Positive labels			
subject	0.8173	**0.8706**	0.7412	0.5992
name	**0.9402**	0.6948	0.9093	0.6928
birthPlace	**0.8370**	0.7174	0.7826	0.4065
rdf-schema#label	**0.9677**	0.6833	0.9326	0.1290
hometown	0.5260	0.7031	**0.8438**	0.2344
abstract	0.9010	0.9167	**0.9271**	0.9167
capital	**0.7410**	0.7122	0.7194	0.5108
hypernym	**0.8489**	0.5971	0.7698	0.4748

Table 6.27 continued from previous page

rdf-schema#comment	**0.9724**	0.9655	0.9655	0.8828
genre	**0.7851**	0.7603	0.5041	0.5207
starring	**0.7647**	0.6387	0.6723	0.6807
isPrimaryTopicOf	**0.9917**	0.3333	0.5083	0.4417
country	0.4952	**0.5810**	0.4667	0.3810
deathPlace	0.6044	**0.7363**	0.6264	0.2747
currency	**0.8481**	0.8228	0.5316	0.5316
location	**0.9157**	0.8193	0.8675	0.6627
language	**0.6707**	**0.6707**	0.5854	0.5366
hasPhotoCollection	**0.4314**	0.3137	0.1569	0.0784
birthDate	**0.9841**	0.8254	0.6667	0.8730
producer	0.8621	**0.8793**	0.7586	0.4138
Total	**0.8279**	0.7571	0.7638	0.5306
	Negative labels			
subject	0.2222	0.1984	0.3571	**0.5159**
name	0.2069	0.4253	0.2759	**0.4713**
birthPlace	0.0606	0.2727	0.2727	**0.8182**
rdf-schema#label	0.0000	0.0000	0.0000	**1.0000**
hometown	0.6667	0.3000	0.2667	**0.7000**
abstract	0.1667	**0.3333**	0.0000	0.1667
capital	0.7931	0.7586	0.6552	**0.8966**
hypernym	0.3810	0.6190	0.4762	**0.8095**
rdf-schema#comment	0.1000	0.1000	0.2000	**0.4000**
genre	0.3684	0.3684	0.7368	**0.8947**
starring	0.5833	0.6667	0.4167	**0.7500**
isPrimaryTopicOf	0.0000	0.0000	0.3333	**0.6667**
country	0.7500	0.2500	0.7500	**1.0000**
deathPlace	0.3333	0.2222	0.3333	**0.8889**
currency	0.6875	0.4375	0.8125	**0.9375**
location	0.2222	0.0000	0.2222	**0.5556**
language	0.3333	0.3333	**1.0000**	**1.0000**
hasPhotoCollection	0.5758	0.6061	0.8485	**0.9394**
birthDate	0.2500	0.0000	0.0000	**0.7500**

Table 6.27 continued from previous page

producer	0.2500	0.0000	0.2500	**0.5000**
Total	0.3399	0.3573	0.4139	**0.6580**

Table 6.27: Decomposition of Baseline accuracy by predicate for DBpedia. For each predicate and each model, the highest scores are highlighted in bold.

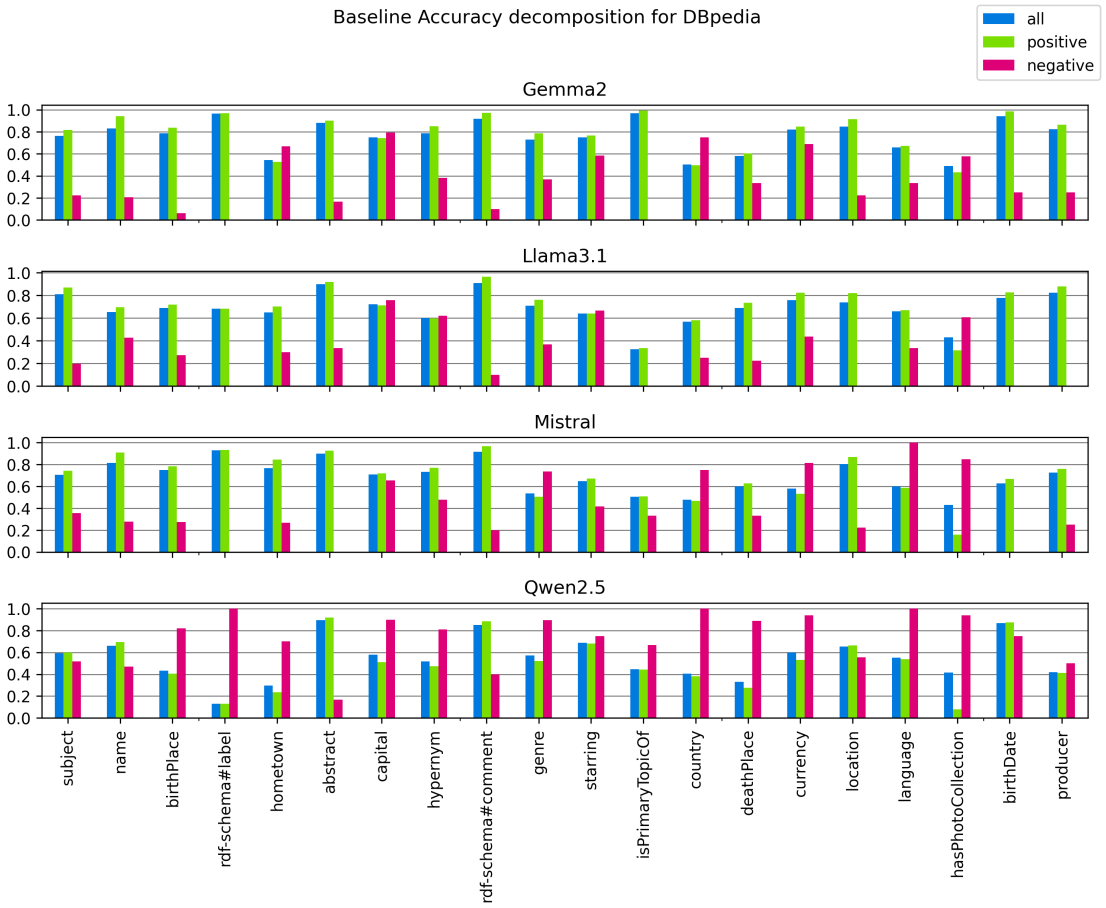

Figure 6.7: Visualization of the decomposition of Baseline Accuracy for DBpedia.

By examining the results, it is clear that Gemma2 and Llama3.1 consistently achieve higher accuracy on positive labeled instances. Mistral shows a mixed performance, still leaning towards a higher accuracy on the positives. In contrast, Qwen2.5 stands out for its strength on the negatives.

The plot also highlights how Gemma2 and Llama3.1 struggle in recognizing false facts, and how Qwen2.5 is lacking on the true ones. As mentioned, Mistral

falls in between: it's very good in the positives, but still succeeds in correctly identifying most of the negatives for some predicates (*e.g.*, `country`, `currency`, `language`).

Taking a closer look, we can identify the best and worst performing predicates for each label (positive and negative). Among the positive labels, the best performing predicates are `subject`, `name`, `abstract`, `capital`, `rdf-schema#comment`, `location` and `birthDate`, while the worst are `isPrimaryTopicOf` and `hasPhotoCollection`.

For the negatives, the best performing predicates are `capital`, `starring`, `country`, `currency` and `hasPhotoCollection`. The worst are `rdf-schema#label`, `rdf-schema#comment`, `birthDate` and `producer`.

6.4.2 FACTBENCH

The results for FactBench are reported in Table 6.28 and Figure 6.8. Here, it is evident that the models struggle a lot less in general (with respect to DBpedia), with the lowest scores concentrating on the positive instances. We attribute the great performance on the negative accuracy to the way in which those instances are constructed. Indeed, FactBench builds them starting from the correct facts by randomly replacing some items in the triples, resulting in erroneous facts that can be detected with greater ease.

Concentrating next on the positive triples, we can see that Gemma2 struggles more on the `foundationPlace` predicate; Llama3.1 achieves lower results in `spouse` and `starring`; Mistral shows some weaknesses in evaluating `award`, `foundationPlace`, `office`, and `team`; and Qwen2.5 frequently (or completely) fails in `foundationPlace`, `office`, `spouse`, `team` and `award`.

Predicate	Gemma2	Llama3.1	Mistral	Qwen2.5
	All labels			
author	**0.7622**	**0.7622**	0.7098	0.7483
award	**0.9424**	0.8957	0.6223	0.6978
birthPlace	0.7963	0.7778	**0.8074**	0.7148
deathPlace	0.7882	0.8194	**0.8438**	0.7951
foundationPlace	0.5878	0.6093	**0.6237**	0.5090
office	0.7266	**0.8993**	0.5791	0.4640
spouse	0.7177	0.6361	**0.7619**	0.5408
starring	0.7536	0.6449	**0.7754**	0.7572
subsidiary	0.7183	0.6620	**0.7641**	0.6408
team	**0.6629**	**0.6629**	0.588	0.6255
Total	**0.7457**	0.7368	0.7086	0.6493
	Positive labels			
author	**0.8733**	0.7533	0.6000	0.6200
award	**0.9533**	0.9200	0.3067	0.4400
birthPlace	**0.7533**	0.7133	0.7400	0.5333
deathPlace	0.6200	0.7533	**0.7733**	0.6733
foundationPlace	0.3667	**0.5000**	0.3933	0.1533
office	0.5933	**0.9400**	0.2933	0.0200
spouse	**0.6333**	0.4533	**0.6333**	0.1067
starring	0.6667	0.4200	**0.7733**	0.6800
subsidiary	**0.9267**	0.7733	0.8600	0.4200
team	**0.6333**	0.5333	0.3333	0.3667
Total	**0.7020**	0.6760	0.5707	0.4013
	Negative labels			
author	0.6397	0.7721	0.8309	**0.8897**
award	0.9297	0.8672	0.9922	**1.0000**
birthPlace	0.8500	0.8583	0.8917	**0.9417**
deathPlace	**0.9710**	0.8913	0.9203	0.9275
foundationPlace	0.8450	0.7364	0.8915	**0.9225**
office	0.8828	0.8516	0.9141	**0.9844**
spouse	0.8056	0.8264	0.8958	**0.9931**
starring	0.8571	**0.9127**	0.7778	0.8492
subsidiary	0.4851	0.5373	0.6567	**0.8881**
team	0.7009	0.8291	0.9145	**0.9573**
Total	0.7962	0.8069	0.8677	**0.9354**

Table 6.28: Decomposition of Baseline accuracy by predicate for the FactBench dataset. For each predicate and each model, the highest scores are highlighted in bold.

