From notes to models: Leveraging LLMs for museum closure data

George A. Wright[1,2][0000−0002−2036−7737], Andrea Ballatore[2][0000−0003−3477−7654], Alexandra Poulovassilis[1][0000−0001−8981−4104], and Peter T. Wood[1][0000−0003−3704−1431]

[1] Birkbeck, University of London, Malet Street, London, WC1E 7HX, UK
{george.wright,a.poulovassilis,p.wood}@bbk.ac.uk
[2] Kings College London, The Strand, London, WC2R 2LS, UK
{george.wright,andrea.ballatore}@kcl.ac.uk

Abstract. GLAM research often involves the collection of unstructured textual data which is semantically rich, but labour-intensive to handle and process. This paper explores the use of Large Language Models (LLMs) to support the transformation of such material into structured data that can be queried and quantitatively analysed. Focusing on a corpus of notes documenting the closure of over 500 UK museums between 2000 and 2025, we present a two-stage pipeline to automate the generation of data models. In the first stage, an LLM proposes schema fragments based on chunks of notes; in the second, the LLM collates these fragments into a coherent data model. As a preliminary evaluation, we introduce a method based on syntactic validity to assess the structure of the generated models in the absence of ground truth. Our experiments with Llama 3.1 show promising results using zero-shot prompting, though ensuring semantic consistency and model integration remain challenging.

Keywords: information extraction · Large Language Models · data modelling · museum data

1 Introduction

Research in the GLAM (Galleries, Libraries, Archives, and Museums) sector frequently involves the collection and analysis of unstructured textual data. In order to support rigorous analysis and querying, such data must often be transformed into more structured formats This transformation remains a significant bottleneck: While some elements of these datasets can be captured through manually designed data models, fully modelling the breadth and complexity of the information is often infeasible within project constraints.

In this context, Large Language Models (LLMs) such as GPT and Llama offer a promising, if imperfect, set of tools for automating aspects of data structuring [7, 12]. Despite ongoing debates around their accuracy and generalizability, LLMs have demonstrated utility in tasks such as information extraction, ontology induction, and schema generation, especially in domains where training data is limited and where bespoke knowledge representations are required [1, 14].

This article reports on our development of a semi-automated pipeline for data modelling, applied to a corpus of textual notes collected as part of a project on museum closures in the UK from 2000 to 2025.[3] The project team has gathered detailed records about the closure of some 500 museums, including the fate of their collections and their buildings [16]. As part of the project, we have constructed a graph-based data model to capture collection dispersal information, but, due to time constraints, parts of the data relating to reasons for closure and buidling use remain unstructured. We have therefore decided to trial approaches to automate data modelling. Museologists will then be able to quantify the types of event that lead to closure; the changing status, use, and ownership of ex-museum buildings; and how these vary according to museum attributes.

Our study here aims to explore the effectiveness of LLMs to assist in generating data models from such unstructured text. After reviewing related work (§2) and describing our museum closure data (§3), We present an LLM-based pipeline of two stages: a *model suggester*, which generates models for chunks of text; and a *model collator*, which merges them into one (§4). We investigate the impact of prompt design and chunking strategies on the syntactic correctness of outputs. (§5). Finally, we draw conclusions and discuss future work (§6).

2 Related work

An LLM can be made to perform a task by providing it with a prompt that describes that task. *Prompt engineering* is the practice of altering prompts to improve performance. Adjustments include changing the way instructions are described, adding example input/output pairs (few-shot prompting [2]), and prepending the prompt with a role description which conditions the model to output tokens appropriate to the role, e.g., "you are a very serious professor" [10].

Progress has been made in reasoning tasks with *chain-of-thought* prompts: in a zero-shot setting, prompts are appended with the instruction "let's think step-by-step" [5]; in a few-shot setting, prompts are structured as worked examples [15]. This conditions an LLM to describe the steps leading to a solution.

With appropriate prompting, LLMs can be made to perform information extraction — to generate structured data such as a knowledge graph from a piece of text. Often, work on this task (see for example Polat *et al.* [12] and Papaluca *et al.* [11]) is tested on general knowledge text-to-triple datasets such as REDFM [4] and NYT [13], which respectively align Wikipedia and New York Times text with Wikidata and Freebase triples. But the prevalence of similar content in LLMs' pre-training data makes these datasets perhaps less challenging than notes collected during novel academic research such as in our setting.

A finding by Papaluca *et al.* [11] is that high performance is possible with what they term a *0.5-shot* prompt. They found highest performance with 5-shot prompts — 5 sentences alongside 5 sets of triples — but their 0.5-shot prompts contain only triples without matching sentences. For such prompts no pre-existing text/triple pairs are required. We have tested a similar approach.

