European Research Studies Journal
Volume XXVII, Special Issue 3, 2024

pp. 414-425

The Effect of the Context Length of Large Language Models on the Quality of Extracted Information in Polish in the Process of Project Management

Submitted 14/09/24, 1st revision 22/09/24, 2nd revision 21/10/24, accepted 30/10/24

Damian Pliszczuk[1], Michał Maj[2], Patryk Marek[3], Weronika Wilczewska[4], Tomasz Cieplak[5], Tomasz Rymarczyk[6]

Abstract:

Purpose: *This paper presents GraphRAG, a novel tool that integrates large language models (LLMs) with knowledge graphs to enhance the precision and consistency of responses generated from unstructured text data. The primary objective is to improve the quality of information retrieval and synthesis for complex user queries requiring comprehensive understanding.*

Desugn/Methodology/Approach: *The GraphRAG framework processes source documents by dividing them into smaller fragments (chunks) to facilitate knowledge extraction. Using community detection algorithms, such as the Leiden algorithm, GraphRAG identifies semantic clusters within the knowledge graph, enabling both local and global information retrieval. The tool employs a multi-stage analysis approach, leveraging prompts to detect entities and relationships in the text, which are then organized into structured graph nodes and edges.*

Findings: *The experimental results reveal that smaller chunk sizes (e.g., 300 tokens) significantly improve the granularity of detected entities and relationships, leading to a more detailed knowledge graph structure. This approach enhances response accuracy for knowledge-intensive queries by enabling the LLM to focus on specific text segments, improving the precision of extracted information.*

Practical Implications: *GraphRAG has practical applications in any domain where accurate and context-rich responses are essential, such as customer support, decision-making, and research analysis. By balancing chunk size and processing efficiency, the tool enables scalable analysis while maintaining high data quality, making it a valuable asset for knowledge-intensive tasks.*

Originality/Value: *This research contributes to the field by demonstrating an effective integration of LLMs with knowledge graphs to process large text corpora. GraphRAG's method of combining local and global retrieval through knowledge graphs represents an*

[1]*Netrix S.A., Poland, WSEI University in Lublin, Poland, damian.pliszczuk@netrix.com.pl;*
[2]*WSEI University in Lublin, Poland, michal.maj@wsei.pl;*
[3]*Netrix S.A., Poland, patryk.marek@netrix.com.pl;*
[4]*Lublin University of Technology, Lublin, Poland, w.wilczewska@pollub.pl;*
[5]*Lublin University of Technology, Lublin, Poland, t.cieplak@pollub.pl;*
[6]*Netrix S.A., Poland, WSEI University in Lublin, Poland, tomasz.rymarczyk@netrix.com.pl;*

advancement over traditional retrieval-augmented generation methods, especially in scenarios requiring detailed information synthesis.

Keywords: *Knowledge graphs, language models, retrieval-augmented generation, semantic clustering, GraphRAG.*

JEL Codes: *C45, C61, M31, L8, D83.*

Paper Type: *Research article.*

1. Introduction

Large language models (LLMs) play a key role in natural language processing, enabling the generation of consistent and contextually relevant responses to a variety of user queries. One area of research related to LLM development is the integration of external knowledge by building knowledge graphs that support language models to better understand and generate information based on real facts (Wu *et al.*, 2022) (Przysucha *et al.*, 2024).

However, the challenge for such models remains to make effective use of huge collections of unstructured text data, especially in cases of queries that require a global understanding of the topic (Lewis *et al.*, 2020).

Traditional Retrieval-Augmented Generation (RAG) approaches rely on direct search and the inclusion of text snippets semantically related to the query. However, these methods often fail when queries require synthesizing information from multiple sources or when data are incomplete (Lewis *et al.*, 2020).

To overcome these limitations, research is increasingly focusing on integrating knowledge graphs with language models to represent individuals and the relationships between them in a more structured way that supports the response generation process (Ji *et al.*, 2022; Hogan *et al.*, 2021; Lewis *et al.*, 2020).