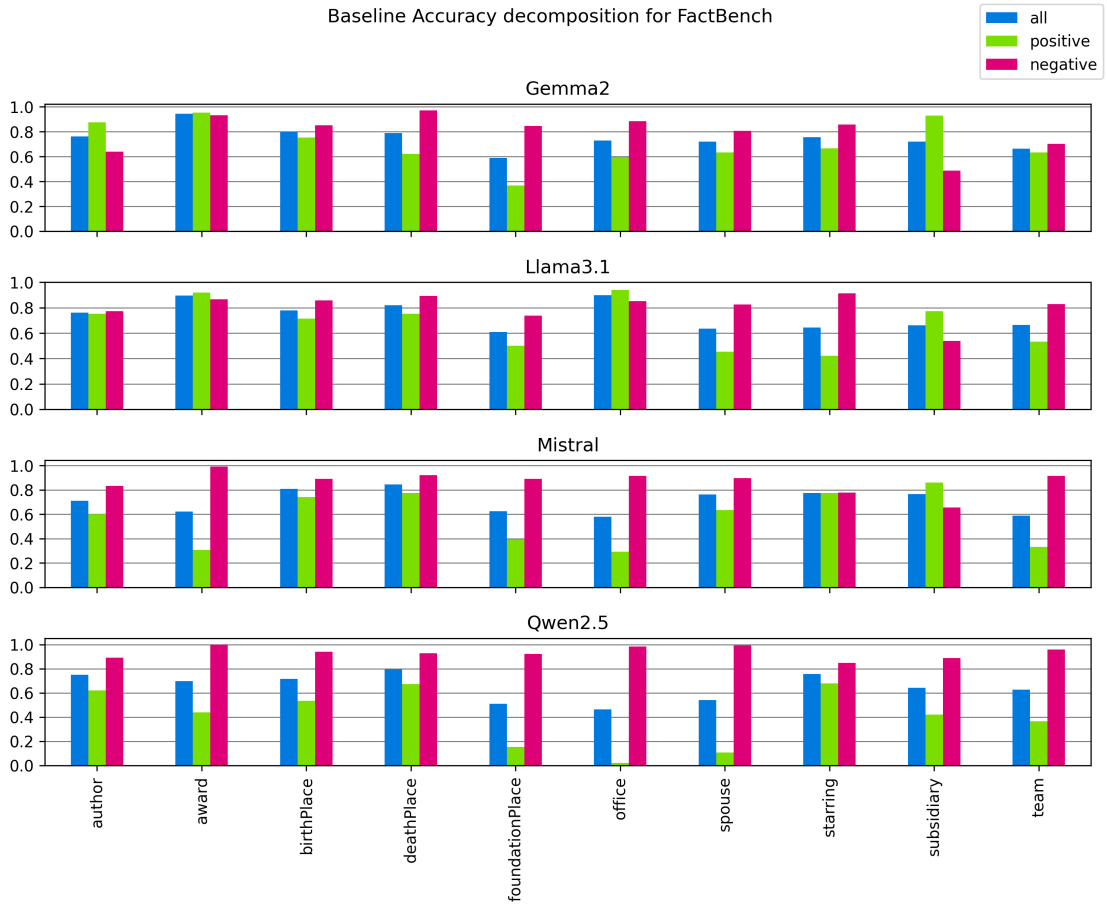

Figure 6.8: Visualization of the decomposition of Baseline Accuracy for FactBench.

6.4.3 NELL

Figure 6.9 makes immediately apparent the imbalances of NELL, as a limited number of predicates lack negative instances. The plots include the gray patches to differentiate actual zeros in performance, *e.g.*, the ones in `stadiumlocatedincity`, from "fake" zeros, caused by the lack of triples.

At a glance, we can see how the performance on the positive instances drops for Mistral and Qwen2.5 when the predicate involve athletes, despite a stronger performance on the negative accuracy.

Qwen2.5 shows again its ability in recognizing negatives better than positives. The low scores on the positive instances confirm the notable amount of false negative predictions. Gemma2 is recognizing both positives and negatives fairly well, although it achieves weaker results in some predicates for both labels (see `coachesteam`, `sportusesstadium` and `worksfor` for the positive labels, and `athletecoach`, `athleteplaysinleague` and `stadiumlocatedincity` for the negative labels). Mistral closely resemble the performance of Gemma2, with few differences regarding the negatives (see `athletecoach`, `athletehomestadium`, `teamhomestadium` and `worksfor`).

Llama3.1 falls in the middle, showing some low scores in both positives (the `athlete` predicates) and negatives (the `sport` and `stadium` predicates). It also fails completely on the instances of `sportsgameloser`, giving the opposite answer in all occurrences. However, a short investigation reveals the presence of just two triples for that predicate.

Another curious predicate is `stadiumlocatedincity`, where all models fail in recognizing the single negative instance.

Predicate	**Gemma2**	**Llama3.1**	**Mistral**	**Qwen2.5**
	All labels			
athletecoach	0.7500	**1.0000**	0.5000	0.7500
athletehomestadium	**0.7273**	0.6364	0.3636	0.4545
athleteledsportsteam	**0.6812**	0.5546	0.6245	0.4760
athleteplaysforteam	**0.5175**	0.4693	0.4649	0.3904
athleteplaysinleague	**0.8428**	0.6601	0.5265	0.2279
athleteplayssport	**0.8428**	0.5167	0.6798	0.2574

Table 6.29 continued from previous page

coachesteam	0.4348	0.5217	**0.5652**	0.5217
leaguestadiums	**0.6988**	0.5422	0.4458	0.3133
organizationhiredperson	**1.0000**	**1.0000**	**1.0000**	**1.0000**
sportsgameloser	**1.0000**	0.0000	**1.0000**	0.5000
sportsgamewinner	**1.0000**	**1.0000**	**1.0000**	0.5000
sportusesstadium	0.1250	**0.5000**	0.2500	0.2500
stadiumlocatedincity	0.7308	0.3846	**0.8077**	0.3846
teamhomestadium	0.7423	0.6392	**0.8351**	0.5155
teamplaysagainstteam	**1.0000**	**1.0000**	0.0000	0.5000
teamplaysincity	0.4828	**0.7931**	**0.7931**	**0.7931**
teamplaysinleague	0.7955	**0.9091**	0.8636	0.6932
worksfor	**0.5556**	0.4444	0.3333	0.4444
Total	**0.7522**	0.5855	0.6075	0.3468
	Positive labels			
athletecoach	**1.0000**	**1.0000**	0.3333	0.6667
athletehomestadium	**0.6667**	**0.6667**	0.3333	0.3333
athleteledsportsteam	**0.6648**	0.5275	0.5989	0.3791
athleteplaysforteam	**0.4528**	0.3836	0.2767	0.1572
athleteplaysinleague	**0.8606**	0.6707	0.5293	0.2222
athleteplayssport	**0.8452**	0.5119	0.6786	0.2500
coachesteam	0.1667	0.2500	**0.5000**	0.1667
leaguestadiums	**0.7037**	0.5309	0.4444	0.2963
organizationhiredperson	**1.0000**	**1.0000**	**1.0000**	**1.0000**
sportsgameloser	**1.0000**	0.0000	**1.0000**	0.0000
sportsgamewinner	**1.0000**	**1.0000**	**1.0000**	0.5000
sportusesstadium	0.1250	**0.5000**	0.2500	0.2500
stadiumlocatedincity	0.7600	0.4000	**0.8400**	0.4000
teamhomestadium	0.7340	0.6383	**0.8511**	0.5000
teamplaysagainstteam	**1.0000**	**1.0000**	0.0000	0.5000
teamplaysincity	0.4828	**0.7931**	**0.7931**	**0.7931**
teamplaysinleague	0.8023	**0.9186**	0.8721	0.6977
worksfor	0.3333	**0.5000**	0.3333	0.1667
Total	**0.7610**	0.5803	0.5945	0.2984

Table 6.29 continued from previous page

	Negative labels			
athletecoach	0.0000	**1.0000**	**1.0000**	**1.0000**
athletehomestadium	**1.0000**	0.5000	0.5000	**1.0000**
athleteledsportsteam	0.7447	0.6596	0.7234	**0.8511**
athleteplaysforteam	0.6667	0.6667	0.8986	**0.9275**
athleteplaysinleague	0.2143	0.2857	**0.4286**	**0.4286**
athleteplayssport	0.6000	**1.0000**	0.8000	**1.0000**
coachesteam	0.7273	0.8182	0.6364	**0.9091**
leaguestadiums	0.5000	**1.0000**	0.5000	**1.0000**
organizationhiredperson	–	–	–	–
sportsgameloser	**1.0000**	0.0000	**1.0000**	**1.0000**
sportsgamewinner	–	–	–	–
sportusesstadium	–	–	–	–
stadiumlocatedincity	0.0000	0.0000	0.0000	0.0000
teamhomestadium	**1.0000**	0.6667	0.3333	**1.0000**
teamplaysagainstteam	–	–	–	–
teamplaysincity	–	–	–	–
teamplaysinleague	**0.5000**	**0.5000**	**0.5000**	**0.5000**
worksfor	**1.0000**	0.3333	0.3333	**1.0000**
Total	0.6584	0.6398	0.7453	**0.8571**

Table 6.29: Decomposition of the Baseline accuracy by predicate for the NELL dataset. For each predicate and each model, the highest scores are highlighted in bold.