[3] https://mapping-museums.bbk.ac.uk/museum-closure-in-the-uk-2000-2025/

As noted above, prompt engineering for information extraction often implicitly defines the ontology or data model by using a ground-truth ontology such as that of Wikidata, but we are automating the creation of a *new* ontology to record newly collected data. Frequently, ontology learning is separated into sub-tasks. These can include generating: lexical terms, types, a taxonomy, non-taxonomic relations, and additional constraints [1]. Lo *et al.* [7] in their *end-to-end* architecture only tackle the generation of taxonomies from text and use post-processing to combine them. We also separate tasks across a pipeline.

3 Museum closure data

The data collected by our project exists in three forms: (i) unstructured notes; (ii) a semi-structured spreadsheet (iii) a formal spreadsheet.

The unstructured notes consist of approximately 760,000 words including content from websites, newspaper clippings, email conversations with stakeholders and other information pertaining to closed museums and collections dispersal.

The semi-structured spreadsheet separates notes into three columns: the *collection* column describes what happened to each museum's collection post-closure; the *reasons* column describes what led to each closure; and the *building* column describes what happened to each museum's buildings post-closure.

To manually model the collection dispersal data, we created a formal spreadsheet with a controlled vocabulary. This is programmatically translated into a Neo4j graph database with 8 node and 13 relation types which serves as a basis for quantitative analysis of collection dispersal [16]. The formal spreadsheet and database do not however model data from the *reasons* or *buildings* columns.

4 Design of a data modelling pipeline

We have designed a pipeline to automate modelling of the *buildings* and *reasons* data, partially inspired by our approach to modelling the collection data, but also influenced by the practical constraints of working with LLMs.

Our manual modelling of collection data was iterative. We first designed a model to describe dispersals from a sample of museums. As we sought to fit new data into the model, it needed adjustment with new features. Once the data model was changed, it was necessary to update representations of previously entered data. This continued until we had modelled all of the data [16].

Such gradual refinement was necessary because it is impossible for humans to read a large body of text and then produce a consistent model of its semantics in one step. A similar problem exists for LLMs: their input has a fixed limit[4] and their performance degrades with larger inputs within this limit [6]. But instead of reproducing the cyclic process we followed when manually creating a data model, we have designed a pipeline with a hierarchical structure. This is easier to control and optimize. The pipeline consists of three main steps:

[4] The LLM we use (Llama 3.1) has a context length of 128 thousand tokens.

Data model suggestion Text describing an aspect of the closure of a set of museums is broken into chunks. Each chunk is given to an LLM tasked with suggesting an appropriate data model of entity and relation types. This results in a collection of (not necessarily compatible) data models, one for each chunk.

Data model collation Next, another LLM is tasked with combining these suggested data models into a single coherent model. The input to this LLM is the concatenation of models generated in the previous step.

Database generation Finally, an LLM uses the resulting data model to generate a graph database. This stage is left for future work.

5 Experiments

We have tested prompts for a data model suggester and have begun work on a model collator, so far using Meta's instruction-tuned 8 billion parameter Llama 3.1[5]. We will test other models in future. We ensure reproducibility by using a downloaded model[6] (and random seeds), not APIs to often updated models.

5.1 Data set and evaluation

As described in Section 3, we have text separated into *collection*, *building*, and *reasons* notes and a data model for the collection notes. We use the collection notes and data model as a support set to generate examples in n-shot prompts. We treat building notes as a development set for prompt optimization and the reasons notes as a test set. The building notes are shorter, so we assume that the reasons notes are more complex and a better test of the pipeline's generalizability.

Lacking a gold standard data model for buildings and reasons notes, we cannot directly measure correctness of a candidate data model. Automated evaluation at this stage is thus limited to syntactic correctness, which is important for gauging how well a relatively unconstrained process outputs models that conform to our JSON template for describing entities and relations (see Figure 1a).

This is done by comparing the keys in a candidate data model with the keys in the template (see Figure 1 for the template and an example model). The data models are represented with hierarchical JSON so must first be flattened into a list, where each element is the concatenation of the keys and array indices required to access the values in the leaves of the structure. Array indices are replaced with 0 since order is unimportant. A similarity score is then calculated for each candidate/template key pair:

$$\mathrm{Sim}(c,t) = 1 - \frac{L(c,t)}{\max(|c|,|t|)}$$

where $L(c,t)$ is the Levenshtein edit distance [9] between candidate and template keys. This is used for calculations of precision, recall, and F1 score (the harmonic mean of precision and recall), with:

[5] Downloaded from https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct.