GraphRAG, a tool developed by Microsoft, is an innovative approach that combines LLM capabilities with knowledge graphs to efficiently process data and generate responses to user queries in a more consistent and precise manner (Bi *et al.*, 2019). Applying the Leiden algorithm to detect communities in a knowledge graph enables better contextual understanding and semantic clustering of data, which supports both local and global information retrieval (Kang *et al.*, 2023).

This approach overcomes the limitations of traditional RAG methods, enabling language models to not only better understand context, but also to provide more comprehensive and complex answers, especially in tasks requiring information synthesis, trend identification, or guiding themes in large text corpora (Lewis *et al.*, 2020; Bi *et al.,* 2019).

2. Building Graphs

GraphRAG is a tool that allows you to build a knowledge graph to efficiently answer queries based on unstructured data. The process consists of several key steps that step-by-step transform raw documents into a structured graph structure for generating consistent answers to user queries.

The entire process begins with source documents, which contain texts that require analysis, such as articles, reports or other resources. These documents are transformed into smaller fragments (called chunks) to facilitate further processing. Each text fragment is subjected to a specially tailored summarization process, taking into account the specific knowledge or area from which the material comes. The purpose of this stage is to extract key information about the structure of the data and text.

The tool then identifies the individual units in the text that constitute the elements of the knowledge graph. These can be people, organizations, places or other relevant elements included in the content. Each of these elements is described in a manner tailored to the specific area under study. Such customization improves the results of the analysis in a contextual sense.

Next, community detection is performed. The tool is designed to group related elements based on their interrelationships. These communities are created using special algorithms, such as the Leiden algorithm, which effectively detects hierarchical structures in large-scale graphs (hundreds of thousands or millions of nodes).

For each detected community, summaries are created that allow the user to better understand the structure of the data and the relationships between graph elements. These reports help in data analysis, especially when there are questions about the totality of available information.

Based on these community summaries, GraphRAG generates answers to user queries. These answers are formulated based on specific questions, and each question can refer to several communities simultaneously. This process allows the creation of fragmented answers, which in subsequent steps are combined into an overall answer.

Ultimately, the tool creates a global answer to the user's question. By collecting fragments of answers from different communities, GraphRAG generates a consistent and complete summary. In this way, the entire knowledge graph structure is used to efficiently process and answer questions that require extensive analysis and synthesis of information.

The GraphRAG tool pipeline built in this way allows the effective use of language models to generate answers to questions based on complex data sets, combining knowledge graphs and LLM.

3. Document Division

In the process of graph construction, the key step is to divide the documents into smaller fragments, called chunks, and identify the relevant units and the relationships between them. This step determines the efficiency of data processing and the quality of the built graph. Each step in the process is aimed at accurately identifying and organizing data to enable later responses to complex user queries.

This is an important aspect of graph design, as the level of detail with which data is shared affects the efficiency of subsequent analysis. By default, GraphRAG divides documents into chunks of about 1,200 tokens. Tokens are the smallest units of text, which can consist of individual words or characters.

Choosing the number of tokens at 1,200 is a compromise that strikes a balance between the accuracy of the analysis and the efficiency of processing large amounts of data in a short period of time. Smaller chunks of text (fewer tokens) allow for more detailed analysis and the detection of more units and relationships, which improves the precision of the results. However, the smaller the chunks, the more computational resources are required to process the entire data set, which can increase the operation time.

On the other hand, larger chunks of text, with more tokens, make it possible to process more data at once, which reduces the number of operations needed for analysis, thus reducing processing time.

However, this may come at the expense of accuracy - in larger chunks, the model may overlook some entities or relationships, especially those that occur in less obvious contexts. The size of the chunks therefore has a direct impact on the number of units (nodes) and relations (edges) detected in the data.

The next step in the graph construction process is the detection of nodes and edges between them. To do this, GraphRAG uses the Large Language Model (LLM), which processes previously segmented text fragments using specially designed prompts (text commands).