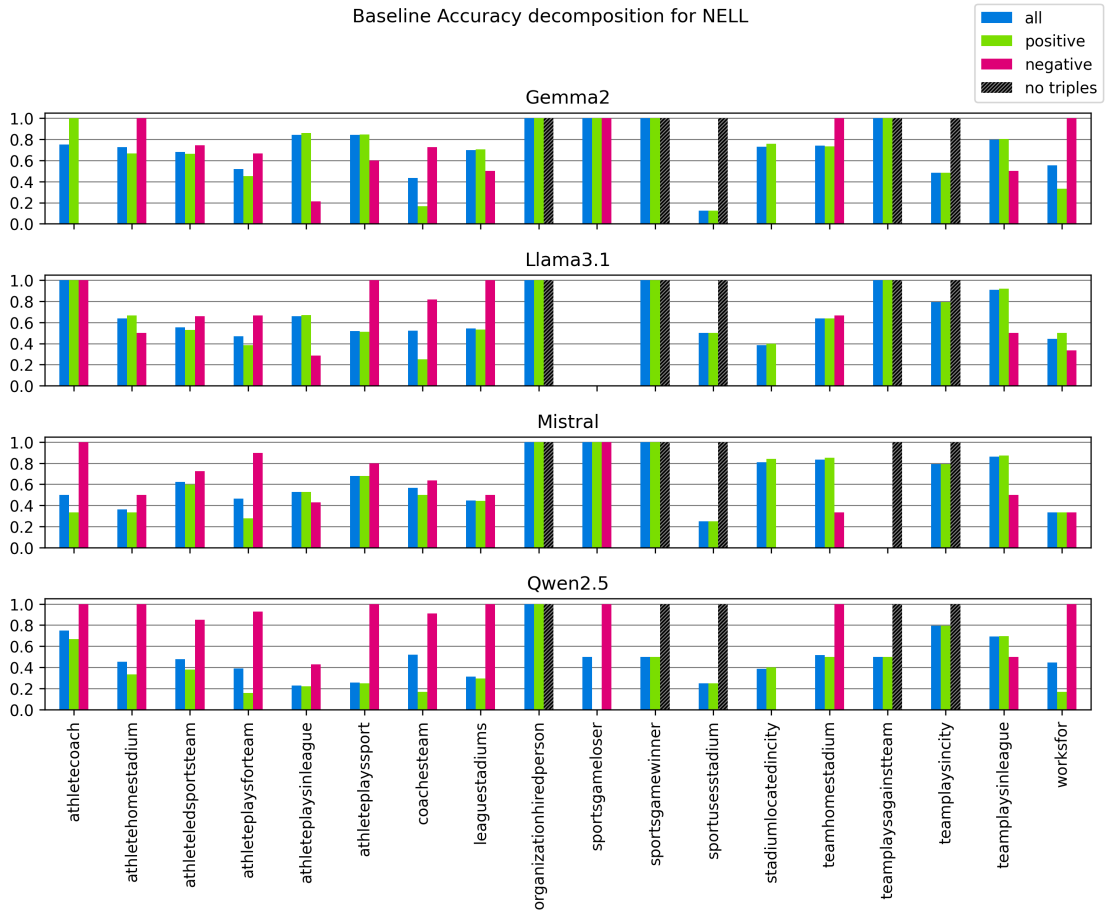

Figure 6.9: Visualization of the decomposition of Baseline Accuracy for NELL.

6.4.4 YAGO

Figure 6.10 shows the imbalance of YAGO, which is even more obvious as YAGO has more predicates than negative labeled triples. In fact, we have negative instances only for 4 predicates out of 16, namely `created`, `directed`, `hasChild` and `produced`.

The models present significant variations in performance. For example, Llama3.1 achieves high accuracy on the positive instances of `hasChild`, but its performance on negative instances is notably poor, as evidenced by the red bars. In constrast, Qwen2.5 achieves high accuracy for the negative instances and lacks in the positives.

Some predicates, *e.g.*, `hasAcademicAdvisor`, show uniformly low performance across models, indicating their difficulty. Others, like `hasChild`, present

relevant gaps between positive and negative accuracy, suggesting biases in how models handle imbalanced data. In practical terms, this means models exhibit different behaviors depending on the type of instance they're evaluating (positive or negative).

This decomposition reveals crucial insights, as Accuracy (represented by the blue bars) may not reflect the true behavior of the models in imbalanced data.

Predicate	**Gemma2**	**Llama3.1**	**Mistral**	**Qwen2.5**
	All labels			
actedIn	0.8158	0.7105	**0.8684**	0.4211
created	0.6048	**0.6129**	0.5403	0.3185
diedIn	**0.3333**	**0.3333**	0.0000	0.0000
directed	0.6466	**0.6552**	0.5733	0.4138
hasAcademicAdvisor	0.0000	**1.0000**	0.5000	0.0000
hasChild	**0.6818**	0.1364	0.1818	0.1364
hasOfficialLanguage	**0.5000**	**0.5000**	**0.5000**	**0.5000**
isCitizenOf	**0.5000**	**0.5000**	0.0000	0.0000
isKnownFor	**1.0000**	**1.0000**	0.8750	0.8750
isLeaderOf	0.0000	0.0000	0.0000	0.0000
isLocatedIn	**1.0000**	**1.0000**	0.5000	**1.0000**
isMarriedTo	**0.7443**	0.4859	0.2999	0.2008
livesIn	**0.2727**	0.1818	0.0000	0.0000
produced	0.7414	**0.7759**	0.7414	0.3793
wasBornIn	**0.6000**	0.0000	0.2000	0.2000
worksAt	**1.0000**	0.5000	**1.0000**	0.5000
Total	**0.6977**	0.5491	0.4221	0.2734
	Positive labels			
actedIn	0.8158	0.7105	**0.8684**	0.4211
created	0.6107	**0.6189**	0.5410	0.3074
diedIn	**0.3333**	**0.3333**	0.0000	0.0000
directed	0.6494	**0.6580**	0.5714	0.4113
hasAcademicAdvisor	0.0000	**1.0000**	0.5000	0.0000
hasChild	**0.7000**	0.0500	0.1500	0.0500
hasOfficialLanguage	**0.5000**	**0.5000**	**0.5000**	**0.5000**

Table 6.30 continued from previous page

isCitizenOf	**0.5000**	**0.5000**	0.0000	0.0000
isKnownFor	**1.0000**	**1.0000**	0.8750	0.8750
isLeaderOf	0.0000	0.0000	0.0000	0.0000
isLocatedIn	**1.0000**	**1.0000**	0.5000	**1.0000**
isMarriedTo	**0.7443**	0.4859	0.2999	0.2008
livesIn	**0.2727**	0.1818	0.0000	0.0000
produced	0.7593	**0.7778**	**0.7778**	0.3519
wasBornIn	**0.6000**	0.0000	0.2000	0.2000
worksAt	**1.0000**	0.5000	**1.0000**	0.5000
Total	**0.7004**	0.5491	0.4218	0.2684
	Negative labels			
actedIn	–	–	–	–
created	0.2500	0.2500	0.5000	**1.0000**
diedIn	–	–	–	–
directed	0.0000	0.0000	**1.0000**	**1.0000**
hasAcademicAdvisor	–	–	–	–
hasChild	0.5000	**1.0000**	0.5000	**1.0000**
hasOfficialLanguage	–	–	–	–
isCitizenOf	–	–	–	–
isKnownFor	–	–	–	–
isLeaderOf	–	–	–	–
isLocatedIn	–	–	–	–
isMarriedTo	–	–	–	–
livesIn	–	–	–	–
produced	0.5000	**0.7500**	0.2500	**0.7500**
wasBornIn	–	–	–	–
worksAt	–	–	–	–
Total	0.3636	0.5455	0.4545	**0.9091**

Table 6.30: Decomposition of the Baseline accuracy by predicate for the YAGO dataset. For each predicate and each model, the highest scores are highlighted in bold.

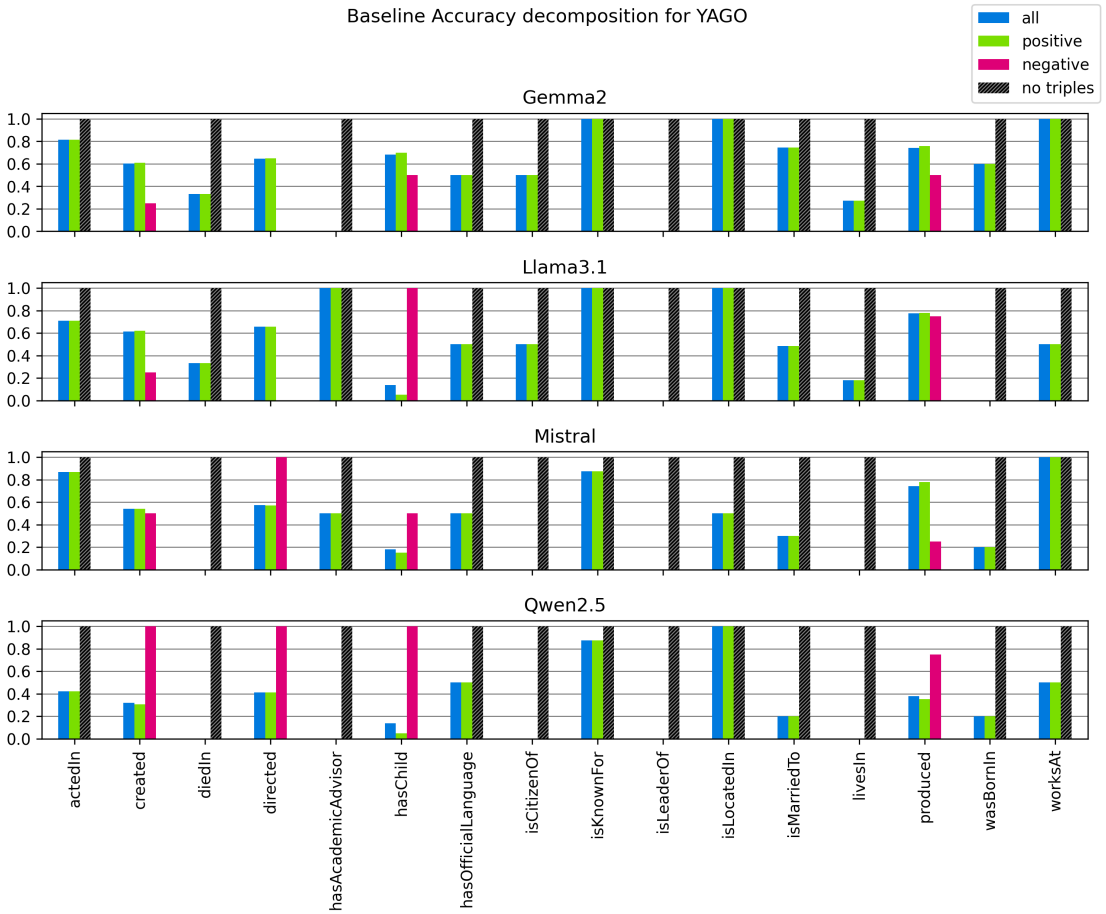

Figure 6.10: Visualization of the decomposition of Baseline Accuracy for YAGO.

