CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Analysis and Generation of Wikidata Descriptions

Master's thesis in Computer science and engineering

Huatai Xu, Yuxiang Cao

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Master's thesis 2025

Analysis and Generation of Wikidata Descriptions

Huatai Xu, Yuxiang Cao

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Analysis and Generation of Wikidata Descriptions
Huatai Xu, Yuxiang Cao

Supervisor: Aarne Ranta, Department of Computer Science and Engineering
Examiner: Peter Ljunglöf, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Analysis and Generation of Wikidata Descriptions
Huatai Xu
Yuxiang Cao
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis explores the structure and generation of descriptive texts for Wikidata entities, focusing on cities, universities, and mathematicians. The goal is to develop a grammar-based, language-independent system for automatic description generation. We begin by analyzing multilingual description patterns across six languages, revealing high structural consistency within languages and substantial cross-language variation, particularly between European and non-European language groups. A detailed property frequency analysis shows that a small number of attributes account for the majority of descriptions. Further label occurrence analysis indicates that while human-readable administrative attributes are well represented, identifiers and spatial data are rarely included in natural language descriptions. To mitigate missing label issues, we design a data augmentation pipeline using GeoNames and OpenStreetMap, significantly improving label coverage across languages. We also compare the grammar-based approach with a Retrieval-Augmented Generation (RAG) system and find that the former performs significantly better in terms of clarity, structural consistency, and multilingual alignment. Our findings inform the design of a multilingual description generation system based on Grammatical Framework (GF), emphasizing clarity, informativeness, and structural consistency. This project is part of a broader collaboration: Bokun Xiao contributed to the development of the core grammar, Imtiaz Ayon focused on building the Bengali grammar, and another team was responsible for the Greek grammar.

Keywords: Wikidata, grammatical framework, multilingual description generation.

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Professor Aarne Ranta, for his continuous guidance, support, and insightful feedback throughout the course of this thesis. His expertise in multilingual grammar and computational linguistics has been instrumental to the success of this project. We are also deeply thankful to Denny Vrandečić, one of the founders of Wikidata, for initiating and supporting this research direction, and for providing us with the opportunity to work on a meaningful and challenging task with real-world impact. Our appreciation goes to the thesis examiner Peter Ljunglöf for his valuable feedback and rigorous supervision, which greatly improved the quality and clarity of this work. Special thanks to Bokun Xiao for his contribution to the development of the grammatical framework modules used in this thesis, and Imtiaz Ayon for his work on Bengali language support, which enriched our multilingual experiments. Finally, we would like to thank everyone involved in this project for their collaboration and support.

Huatai Xu, Yuxiang Cao, Gothenburg, 2025-06-29

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Wikidata is an open knowledge base with a structured format that serves as a primary data source for various applications, including artificial intelligence and Wikipedia. Each item in Wikidata has a unique ID (QID), along with labels and descriptions available in multiple languages, and contains multiple facts related to that entry in the form of attributes that also contain a unique identifier (PID). However, these descriptions intended to provide brief explanations are manually created and often lack uniformity across different languages. This inconsistency contradicts Wikidata's fundamental goal of delivering stable and homogeneous multilingual knowledge.

One of the key challenges in maintaining Wikidata's integrity lies in the variation of language descriptions, which often exhibit significant inconsistencies across languages.

For example, the item **Göteborg** (Q25287) has the following descriptions:

English: Sweden's second-largest city and the capital of West Götaland County

Swedish: the city centre of Gothenburg, Sweden

Chinese: a city in Sweden

These discrepancies arise because descriptions are manually entered in each language independently, sometimes resulting in missing descriptions in specific languages. Given the massive scale of Wikidata containing over 116 million entries across more than 300 languages. There is a pressing need for an automated, scalable, and semantically consistent solution. The work presented in this thesis builds on collaborative work with Bokun Xiao. In our collaboration, the grammar-based generation component was primarily developed by him, while the data collection, analysis pipeline, and evaluation framework are original contributions of this thesis. Based on the results of data analysis, Bokun continues to refine the grammar-based generation method. At the same time, this thesis explores a data-driven generation method. The outcomes of both methods are then compared and evaluated.

The Wikidata Query Service (WDQS [1]) provides a flexible system for querying and retrieving structured data. Building on this capability, this paper aims to develop a

method for automatically generating multilingual descriptions based on structured facts in the entries. The method uses the Grammatical Framework (GF [2]) to build a unified syntax specification to generate consistent and informative descriptions across languages. By extracting key attributes of an entry, such as category, geographic location, entity affiliation, etc., and combining them with GF's multilingual generation capabilities, the methodology enables the generation of automated, multilingual descriptions that are semantically accurate, thus improving the consistency and usability of Wikidata. In addition, we will explore the use of large language models (LLMs [3]) to generate similar brief descriptions and compare their output with that produced by GF. This comparison aims to identify the strengths and limitations of the two methods.

1.2 Research Question and Objectives

This thesis aims to address the following central research question: How can an automated system be developed to generate semantically consistent and linguistically uniform descriptions for Wikidata items across multiple languages? To explore this, we investigate the following sub-questions:

- What are the primary sources of inconsistencies in multilingual Wikidata descriptions?
- How can structured data extraction methods help standardize descriptions?
- What are the relative advantages and drawbacks of rule-based and data-driven generation techniques in terms of cross-linguistic consistency?
- How can an automated system be designed to scale effectively across different categories and languages?

The primary objectives of this research are:

- To analyze linguistic and structural inconsistencies in existing Wikidata descriptions.
- To develop a structured data extraction pipeline using WDQS.
- To implement and compare two generation approaches: a rule-based method and a data-driven method.
- To evaluate the accuracy and consistency of the generated descriptions.

By achieving these objectives, this thesis aims to strengthen Wikidata's role as a reliable multilingual knowledge base [4]. Automating description generation will improve knowledge representation, reduce redundancy, and enhance accessibility for users from diverse linguistic backgrounds. Furthermore, the methodology developed in this thesis can be extended to other structured knowledge bases that require multilingual consistency.

1.3 Methodological Overview

This thesis follows a structured methodology integrating data collection, analysis, and two different methods to generate multilingual descriptions. The key steps include:

- Data Collection: Retrieve structured data from Wikidata using WDQS, focusing on selected categories such as city, university and mathematician in several languages.
- Data Analysis: Analyse the variation in information content and structure of existing Wikidata descriptions.
- Description Generation: Implement two generation method:
  - A rule-based method using GF to generate descriptions.
  - A data-driven method using LLMs to generate descriptions.
- Evaluation: Assessing the quality and consistency of the generated descriptions.

In general, we present a scalable and automated approach to improve the consistency and completeness of Wikidata descriptions across multiple languages. By combining structured data analysis with a multilingual grammar framework, we will contribute to the broader knowledge representation and information retrieval field in multilingual environments. The findings and methods presented in this thesis might also be applied to other structured knowledge bases to promote better multilingual data integration and standardisation in various domains.

2

Theory

This chapter introduces the technical foundations and tools that support our approach to multilingual description generation from Wikidata. It begins with an overview of the Wikidata Query Service for structured data retrieval, the GF is then introduced, which serves as the core engine for grammar-based multilingual generation, including its abstract and concrete syntactic layers. To support linguistic analysis, we incorporate UDPipe and the CoNLL-U format for dependency parsing. Cosine similarity is applied for measuring lexical alignment between descriptions in different languages. Additionally, relevant key libraries such as SPARQLWrapper used in our data collection process are briefly described. Finally, we introduce the Retrieval-Augmented Generation (RAG) framework, which integrates external knowledge retrieval into the text generation process to enhance factual accuracy and contextual richness in description generation. A brief review of related work in multilingual text generation is also included at the end of the chapter. These components together form the theoretical and technical basis for the methodology proposed in later chapters.

2.1 Wikidata Query Service

Operational since mid-2015, the Wikidata Query Service (WDQS) provides access to Wikidata through SPARQL, a standardized query language for Resource Description Framework (RDF) data, and is powered by the BlazeGraph RDF [5] store and graph database. It adopts the W3C-standard RDF data model, with each fact stored as a <subject–predicate–object> triple. As the largest open knowledge graph query service currently available, WDQS offers full support for the W3C SPARQL 1.1 [6] standard, and additionally provides unique label services and entity resolution capabilities through its Wikibase extensions. This backend was selected for its comprehensive documentation, reliable support, high availability, and adherence to standard query languages. The service delivers live data with minimal latency, ensuring real-time updates. By combining predicates from different knowledge bases and ontologies [7], users can construct knowledge retrieval querys across datasets.

As shown in Figure 2.1, this SPARQL query retrieves city-related data from Wikidata by selecting all entities classified as cities along with their English labels and corresponding Geonames identifiers. The query structure begins by identifying city entities through the property path `wdt:P31/wdt:P279*`, which captures both direct

```
SELECT ?city ?cityLabel ?geonamesID WHERE {
  ?city wdt:P31/wdt:P279* wd:Q515.
  OPTIONAL { ?city wdt:P1566 ?geonamesID. }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

Figure 2.1: SPARQL query to retrieve city-related data

instances of cities (using `wdt:P31` for "instance of") and any subclasses of cities (via the recursive `wdt:P279*` for "subclass of"), with `wd:Q515` specifically denotes the Wikidata item for "city" as the root category. This ensures comprehensive coverage of all urban settlements in Wikidata.

The query then optionally includes Geonames IDs (through property `wdt:P1566`) where available, acknowledging that not all city entries in Wikidata may have this external database linkage. Finally, the `SERVICE wikibase: label` clause generates human-readable English names for the cities by leveraging Wikidata's multilingual labeling system, with the language parameter specifically set to English.

2.2 Grammatical Framework

The Grammatical Framework (GF) offers an approach for generating semantically consistent multilingual descriptions. GF is a specialized programming language and framework designed for multilingual grammar applications and natural language processing. The core idea of GF is to describe the grammar of different languages with unified rules. It first abstracts the meaning of the sentence into a language-independent logical structure, and then generates grammatically specific expressions based on the characteristics of each language (such as word form changes, word order, gender, etc.). They correspond to abstract syntax and concrete syntax respectively.

2.2.1 Abstract Syntax

Defining what meanings can be expressed. Figure 2.2 is a simple greeting syntax, the code consists of the following components:

startcat: The program starts generating sentences from the Greeting category.

cat: Declare semantic categories (similar to the classification of "noun" and "verb", but more abstract).

fun: Define how to combine meanings (such as Hello + Mum to form a greeting).