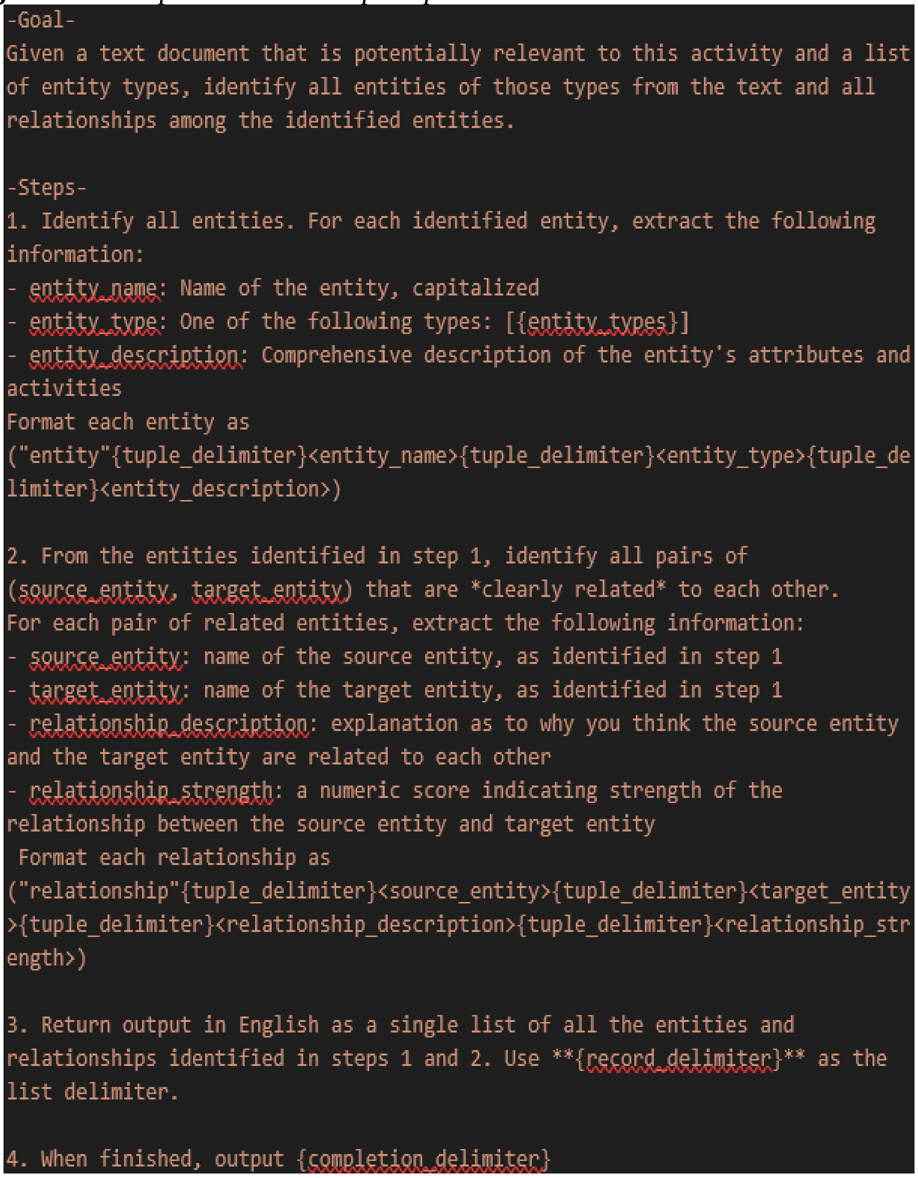
The main goal of this step is to identify all relevant elements in the text. These entities, as mentioned earlier, can represent different entities (people, organizations, products, places or other categories depending on the specifics of the document).

The detection process is carried out by means of a multi-part prompt, which guides the model step by step through the analysis of the text.

The basic prompt consists of the following sections:

Purpose of action: Explains the task, which is to identify all units of specific types from the document and all relationships between these units.