7 Implementation

This chapter provides an overview of the software tools and libraries employed for this thesis, discussing the frameworks used for the implementation of the annotation, evaluation and plotting components. Following this, we describe the three pipelines (Baseline, Iterative, and Stepwise) presenting their underlying algorithms by means of pseudocode.

7.1 Technologies and Tools

The implementation of this thesis was carried out using Python as the sole programming language, making use of both standard and third-party libraries. To develop the code for the annotation task the following libraries were utilized:

- `transformers`[1] library: third-party library provided by Hugging Face[2], used to load and prompt the models;
- `argparse` library: it manages command line parameters, used for the run configuration;
- `json` and `pathlib` libraries: for file I/O operations;
- `time` library: it provides a straightforward way to perform time measurements;

[1] `https://huggingface.co/docs/transformers/index`
[2] `https://huggingface.co/`

- `tqdm`[3] library: third party library employed for progress visualization purposes only.

The code for the computation of the evaluation metrics relies on `json`, `pathlib`, and `pandas`[4] for the file management, and on `numpy`[5] and `scikit-learn`[6] for the metrics themselves.

Lastly, `pandas` and `matplotlib`[7] were employed to plot the results.

7.2 Baseline pipeline

The Baseline annotation process algorithm annotates the KG in exam using the Baseline prompt. We report pseudocode in Algorithm 7.1.

The inputs are: (a) the KG, represented as a dictionary of triples composed by fact IDs and corresponding facts; (b) an instance of an LLM; (c) a prompt template $\mathcal{P}$. The results of the procedure are stored in a dictionary, that will be saved on disk at the end of the annotation process.

For each fact in the KG, we inject the fact into the prompt $\mathcal{P}$ and we query the LLM using the formatted prompt $\mathcal{P}_f$. Finally, the response is collected into the `results` dictionary using the fact ID as the dictionary key.

[3] `https://tqdm.github.io/`
[4] `https://pandas.pydata.org/`
[5] `https://numpy.org/`
[6] `https://scikit-learn.org/stable/`
[7] `https://matplotlib.org/stable/`

Algorithm 7.1 Baseline annotation process

Input:
KG $\mathcal{G}$: a dictionary of facts $\{(\texttt{id}_1, f_1), \ldots, (\texttt{id}_n, f_n)\}$
LLM $\mathcal{L}$: a Large Language Model instance
Prompt $\mathcal{P}$: the prompt template
Output: Results saved as a JSON file

1: `results` $\leftarrow \emptyset$ ▷ *dictionary to store results (initially empty)*
2: **for each** (`id`, f) $\in \mathcal{G}$ **do**
3: $\mathcal{P}_f \leftarrow$ `format`($\mathcal{P}$, f) ▷ *format prompt with fact*
4: `resp` $\leftarrow$ `query`($\mathcal{L}$, $\mathcal{P}_f$)
5: `results` $\leftarrow$ `results` $\cup$ {(`id`, `resp`)} ▷ *store the response*
6: Save `results` to a JSON file

7.3 Iterative pipeline

The Iterative annotation process builds on the Baseline approach by introducing the retry mechanism to handle invalid responses. Similarly to the Baseline algorithm, it processes all the facts contained in a KG and generates annotations using an LLM. Pseudocode is provided in Algorithm 7.2.

The inputs are the same of the Baseline algorithm, with the addition of a second prompt $\mathcal{R}$ tailored for the retry mechanism and a maximum number of attempts allowed A_{max}. The retry mechanism corresponds to the `while` loop (lines 7–14).

For each fact in the KG, we start by formatting the prompt $\mathcal{P}$ with the fact and querying the model with the formatted prompt $\mathcal{P}_f$. If the response is valid, it gets saved in the results dictionary and the for loop moves onto the next fact. If the response is invalid, the retry mechanism is triggered.

When this happens, a list containing the chat history, *i.e.*, the whole conversation, is created. This list serves as the model's memory, providing it with all the previous (erroneous) answers.

In the retry mechanism we employ the so-called "retry" prompt $\mathcal{R}$. This prompt differs from the initial one: it explains that the previous answer didn't meet the guidelines, reminds how many attempts are still available, and recaps

the guidelines with a short summary. Each time the model is queried, the chat history is updated. The loop continues until: (a) the model provides a valid answer or (b) the maximum number of attempts is reached, leaving the response invalid. In both cases, the response and the number of attempts are collected into the results dictionary, which will be saved to disk at the end of the procedure.

Algorithm 7.2 Iterative annotation process

Input:
KG $\mathcal{G}$: a dictionary of facts $\{(\texttt{id}_1, f_1), \ldots, (\texttt{id}_n, f_n)\}$
LLM $\mathcal{L}$: a Large Language Model instance
Prompt $\mathcal{P}$: the prompt template
Prompt $\mathcal{R}$: the "retry" prompt template
A_{max}: max number of attempts allowed
Output: Results saved as a JSON file

1: `results` $\leftarrow \emptyset$ ▷ *dictionary to store results (initially empty)*
2: **for each** $(\texttt{id}, f) \in \mathcal{G}$ **do**
3: $\mathcal{P}_f \leftarrow \texttt{format}(\mathcal{P}, f)$ ▷ *format prompt with fact*
4: `att` $\leftarrow 0$ ▷ *init attempts counter*
5: `resp` $\leftarrow \texttt{query}(\mathcal{L}, \mathcal{P}_f)$
6: `history` $\leftarrow [\mathcal{P}_f]$ ▷ *list to store chat history (initialized with the first prompt)*
7: **while** `resp` $\notin \{\text{T}, \text{F}\}$ **do**
8: **if** `att` $== A_{max}$ **then**
9: `resp` $\leftarrow$ NA ▷ *attempts exhausted: invalid response*
10: **break** ▷ *exit from the while loop*
11: $\mathcal{R}_f \leftarrow \texttt{format}(\mathcal{R}, f, A_{max} - \texttt{att})$ ▷ *format retry prompt with fact and remaining attempts*
12: Add `resp` and $\mathcal{R}_f$ to `history`
13: `resp` $\leftarrow \texttt{query}(\mathcal{L}, \texttt{history})$ ▷ *ask LLM again using retry prompt*
14: `att` $\leftarrow$ `att` $+ 1$ ▷ *update attempts counter*
15: `results` $\leftarrow$ `results` $\cup \{(\texttt{id}, \texttt{resp}, \texttt{att})\}$ ▷ *store the response*
16: Save `results` to a JSON file

7.4 Stepwise pipeline

The Stepwise annotation process extends the annotation workflow by dividing the task into multiple steps, querying the the LLM in a step-by-step manner. The procedure is described in Algorithm 7.3.

The inputs are: (a) the KG, (b) the LLM, (c) the "rewrite" prompt $\mathcal{R}$, which is employed in the first step of the pipeline (statement production), (d) the "confidence" prompt $\mathcal{C}$, to be used in the confidence assessment step, and (e) the prompt $\mathcal{P}$ for the truthfulness evaluation.

In contrast to both Baseline and Iterative, we now have two dictionaries in output: one contains the statements and one contains the responses and the confidence levels. Both dictionaries use the fact IDs as keys.

For each fact, we format the "rewrite" prompt $\mathcal{R}$ and query the model with the prompt $\mathcal{R}_f$. The purpose of this prompt is to obtain a statement `stat` in natural language representing the fact. If the model is able to generate a statement, we proceed with the confidence evaluation, otherwise an invalid response for the fact is registered and the process advances to the next fact.

In the confidence assessment step, the "confidence" prompt $\mathcal{C}$ gets formatted with the statement, and the chat history[8] is filled with $\mathcal{R}_f$, the statement `stat` and the formatted "confidence" prompt $\mathcal{C}_s$. Then, we query the model with the chat history, asking for a confidence level `conf`.

If the model exhibits high confidence, we format the evaluation prompt $\mathcal{P}$ with the statement, we add the confidence level `conf` and the prompt $\mathcal{P}_s$ to the chat history, and we query the model with the entire conversation. Conversely, on the low confidence scenario, we ignore the chat history and we query the model using only $\mathcal{P}_s$. Eventually, we save the results, *i.e.*, the response and the confidence level. At the end of the for loop we save to disk both the statements and the results dictionaries.

[8] As in Algorithm 7.2, the chat history stores the whole conversation and acts as the model's memory.