```
abstract Hello = {

      flags startcat = Greeting ;

      cat Greeting ; Recipient ;

      fun
        Hello : Recipient -> Greeting ;
        World, Mum, Friends : Recipient ;
}
```

Figure 2.2: Abstract syntax for greeting

2.2.2 Concrete Syntax

The specific grammar of English is as Figure 2.3 shows:

```
concrete HelloEng of Hello = {

  lincat Greeting, Recipient = {s : Str} ;

  lin
    Hello recip = {s = "hello" ++ recip.s} ;
    World = {s = "world"} ;
    Mum = {s = "mum"} ;
    Friends = {s = "friends"} ;
}
```

Figure 2.3: Concrete syntax for English greetings

The major parts of this code are:

Module declaration header: The concrete keyword is used to indicate that this is a concrete syntax implementation. The naming convention is HelloEng, indicating that this is an English implementation of the abstract syntax Hello.

lincat: Specifies that both Greeting and Recipient categories are implemented as record types containing a string field s.

lin: Specify a specific string combination method for each semantic structure defined in the abstract syntax.

To avoid manually implementing the specific grammar of each language, GF provides the Resource Grammar Library [8] (RGL), a library containing rich language rules and encapsulating constructors of common structures such as noun phrases (`CN`) and verb phrases (`VP`). Listing A.1 shows a concrete syntax module for English city descriptions, which builds on several submodules and uses constructors from

the RGL such as `mkCN`, `mkN`, and `mkAdv` to build compositional noun phrases and adverbials. The rules combine different grammatical elements—such as location information and city types (e.g., capital) into complete descriptions. Taking the code in Listing A.1 as an example:

```
CityDescription kind location = mkCN kind location ;
```

`mkCN` is a function provided by RGL that constructs a common noun phrase (`CN`), which combines a phrase kind that indicates a city type (such as "capital") with a modifier location that indicates geographic location information (such as "in Sweden"). Since `mkCN` implements the corresponding concrete grammatical rules for each language in RGL, the same abstract structure can be automatically adapted in different languages. For example, in English, "capital city in Sweden" is generated, while in French it is "ville capitale en Suède". This mechanism allows us to obtain grammatically natural and correct multilingual descriptions by defining the structure only once at the abstract level, greatly simplifying the implementation complexity of cross-language text generation.

GF enables the generation of grammatically accurate and semantically aligned text across languages by mapping a shared abstract syntax to concrete syntaxes in multiple languages. This project aims to investigate whether language-independent descriptions can be generated using GF from Wikidata facts.

2.3 UDPipe

UDPipe [9] is a specific natural language processing tool used to automatically convert raw natural language text into structured data rich in linguistic information, providing reliable basic analysis results for downstream applications such as machine translation. The complete process of UDPipe processing text can be summarized into three stages: word segmentation splits continuous text into individual words and symbols, part-of-speech tagging assigns a tag to each token based on the Universal Dependency Standard (UD), and dependency syntactic analysis to identify the interdependence between words in a sentence. The UDPipe tool is used to analyze the similarity of sentence structures in different languages. This analysis helps assess the feasibility of reusing common grammatical patterns in multilingual generation for grammar groups.

2.3.1 CoNLL-U Format

UDPipe outputs result in **CoNLL-U**, a standardized tabular format where each row represents a token and its annotations. The simplified example below is the English sentence: **The black cat sees us now.**

ID	FORM	LEMMA	UPOS	HEAD	DEPREL
1	the	the	DET	3	det
2	black	black	ADJ	3	amod
3	cat	cat	NOUN	4	nsubj
4	sees	see	VERB	0	root
5	us	we	PRON	4	obj
6	now	now	ADV	4	advmod

Key columns include:

ID: Token position.

FORM: Surface form (e.g., ”sees”).

LEMMA: Base form (e.g., ”see”).

UPOS: Universal POS tag (e.g., VERB).

HEAD: ID of the governing word (e.g., 0 for ”sees” as root).

DEPREL: Dependency relation (e.g., nsubj).

The *base form* is the standard dictionary form of a word, which groups together different inflected variants. For example, ”eats”, ”ate”, and ”eating” all share the lemma ”eat”. The *POS tag* (part-of-speech tag) assigns a grammatical category to each word, such as NOUN, VERB, or ADJ. In dependency grammar, each word (except the root) depends on another word, forming a tree structure. The *HEAD* of a word is the word it syntactically depends on (its governor). The syntactic relation between a word and its head is specified by a *dependency relation label* (`DEPREL`), such as `nsubj` (nominal subject), `obj` (object), or `amod` (adjectival modifier). These annotations together define the grammatical structure of the sentence.

2.4 Cosine Similarity

Cosine similarity [10] is a commonly used method for measuring the similarity between two non-zero vectors. It is widely applied in fields such as text mining and recommendation systems. This algorithm evaluates the similarity by calculating the cosine of the angle between the two vectors. In theory, the closer the directions of the vectors are, the higher their cosine similarity. A cosine value of 1 indicates that the two vectors are in exactly the same direction, 0 means they are orthogonal, and -1 signifies they are in completely opposite directions. It is a normalized measure that focuses on the angle between vectors rather than their magnitude, allowing it to effectively assess directional similarity even when the vectors differ in length. It is widely used across various fields, especially in information retrieval, text mining, and recommendation systems. Additionally, its relatively simple computation compared to other similarity metrics makes it ideal for large-scale data processing. In

our system, cosine similarity is used to compare the frequency distributions of dependency relation labels (DEPREL) across different languages, the quantitative output of cosine similarity provides guidance on the extent to which previously designed grammars can be reused when developing grammars for new languages.

2.4.1 Formula

The cosine similarity between two vectors $\mathbf{A}$ and $\mathbf{B}$ is calculated as:

$$\text{cosine similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.1}$$

- $\mathbf{A} \cdot \mathbf{B}$ represents the **dot product** of vectors $\mathbf{A}$ and $\mathbf{B}$.
- $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ denote the **Euclidean norms** (magnitudes) of the vectors.
- θ is the angle between the two vectors in the vector space.
- The output range is $[0, 1]$, where:
  - 1 indicates identical direction (maximum similarity).
  - 0 means orthogonal (no correlation).
  - In this project, cosine similarity values are always greater than 0, because the vectors are based on frequency counts, which are nonnegative. Therefore, the angle between vectors is never greater than 90 degrees, and the similarity remains within $[0, 1]$.

2.5 SPARQLWrapper

SPARQLWrapper is a Python library designed to simplify interactions with SPARQL endpoints. It is particularly valuable for retrieving structured data from Resource Description Framework [11] (RDF) knowledge bases, including prominent open-source projects like Wikidata. This library handles the underlying Hypertext Transfer Protocol (HTTP) communications transparently while supporting multiple standardized response formats:

- JavaScript Object Notation (JSON)
- Extensible Markup Language (XML)

Among these formats, JSON is especially convenient for integration with libraries like pandas or json, making it easier to incorporate query results into data processing or analysis workflows.

2.6 Retrieval-Augmented Generation

Retrieval Augmented Generation (RAG) is a combined framework that merges information retrieval with text generation. Unlike conventional models that depend solely on their internal parameters to create text, RAG incorporates an external knowledge base into the generation process, which improves the accuracy and breadth of the content produced by the model. The system retrieves relevant documents from the knowledge base—specifically from Wikidata in this thesis—to gather pertinent information during the text generation process.

The RAG has two primary components:

- **Retriever**: When an input query is received, it is initially transformed into a vector representation, followed by a similarity computation with a pre-established knowledge anticipation base (also represented as vectors) to identify the top k most applicable text segments. For this thesis, JSON files will serve as the corpus knowledge base.
- **Generator**: The generator can be LLMs. It takes the original input, along with the retrieved text, and combines both to produce the desired text as context.

Overall, RAG extracts contextual information based on the input and generates coherent natural language text.

2.7 Previous work

Natural Language Generation (NLG) has a long history, with early systems appearing as early as the 1960s. One of the first well-known examples is ELIZA, a computer program developed by Joseph Weizenbaum in 1966. ELIZA was designed to simulate conversation by matching keywords in the user's input and replying with fixed templates. Although the program was quite simple, it showed that basic techniques could be used to create the appearance of meaningful conversation.[12].

Years later, Ehud Reiter and Robert Dale made a substantial impact on the field with their publication, Building Natural Language Generation Systems. In this work, they presented a structured framework for building NLG systems, outlining essential tasks such as document planning, microplanning, and surface realisation. They also emphasised the important role of requirements analysis and system architecture in creating practical NLG applications [13].

Some of the early ideas that influenced NLG came from theoretical linguistics. In particular, Chomsky's Syntactic Structures introduced the concept that language could be described using formal rules, such as phrase structure rules and transformations. These ideas helped shape early rule-based systems for generating sentences, where the focus was on building grammatically correct outputs [14].

Neural networks have led to significant improvements in the field. OpenAI's GPT-2

shows excellent performance in different languages [15]. Another important development is Facebook AI's BART model. It combines BERT's bidirectional encoder with GPT's autoregressive decoder, which improves text generation [16].

Datasets with a structured format, such as Wikidata multilingual knowledge base with multiple layers have become essential assets for NLG. The WebNLG Challenge, which evaluates systems that convert RDF triples into coherent text, continues to foster research in this area [17]. Additionally, the Topic-Guided Wikipedia Abstract Generator (TWAG) enhances the quality of generated Wikipedia articles by utilising cues that are specific to the topic [18].

In contrast to neural models, linguistically driven frameworks like GF offer greater precision and robust multilingual control; Angelov et al. introduce a system that integrates GF with resources like WordNet and Wikidata to produce encyclopaedic content across multiple languages. This method emphasizes grammatical fidelity and consistent multilingual rendering, making it especially useful for languages with limited computational resources [19].

Moreover, the prospect of a multilingual, semantically coherent encyclopaedia has inspired new work at the intersection of NLG and structured knowledge bases; Vrandečić, for example, describes the architectural and philosophical principles of an Abstract Wikipedia, whose goal is to decouple content and language by encoding entries as language-independent abstractions [20]. This method allows the creation of coherent entries across languages by using structured databases like Wikidata.

Inspired by this, Ranta investigated how multilingual text generation can be achieved in an abstract Wikipedia using GF, which can render content accurately and faithfully into a variety of natural languages, which is particularly important for resource-poor language support [21]; he also emphasised the importance of grammatical coverage and contributor availability and suggested practical ways of extending the Grammar Library with existing knowledge bases.

In contrast to grammar-based methods, Gutman, Ivanov, and Ramírez propose a hybrid system that integrates dependency grammars with template-based generation; their system uses dependency relations to constrain and guide the construction of natural language outputs from structured inputs, improving grammaticality while retaining some of the flexibility of neural model [22]s. This middle ground balances control and fluency and suggests promising directions for multilingual and domain-specific NLG.

3

Methods

This chapter outlines the methodology adopted to analyze multilingual entity descriptions in Wikidata, with a focus on identifying patterns and developing a scalable generation workflow. We begin by providing an overview of the end-to-end workflow, followed by detailed procedures for data collection and preparation. This includes resolving missing labels across languages and merging relevant property values. Next, we conduct a multi-level analysis of property usage, including syntactic structure comparison, frequency-based analysis, and keyword-level matching. Lastly, we also experiment with using LLMs to generate descriptions based on structured data. This helps evaluate the strengths and limitations of different methods. The methods described in this chapter form the empirical basis for the generation and evaluation experiments presented in the subsequent chapter.

3.1 Overview of the workflow

The system for automatically generating multilingual descriptions for several categories from Wikidata follows a structured pipeline designed to support extensibility and language independence. The workflow consists of four major stages: data collection, property analysis, multilingual grammar development, and description generation. The overall workflow of our system is illustrated in Figure 3.1. Each component is designed to handle the challenges associated with Wikidata's massive

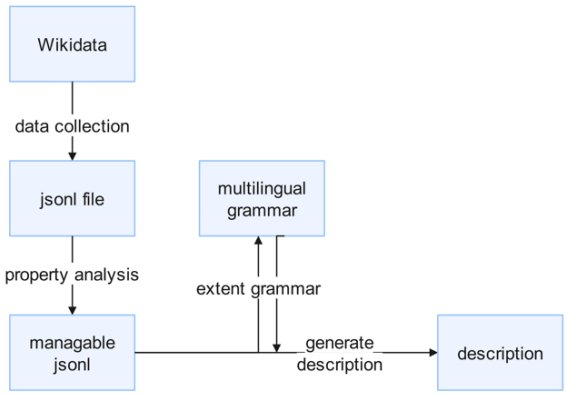

Figure 3.1: Workflow of the Description Generation Pipeline

data volume and multilingual structure while ensuring that the output descriptions are both semantically accurate and linguistically natural across different languages. The process is divided into several key stages:

- Data Collection from Wikidata: The first step involves retrieving relevant information from Wikidata using SPARQL queries. This structured query language allows us to target specific classes of items (in this case, cities) and request associated property-value pairs. We extract a broad range of multilingual facts for each item, including its labels and descriptions in multiple languages (e.g., English, Swedish, Chinese), as well as domain-specific facts such as population, country, coordinates, and administrative divisions.

  To make the process scalable and repeatable, we implemented a Python-based data collection pipeline that interacts directly with the Wikidata Query Service (WDQS). The tool handles large result sets by paginating queries with parameters like LIMIT and OFFSET, allowing us to efficiently extract thousands of entries while avoiding query timeouts. The results are stored in JSON Lines (.jsonl) format, where each line represents a single entity and its associated data. This structure makes the dataset easy to parse, transform, and process in subsequent steps.

- Property Analysis: The data collected are analyzed to identify the properties that are most frequently used and linguistically useful to generate descriptions. This step filters out less relevant information and ensures that the final dataset is manageable and well structured. The result is a curated and normalized.jsonl file suitable for generation. The analysis process begins by extracting all property identifiers (PIDs) associated with corresponding entities in Wikidata. Each property, such as `P17` for "country" or `P1082` for "population", represents a fact that may potentially be expressed in a natural language description.

  To prioritize the most relevant properties, we perform a frequency-based ranking: the top 20 most common PIDs across the dataset are selected for further analysis. These high-frequency properties form the initial candidate set for description generation.

  We then match each property against its textual occurrence. Specifically, for each property-value pair, we check whether the label of the property (e.g., "country", "capital of", "located in") appears explicitly in the description. In the next stage, we analyze the corpus of descriptions to extract high-frequency content words, especially nouns and adjectives, which typically carry the semantic weight of descriptions.

- Multilingual Grammar Development: To generate fluent and structurally consistent multilingual descriptions, we rely on the Grammatical Framework (GF), a grammar formalism and programming language for multilingual natural language generation. GF separates abstract syntax from concrete syntaxes, which is particularly well-suited to our use case. The abstract syntax defines the semantic structure of domain-specific statements (e.g., function `LocatedIn :`

`City` → `Country` → `Fact`), while the concrete syntaxes specify how these structures are realized in different natural languages.

Based on the results of the property analysis, we identify the most commonly used and linguistically meaningful facts in descriptions. These high-frequency properties are then used to determine the core sentence patterns needed to describe an entity. For example, the property `P17` ("country") appears prominently across both the structured data and the natural language descriptions. This motivates the creation of a core sentence structure such as "a city located in a country". The abstract function **LocatedIn : City -> Country -> Fact** models the idea that a city is located in a country, independently of language. This design ensures that once an abstract syntax tree is built from structured data (e.g., JSON), we can systematically generate consistent and fluent descriptions across multiple languages. Listing A.1 presents the city grammar developed by the grammar team.

- Description Generation: The curated JSONL file serves two purposes in the system. First, it identifies key properties that need to be presented, which guides the grammar development team in implementing appropriate abstract and concrete syntax rules in GF. Second, it provides concrete data that are used to generate natural-language sentences. Based on the multilingual grammar, natural language descriptions are automatically generated for each item in the dataset. Listing A.2 shows an example of descriptions for various categories generated by the program.

This pipeline is designed to be modular and easy to extend, which makes it possible to generate descriptions in different languages and at a larger scale. It also helps keep the descriptions on Wikidata more consistent. The first case study looks at cities, which is the type of entity the pipeline was originally built and tested on. After that, we used the same approach for other types of entities, including universities and mathematicians, to explore how different categories may have different patterns in how properties are used and how descriptions are written. Although this method can be applied to various entity types, the analysis in this work mainly focuses on city data to explain the ideas more clearly.

3.1.1 Assumptions on Relevance and Frequency

In this thesis, we worked with two main assumptions during the data selection and analysis process: First, we assumed that the frequency of property use is related to its relevance in natural language descriptions. That is, the more often a property appears in Wikidata, the more likely it is to carry basic and commonly understood meanings that can be expressed in language. So, we considered these high-frequency properties as more important for generating descriptions. This assumption enabled us to focus on key information within a large dataset. It also fits what we observed in our experiments—for example, properties like "country" (P17) and "capital of" (P1376) often appeared in human-written descriptions.

Second, we assumed that the English label of a property could be used to judge

whether the property appears in a description. In our keyword matching step, we directly matched the English label (such as "country") with the English description of the entity to estimate if and how the property was mentioned. This method does not consider synonyms or changes in wording, but for our purpose—understanding which properties are often used in text—it was effective enough. It also allowed us to handle a large number of attributes with limited computing resources.

We also considered the Wikidata project founder, who emphasised the platform's guidelines that encourage short, single-sentence descriptions. Based on these discussions, we decided to focus on high-frequency attributes that not only appear often but also carry clear meanings and are easy to express in natural language. These properties were used as the main elements for building the final descriptions.

3.2 Data Collection

This section outlines the data collection process using SPARQL queries, as well as the subsequent steps for cleaning and transformation. We utilized the Wikidata Query Service (WQS) endpoint of Wikidata to extract relevant information for city entities (instances of `wd:Q515`). A Python class named `WikiDataQueryResults` was implemented to handle queries and convert results into `pandas DataFrame` objects, a tabular data structure widely used for data analysis in Python, for further manipulation.

We first collected the following information:

- QIDs of cities to serve as unique keys.
- Labels of cities in six languages: English (en), Chinese (zh), Japanese (ja), Greek (el), Bengali (bn), and Swedish (sv).
- Descriptions in the same six languages.

Working directly with the Wikidata online service through their website is efficient for small or medium-sized data queries. However, attempting to read large sets of data, such as city names and descriptions in six languages simultaneously, may fail due to resource limitations or timing out. One common approach is to send a query for each language separately and then manually combine the results. This method is simple but requires a lot of work because it involves doing several runs and processing the results afterwards.

To illustrate a typical query used during data collection, Figure 3.2 shows a SPARQL example that retrieves labels and descriptions for cities in English only. Similar queries were executed separately for each of the six different languages, and the results were later merged during preprocessing.

```
SELECT ?city ?cityLabel ?cityDescription WHERE {
  ?city wdt:P31/wdt:P279* wd:Q515.

  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en".
  }
}
LIMIT 1000 OFFSET 0
```

Figure 3.2: Query to retrieve labels and descriptions in a single language

To enhance scalability and efficiency, this project employs a Python-based programmatic strategy. By interacting directly with the Wikidata endpoint, we can automate queries, handle complex requests, and fetch large data volumes using pagination with 'LIMIT' and 'OFFSET'. For example, a query with `LIMIT 1000 OFFSET 0` retrieves the first 1000 results, while `LIMIT 1000 OFFSET 1000` fetches the next batch. This method simplifies the extraction of multilingual and high-volume data, ensuring efficiency and reproducibility.

3.2.1 Missing label and Filling

Wikidata categorizes entities using the instance of property (`P31`). A basic SPARQL query like the following attempts to retrieve all items that are directly instances of city (`Q515`):

```
SELECT ?item ?itemLabel WHERE {
  ?item wdt:P31 wd:Q515.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

However, The query only returns 9,203 entries and it fails to retrieve cities classified under more specific subclasses, such as "port city" (`Q2264924`), which is defined as a subclass of city. For example, the Wikidata item `Q35765`, corresponding to Osaka, is typed as:

```
wd:Q35765 wdt:P31 wd:Q2264924. #port city
```

Thus, Osaka is not returned by the basic query, even though it is clearly a city. To solve this, the following filter was used: `wdt:P31/wdt:P279* wd:Q515` . This adjustment allowed us to retrieve a much broader set of city entities. In this way we got over 30k cities.