Figure 1. *Multi-part base search prompt*



```
-Goal-
Given a text document that is potentially relevant to this activity and a list
of entity types, identify all entities of those types from the text and all
relationships among the identified entities.

-Steps-
1. Identify all entities. For each identified entity, extract the following
information:
- entity_name: Name of the entity, capitalized
- entity_type: One of the following types: [{entity_types}]
- entity_description: Comprehensive description of the entity's attributes and
activities
Format each entity as
("entity"{tuple_delimiter}<entity_name>{tuple_delimiter}<entity_type>{tuple_de
limiter}<entity_description>)

2. From the entities identified in step 1, identify all pairs of
(source_entity, target_entity) that are *clearly related* to each other.
For each pair of related entities, extract the following information:
- source_entity: name of the source entity, as identified in step 1
- target_entity: name of the target entity, as identified in step 1
- relationship_description: explanation as to why you think the source entity
and the target entity are related to each other
- relationship_strength: a numeric score indicating strength of the
relationship between the source entity and target entity
 Format each relationship as
("relationship"{tuple_delimiter}<source_entity>{tuple_delimiter}<target_entity
>{tuple_delimiter}<relationship_description>{tuple_delimiter}<relationship_str
ength>)

3. Return output in English as a single list of all the entities and
relationships identified in steps 1 and 2. Use **{record_delimiter}** as the
list delimiter.

4. When finished, output {completion_delimiter}
```

Source: *Own study.*

The base version of the Prompt is shown in Figure 1.

Steps: Specifies the steps the model is to take to complete the task. The model starts by identifying the units and then moves on to identifying the relationships between them.

- ➢ Identification of units: For each entity detected, the model extracts its name, type and detailed description. For example, for an entity representing a person, this could be data on name, function and work activity.
- ➢ Relationship identification: Once entities are detected, the model analyzes which entities are related to each other. If a relationship is detected between two units, the model creates a description of that relationship and assigns a strength (in the form of a numerical score) that determines how strong the link between them is.
- ➢ Data format: All units and relationships are returned as a list, with the data formatted accordingly. The model extracts the units and relationships in a structured form, which allows for further processing and building a knowledge graph.

To increase the accuracy of unit detection, GraphRAG uses additional procedures, based on heuristics, to identify as many elements as possible that the model can miss in the first phase of analysis. In the multi-stage processing, the model is asked whether all units have already been detected. If the model finds that any elements have been missed, it repeats the search and turns its attention to detecting all possible units.

This multiple detection cycle increases accuracy, especially in cases where some entities are difficult to identify due to ambiguous context or specific linguistic forms. Through this process, the model is able to build a more complete and accurate knowledge graph.

After the initial phase of detecting entities and relationships, the next step in the graph building process is to organize the detected elements into community structures that reflect the connections between nodes in the graph. This process uses algorithms for community detection, in which nodes with stronger connections are clustered together to better understand the interrelationships between individuals.

The graph created from the data of the previous stages is a homogeneous, undirected and weighted model. This means that nodes (representing detected entities, e.g., people, organizations, places) are connected by edges (relations) of varying strength, which depends on the number and quality of detected relation instances.

The weights of the edges represent normalized counts of relation instances between entities. At this stage, various community detection algorithms are used, such as the Leiden algorithm, which effectively handles the analysis of large-scale graphs.

This algorithm is particularly good at detecting the hierarchical structure of a community, making it possible to create complex, multi-level community models.

The hierarchical structure allows for more detailed analysis at different levels of generality, which is crucial for extensive datasets. Each level of the hierarchy provides a partition of the community that encompasses graph nodes in a mutually exclusive and collective manner, which allows the entire dataset to be effectively divided into smaller, thematically related groups.

This allows for global summarization, a so-called "divide-and-conquer" approach, in which smaller groups of data are analyzed separately and then combined into a larger whole. Once the community is formed, each detected cluster is subjected to a summarization process.