Algorithm 7.3 Stepwise annotation process

Input:
KG $\mathcal{G}$: a dictionary of facts $\{(\texttt{id}_1, f_1), \ldots, (\texttt{id}_n, f_n)\}$
LLM $\mathcal{L}$: a Large Language Model instance
Prompt $\mathcal{R}$: the "rewrite" prompt template
Prompt $\mathcal{C}$: the "confidence" prompt template
Prompt $\mathcal{P}$: the prompt template for the truthfulness evaluation

Output:
Statements saved as a JSON file
Results saved as a JSON file

```
statements ← ∅                                  ▷ dictionary to store statements (initially empty)
results ← ∅                                     ▷ dictionary to store results (initially empty)
for each (id, f) ∈ G do
    history ← [ ]                               ▷ list to store chat history (initially empty)
    R_f ← format(R, f)                          ▷ format "rewrite" prompt with fact
    stat ← query(L, R_f)                        ▷ statement generation
    statements ← statements ∪ {(id, stat)}      ▷ store the statement
    if stat ≠ IDK then                          ▷ statement is valid
        C_s ← format(C, s)                      ▷ format "confidence" prompt with statement
        Add R_f, stat and C_s to history
        conf ← query(L, history)                ▷ confidence assessment
        P_s ← format(P, s)                      ▷ format prompt with statement
        Add conf and P_s to history
        if c == H then
            resp ← query(L, history)            ▷ truthfulness evaluation (high conf.)
        else
            resp ← query(L, P_s)                ▷ truthfulness evaluation (low conf.)
        results ← results ∪ {(id, resp, conf)}  ▷ store the response
    else
        results ← results ∪ {(id, NA, NA)}      ▷ store NA (invalid statement)
Save statements to a JSON file
Save results to a JSON file
```

8

Conclusions and Future Works

The purpose of this thesis was to investigate on the potential of Large Language Models as KGs Accuracy Estimators, employing pre-trained models and leveraging only their internal knowledge for the annotation process of a set of triples from a Knowledge Graph.

We started by defining the building blocks of our research, namely the Resource Description Framework, Knowledge Graphs and Large Language Models. RDF is the foundational framework to represent data in the Semantic Web, providing a standardized way to structure and link data. This framework is also critical for the creation of Knowledge Graphs, which, simplifying a lot, are collections of interconnected data. We concluded this introductory part with an overview on Large Language Models, highlighting their key features and limitations.

We continued by introducing the Accuracy Evaluation task: for each triple (*i.e.*, fact) in a KG, the goal is to assess its veracity by asking whether the triple contains a true fact or a false fact. At the core of this thesis lies the exploration of Large Language Models as annotators for Knowledge Graphs, investigating their potential to assess and validate the accuracy of such data.

Our main contribution in this regard consisted in the development of three prompting techniques: the Baseline, the Iterative, and the Stepwise technique. The Baseline technique is the simplest one: it uses a direct and concise prompt, serving as a reference point in the analysis of the strengths and the weaknesses of the other two techniques. The Iterative technique builds upon the Baseline by

enriching the prompt with more detailed instructions and incorporating a retry mechanism designed to resolve initial invalid responses within a predefined number of retries. This technique was tested using both zero-shot and few-shot prompt, as well as some variants of the style of the prompt. Lastly, the Stepwise technique approaches the Accuracy Evaluation task from another perspective, analyzing the capabilities of the models in evaluating natural language statements, with insights on their confidence about their own evaluation.

We conducted experiments on four models within the 7–9 billions parameters range, namely Gemma2, Llama3.1, Mistral and Qwen2.5, evaluating their performance on four datasets: DBpedia, FactBench, NELL, and YAGO. The evaluation comprises standard metrics for all techniques (Accuracy, Balanced Accuracy, Precision, Recall, F1 Score, Informativeness, Truthfulness) as well as time measurements (average and total), with the addition of customized metrics for the Iterative technique (Compliance and Average Number of Retries per Fact) and Stepwise technique (Statement rate, Give-up rate, Right|G and Right|¬G). We describe the main findings below, dividing them by the prompting technique.

The Baseline sees Gemma2 as the top performer, achieving consistent results across all datasets, while the other models present deterioration in performance on NELL and YAGO. This pipeline was the quickest one, as a consequence of the short prompt.

For the Iterative technique, the few-shot variants outperformed the zero-shot ones. This is particularly evident when considering Llama3.1, and confirms the validity of the few-shot examples, demonstrating their tangible support in helping models assess the veracity of facts. Both the style variants (*i.e.*, NI and SN) prove reliable, showing that the instructions provide solid guidance regardless of the way in which they are integrated into prompts. The improvements are also noticeable in the comparison against the Baseline technique. The tailored metrics showed that the models rarely need extra attempts. Time-wise, the Iterative pipeline took just over double the amount of the Baseline in the few-shot prompts, and we believe that its increased costs are justified by the improvements it brought.

In the Stepwise technique all models but Qwen2.5 achieve high accuracy across all datasets, with Gemma2 standing out again as the most consistent

model. The results show widespread improvements with respect to the Baseline technique, however, the Stepwise technique falls short when compared to the Iterative one. The tailored metrics delved into the model knowledge boundary: all models were almost always able to produce and evaluate statements. Focusing on the confidence, Llama3.1 and Mistral were the two most confident models, while Gemma2 and Qwen2.5 showed higher give up ratios. Processing times are the major setback for the Stepwise technique, as they are significantly higher than both those of Baseline and Iterative pipelines (respectively up to 12 and 5 times higher on DBpedia). The causes of these increased times are to be attributed to the Statement production step, making up for half the time needed for each fact. Overall, despite the improvements over the Baseline, the substantially higher processing times cancel out every positive aspect of this technique: for this reason, we strongly believe it is not well-suited for the task we've pursued.

We concluded our analysis with an ablation study analyzing the Accuracy results of the Baseline technique by predicate, to get more insights on the models' strenghts and weaknesses.

In DBpedia, Gemma2 and Llama3.1 outperform on positive instances compared to the negatives, while Qwen2.5 exhibits the opposite behavior, achieving a high accuracy when identifying negatives. FactBench stands out as the dataset where all models achieve strong performance on both positive and negative instances. We attribute this to the synthetic nature of its negative instances, which makes them easier to recognize. In NELL, Qwen2.5 continues to shine in handling negatives, while Gemma2 demonstrates consistent performance across both positive and negative triples. Mistral closely follows Gemma2 overall, but performs better on the negatives. Lastly, YAGO reveals significant variations in performance across models, with Qwen2.5 once again achieving high accuracy on the negative instances.

The fact that these pipelines were designed to leverage the internal knowledge of the models leaves several paths unexplored, which we outline below:

1. **Retrieval-Augmented Generation pipelines**: RAG pipelines emerged as an effective method to enhance the performance of LLMs by integrating external context into prompts. The added context is retrieved from external sources of information, such as other KGs, document collections, or the

Web. RAG overcomes the limitation of pre-trained knowledge, and helps providing grounded responses, improving the model capabilities;

2. **Fine-tuning**: fine-tuning techniques allow for the transformation of a general-purpose model into a specialized tool tailored for Fact Validation. Fine-tuning consists in an additional training on a task- or domain-specific dataset, enabling the model to learn context-specific patterns and nuances, hence improving their accuracy;

3. **Exploring larger models**: the experiments in this thesis were conducted on models within the 7–9 billions parameters range. We believe testing larger models could provide valuable insights, specifically on the relationship between model size and model accuracy (*i.e.*, the amount of factual knowledge stored by the model). Larger models have increased capacity to store knowledge, due to the higher number of parameters, and they may exhibit better accuracy. Speaking of larger models, we include open-source, research-oriented, and commercial models as well (*e.g.*, the recent GPT-4o [34]).

The investigation proposed in this thesis underscores the potential of Large Language Models in assessing the accuracy of Knowledge Graphs, demonstrating both their promises and their limitations. With the future directions outlined, we want to draw a path that addresses these limitations but also goes beyond the scope of this research to explore novel methodologies, in the hope of further enhancing the capabilities of Large Language Models in Fact Validation tasks.

Prompts

Appendix A provides the prompts that were used for the experiments. For the evaluation of DBpedia, the prompts specify that the model should access only knowledge available prior to 2016, as the dataset is based on a 2015-10 snapshot.

A.1 Baseline prompts

This section provides the prompt for the Baseline pipeline. We will first present the regular version (Prompt A.1), and then the DBpedia-specific version (Prompt A.2).

```
Only reply with "T" or "F" and nothing else.
If the user provides a fact you judge accurate, reply with the letter "T".
If the user provides a fact you judge inaccurate, reply with the letter "F".

Fact: {s} {p} {o}

Judgment:
```

Prompt A.1: Baseline prompt (regular).

```
Your knowledge is limited to the year 2015. Only reply with "T" or "F" and
nothing else.
If the user provides a fact you judge accurate (based on your knowledge up to
2015), reply with the letter "T".
If the user provides a fact you judge inaccurate (based on your knowledge up to
2015), reply with the letter "F".

Fact: {s} {p} {o}

Judgment:
```

Prompt A.2: Baseline prompt (DBpedia).

A.2 Iterative prompts

This section shows the prompts for the Iterative pipeline. The section is organized as follows: first, we introduce the examples used in the few-shot prompts. Then, we present all the prompts, starting with the zero-shot and few-shot variants in their regular form, followed by their DBpedia-specific counterparts.

A.2.1 Few-shot prompts Examples

We designed two sets of examples, each consisting of six correctly classified triples. The sets reflect the topics of the datasets: the first set, designed for DBpedia, FactBench, and YAGO, contains general-purpose examples, while the second set is tailored for NELL which focuses entirely on sports.

It is also important to mention that style in which examples re injected into the prompts matches both the dataset and the fact formatting. For brevity, the general-purpose examples are formatted for FactBench, while both sets are formatted for the FNI variant.

General-purpose examples

```
### Examples
# Fact
Curiosity (rover) location Mars

Judgment: T

# Fact
Isaac Newton knownFor Universal gravitation

Judgment: T

# Fact
Monica Bellucci birthPlace Città di Castello, Italy

Judgment: T

# Fact
Hyundai Motor Company foundationPlace Rio de Janeiro

Judgment: F

# Fact
J. K. Rowling author Lord of The Rings

Judgment: F

# Fact
La Rambla location Berlin

Judgment: F
```

Prompt A.3: General-purpose examples for the few-shot prompts of the Iterative pipeline.