[6] The experiments were run on the King's College London HPC system [8].

```
{
    "entities": [
        {
            "name": "...",
            "properties": [
                {"name": "...", "data_type": "..."},
                ...
            ]
        },
        ...
    ],
    "relations": [
        {
            "name": "...",
            "source": entity_type,
            "target": entity_type,
            "properties": [
                {"name": "...", "data_type": "..."},
                ...
            ]
        },
        ...
    ],
    "enumerated_types": [
        {
            "name": "...",
            "values": [
                "..."
            ]
        },
        ...
    ]
}
```

(a) The JSON template included in the LLM's prompt.

```
{
  "entities": [
    {
      "name": "Building",
      "properties": [
        {"name": "Status", "data_type": "enumerated type"},
        {"name": "Current use", "data_type": "string"},
        {"name": "Previous use","data_type": "string"},
        ...
      ]
    },
    ...
  ],
  "relations": [
    {
      "name": "USED_AS",
      "source": "Building",
      "target": "Current_use",
      "properties": []
    },
    ...
  ],
  "enumerated_types": [
    {
      "name": "Status",
      "values": [
        "open",
        "closed",
        ...
      ]
    },
    ...
  ]
}
```

(b) An extract from the model collator's output.

Fig. 1: The JSON template and an example output of an LLM that models building notes. All flattened keys in the candidate data model (e.g. "entities[0].name", "entities[0].properties[0].data_type") match with keys in the template.

$$\text{Prec}(C,T) = \frac{1}{|C|}\sum_{c\in C}\max_{t\in T}\text{Sim}(c,t) \qquad \text{Rec}(C,T) = \frac{1}{|T|}\sum_{t\in T}\max_{c\in C}\text{Sim}(t,c)$$

where C is the set of candidate keys and T the set of template keys.

5.2 The data model suggester

The data model suggester must generate entity and relation types for modelling the semantics of a chunk of notes. In developing a model suggester, we varied the size of chunks and the text of the prompt. LLM temperature was fixed at 0 but future work will test other values. Table 1 sets out the parameters tested.

We tested prompts containing: an optional role description, a description of the input data, a task description, an optional example using the collection notes and data model, a chunk of notes with average lengths ranging between 1 and 4 thousand characters, and an optional "let's think step-by-step" instruction.

Roles included: *data modeller*, intended to condition the LLM to use terms used in data modelling; and *machine*, intended to encourage only JSON output.

We tested three task descriptions with accompanying JSON templates: *basic* requesting only entities and relations with a JSON template for representing them; *basic+hierarchy*, explicitly instructing the model to "include entities and relations necessary for making type hierarchies if appropriate"; and *basic+hierarchy+enums*, with a field for enumerated types in the JSON template. These test the need for and effect of explicitly requesting types and taxonomies.

Parameter	Options	Importance
Chunk size	1k, 2k, 3k, 4k	0.05
Role *You are a...*	none *data modeller interested in the aftermath of museum closure.* *machine which communicates only in JSON.*	0.25
Task	basic, basic+hierarchy, basic+hierarchy+enums	0.06
N-Shots	0, 0.5, 1	**0.55**
Chain-of-thought	none *let's think step-by-step* worked through example (with 1-shot)	0.09

Table 1: Parameters tested while developing the model suggester and their feature importance found by random forest regression.

Zero-shot prompts have no examples. 0.5-shot prompts (cf. [11]) have no example notes but do have our JSON-formatted collection data model as an example output. 1-shot prompts include a chunk of collection notes for five museums followed by our collection data model as an example output.

In addition to the different combinations of features described above, we tested a 1-shot chain-of-thought prompt which replaces the task description with steps of reasoning starting from a chunk of notes and showing how lists of entities, properties, and relations can be found and then used to assemble a data model.

Results We instantiated 228 configurations of model suggester by combining different chunk sizes and prompting methods and ran them with the development set (building notes). A table of results is in our GitHub repository[7]. Some configurations performed well: the highest macro-averaged F1 score[8] was 0.95. Others output only English descriptions of the data, but no JSON, thus scoring 0. Using random forest regression to measure the importance of different prompt parameters, we found that the number of examples used in the prompt was the most important determiner of performance, followed by the role description, and then the presence or absence of chain-of-thought instructions (see Table 1).

On average, zero-shot prompts achieved higher F1 (0.79 for 0-shot, 0.71 for 0.5-shot, 0.28 for 1-shot), likely because examples of collection dispersal modelling are irrelevant to the task at hand. One would expect examples to improve performance, but that is often because they provide the LLM with vocabulary appropriate to the task (such as when Polat *et al.* [12] use semantic similarity to pick examples). Collection dispersal examples do not contain useful vocabulary beyond that found in the JSON template, but do significantly lengthen the context and thus hinder rather than help. They can also result in parts of the collection dispersal model being copied into the new building use model.