This is a key part of the analysis, providing the user with summary reports that enable a deeper understanding of the structure of the data and the relationships between nodes in the graph. Such summaries are of significant value, especially when analyzing large data sets, as they make it easier to understand the overall picture and detect key patterns.

These summaries are created using a specially designed method that can scale to very large data sets. The primary purpose of summary reports is to provide information about the overall structure of the community, the relationships between individuals, and the most important conclusions about the analyzed data. This makes it possible to quickly see the themes and relationships in the data even when the user does not have a pre-specified question.

These reports can also be used for global questions on the entire dataset. The user has the ability to query the entire graph, and the generated community summaries can be used as data for answering. This approach allows for automation of the analysis process and faster results, even for complex queries. As in previous stages of analysis, this one also uses the Large Language Model (LLM).

This model is responsible for generating detailed reports summarizing each community. The report generation process uses specially designed prompts that define the structure of the report, the purpose of its creation and how the results are presented. The basic prompt (Figure 2) used to generate community reports includes several key sections:

➢ The purpose of the report (Goal): Explains that the purpose of the model is to produce a detailed report on the community, including its entities, the relationships between them, and the relevant claims associated with those entities. The report is to be used by decision-makers, so it includes information on legal compliance, reputation, technical capabilities, and other relevant community data.

- Report Structure: The model generates a report containing:
  Title: The name of the community, representing its key entities.
  Summary: A brief summary of the community's structure and key information about the relationships between entities.
- Rating of community importance: A numeric value (from 0 to 10) representing the importance of the community in the context of the data.
- Detailed results: A list of 5-10 key findings about the community, with a description of each.
- Data referencing rules (Grounding Rules): Each proposal should be supported by data, with reference to relevant records in the dataset. The most important data that support the conclusion should be identified, and in the case of multiple links, be limited to the five most important.

4. Research Results and Discussion

This chapter will present the results of comparing different model configurations, as well as the effect of chunk size on the number of extracted entities and relationships. These results have important implications for further optimizing the knowledge graph construction process for more comprehensive data analysis.

The knowledge graph construction process was carried out in two iterations, using different model configurations to generate and tune the results. The main purpose of the comparison was to evaluate the impact of different configurations and sizes of text chunks on the quality of entity and relationship extraction.

This section of the article will present the results of these experiments and discuss the effects of changing the size of the chunks on the quality and completeness of the extracted data.