Another problem is that a significant number of cities in Wikidata lacked labels in one or more of the target languages. This posed a serious challenge for our subsequent steps, particularly for multilingual generation, where labels often serve as essential lexical anchors in the generated sentences. Although city descriptions themselves typically do not mention the city's own label, these labels are frequently

referenced in the descriptions of other entities—such as universities and individuals—where cities appear as places of birth or location. Therefore, ensuring label completeness is vital for maintaining fluency and correctness across different types of entities.

To address this issue, we incorporated external trusted sources to supplement the missing labels. Specifically, we retrieved data from GeoNames and OpenStreetMap, both of which provide structured geographical information and multilingual place names. For each city with missing labels, our process involved the following steps:

- Entity Matching: For each city with missing labels, we attempted to match it against two trusted data resources: GeoNames and OpenStreetMap data using one of:
  - Exact match on the English label or normalized name.
  - Use of external IDs if available (e.g., GeoNames ID via `P1566`).
- Multilingual Label Extraction: Once a match was found, we extracted names in the required languages (e.g., ”name:sv”, ”name:el” from OSM), if available.

Once retrieved, these external labels were cleaned, normalized, and merged back into the main dataset. Care was taken to ensure consistency with existing data. As a result, the label coverage across all six target languages was improved, enhancing the reliability of the dataset and the quality of the final descriptions.

3.2.2 Property collection and merging

We further extracted some more information from city entities to support property analyses. Specifically, we focused on gathering the properties associated with each city in Wikidata. This process was also completed through SPARQL queries, and we illustrate it here using city entities as examples.

The first step in our process was to collect the full set of properties associated with each city, to support property-level analysis and eventual description generation (Figure 3.3). After identifying the 20 most frequently occurring properties among city entities, we conducted a deeper analysis to assess which of these properties are suitable for inclusion in generated descriptions. Based on this analysis, we eventually select the following properties to be used in our generation system:

- `P17` – country
- `P1082` – population
- `P1376` – capital of

```
SELECT DISTINCT ?city ?prop WHERE {
  ?city wdt:P31/wdt:P279* wd:Q515.
  ?city ?prop ?value.
  FILTER(STRSTARTS(STR(?prop), "http://www.wikidata.org/prop"))
}
```

Figure 3.3: Query all cities and all the properties they have

And we used the query shown in Figure 3.4:

```
SELECT ?city ?cityLabel ?country ?countryLabel ?population
?capitalOf ?capitalOfLabel WHERE {
  ?city wdt:P31/wdt:P279* wd:Q515.
  OPTIONAL { ?city wdt:P17 ?country. }
  OPTIONAL { ?city wdt:P1082 ?population. }
  OPTIONAL { ?city wdt:P1376 ?capitalOf. }

  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en".
  }
}
LIMIT 1000 OFFSET 0
```

Figure 3.4: Query to retrieve selected properties

The query includes optional clauses for each property to account for missing values. The service clause ensures that English labels are retrieved alongside entity identifiers, improving interpretability. The results from this query will later be merged into the original dataset and then delivered to grammar groups. In this stage, we focused on extracting the raw values associated with each selected property—such as country identifiers (QIDs), numeric population values, or referenced entities—rather than natural language strings. These structured values serve as input for further mapping in the description generation pipeline.

Finally, we merged these property values with the previously collected multilingual data. This produced a standardised dataset in JSON Lines (`.jsonl`) format, where each entry contains the following fields:

- QIDs of cities.
- Labels of cities in six languages.
- Descriptions in the same six languages.
- Values of selected properties (country, population, and capital-of).

To illustrate the final structure of the merged dataset, the following JSONL entry shows the information collected for the city `Stockholm` (`Q1754`):

```
{
  "url": "http://www.wikidata.org/entity/Q1754",
  "labels": {
    "en": "Stockholm",
    "el": "Στοκχόλμη",
    "jp": "ストックホルム",
    "sv": "Stockholm",
    "zh": "斯德哥尔摩",
    "bn": "স্টকহোম"
  },
  "descriptions": {
    "en": "capital and largest city of Sweden",
    "el": "πρωτεύουσα και μεγαλύτερη πόλη της Σουηδίας",
    "jp": "スウェーデンの首都",
    "sv": "Sveriges huvudstad",
    "zh": "瑞典首都",
    "bn": "সুইডেনের রাজধানী"
  },
  "prop": {
    "country": "http://www.wikidata.org/entity/Q34",
    "population": "984748",
    "capital of": "http://www.wikidata.org/entity/Q34"
  },
  "domain": "https://www.wikidata.org/wiki/Q515"
}
```


The data processing was implemented in Python using the pandas library. The function `process_city` standardizes and merges multilingual labels and descriptions for each city, while also collecting structured Wikidata properties such as country, population, and capital status. Missing values are tracked to evaluate data completeness across languages and properties. The core pseudocode is shown in Algorithm 1:

Algorithm 1 Process City Item

```
procedure ProcessCity(item, missing_counts)
    Initialize empty dictionaries: labels, descriptions, properties
    for each language in {en, el, jp, sv, zh, bn} do
        Retrieve label and description for the given language from the item
        if label is missing or invalid then
            Set label to null
            Increment missing_counts["missing_labels"][language]
        end if
        if description is missing or invalid then
            Set description to null
            Increment missing_counts["missing_descriptions"][language]
        end if
        Store label in labels[language]
        Store description in descriptions[language]
    end for
    Extract properties:
      properties["country"] ← value of property P17 from item, or null if missing
      properties["population"] ← value of property P1082 from item, or null if missing
      properties["capital_of"] ← value of property P1376 from item, or null if missing
    return a dictionary containing url, labels, descriptions, and properties
end procedure
```

This step completed the preparation of a multilingual and property-rich dataset, which served as input for both statistical analysis and multilingual description generation in the following stages.

3.3 Property Analysis

In this system, property analysis serves two purposes. First, it helps determine which properties are most relevant and commonly used, thereby guiding the creation of templates for multilingual description generation. Second, it reveals semantic biases and limitations in existing Wikidata descriptions, providing a basis for designing more informative and balanced generation strategies. To support description generation for Wikidata entities, we conducted a structured analysis of properties associated with city items. Our methodology consisted of three main steps: dependency and part-of-speech (POS) structure comparison, property identifier (PID) frequency analysis, and label occurrence verification. These steps aimed to understand the linguistic and semantic composition of Wikidata descriptions and identify which elements contribute most consistently to their construction.

3.3.1 Dependency and POS-based Structural Comparison

We used UDPipe to analyze syntactic and morphological patterns of city and university descriptions in five languages: English, Chinese, Japanese, Swedish, and Greek. Descriptions were processed using language-specific models to extract tokenization, POS tags (UPOS), and universal dependency relations (DEPREL), yielding outputs in CoNLL-U format. This allowed us to capture abstract syntactic structures independent of vocabulary. For each language, we aggregated DEPREL and UPOS frequencies for both city and university domains. We then computed pairwise cosine similarity scores across languages and domains to quantify syntactic alignment.

3.3.2 Property Frequency Extraction

To identify which properties are most commonly associated with city entities in Wikidata, we executed a SPARQL query (Figure 3.3) that retrieved all properties used in city entries. This yielded a comprehensive list of property identifiers (PIDs), which were then resolved to their English labels using the Wikidata API. The output was a frequency table ranking properties by their occurrence counts across city entities. This list was later used to determine the core attributes for description generation.

In detail, we divide the process of handling properties into two stages. In the first stage, we extracted all property identifiers (PIDs) from the collected city dataset and counted the frequency with which each one appeared across different entities. This gave us a ranked list of properties based on frequency. We used this overall frequency analysis to understand which properties are more common and might be useful for generating descriptions.

In the second stage, we manually chose a few properties that appeared often and were also suitable for language generation. These include `P17` (country), `P1082` (population), and `P1376` (capital of). We then used the values of these selected properties in the final dataset. These properties are also used in Algorithm 1 and the generation experiments to serve as examples. Although only a small number of properties are used in later steps, they were chosen based on a broader analysis of the full set.

To make these properties easier to read, we used batch queries to the Wikidata API to get their English labels. We tried to follow the API rate limits during this step. The main steps of this process are shown in Algorithm 2.

Algorithm 2 Retrieve English Labels for Property IDs from Wikidata

```
procedure GET_PID_LABELS(pids)
    Set batch_size ← 50
    Initialize empty dictionary pid_labels
    for each batch of pids (size = batch_size) do
        Construct API request with:
          url ← "https://www.wikidata.org/w/api.php"
          params ← {
            action: "wbgetentities",
            format: "json",
            ids: join(batch, "|"),
            props: "labels",
            languages: "en"
        }
        response ← HTTP_GET(url, params)
        if response.status_code = 200 then
            data ← PARSE_JSON(response)
            for each pid in data["entities"] do
                if English label exists then
                    pid_labels[pid] ← label
                else
                    pid_labels[pid] ← pid
                end if
            end for
        else
            Log API request error
        end if
        Sleep for 0.5 seconds                    ▷ To respect API rate limits
    end for
    return pid_labels
end procedure
```

3.3.3 Label-Matching Analysis with Description Texts

To assess the extent to which property information is already reflected in existing descriptions, we performed keyword-based matching between property labels and English description texts. This approach assumes that, for some properties, the label or a close lexical variant may appear explicitly in the description.