NELL-specific examples

```
# Fact
fernando tatis jr athleteplaysforteam san diego padres

Judgment: T

# Fact
lebron james athleteplayesforteams los angeles lakers

Judgment: T

# Fact
mike trout athleteplaysinleague mlb

Judgment: T

### Examples
# Fact
derek jeter athleteplaysforteam chicago cubs

Judgment: F

# Fact
pete rose athleteplaysforteam red sox

Judgment: F

# Fact
red sox teamplaysincity new york

Judgment: F
```

Prompt A.4: NELL-specific examples for the few-shot prompts of the Iterative pipeline.

A.2.2 Regular prompts

Zero-shot prompts

```
You are an expert annotator for the fact verification task.
You will be provided a fact in the form of a subject, predicate, and object.
Your job is to assess the fact based solely on the knowledge you have been
trained on, without using any external information.

### Guidelines
The answer should be "T" (true) if the fact is accurate based on your
knowledge, or "F" (false) if the fact is inaccurate based on your knowledge.

Please do not spread incorrect information. Your judgment must rely only on
your internal knowledge.
You must always answer in English.
The answer must contain only the judgment ("T", "F"). Lastly, keep your answer
concise.

### Retry Mechanism
If your response does not comply with the guidelines above, you will be asked
to retry.
In the retry follow-up, you must meet the same guidelines and provide a
compliant answer.
You will have a total of {max_attempts} retry attempts.
If you fail to provide a compliant answer within the attempts, your final
response for the fact will be set to "NA" (Not an Answer).

### Input format
The fact is given in one line: first the subject, then the predicate and
finally the object.

### Fact
{s} {p} {o}

Judgment:
```

Prompt A.5: Iterative prompt for the ZNI variant (regular).

```
Your previous response did not meet the guidelines, please try again and
provide a compliant answer.
You have {attempts} remaining attempts.
If you fail to provide a compliant answer within the remaining attempts, your
final response for the fact will be set to "NA" (Not an Answer).

### Reminder of the Guidelines
The answer should be "T" (true), or "F" (false) based on your internal
knowledge. Limit the use of "N" as much as possible.
Provide your judgment in English and keep it concise. Do not spread incorrect
information.

### Input format
The fact is given in one line: first the subject, then the predicate and
finally the object.

### Fact
{s} {p} {o}

Judgment:
```

Prompt A.6: Iterative Retry prompt for the ZNI variant (regular).

```
You are an expert annotator for the fact verification task.
You will be provided a fact in the form of a subject, predicate, and object.
Your job is to assess the fact based solely on the knowledge you have been
trained on, without using any external information.

### Guidelines
1. The answer should be:
- "T" (true) if the fact is accurate based on your knowledge.
- "F" (false) if the fact is inaccurate based on your knowledge.

- Please do not spread incorrect information.
- Your judgment must rely only on your internal knowledge.
- You must always answer in English.
- The answer must contain only the judgment ("T", "F").
- Keep your answer concise.

### Retry Mechanism
If your response does not comply with the guidelines above, you will be asked
to retry.
In the retry follow-up, you must meet the same guidelines and provide a
compliant answer.
You will have a total of {max_attempts} retry attempts.
If you fail to provide a compliant answer within the attempts, your final
response for the fact will be set to "NA" (Not an Answer).

### Input format
The fact is given in an unordered list: first the subject, then the predicate
and finally the object.

### Fact
- {s}
- {p}
- {o}

Judgment:
```

Prompt A.7: Iterative prompt for the ZSN variant (regular).

```
Your previous response did not meet the guidelines, please try again and
provide a compliant answer.
You have {attempts} remaining attempts.
If you fail to provide a compliant answer within the remaining attempts, your
final response for the fact will be set to "NA" (Not an Answer).

### Reminder of the Guidelines

- The answer should be "T" (true), or "F" (false) based on your internal
knowledge.
- Limit the use of "N" as much as possible.
- Provide your judgment in English and keep it concise.
- Do not spread incorrect information.

### Input format
The fact is given in an unordered list: first the subject, then the predicate
and finally the object.

### Fact
- {s}
- {p}
- {o}

Judgment:
```

Prompt A.8: Iterative Retry prompt for the ZSN variant (regular).

Few-shot prompts

```
You are an expert annotator for the fact verification task.
You will be provided a fact in the form of a subject, predicate, and object.
Your job is to assess the fact based solely on the knowledge you have been
trained on, without using any external information.

### Guidelines
The answer should be "T" (true) if the fact is accurate based on your
knowledge, or "F" (false) if the fact is inaccurate based on your knowledge.

Please do not spread incorrect information. Your judgment must rely only on
your internal knowledge.
You must always answer in English.
The answer must contain only the judgment ("T", "F"). Lastly, keep your answer
concise.

### Retry Mechanism
If your response does not comply with the guidelines above, you will be asked
to retry.
In the retry follow-up, you must meet the same guidelines and provide a
compliant answer. You will have a total of {max_attempts} retry attempts.
If you fail to provide a compliant answer within the attempts, your final
response for the fact will be set to "NA" (Not an Answer).

### Examples
...

### Input format
The fact is given in one line: first the subject, then the predicate and
finally the object.

### Fact
{s} {p} {o}

Judgment:
```

Prompt A.9: Iterative prompt for the FNI variant (regular).

```
Your previous response did not meet the guidelines, please try again and
provide a compliant answer.
You have {attempts} remaining attempts.
If you fail to provide a compliant answer within the remaining attempts, your
final response for the fact will be set to "NA" (Not an Answer).

### Reminder of the Guidelines
The answer should be "T" (true), or "F" (false) based on your internal
knowledge. Limit the use of "N" as much as possible.
Provide your judgment in English and keep it concise. Do not spread incorrect
information.

### Examples
...

### Input format
The fact is given in one line: first the subject, then the predicate and
finally the object.

### Fact
{s} {p} {o}

Judgment:
```

Prompt A.10: Iterative Retry prompt for the FNI variant (regular).

```
You are an expert annotator for the fact verification task.
You will be provided a fact in the form of a subject, predicate, and object.
Your job is to assess the fact based solely on the knowledge you have been
trained on, without using any external information.

### Guidelines
1. The answer should be:
- "T" (true) if the fact is accurate based on your knowledge.
- "F" (false) if the fact is inaccurate based on your knowledge.

- Please do not spread incorrect information.
- Your judgment must rely only on your internal knowledge.
- You must always answer in English.
- The answer must contain only the judgment ("T", "F").
- Keep your answer concise.

### Retry Mechanism
If your response does not comply with the guidelines above, you will be asked
to retry.
In the retry follow-up, you must meet the same guidelines and provide a
compliant answer.
You will have a total of {max_attempts} retry attempts.
If you fail to provide a compliant answer within the attempts, your final
response for the fact will be set to "NA" (Not an Answer).

### Examples
...

### Input format
The fact is given in an unordered list: first the subject, then the predicate
and finally the object.

### Fact
- {s}
- {p}
- {o}

Judgment:
```

Prompt A.11: Iterative prompt for the FSN variant (regular).

```
Your previous response did not meet the guidelines, please try again and
provide a compliant answer.
You have {attempts} remaining attempts.
If you fail to provide a compliant answer within the remaining attempts, your
final response for the fact will be set to "NA" (Not an Answer).

### Reminder of the Guidelines

- The answer should be "T" (true), or "F" (false) based on your internal
knowledge.
- Limit the use of "N" as much as possible.
- Provide your judgment in English and keep it concise.
- Do not spread incorrect information.

### Examples
...

### Input format
The fact is given in an unordered list: first the subject, then the predicate
and finally the object.
### Fact
- {s}
- {p}
- {o}

Judgment:
```

Prompt A.12: Iterative Retry prompt for the FSN variant (regular).

A.2.3 DBpedia prompts

Zero-shot prompts

```
You are an expert annotator for the fact verification task.
You will be provided a fact in the form of a subject, predicate, and object.
Your job is to assess the fact based solely on the knowledge you have been
trained on, without using any external information.
Your knowledge is limited to the year 2015.

### Guidelines
The answer should be "T" (true) if the fact is accurate based on your knowledge
up to 2015, or "F" (false) if the fact is inaccurate based on your knowledge up
to 2015.

Please do not spread incorrect information. Your judgment must rely only on
your internal knowledge.
You must always answer in English.
The answer must contain only the judgment ("T", "F"). Lastly, keep your answer
concise.

### Retry Mechanism
If your response does not comply with the guidelines above, you will be asked
to retry.
In the retry follow-up, you must meet the same guidelines and provide a
compliant answer.
You will have a total of {max_attempts} retry attempts.
If you fail to provide a compliant answer within the attempts, your final
response for the fact will be set to "NA" (Not an Answer).

### Input format
The fact is given in one line: first the subject, then the predicate and
finally the object.

### Fact
{s} {p} {o}

Judgment:
```

Prompt A.13: Iterative prompt for the ZNI variant (DBpedia).

```
Your previous response did not meet the guidelines, please try again and
provide a compliant answer.
You have {attempts} remaining attempts.
If you fail to provide a compliant answer within the remaining attempts, your
final response for the fact will be set to "NA" (Not an Answer).

### Reminder of the Guidelines
Your knowledge is limited to the year 2015. The answer should be "T" (true), or
"F" (false) based on your internal knowledge. Limit the use of "N" as much as
possible.
Provide your judgment in English and keep it concise. Do not spread incorrect
information.

### Input format
The fact is given in one line: first the subject, then the predicate and
finally the object.

### Fact
{s} {p} {o}

Judgment:
```