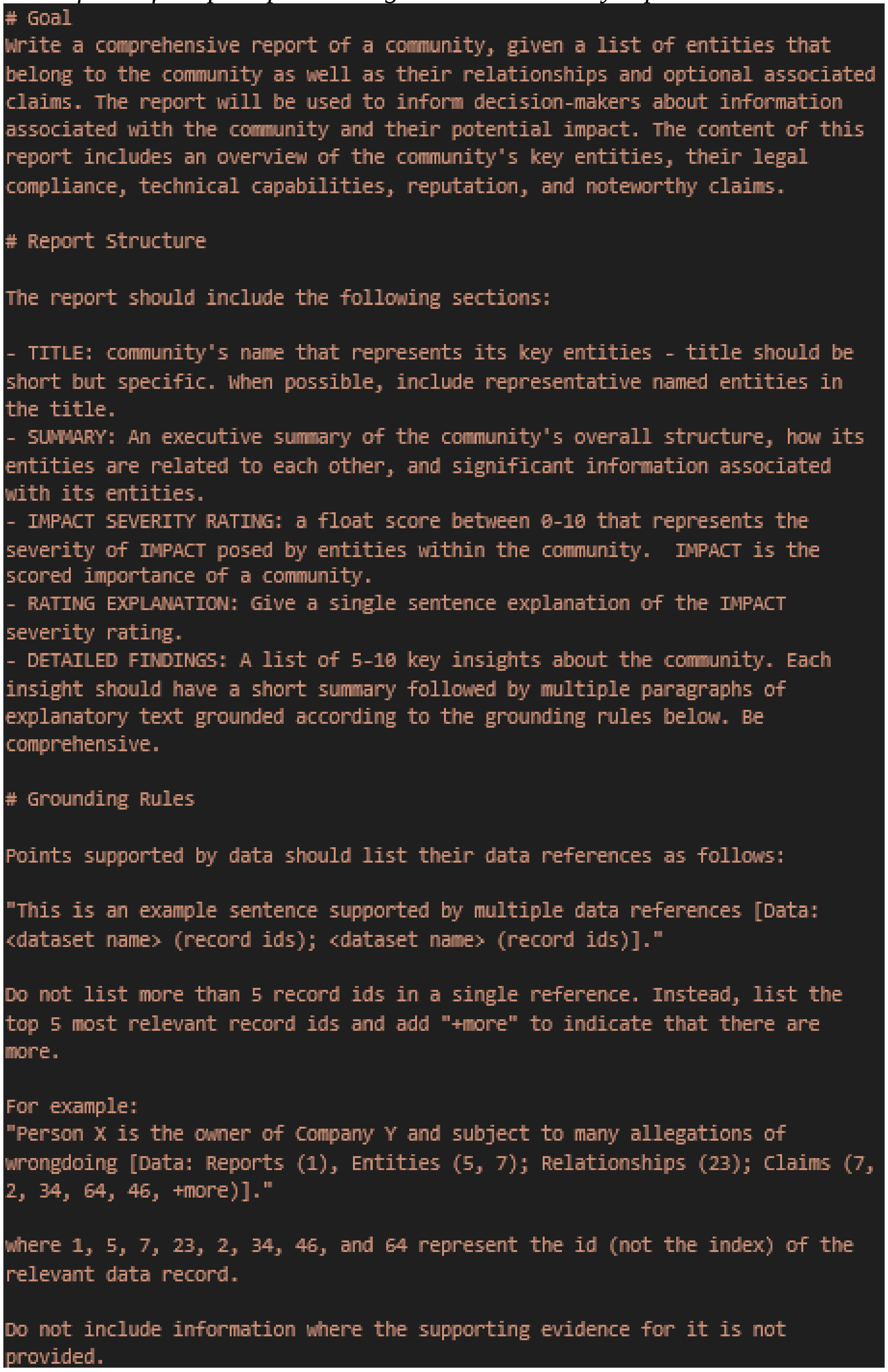
The following models were used to create knowledge graphs:

- Gemma 2 27b
- Llama 3.1 8b
- Llama 3 70b-instruct

In each case, the process involved adjusting the prompts. In some cases, a specific area was indicated (e.g., project management), while in other cases no area specification was used.

Experiments were conducted using two different text chunk sizes: 1200 tokens and 300 tokens. The number of extracted entities and relations was recorded for each configuration. Below is a comparison of the graphs created using different models and chunk sizes.

Figure 2. *The form of the prompt used to generate community reports*



```
# Goal
Write a comprehensive report of a community, given a list of entities that
belong to the community as well as their relationships and optional associated
claims. The report will be used to inform decision-makers about information
associated with the community and their potential impact. The content of this
report includes an overview of the community's key entities, their legal
compliance, technical capabilities, reputation, and noteworthy claims.

# Report Structure

The report should include the following sections:

- TITLE: community's name that represents its key entities - title should be
short but specific. When possible, include representative named entities in
the title.
- SUMMARY: An executive summary of the community's overall structure, how its
entities are related to each other, and significant information associated
with its entities.
- IMPACT SEVERITY RATING: a float score between 0-10 that represents the
severity of IMPACT posed by entities within the community.  IMPACT is the
scored importance of a community.
- RATING EXPLANATION: Give a single sentence explanation of the IMPACT
severity rating.
- DETAILED FINDINGS: A list of 5-10 key insights about the community. Each
insight should have a short summary followed by multiple paragraphs of
explanatory text grounded according to the grounding rules below. Be
comprehensive.

# Grounding Rules

Points supported by data should list their data references as follows:

"This is an example sentence supported by multiple data references [Data:
<dataset name> (record ids); <dataset name> (record ids)]."

Do not list more than 5 record ids in a single reference. Instead, list the
top 5 most relevant record ids and add "+more" to indicate that there are
more.

For example:
"Person X is the owner of Company Y and subject to many allegations of
wrongdoing [Data: Reports (1), Entities (5, 7); Relationships (23); Claims (7,
2, 34, 64, 46, +more)]."

where 1, 5, 7, 23, 2, 34, 46, and 64 represent the id (not the index) of the
relevant data record.

Do not include information where the supporting evidence for it is not
provided.
```