To ensure consistency, all texts were lowercased before matching, and we used exact word-level matches rather than partial or fuzzy matching. While more advanced methods (e.g., semantic similarity via embeddings) were considered, they would have significantly increased the complexity and computational cost due to the need for tokenization and vector representations. Given that our main goal is to evaluate the feasibility and extensibility of the pipeline, we select a simpler matching strategy.

3.3.4 Semantic Tag Frequency for P31

As a sub-analysis, we investigated how entity type information—captured by the `P31` (instance of) property—is reflected in the English descriptions of city entities.

Figure 3.5: P31 (instance of) values for Gothenburg

Figure 3.5 shows example values of `P31` for Gothenburg, illustrating the diversity of entity types used in Wikidata.

Specifically, we sampled 10,000 cities and extracted the English labels of their `P31` values (e.g., "town", "municipality", "human settlement"). We then applied the same keyword-based matching method described in Section 3.3.3 to check whether these terms appeared explicitly in the descriptions.

This analysis aimed to identify common lexical realizations (e.g., "largest city") that might be useful as candidate phrases for grammar construction. However, due to the variability and potential redundancy introduced by including this information, we chose not to incorporate `P31`-based labels in the current generation phase. Instead, integrating this dimension is considered future work, as it may increase sentence complexity without always contributing essential content. Our current goal is to generate short, concise, and informative descriptions based on a minimal set of core properties.

3.4 LLMs-Based Description Generation

In addition to generating descriptions using GF, this thesis also implements a retrieval enhancement process based on LLMs for generating multilingual descriptions. The specific results of generating descriptions will be shown and discussed in the Chapter 4. This section outlines the implementation process. An overview of the RAG architecture is illustrated in Figure 3.6.

The first task was data preparation and vector indexing, where we merged the three datasets (Cities, Universities and Mathematicians) into a single corpus. Then, using the `all-MiniLM-L6-v2` model, a sentence-transformers model that transforms sentences and paragraphs into dense vector space [23]. These vectors were indexed

using FAISS (Facebook AI Similarity Search), a highly efficient, open-source library for rapid similarity search and clustering of dense vectors [24]. The specific core code is provided in the Appendix B.1.

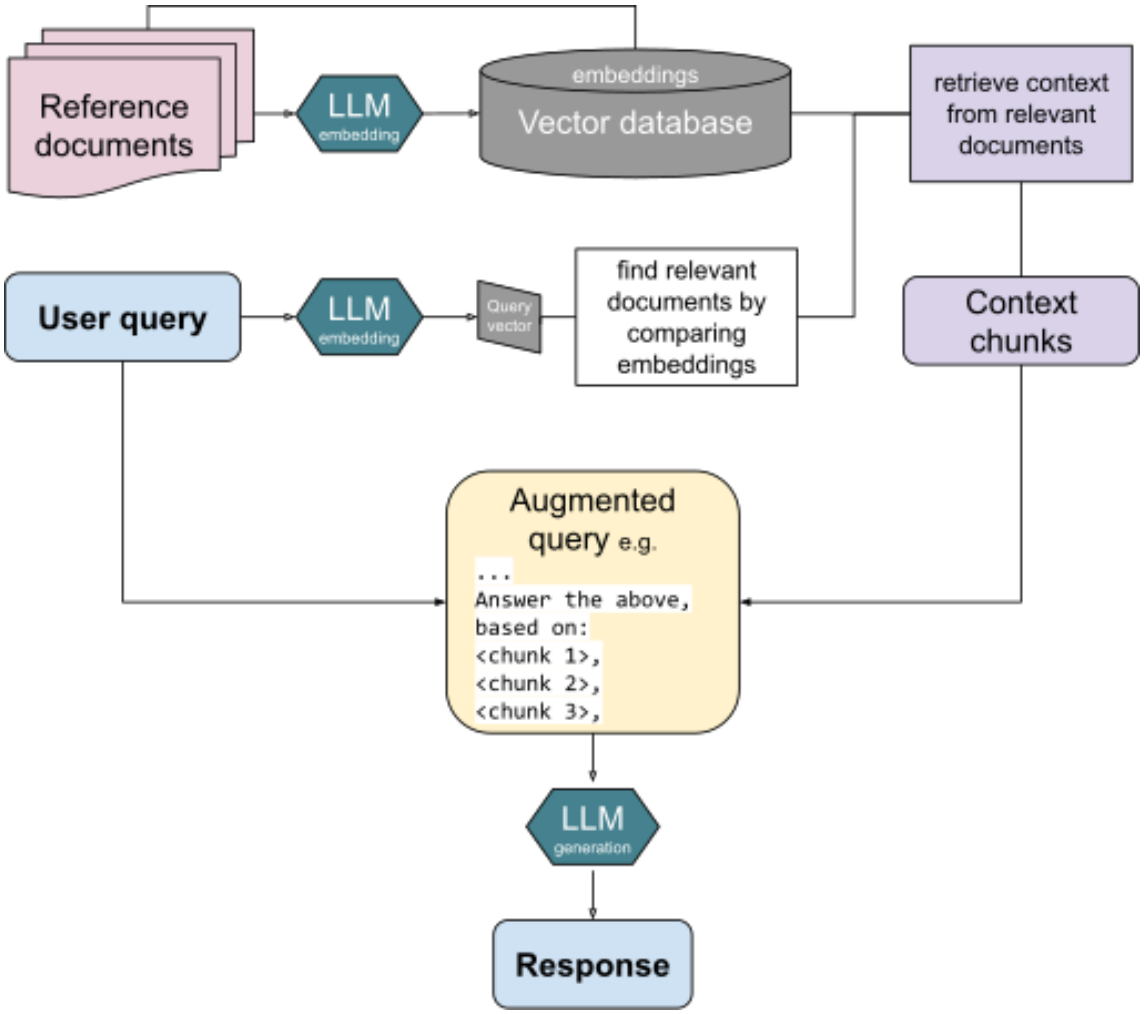

Figure 3.6: An illustration of the Retrieval-Augmented Generation process.[1]

When a user enters a query (such as "Describe David Hilbert?" or "Describe Stockholm"), the developed system in this thesis first tries to detect the language of the input using the Python library `langdetect`. After that, it turns the query into vector embeddings (query vectors) by using the `all-MiniLM-L6-v2` model. These query vectors are compared with vectors indexed in the database created earlier during the data preparation step, retrieving the most relevant context chunks from the knowledge base.

Next, these context chunks are combined with the original query to form an augmented query. This augmented query explicitly instructs the LLM (Mistral model) to generate the response based solely on the retrieved entity's information, includ-

[1]Source: Turtlecrown - Own work, CC BY-SA 4.0, `https://commons.wikimedia.org/w/index.php?curid=150390279`.

ing its name, domain, description, and other structured properties. For example, an English augmented query prompt may look like:

```
You are a helpful assistant. Based on the following information,
write a one-sentence factual description.

Name: Stockholm
Domain: https://www.wikidata.org/wiki/Q515
Description: capital and largest city of Sweden
Properties:
{
  "country": "Q34",
  "population": "984748",
  "capital of": "Q34"
}

Question: Describe Stockholm?
Answer:
```

The request prompt is forwarded to the Ollama platform, a lightweight local inference engine for LLms. In this thesis, it operates with the Mistral model—a fast, open-weight transformer-based LLM developed for efficient text generation. Together, they generate the final response in natural language based on the retrieved entity information.

4

Results

This chapter presents the results of our empirical analysis on Wikidata descriptions, focusing on identifying patterns in property usage. First, we examine the frequency distribution of all properties associated with city entities in Wikidata, revealing a small subset of high-frequency properties that dominate the dataset. This finding is crucial for guiding template-based generation system, as it identifies which properties are most likely to be present. Second, we assess the degree to which these high-frequency property labels actually appear in human-written English descriptions. The results uncover a strong semantic preference for administrative and geographic information, while identifiers and spatial data are rarely expressed. Finally, we analyze the lexical distribution of instance type tags (`P31`) in city descriptions. The analysis reveals a hierarchical structure in the use of urban classification terms, with "city" dominating the distribution and specialized tags appearing only occasionally.

4.1 Label Coverage Improvement

Lang	Raw	Geo Local	Geo API	OSM
en	1,342	1,339 (-3)	1,342 (0)	1,340 (-2)
zh	9,966	9,819 (-147)	9,827 (-139)	9,937 (-29)
jp	15,409	15,273 (-136)	15,155 (-254)	15,362 (-47)
sv	13,007	12,949 (-58)	12,848 (-159)	13,003 (-4)
el	22,179	22,113 (-66)	21,958 (-221)	22,155 (-24)
bn	26,397	26,392 (-5)	26,264 (-133)	26,352 (-45)

Table 4.1: Missing label counts and improvements (Δ = reduction from raw)

To reduce label sparsity and improve coverage across multilingual descriptions, we introduced external data augmentation from three sources: GeoNames (local dump), GeoNames (API), and OpenStreetMap (OSM). The table above compares the raw missing label counts with those after each enrichment step for six languages. The GeoNames API proved most effective, filling 471 missing values across the corpus, followed by the local GeoNames dump (228) and OSM (108). For instance, Japanese and Greek benefited the most, with the API-based method reducing Japanese missing labels by 254 and Greek by 221, respectively.

4.2 Dependency and POS-based Analysis

For deeper linguistic processing, we employed UDPipe's trained models to handle the syntactic and morphological analysis of descriptions in five key languages: English, Chinese, Japanese, Swedish, and Greek. Each description underwent tokenization, part-of-speech tagging, lemmatization, and dependency parsing through language-specific models. The system output structured CoNLL-U format data that revealed not just surface-level vocabulary differences but also deeper grammatical patterns across languages. Parsing this output allowed us to examine how different languages construct descriptions through their characteristic syntactic structures and lexical choices.