Prompt A.14: Iterative Retry prompt for the ZNI variant (DBpedia).

```
You are an expert annotator for the fact verification task.
You will be provided a fact in the form of a subject, predicate, and object.
Your job is to assess the fact based solely on the knowledge you have been
trained on, without using any external information.
Your knowledge is limited to the year 2015.

### Guidelines
1. The answer should be:
- "T" (true) if the fact is accurate based on your knowledge up to 2015.
- "F" (false) if the fact is inaccurate based on your knowledge up to 2015.

- Please do not spread incorrect information.
- Your judgment must rely only on your internal knowledge.
- You must always answer in English.
- The answer must contain only the judgment ("T", "F").
- Keep your answer concise.

### Retry Mechanism
If your response does not comply with the guidelines above, you will be asked
to retry.
In the retry follow-up, you must meet the same guidelines and provide a
compliant answer.
You will have a total of {max_attempts} retry attempts.
If you fail to provide a compliant answer within the attempts, your final
response for the fact will be set to "NA" (Not an Answer).

### Input format
The fact is given in an unordered list: first the subject, then the predicate
and finally the object.

### Fact
- {s}
- {p}
- {o}

Judgment:
```

Prompt A.15: Iterative prompt for the ZSN variant (DBpedia).

```
Your previous response did not meet the guidelines, please try again and
provide a compliant answer.
You have {attempts} remaining attempts.
If you fail to provide a compliant answer within the remaining attempts, your
final response for the fact will be set to "NA" (Not an Answer).

### Reminder of the Guidelines
- Your knowledge is limited to the year 2015.
- The answer should be "T" (true), or "F" (false) based on your internal
knowledge.
- Limit the use of "N" as much as possible.
- Provide your judgment in English and keep it concise.
- Do not spread incorrect information.

### Input format
The fact is given in an unordered list: first the subject, then the predicate
and finally the object.

### Fact
- {s}
- {p}
- {o}

Judgment:
```

Prompt A.16: Iterative Retry prompt for the ZSN variant (DBpedia).

Few-shot prompts

```
You are an expert annotator for the fact verification task.
You will be provided a fact in the form of a subject, predicate, and object.
Your job is to assess the fact based solely on the knowledge you have been
trained on, without using any external information.
Your knowledge is limited to the year 2015.

### Guidelines
The answer should be "T" (true) if the fact is accurate based on your knowledge
up to 2015, or "F" (false) if the fact is inaccurate based on your knowledge up
to 2015.

Please do not spread incorrect information. Your judgment must rely only on
your internal knowledge.
You must always answer in English.
The answer must contain only the judgment ("T", "F"). Lastly, keep your answer
concise.

### Retry Mechanism
If your response does not comply with the guidelines above, you will be asked
to retry.
In the retry follow-up, you must meet the same guidelines and provide a
compliant answer.
You will have a total of {max_attempts} retry attempts.
If you fail to provide a compliant answer within the attempts, your final
response for the fact will be set to "NA" (Not an Answer).

### Examples
...

### Input format
The fact is given in one line: first the subject, then the predicate and
finally the object.

### Fact
{s} {p} {o}

Judgment:
```

Prompt A.17: Iterative prompt for the FNI variant (DBpedia).

```
Your previous response did not meet the guidelines, please try again and
provide a compliant answer.
You have {attempts} remaining attempts.
If you fail to provide a compliant answer within the remaining attempts, your
final response for the fact will be set to "NA" (Not an Answer).

### Reminder of the Guidelines
Your knowledge is limited to the year 2015. The answer should be "T" (true), or
"F" (false) based on your internal knowledge. Limit the use of "N" as much as
possible.
Provide your judgment in English and keep it concise. Do not spread incorrect
information.

### Examples
...

### Input format
The fact is given in one line: first the subject, then the predicate and
finally the object.

### Fact
{s} {p} {o}

Judgment:
```

Prompt A.18: Iterative Retry prompt for the FNI variant (DBpedia).

```
You are an expert annotator for the fact verification task.
You will be provided a fact in the form of a subject, predicate, and object.
Your job is to assess the fact based solely on the knowledge you have been
trained on, without using any external information.
Your knowledge is limited to the year 2015.

### Guidelines
1. The answer should be:
- "T" (true) if the fact is accurate based on your knowledge up to 2015.
- "F" (false) if the fact is inaccurate based on your knowledge up to 2015.

- Please do not spread incorrect information.
- Your judgment must rely only on your internal knowledge.
- You must always answer in English.
- The answer must contain only the judgment ("T", "F").
- Keep your answer concise.

### Retry Mechanism
If your response does not comply with the guidelines above, you will be asked
to retry.
In the retry follow-up, you must meet the same guidelines and provide a
compliant answer.
You will have a total of {max_attempts} retry attempts.
If you fail to provide a compliant answer within the attempts, your final
response for the fact will be set to "NA" (Not an Answer).

### Examples
...

### Input format
The fact is given in an unordered list: first the subject, then the predicate
and finally the object.

### Fact
- {s}
- {p}
- {o}

Judgment:
```

Prompt A.19: Iterative prompt for the FSN variant (DBpedia).

```
Your previous response did not meet the guidelines, please try again and
provide a compliant answer.
You have {attempts} remaining attempts.
If you fail to provide a compliant answer within the remaining attempts, your
final response for the fact will be set to "NA" (Not an Answer).

### Reminder of the Guidelines
- Your knowledge is limited to the year 2015.
- The answer should be "T" (true), or "F" (false) based on your internal
knowledge.
- Limit the use of "N" as much as possible.
- Provide your judgment in English and keep it concise.
- Do not spread incorrect information.

### Examples
...

### Input format
The fact is given in an unordered list: first the subject, then the predicate
and finally the object.

### Fact
- {s}
- {p}
- {o}

Judgment:
```

Prompt A.20: Iterative Retry prompt for the FSN variant (DBpedia).

A.3 Stepwise prompts

This section illustrates the prompts used for the Stepwise pipeline. Note that only the prompt for the "Truthfulness evaluation" step is dataset-dependent, resulting in two variants.

```
You are an expert annotator for the fact verification task.
Your task is to rewrite a fact as a natural language statement, and then, if
you feel confident enough, evaluate the truthfulness of the statement.
The process unfolds in 3 steps: statement production, confidence assessment,
and truthfulness evaluation.
Each step of the process is distinct and will be addressed at different times.

Step: statement production.
Rewrite the following fact as a natural language statement.
Make sure the statement includes each part of the fact, as the statement will
undergo a truthfulness evaluation.
If you aren't able to produce a statement, answer with "IDK" (but only if you
truly can't generate a statement).
You only need to produce the statement.

Fact: {s} {p} {o}

Statement:
```

Prompt A.21: Stepwise prompt for the "Statement production" step.

```
Step: confidence assessment.
Do you feel confident enough to evaluate the following statement?
Reply with "H" if you feel confident enough, otherwise reply with "L".

Statement: {statement}

Confidence:
```

Prompt A.22: Stepwise prompt for the "Confidence assessment" step.

```
Step: truthfulness evaluation.
Evaluate the following statement with a concise judgment.
The judgment must be "T" (true) if the statement is about an accurate fact, or
else "F" (false) if the mentioned fact is inaccurate.
Rely on your internal knowledge only.

Statement: {statement}

Judgment:
```

Prompt A.23: Stepwise prompt for the ″Truthfulness evaluation″ step (regular).

```
Step: truthfulness evaluation.
Your knowledge is limited to the year 2015. Evaluate the following statement
with a concise judgment.
The judgment must be "T" (true) if the statement is about an accurate fact
(based on your knowledge up to 2015), or else "F" (false) if the mentioned fact
is inaccurate (based on your knowledge up to 2015).
Rely on your internal knowledge only.

Statement: {statement}

Judgment:
```

Prompt A.24: Stepwise prompt for the ″Truthfulness evaluation″ step (DBpedia).

References

[1] Alan Agresti and Brent A. Coull. "Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions". In: *The American Statistician* 52.2 (1998), pp. 119–126. DOI: 10.1080/00031305.1998.10480550. eprint: https://doi.org/10.1080/00031305.1998.10480550. URL: https://doi.org/10.1080/00031305.1998.10480550.

[2] Sören Auer et al. "DBpedia: A Nucleus for a Web of Open Data". In: *The Semantic Web*. Ed. by Karl Aberer et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 722–735. ISBN: 978-3-540-76298-0.

[3] Aishwarya Bhandare et al. *Efficient 8-Bit Quantization of Transformer Neural Machine Language Translation Model*. 2019. arXiv: 1906.00532 [cs.LG]. URL: https://arxiv.org/abs/1906.00532.

[4] Kurt Bollacker et al. "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD '08. Vancouver, Canada: Association for Computing Machinery, 2008, pp. 1247–1250. ISBN: 9781605581026. DOI: 10.1145/1376616.1376746. URL: https://doi.org/10.1145/1376616.1376746.

[5] Angela Bonifati et al. *Querying Graphs*. Morgan & Claypool Publishers, 2018. ISBN: 168173432X.

[6] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[7] Andrew Carlson et al. "Toward an architecture for never-ending language learning". In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI'10. Atlanta, Georgia: AAAI Press, 2010, pp. 1306–1313.

[8] G. Casella and R.L. Berger. *Statistical Inference*. Duxbury advanced series in statistics and decision sciences. Thomson Learning, 2002. ISBN: 9780534243128. URL: https://books.google.it/books?id=0x_vAAAAMAAJ.

[9] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. arXiv: 2204.02311 [cs.CL]. URL: https://arxiv.org/abs/2204.02311.

[10] Philipp Cimiano and Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". In: *Semant. Web* 8.3 (Jan. 2017), pp. 489–508. ISSN: 1570-0844. DOI: 10.3233/SW-160218. URL: https://doi.org/10.3233/SW-160218.

[11] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[12] Xin Luna Dong. *Generations of Knowledge Graphs: The Crazy Ideas and the Business Impact*. 2023. arXiv: 2308.14217 [cs.DB]. URL: https://arxiv.org/abs/2308.14217.