➢

Source: *Own study.*

Table 1. *Comparison of graphs generated with chunks of 1200 tokens*

Graph creating model	Prompt tuning	Nodes	Edges
Gemma 2 27b	Llama 3 70b instruct	25	20
Gemma 2 27b	Llama 3.1 8b	15	13
Gemma 2 27b	Llama 3 70b instruct nd	16	12
Gemma 2 27b	Gemma 2 27b	19	14
Llama 3.1 8b	Gemma 2 27b	9	4
Gemma 2 27b	-	22	13
Llama 3.1 8b	-	10	8

Source: *Authors' calculations.*

Table 2. *Comparison of graphs generated with chunks of 300 tokens*

Graph creating model	Prompt tuning	Nodes	Edges
Gemma 2 27b	Llama 3 70b instruct	106	120
Gemma 2 27b	Llama 3.1 8b	100	80
Gemma 2 27b	Llama 3 70b instruct nd	102	96
Gemma 2 27b	Gemma 2 27b	81	57
Llama 3.1 8b	Gemma 2 27b	24	0
Gemma 2 27b	-	129	31
Llama 3.1 8b	-	24	5

Source: *Authors calculations.*

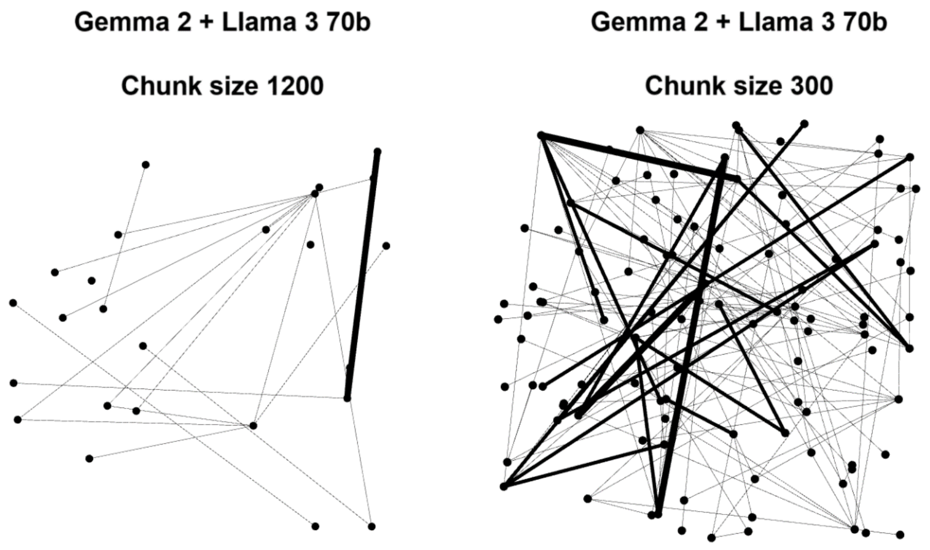
One of the key findings of the study is that smaller chunks (300 tokens) resulted in a significant increase in the number of extracted entities and relationships compared to larger chunks (1,200 tokens). This result can be attributed to the fact that smaller chunks allow the model to focus on specific segments of text in more detail, which improves the ability to identify entities and relationships.

This trend is confirmed by the results of experiments with 1200-token chunks, shown in Table 1, where the number of extracted entities and relations is much lower. Although larger chunks allow for faster text processing due to fewer segments, they may miss details that would have been detected with smaller and more concentrated chunks of text.