Our analysis focused on quantifying structural similarities between Wikidata descriptions across five languages (English, Swedish, Chinese, Japanese, and Greek) using two key linguistic features: universal dependency relations (DEPREL) and part-of-speech tags (UPOS). For each language pair, we computed cosine similarity scores of these features in two domains: city descriptions and university descriptions.

cos_similarity	**city_en**	**city_sv**	**city_zh**	**city_jp**	**city_el**
city_en	1	-	-	-	-
city_sv	0.86	1	-	-	-
city_zh	0.47	0.51	1	-	-
city_jp	0.85	0.79	0.55	1	-
city_el	0.64	0.65	0.65	0.67	1

Table 4.2: City Cosine Similarity

cos_similarity	**uni_en**	**uni_sv**	**uni_zh**	**uni_jp**	**uni_el**
uni_en	1	-	-	-	-
uni_sv	0.91	1	-	-	-
uni_zh	0.44	0.51	1	-	-
uni_jp	0.79	0.80	0.37	1	-
uni_el	0.79	0.83	0.56	0.65	1

Table 4.3: University Cosine Similarity

	EN	**SV**	**ZH**	**JP**	**EL**
cos_similarity	0.94	0.94	0.92	0.93	0.85

Table 4.4: University vs City Cosine Similarity

The similarity matrix revealed distinct patterns across language families. Germanic languages (English and Swedish) showed particularly high similarity, with scores of 0.86 for city descriptions and 0.91 for university descriptions. Japanese descriptions

demonstrated unexpected alignment with English (0.85 for cities, 0.79 for universities), despite being from different language families. Chinese descriptions showed the lowest cross-language similarity, ranging from 0.37-0.51 with other languages, reflecting its more distinct syntactic patterns.

Notably, within-language similarity between city and university descriptions was consistently high across all languages (0.85-0.94), suggesting Wikidata maintains strong structural consistency in how it describes different types of entities within the same language. The Greek data showed moderate similarity with all other languages (0.56-0.83), occupying an intermediate position between the Germanic/Japanese and Chinese patterns.

4.3 Key property Analysis

Property analysis is an important step to understand data distribution and optimize production quality. This section adopts a two-step analysis method: PID frequency analysis and label occurrence verification. Through a systematic analysis of the attribute distribution characteristics of city entities in Wikidata, this study reveals the underlying metadata patterns for descriptive text generation.

4.3.1 PID Frequency Analysis

The analysis process begins by extracting all property identifiers (PIDs) associated with city entities, using the regular expression r'P\\d+' to capture property codes such as "`P31`" and "`P17`." The official Wikidata API is then used to batch-fetch their English semantic labels (e.g., "`P31` → instance of"). A frequency analysis is performed on the 1,064 properties, revealing that approximately 80% of entity descriptions rely on the top 20 high-frequency properties, among which "country" (17,696 occurrences), "coordinate location" (16,400 occurrences), and "located in the administrative territorial entity" (16,204 occurrences) form the core metadata framework, reflecting the fundamental roles of entity type, geographic location, and founding time. To achieve efficient data processing, this study designs a batch

Property Name	**Frequency**
country	17,696
coordinate location	16,400
located in the administrative territorial entity	16,204
service retirement	2
conflict	2
ISBN-10	1

Table 4.5: Top 3 and Bottom 3 Properties with Frequencies

API request mechanism (50 PIDs per batch) and introduces a 0.5–second delay strategy, successfully avoiding the rate limit imposed by the Wikidata interface.

The resulting property frequency distribution table shows that the top 5% of high-frequency properties cover 89.7% of the entities, providing a design basis for filling templates with high-frequency properties in an automated generation system.

4.3.2 Label Occurrence Verification

The top 20 high-frequency PIDs (e.g., '`P17`' corresponds to "country") are extracted from structured data. Then, for each city entity, we check whether the values of these properties, like country names or administrative regions, appear in the English description text as exact word matches. By counting how often each PID's values are found in the text, we can see how much the description mentions the information contained in the structured data.

The results show that there are significant differences in the coverage of different property labels. Administrative geography attributes (e.g., "country") appear most frequently in descriptions, with 12,096 matches, indicating that city descriptions generally contain administrative affiliation information. In contrast, identifier attributes (e.g., "GeoNames ID", "VIAF ID", etc.) have extremely low coverage, with less than 10 matches, reflecting that the description text has a serious lack of references to unique identifiers and coordinate information is almost never integrated into the text description. These findings indicate that existing descriptions have a clear semantic bias, preferring to include human-readable administrative information while ignoring machine-readable identifiers and spatial data.

Property Name	**Frequency**
country	12,096
located in the administrative territorial entity	4,467
capital of	2,536
located in time zone	7
topic's main category	7
coordinate location	0

Table 4.6: Top 3 and Bottom 3 Properties Appeared in Descriptions

Although the exact matching used in this study is relatively simple, it provides a conservative estimate of the overlap between property labels and description texts. It avoids false positives but may miss semantically similar expressions. This analysis offers insight into how frequently certain types of information are explicitly mentioned in existing English descriptions.

4.3.3 Frequency of P31 Labels in Descriptions

Based on the analysis in Section 3.3.4, we looked further into how often the English labels of `P31` values (such as "city," "town," or "municipality") appear in the description texts of city entities. This analysis aims to help us decide whether it is useful to include type labels directly in automatically generated descriptions.

For each entity, we checked whether the English label(s) of its P31 types appeared exactly the same in its English description. For instance, for Leipzig (Q2079), we checked if the description used words like "city" or "urban municipality."

Table 4.7 shows how often the most common and least common P31 labels were found in these descriptions.

Instance of	**Frequency**
city	6,831
town in Hungary	351
largest city	218
hillfort	1
municipality	1
ghost town	1

Table 4.7: Top 3 and Bottom 3 Labels Appeared in Descriptions

The basic type "city" appears most often, which suggests it is important in describing urban entities. However, more specific labels like "town in Hungary" or "largest city" show up less frequently. This might mean that although Wikidata has detailed types, they are not always directly reflected in the text descriptions.

These findings support our earlier choice to focus on a small set of main type labels when creating short entity descriptions. Although using rare labels might add more detail, their low frequency and the possibility of making sentences more complex suggest they might not be very practical to include at this stage.

4.4 Other categories

While this chapter primarily focuses on city entities, we also extended our analysis to two additional Wikidata item types: universities and mathematicians, to examine whether similar property selection strategies can be applied across different domains.

For universities, we evaluated several potential attributes, including "instance of", "country", and "inception". After filtering for coverage and semantic relevance, we identified "country" (P17) and "inception" (P571, i.e., founding time) as the two most informative and consistently available properties. These are widely used in existing university descriptions and form a robust foundation for template-based generation.

For mathematicians, the focus shifted to biographical attributes. The properties "date of birth" (P569), "sex or gender" (P21), and "country of citizenship" (P27) were selected based on frequency and relevance. These properties are well-suited for constructing brief introductions (e.g., "French mathematician born in 1903").

The frequency distributions of these properties within their respective categories are provided in Table A.1 and A.2. These findings support the feasibility of applying our property analysis framework to diverse Wikidata item types.

4.5 Comparison of description results

In this section, we conduct a manual evaluation to compare and analyse the two description generation methods, with their specific results depicted in the following three figures 4.1, 4.2 and 4.3. We selected 5 examples in 3 different types of entities and generated descriptions in both Chinese and English languages. Among these, the beginnings labelled as RAG and GF refer to the RAG generation method and the GF generation method, respectively; those ending with "en" are in English, while those ending with "zh" are in Chinese. The subsequent analysis is conducted for city and university, focusing on three perspectives: consistency, redundancy, and simplicity.

City

Consistency: RAG exhibits serious consistency issues. For instance, London is incorrectly described as a town in Wisconsin, USA, rather than the capital of the UK. Similarly, Beijing and Guangzhou are misidentified as universities. These outputs are inconsistent with the corresponding Chinese descriptions, which are more accurate. In contrast, GF maintains consistency in both English and Chinese outputs, accurately reflecting elements such as city name, administrative level, and the country to which the city belongs.

Redundancy: RAG descriptions tend to be verbose and sometimes misleading. For example, the phrase "not the capital of any country" is both incorrect and misleading when applied to entities like London, which is indeed the capital of the UK. The Chinese RAG output also includes machine-readable codes like "belongs to Q39." However, these codes may not make sense to most human readers without looking them up elsewhere. Because they don't provide enough context on their own, they might not help make the description more useful. On the other hand, GF provides concise descriptions in both languages, focusing on key geographical identities such as "capital of the People's Republic of China."

Simplicity: RAG attempts free-form generation, leading to stylistic variety but also confusion. Statements like "known for its rich history and vibrant culture" are vague and offer no precise locational data. GF's approach is more structured, using a fixed sentence pattern such as "[city name]: the [capital/city] of [country]," which is both clear and easy to understand.

Label	RAG_en	GF_en	RAG_zh	GF_zh
Beijing	Beijing Huajia University is a university located in China, founded in 1985.	Beijing: capital of People's Republic of China	北京是中国的首都，人口为1538万。	北京市：中华人民共和国的首都
Guangzhou	The Guangzhou Open University is a university based in China, which was established in the year 1961.	Guangzhou: capital of Guangdong, People's Republic of China	广州是中国广东省的首都，人口为1308万	广州市：中华人民共和国广东省的首府
London	London is a city in Outagamie and Waupaca Counties, Wisconsin, United States with a population of 7348. It is not the capital of any country.	London: capital of United Kingdom	伦敦是英国的首都，属于Q39。	倫敦：英国的首都
Gothenburg	Gothenburg is a city located in Sweden (Wikidata Q6580), known for its rich history and vibrant culture.	Gothenburg: capital of Västra Götaland County, Sweden	哥德堡是阿尔巴尼亚科尔察州的一个市镇，人口为约20,848人。	哥德堡：瑞典西约塔兰省的首府
Stockholm	Stockholm is a city that has been featured in various works of fiction.	Stockholm: capital of Sweden	斯德哥尔摩位于瑞典（Sweden），居民为350,859人。	斯德哥尔摩：瑞典的首都

Figure 4.1: Comparison of city descriptions

University

Consistency: RAG performs poorly in maintaining consistent information. In the Chinese version, Tsinghua University is erroneously described as a branch of the University of Minnesota in the US. Chalmers University's foundation year is stated as 1967 instead of the correct 1829, and it is even mislocated in Austria in the Chinese text. Additionally, the Chinese description of MIT is oddly given in English. In contrast, GF uses templated outputs in both languages, consistently presenting school type, location, and founding year.