[13] Abhimanyu Dubey et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[14] Jeffrey L. Elman. "Finding structure in time". In: *Cognitive Science* 14.2 (1990), pp. 179–211. ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E. URL: https://www.sciencedirect.com/science/article/pii/036402139090002E.

[15] Junyang Gao et al. *Efficient Knowledge Graph Accuracy Evaluation*. 2019. arXiv: 1907.09657 [cs.DB]. URL: https://arxiv.org/abs/1907.09657.

[16] Daniel Gerber et al. "DeFacto—Temporal and multilingual Deep Fact Validation". In: *Journal of Web Semantics* 35 (2015). Machine Learning and Data Mining for the Semantic Web (MLDMSW), pp. 85–101. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2015.08.001. URL: https://www.sciencedirect.com/science/article/pii/S1570826815000645.

[17] Dan Hendrycks et al. "Measuring Massive Multitask Language Understanding". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=d7KBjmI3GmQ.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML]. URL: https://arxiv.org/abs/1503.02531.

[19] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[20] Johannes Hoffart et al. "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia". In: *Artificial Intelligence* 194 (2013). Artificial Intelligence, Wikipedia and Semi-Structured Resources, pp. 28–61. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2012.06.001. URL: https://www.sciencedirect.com/science/article/pii/S0004370212000719.

[21] Jordan Hoffmann et al. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL]. URL: https://arxiv.org/abs/2203.15556.

[22] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825.

[23] G. Kuck. "Tim Berners-Lee's Semantic Web". In: *South African Journal of Information Management* 6.1 (2004). ISSN: 1560-683X. DOI: 10.4102/sajim.v6i1.297. URL: https://sajim.co.za/index.php/sajim/article/view/297.

[24] Nayeon Lee et al. *Language Models as Fact Checkers?* 2020. arXiv: 2006.04102 [cs.CL]. URL: https://arxiv.org/abs/2006.04102.

[25] Stephanie Lin, Jacob Hilton, and Owain Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. 2022. arXiv: 2109.07958 [cs.CL]. URL: https://arxiv.org/abs/2109.07958.

[26] Stefano Marchesin and Gianmaria Silvello. "Efficient and Reliable Estimation of Knowledge Graph Accuracy". In: *Proc. VLDB Endow.* 17.9 (Aug. 2024), 2392â€"2403. ISSN: 2150-8097. DOI: 10.14778/3665844.3665865. URL: https://doi.org/10.14778/3665844.3665865.

[27] Stefano Marchesin, Gianmaria Silvello, and Omar Alonso. "Utility-Oriented Knowledge Graph Accuracy Estimation with Limited Annotations: A Case Study on DBpedia". In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 12.1 (2024), pp. 105–114. DOI: 10.1609/hcomp.v12i1.31605. URL: https://ojs.aaai.org/index.php/HCOMP/article/view/31605.

[28] Stefano Marchesin, Gianmaria Silvello, and Omar Alonso. "Veracity Estimation for Entity-Oriented Search with Knowledge Graphs". In: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. CIKM '24. Boise, ID, USA: Association for Computing Machinery, 2024, pp. 1649–1659. ISBN: 9798400704369. DOI: 10.1145/3627673.3679561. URL: https://doi.org/10.1145/3627673.3679561.

[29] George A. Miller. "WordNet: a lexical database for English". In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: https://doi.org/10.1145/219717.219748.

[30] T. Mitchell et al. "Never-ending learning". In: *Commun. ACM* 61.5 (Apr. 2018), pp. 103–115. ISSN: 0001-0782. DOI: 10.1145/3191513. URL: https://doi.org/10.1145/3191513.

[31] Vishwas Mruthyunjaya et al. *Rethinking Language Models as Symbolic Knowledge Graphs*. 2023. arXiv: 2308.13676 [cs.CL]. URL: https://arxiv.org/abs/2308.13676.

[32] Prakhar Ojha and Partha Talukdar. "KGEval: Accuracy Estimation of Automatically Constructed Knowledge Graphs". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 1741–1750. DOI: 10.18653/v1/D17-1183. URL: https://aclanthology.org/D17-1183.

[33] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[34] OpenAI et al. *GPT-4o System Card*. 2024. arXiv: 2410.21276 [cs.CL]. URL: https://arxiv.org/abs/2410.21276.

[35] Fabio Petroni et al. *Language Models as Knowledge Bases?* 2019. arXiv: 1909.01066 [cs.CL]. URL: https://arxiv.org/abs/1909.01066.

[36] Mohammad Taher Pilehvar and Jose Camacho-Collados. "WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 1267–1273. DOI: 10.18653/v1/N19-1128. URL: https://aclanthology.org/N19-1128.

[37] Yifan Qi et al. "Evaluating Knowledge Graph Accuracy Powered by Optimized Human-machine Collaboration". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD '22. Washington DC, USA: Association for Computing Machinery, 2022, pp. 1368–1378. ISBN: 9781450393850. DOI: 10.1145/3534678.3539233. URL: https://doi.org/10.1145/3534678.3539233.

[38] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

[39] Jack W. Rae et al. *Scaling Language Models: Methods, Analysis & Insights from Training Gopher*. 2022. arXiv: 2112.11446 [cs.CL]. URL: https://arxiv.org/abs/2112.11446.

[40] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG]. URL: https://arxiv.org/abs/1910.10683.

[41] Ruiyang Ren et al. *Investigating the Factual Knowledge Boundary of Large Language Models with Retrieval Augmentation*. 2023. arXiv: 2307.11019 [cs.CL]. URL: https://arxiv.org/abs/2307.11019.

[42] Adam Roberts, Colin Raffel, and Noam Shazeer. "How Much Knowledge Can You Pack Into the Parameters of a Language Model?" In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 5418–5426. DOI: 10.18653/v1/2020.emnlp-main.437. URL: https://aclanthology.org/2020.emnlp-main.437.

[43] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL]. URL: https://arxiv.org/abs/1910.01108.

[44] Amit Singhal. *Introducing the Knowledge Graph: things, not strings*. 2012. URL: https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[45] Aarohi Srivastava et al. *Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models*. 2023. arXiv: 2206.04615 [cs.CL]. URL: https://arxiv.org/abs/2206.04615.

[46] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge". In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Banff, Alberta, Canada: Association for Computing Machinery, 2007, pp. 697–706. ISBN: 9781595936547. DOI: 10.1145/1242572.1242667. URL: https://doi.org/10.1145/1242572.1242667.

[47] Kai Sun et al. *Head-to-Tail: How Knowledgeable are Large Language Models (LLMs)? A.K.A. Will LLMs Replace Knowledge Graphs?* 2024. arXiv: 2308.10168 [cs.CL]. URL: https://arxiv.org/abs/2308.10168.

[48] Gemma Team et al. *Gemma 2: Improving Open Language Models at a Practical Size*. 2024. arXiv: 2408.00118 [cs.CL]. URL: https://arxiv.org/abs/2408.00118.

[49] Qwen Team. *Qwen2.5: A Party of Foundation Models*. 2024. URL: https://qwenlm.github.io/blog/qwen2.5/.

[50] Romal Thoppilan et al. *LaMDA: Language Models for Dialog Applications*. 2022. arXiv: 2201.08239 [cs.CL]. URL: https://arxiv.org/abs/2201.08239.

[51] James Thorne et al. *FEVER: a large-scale dataset for Fact Extraction and VERification*. 2018. arXiv: 1803.05355 [cs.CL]. URL: https://arxiv.org/abs/1803.05355.

[52] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[53] Denny Vrandečić and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase". In: *Commun. ACM* 57.10 (Sept. 2014), pp. 78–85. ISSN: 0001-0782. DOI: 10.1145/2629489. URL: https://doi.org/10.1145/2629489.

[54] Chenguang Wang, Xiao Liu, and Dawn Song. *Language Models are Open Knowledge Graphs*. 2020. arXiv: 2010.11967 [cs.CL]. URL: https://arxiv.org/abs/2010.11967.

[55] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL]. URL: https://arxiv.org/abs/2201.11903.

[56] Edwin B. Wilson. "Probable Inference, the Law of Succession, and Statistical Inference". In: *Journal of the American Statistical Association* 22.158 (1927), pp. 209–212. DOI: 10.1080/01621459.1927.10502953. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1927.10502953. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1927.10502953.

[57] An Yang et al. "Qwen2 Technical Report". In: *arXiv preprint arXiv:2407.10671* (2024).

[58] Xiao Yang et al. *CRAG – Comprehensive RAG Benchmark*. 2024. arXiv: 2406.04744 [cs.CL]. URL: https://arxiv.org/abs/2406.04744.

[59] Shunyu Yao et al. *ReAct: Synergizing Reasoning and Acting in Language Models*. 2023. arXiv: 2210.03629 [cs.CL]. URL: https://arxiv.org/abs/2210.03629.

[60] Xuan Zhang and Wei Gao. *Towards LLM-based Fact Verification on News Claims with a Hierarchical Step-by-Step Prompting Method*. 2023. arXiv: 2310.00305 [cs.CL]. URL: https://arxiv.org/abs/2310.00305.

[61] Jeffrey Zhou et al. *Instruction-Following Evaluation for Large Language Models*. 2023. arXiv: 2311.07911 [cs.CL]. URL: https://arxiv.org/abs/2311.07911.

Acknowledgments

I would like to express my gratitude to all those who supported me throughout this incredible journey, spanning my years at university.

Firstly, I am deeply thankful to my supervisors, Prof. Stefano Marchesin and Prof. Gianmaria Silvello, for their invaluable guidance, availability, and mentorship, helping me in shaping this research project.

I would like to thank my family next, for this opportunity they gave me and the sacrifices they made, always standing by me with unwavering support and encouragement.

To my friends at university, thank you for being part of this shared journey and making these years special, for supporting me through the study sessions and the group projects, and for celebrating the milestones we achieved together. To my friends outside university, thank you for supporting and motivating me, and reminding me to cherish the moments beyond my studies.

Finally, I want to acknowledge myself. Without the effort and perseverance I've put during these years, none of this would have been possible.