The choice of chunk size plays a key role in balancing the speed of processing with the quality of the extracted information. With larger chunks, models can process text faster, but may miss important entities and relationships, resulting in fewer nodes and edges in the generated graph. Smaller chunks, on the other hand, significantly improve the detail of the extracted data, leading to more comprehensive graphs.

The results indicate that the use of chunks with a size of 300 tokens provides a much more detailed graph structure, as illustrated in Figure 5, which shows the visualization of graphs generated with different chunk sizes. Figure 5 clearly shows a significant increase in the number of entities and relationships detected in scenarios with smaller chunks.

Figure 3. *Graph visualization for different chunk sizes*



Source: *Own study.*

5. Conclusions

This paper presents a complex knowledge graph construction process using the GraphRAG tool, which uses large language models (LLM) to extract information from unstructured textual data. The tool uses a search-assisted generation (RAG) method, enhancing traditional approaches with a complex knowledge graph structure that enables more precise answers to user queries. Special attention was paid to important pipeline steps, such as dividing documents into chunks, detecting entities and relationships, and creating communities using the Leiden algorithm.

Experimental results show that the size of text chunks has a key impact on the quality of data extraction. Smaller chunks (300 tokens) allow for more detailed analysis, leading to the detection of more entities and relationships, resulting in more complete knowledge graphs.

Larger chunks (1,200 tokens), on the other hand, allow for faster data processing, but may omit important entities and relationships. Different configurations of the models and prompts used to generate the graphs also affected the results, highlighting the importance of optimizing these elements in the graph generation process.

References:

Bi, B., Wu, C., Yan, M., Wang, W., Xia, J., Li, C. 2019. Incorporating External Knowledge into Machine Reading for Generative Question Answering. arXiv preprint

arXiv:1909.02745. https://doi.org/10.48550/arXiv.1909.02745.
Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D. 2020. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, 33, 1877-1901.
Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Larson, J. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv preprint arXiv:2404.16130.
Gupta, V., Lehal, G.S. 2010. A Survey of Text Summarization Extractive Techniques. Journal of Emerging Technologies in Web Intelligence, 2(3), 258-268.
Hogan, A., Gutierrez, C., Cochez, M., de Melo, G., Kirrane, S., Polleres, A., Navigli, R., Ngonga Ngomo, A.C., Rashid, S.M., Schmelzeisen, L., Staab, S., Blomqvist, E., d'Amato, C., Labra Gayo, J.E., Neumaier, S., Rula, A., Sequeda, J., Zimmermann, A. 2021. Knowledge Graphs. Synthesis Lectures on Data, Semantics, and Knowledge, 12(2), 1-257.
Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S. 2022. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. IEEE Transactions on Neural Networks and Learning Systems, 33(2), 494-514.
Kang, M., Kwak, J.M., Baek, J., Hwang, S.J. 2023. Knowledge Graph-Augmented Language Models for Dialogue Generation. arXiv preprint arXiv:2305.18846. https://doi.org/10.48550/arXiv.2305.18846.
Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.T., Rocktäschel, T., Riedel, S., Kiela, D. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.
Przysucha, B., Kaleta, P., Dmowski, A., Piwkowski, J., Czarnecki, P., Cieplak, T. 2024. Product Knowledge Graphs – Creating a Knowledge System for Customer Support. European Research Studies Journal, 27(2), 150-159. DOI: 10.35808/ersj/3395.
Traag, V.A., Waltman, L., Eck, N.J. 2019. From Louvain to Leiden: Guaranteeing Well-Connected Communities. Scientific Reports, 9(1), 5233.
Wu, L., Chen, Y., Shen, K., Guo, X., Gao, H., Li, S., Pei, J., Long, B. 2022. Graph Neural Networks for Natural Language Processing: A Survey. arXiv preprint arXiv:2106.06090. https://doi.org/10.48550/arXiv.2106.06090.