Redundancy: RAG's English outputs frequently include unnecessary elements such as hyperlinks, QIDs, and redundant subject phrases. These additions dilute the clarity of the description. GF avoids such issues by producing brief and clean summaries without extraneous links or explanations.

Simplicity: The language used in RAG outputs, especially in English and Chinese, tends to be overly flexible and structurally complex, often obscuring the core message. GF, by contrast, employs a formal and consistent style. Phrases like "public university in [country] founded in [year]" recur throughout the output, reinforcing logical clarity and ease of comprehension.

Label	RAG_en	GF_en	RAG_zh	GF_zh
Tsinghua University	Tsinghua University is a university located in Beijing, China (Q2541), established in 1911 (P580).	Tsinghua University: public university in Beijing, People's Republic of China founded in 1911	该实体是美国休斯顿大学清湖分校（University of Minnesota Twin Cities），位于美国米荷州（Minnesota），成立于1971年，属于美国的一所公立大学。	清华大学：建立于1911年的中华人民共和国北京市的公立大学
University of Cambridge	The University of Cambridge is a higher education institution located in England (Q878), more specifically, in the city of Cambridge (Q4091) with no recorded year of inception (inception: null).	University of Cambridge: public university in United Kingdom founded in 1209	剑桥大学位于英国，成立于1502年	剑桥大学：建立于1209年的英国的公立大学
Chalmers University of Technology	Chalmers University of Technology is a Swedish technical university located at http://www.wikidata.org/entity/Q3403 (Göteborg, Sweden), founded in 1967.	Chalmers University of Technology: university in Sweden founded in 1829	查尔姆斯理工大学位于奥地利（Q40），在维也纳市内（Q131266）。该大学于1995年成立。	查爾摩斯工學院：建立于1829年的瑞典的大学
ETH Zurich	ETH Zurich, located in Switzerland (Q39), has an architecture faculty that was established in 1854 (Q3918).	ETH Zurich: public university in Zurich, Switzerland founded in 1855	苏黎ce联邦美术大学位于瑞士的卑尔城市（Q72），成立于2007年。	苏黎世联邦理工学院：建立于1855年的瑞士苏黎世的公立大学
Massachusetts Institute of Technology	The Massachusetts Institute of Technology (MIT) is a prestigious higher education institution based in Cambridge, United States, founded in 1861.	Massachusetts Institute of Technology: private university in Cambridge, United States founded in 1861	Massachusetts Institute of Technology is an educational institution located in the United States, with the Wikidata entity Q143.	麻省理工學院：建立于1861年的美国剑桥的私立大学

Figure 4.2: Comparison of university descriptions

A comparison of the results of the mathematicians' descriptions is also provided here in Figure 4.3, but this is not discussed, as the content of the results of their discussion is similar to that of the city and university discussions.

Label	RAG_en	GF_en	RAG_zh	GF_zh
Alexander Grothendieck	Alexander Grothendieck is a male mathematician from France (Q30), known for his Ph.D. from Rutgers University, New Brunswick in 1957 (not John Grothendieck as stated).	Alexander Grothendieck: French male mathematician (1928 – 2014)	亚历山德罗·格罗滕迪克是1691年8月25日出生的男性，没有国籍。	亚历山大·格罗滕迪克：法国的男數學家 (1928 – 2014)
Alan Turing	Alan B. Tayler, born on September 5, 1931, is a male British mathematician (Q170790).	Alan Turing: British male computer scientist and mathematician (1912 – 1954)	阿兰·图灵（Alan Turing）是英国计算机科学的奠基人，被称为“计算机世界之父	艾倫·圖靈：英国的男電腦科學家、數學家 (1912 – 1954)
Hua Luogeng	Hua Luogeng is a Chinese mathematician (Q188635) from Guangdong, China. His properties include birth and death dates, fields of work, and notable achievements in mathematics.	Hua Luogeng: Chinese male mathematician and politician (1910 – 1985)	华罗庚（沃克茨維爾）是英国的政治家。他出生在英国。	华罗庚：中国的男數學家、政治人物 (1910 – 1985)
David Hilbert	David Hilbert was a German mathematician (country of citizenship: Q30), born on February 23, 1864 (date of birth not provided for David Frank Parkhurst) and male (sex or gender).	David Hilbert: German male mathematician born in Russia (1862 – 1943)	约翰·布朗大学是美国华盛顿特区的一所知名大学，创立于1919年。位于美国州德лавァ利（Q80016），属于美国（Q30）。	大卫·希尔伯特：德国的男數學家 (1862 – 1943)，出生于俄罗斯
John von Neumann	Hungarian-American mathematician (1903-1957)	John von Neumann: Hungarian male mathematician, computer scientist and physicist born in Kingdom of Hungary (1903 – 1957)	约翰·冯·诺伊曼（John von Neumann，1894年9月30日出生于Австро-匈牙利，国籍为美国）是一位重要的数学家和计算机科学家	约翰·冯·诺伊曼：匈牙利的男數學家、電腦科學家、物理学家 (1903 – 1957)，出生于匈牙利王国

Figure 4.3: Comparison of mathematician descriptions

5

Conclusion

5.1 Discussion

This thesis aimed to address the issue of inconsistent multilingual descriptions in Wikidata by proposing an automated approach to create semantically coherent and linguistically unified descriptions across various languages. Through a methodical collection and analysis of data from Wikidata and the application of the GF, the study achieved standardised descriptions for a range of entity types, such as cities, universities, and mathematicians. The findings demonstrate a noticeable enhancement in label coverage, particularly in English, which is primarily attributed to the successful incorporation of external data sources, such as GeoNames and OpenStreetMap.

Additionally, the thesis uncovered structural consistency in multilingual descriptions through dependency and POS-based analysis, showing significant alignment between English and Swedish. This offers both a theoretical foundation and practical evidence for the potential reuse of grammatical rules across languages in the future. Property analysis further emphasised the frequent occurrence of fundamental attributes in the descriptions of various entity types (such as cities, universities, and mathematicians), particularly in terms of geographic and administrative details, as well as personal accomplishments. Conversely, attributes like coordinate locations were included less often, a result established through the statistical analysis of attribute occurrence rates in the gathered Wikidata data.

Through comparative evaluations of three types of entities—cities, universities, and mathematicians—this thesis highlights notable performance disparities between RAG-based and GF-based systems. Regarding consistency, RAG outputs are plagued by significant factual inaccuracies and inconsistencies across languages. For instance, misidentifying London as a town in the US and Tsinghua University as an American institution not only compromises multilingual alignment but also misguides users. Conversely, GF consistently provides a well-structured and accurate output that faithfully retains essential facts such as geographic location and founding year. From the standpoint of redundancy, RAG descriptions frequently include superfluous metadata (such as Wikidata QIDs or extraneous details) that fail to enhance comprehension and often lead to confusion; GF sidesteps this issue by adhering to a minimalist format that emphasises only critical information. Furthermore, while RAG's stylistic flexibility might allow for greater expressiveness, it can result in

issues with conciseness, as free-form sentences often neglect important details or create ambiguities. In contrast, GF's templated format yields predictable, clear, and dependable outputs in both English and Chinese texts.

A significant aspect of this research is its contribution to enhancing the scalability and language independence of automated description generation. The suggested pipeline, which combines the WDQS with GF, effectively mitigates the semantic inconsistency issues associated with manual entries and substantially reduces the need for manual upkeep, showcasing considerable practical benefits.

Nevertheless, the thesis does present some limitations regarding data coverage, attribute completeness, language diversity, and the broader applicability of GF grammar rules. For example, specific low frequency attributes were not sufficiently utilised. A major constraint lies in language coverage: while Wikidata spans over 300 languages, the current implementation of GF relies primarily on the Resource Grammar Library (RGL), which supports only about 40 languages. This leaves significant gaps, especially for underrepresented or low-resource languages. Even widely used languages like English and Chinese exhibit noticeable data deficiencies in Wikidata. Notably, the complexity and extensibility of GF grammar rules also pose challenges for broader adoption.Furthermore, the types of data analysed in the study were somewhat restricted—primarily focusing on cities, universities, and mathematicians—leaving broader entity categories unexplored. These limitations collectively highlight clear opportunities for future enhancement in both data scope and linguistic scalability.

5.2 Conclusion

This thesis directly addresses the main question posed at the beginning: "What automated approach can be developed to produce Wikidata descriptions that are consistent in meaning and linguistically coherent in various languages?" In particular, it has pinpointed the sources of linguistic and structural discrepancies in existing Wikidata descriptions—especially those that stem from manual edits and differences in attribute selection; devised a structured data extraction process utilising WDQS to improve data completeness and standardisation; and implemented GF to facilitate the generation of grammatically correct, semantically reliable multilingual descriptions. The accuracy and consistency of the generated descriptions were assessed using dependency structure and part-of-speech tagging techniques.

In addition, a comparison of the findings presented in this thesis reveals that GF-based method significantly beats RAG-based system in terms of consistency, clarity, and cross-linguistic alignment. Although RAG systems can produce diverse and adaptable texts, their lack of factual stability and verbosity restricts their use in knowledge-focused multilingual contexts. GF, with its organised and deterministic generation approach, can offer more reliable and easily understood summaries, making it particularly suitable for situations that require precise and succinct factual descriptions.

The key contributions of this thesis comprise the introduction and execution of an automated method for generating multilingual descriptions, a comprehensive examination of the connection between property usage frequency and descriptive texts in cities, universities, and mathematicians, enhancements in label coverage, and a clear illustration of the possibility of reusing grammatical rules across languages due to structural similarities. These findings strengthen both the theoretical significance and practical applicability of Wikidata as a global multilingual knowledge repository.

Bibliography

[1] S. Malyshev, M. Krötzsch, L. González, J. Gonsior, and A. Bielefeldt, "Getting the most out of Wikidata: Semantic technology usage in Wikipedia's knowledge graph," in *The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part II 17*, Springer, 2018, pp. 376–394.

[2] A. Ranta, "Grammatical framework," *Journal of Functional Programming*, vol. 14, no. 2, pp. 145–189, 2004.

[3] R. Patil and V. Gudivada, "A review of current trends, techniques, and challenges in Large Language Models (LLMs)," *Applied Sciences*, vol. 14, no. 5, p. 2074, 2024.

[4] G. De Melo and G. Weikum, "Towards universal multilingual knowledge bases," in *Principles, Construction, and Applications of Multilingual Wordnets. Proceedings of the 5th Global WordNet Conference (GWC 2010)*, Narosa Publishing, New Delhi, India, 2010, pp. 149–156.

[5] S. Decker, S. Melnik, F. Van Harmelen, *et al.*, "The semantic web: The roles of XML and RDF," *IEEE Internet computing*, vol. 4, no. 5, pp. 63–73, 2000.

[6] W. W. W. Consortium *et al.*, "SPARQL 1.1 overview," 2013.

[7] D. Dannélls, M. Damova, R. Enache, and M. Chechev, "Multilingual online generation from semantic web ontologies," in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12 Companion, Lyon, France: Association for Computing Machinery, 2012, pp. 239–242, ISBN: 9781450312301. DOI: 10.1145/2187980.2188018.

[8] A. Ranta, "Towards multilingual Autoformalization and Informalization of Mathematics," in *Swedish Language Technology Conference SLTC2024*, 2024. [Online]. Available: https://sltc2024.github.io/abstracts/ranta.pdf.

[9] M. Straka, J. Hajič, and J. Straková, "UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, N. Calzolari, K. Choukri, T. Declerck, *et al.*, Eds., Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 4290–4297. [Online]. Available: https://aclanthology.org/L16-1680/.

[10] P. Xia, L. Zhang, and F. Li, "Learning similarity with cosine similarity ensemble," *Information sciences*, vol. 307, pp. 39–52, 2015.

[11] S. Decker, S. Melnik, F. van Harmelen, *et al.*, "The semantic web: The roles of XML and RDF," *IEEE Internet Computing*, vol. 4, no. 5, pp. 63–73, 2000. DOI: 10.1109/4236.877487.

[12] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966, ISSN: 0001-0782. DOI: 10.1145/365153.365168.

[13] E. Reiter and R. Dale, *Building Natural Language Generation Systems* (Studies in Natural Language Processing). Cambridge University Press, 2000.

[14] N. Chomsky, *Syntactic Structures*. The Hague: Mouton, 1957.

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[16] M. Lewis, Y. Liu, N. Goyal, *et al.*, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703.

[17] C. Gardent, A. Shimorina, S. Narayan, and L. Perez-Beltrachini, "The WebNLG challenge: Generating text from RDF data," in *Proceedings of the 10th International Conference on Natural Language Generation*, J. M. Alonso, A. Bugarín, and E. Reiter, Eds., Santiago de Compostela, Spain: Association for Computational Linguistics, Sep. 2017, pp. 124–133. DOI: 10.18653/v1/W17-3518.

[18] F. Zhu, S. Tu, J. Shi, J. Li, L. Hou, and T. Cui, *Twag: A topic-guided wikipedia abstract generator*, 2021. arXiv: 2106.15135 [cs.CL].

[19] K. Angelov, A. Carrión del Fresno, E. Voloshina, and A. Ranta, "Leveraging Grammatical Framework and WordNet for Natural Language Generation from Wikidata," in *Distributed Computing and Artificial Intelligence, Special Sessions I, 21st International Conference*, R. Mehmood, G. Hernández, I. Praça, *et al.*, Eds., Cham: Springer Nature Switzerland, 2025, pp. 173–184, ISBN: 978-3-031-76459-2.

[20] D. Vrandečić, "Building a Multilingual Wikipedia," *Communications of the ACM*, vol. 64, no. 4, pp. 38–41, 2021. [Online]. Available: https://cacm.acm.org/magazines/2021/4/251343-building-a-multilingual-wikipedia/fulltext.

[21] A. Ranta, "Multilingual text generation for abstract wikipedia in Grammatical Framework: Prospects and challenges," in *Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021)*, R. Loukanova, P. L. Lumsdaine, and R. Muskens, Eds., Cham: Springer Nature Switzerland, 2023, pp. 125–149.

[22] A. Gutman, A. Ivanov, and J. S. Ramírez, *Using dependency grammars in guiding templatic natural language generation*, Google Research Technical Report, available online, 2022. [Online]. Available: https://research.google/pubs/using-dependency-grammars-in-guiding-templatic-natural-language-generation/.

[23] N. Reimers and I. Gurevych, *All-MiniLM-l6-v2*, https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2, Pretrained sentence-transformer

model for semantic textual similarity and clustering, released on Hugging Face, 2021.

[24] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

A

Appendix A

```
concrete DescriptionsEng of Descriptions = CountriesEng,
   CitiesEng, ProvincesEng, CityKindsEng**

lincat
  Description = CN ;
  Location = Adv ;
  UniversityKinds = CN ;
  Attribute = Adv ;

lin
-- common grammars

  CityCountryLocation city country = SyntaxEng.mkAdv in_Prep (
     joinByComma city country.s) ;
  ProvinceCountryLocation province country = SyntaxEng.mkAdv
     in_Prep (joinByComma province country.s) ;
  CountryLocation country = SyntaxEng.mkAdv in_Prep country.s ;
  noLocation = ParadigmsEng.mkAdv "" ;

-- city grammars
  capitalKind = mkN "capital" ;
  ProvinceForCaptial province country = mkCN (mkN "capital") (
     SyntaxEng.mkAdv (mkPrep "of") (joinByComma province country
     .s)) ;
  CountryForCaptial country = mkCN (mkN "capital") (SyntaxEng.
     mkAdv (mkPrep "of") country.s) ;
  CityDescription kind location = mkCN kind location ;
```

Listing A.1: concrete syntax for city

```


(3.11.2) heluxue@heluxue:~/gf-wikidata-tools/scripts$ python3
   description.py Q1754
[city] capital_of_qid: Q34, capital_type: country
[city] CountryForCapital: Q34_Sweden_Country
Stockholm: Sveriges huvudstad
Stockholm: capital of Sweden
```

```
8
9  (3.11.2) heluxue@heluxue:~/gf-wikidata-tools/scripts$ python3
       description.py Q882
10 [build_human_expr] SameNationalityBuilding:
       SameNationalityBuilding (PersonBuilding "Charlie Chaplin"
       Male) Q145_United_Kingdom_Country (
       Q2526255_film_director_Profession) (BornAndDied "1889" "1977"
       )
11 Charlie Chaplin: brittisk manlig filmregissör -(18891977)
12 Charlie Chaplin: British male film director (1889 - 1977)
13
14 (3.11.2) heluxue@heluxue:~/gf-wikidata-tools/scripts$ python3
       description.py Q836805
15 [university] city_qid: None, country_qid: Q34
16 [university] CountryLocation: Q34_Sweden_Country
17 [university] expr_str: UniversityDescription university_Kind (
       CountryLocation Q34_Sweden_Country) (FoundedIn "1829")
18 Chalmers tekniska högskola i Sverige, grundad 1829
19 Chalmers University of Technology in Sweden founded in 1829
20
21 (3.11.2) heluxue@heluxue:~/gf-wikidata-tools/scripts$ python3
       description.py Q1325374
22 [build_human_expr] SameNationalityBuilding:
       SameNationalityBuilding (PersonBuilding "Yoshiyuki Tomino"
       Male) Q17_Japan_Country (ConjProfession (BaseProfession
       Q2526255_film_director_Profession) (BaseProfession
       Q6625963_novelist_Profession)) (OnlyBorn "1941")
23 Yoshiyuki Tomino: japansk regissör och romanförfattare (född
       1941)
24 Yoshiyuki Tomino: Japanese male film director and novelist (1941
       - )
```

Listing A.2: Sample outputs of the `description.py` script

Property Name	**Frequency**
instance of	28,432
country	26,609
inception	23,333
floors above ground	2
head of government	2
Mastodon instance URL	1

Table A.1: Property frequency of Universities

Property Name	**Frequency**
occupation	17,070
instance of	17,067
sex or gender	17,067
represents	2
Artists in Canada record number	2
Pinterest username	1

Table A.2: Property frequency of Mathematicians

B

Appendix B

```
def normalize_entity_multi(data):
    """
    Normalize a list of Wikidata entities by extracting relevant
        labels and descriptions
    in both Chinese and English. Filters out entities without
       either label.

    Args:
        data (list): List of entity dictionaries from Wikidata.

    Returns:
        list: A list of dictionaries containing normalized
           entity info.
    """
    result = []
    for item in data:
        # Extract and clean Chinese and English labels
        label_zh = str(item.get('labels', {}).get('zh', '') or
           '').strip()
        label_en = str(item.get('labels', {}).get('en', '') or
           '').strip()

        # Skip the entity if it lacks both Chinese and English
           labels
        if not label_zh and not label_en:
            continue

        # Extract and clean Chinese and English descriptions
        desc_zh = str(item.get('descriptions', {}).get('zh', '')
            or '').strip()
        desc_en = str(item.get('descriptions', {}).get('en', '')
            or '').strip()

        # Construct a normalized dictionary for the entity
        result.append({
            'entity_zh': label_zh,
            'entity_en': label_en,
            'desc_zh': desc_zh,
```

```
            'desc_en': desc_en,
            'prop': item.get('prop', {}),            # property-value dictionary
            'domain': item.get('domain', 'unknown'), # domain category (e.g., city, person)
        })
    return result

def get_embedding(text):
    """
    Generate a vector embedding for the input text using a pretrained encoder.

    Args:
        text (str): The input text string.

    Returns:
        np.ndarray: A 1D numpy array representing the text embedding.
    """
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True).to(device)
    with torch.no_grad():
        output = encoder(**inputs).last_hidden_state.mean(dim=1)
    return output.cpu().numpy()[0]

# Generate embeddings for all entities using their Chinese or English labels
doc_embeddings = np.array([
    get_embedding(doc['entity_zh'] or doc['entity_en']) for doc in all_docs
])

# Create a FAISS index for fast nearest neighbor search based on L2 distance
index = faiss.IndexFlatL2(doc_embeddings.shape[1])
index.add(doc_embeddings)
```