Role description was the second most important parameter: eight of the top-ten best performing configurations use the *data modeller* role and the other two

[7] https://github.com/Birkbeck/museum-object-flows/tree/main/llm-data-modelling
[8] The mean F1 for each of chunk calculated according to the method in Section 5.1.

use the *machine* role. The *machine* role, which instructs the LLM to output only JSON, did not prevent large blocks of English description being output.

Prompts with "let's think step-by-step" led to slightly higher performance (mean F1 of 0.62) compared with no chain-of-thought prompting (mean F1 of 0.59). Our one-shot chain-of-thought prompt resulted in very low performance (mean F1 of 0.06). Chain-of-thought prompts often led to long outputs stepping through incremental changes to the data model which were cut-off if they exceeded a maximum new tokens limit of 2000 (set for time-efficiency reasons). Differently worded prompts and looser output limits might improve performance.

The task description and chunk size had little effect on performance. Future tests will include larger chunk sizes to find when performance begins to decline.

The highest performing model suggester had a 0.5-shot prompt, the *data modeller* role, no chain-of-thought instruction, a task description with hierarchy and enums, and chunks of 4000 characters. Its F1 was 0.95 on the development set. We also ran it on the reasons test set where its F1 was 0.90.

5.3 The data model collator

As a preliminary investigation into developing a data model collator, we used the suggested data models from the best model suggester as input to an LLM tasked with creating a single coherent data model. The prompt to the LLM was: "combine these suggested entity and relation models into a single model which incorporates characteristics from each of the suggestions" followed by a list of the data models output by the model suggester for each chunk of notes.

The F1 score for the development set was 0.98. The output model had five entity types including *museum* and *building*, 21 relation types, and 3 lists of enumerated types. An extract of this data model is shown in Figure 1b and the full model is available in our GitHub repository[9]. Although the data model conforms to the JSON template (it has a precision of 1.0), it contains a number of inconsistencies. For example, *current use* and *previous use* are included as properties of the *building* type, but there are also *used as* relations from source *building* to target *current use*/*previous use* which would require current and previous use to be entities rather than properties. This indicates the need for a more robust automated evaluation to check that models are consistent. Another problem with the data model is that it contains a number of entities and relations copied from the collection dispersal model provided in the model suggester's 0.5-shot prompt. Additional evaluation of conciseness should check for the existence of such superfluous content in outputs.

We also ran this prompt with the outputs of the model suggester for the test set. But the prompt was about twice the length of the prompt for the development set and caused the GPU to run out of memory. This shows that a more generalizable pipeline will need to chunk the outputs of the data model suggester in a more gradual process of model collation.

[9] https://github.com/Birkbeck/museum-object-flows/tree/main/llm-data-modelling

6 Conclusions and future work

Our initial trial of an LLM-based data modelling pipeline shows that a data model can be produced by a sequence of two LLMs. This data model is not without errors, but is possible for a human to understand. Future work will investigate more robust automated evaluation that can detect these errors. In tests of the *data model suggester*, we found that use of examples and role descriptions had the largest effect on LLM performance with the highest performance achieved by 0- or 0.5-shot prompts and a data modeller role. We did not find that the chunk sizes tested had a significant impact on performance and will therefore test larger chunks in future, resulting in fewer suggestions for the model collator to process. Our initial test of the *model collator* shows some promise, with syntactically correct JSON (precision of 1.0 and recall of 0.96) being output for the development set. In ongoing work we are conducting more extensive tests of model collation, again considering different prompts. We will also test a more hierarchical approach to collation with small groups of data models being combined in a multi-step process. This will allow us to test whether including the English descriptions output by the model suggester aid the model collator.

For a more comprehensive study, we also plan to test a wider range of LLMs. Babaei Giglou *et al.* [1] found that LLMs with different architectures performed differently in their five ontology generation tasks: the best LLM at one task was not necessarily the best LLM for another. We will also test different temperatures and sampling strategies. Higher temperatures generally lead to more novel outputs and are appropriate for open-ended tasks (although it requires the accurate processing of factual information, our data modelling task is also quite open-ended). Different sampling strategies can also affect the overall coherence of the output [3], so will also be tested.

Once we have selected a configuration for data model generation, our next step will be to prompt another LLM with the data model and notes so that it can apply the data model to the notes and represent their semantics in a graph database. This is the final step in our pipeline from unstructured text to structured data. At this stage, we can further evaluate data models' conciseness by measuring the proportion of notes for which entity and relation types are instantiated. If we allow the LLM at the database generating stage to also suggest entities and relations which do not conform to a data model, we will also be able to evaluate model comprehensiveness by measuring the proportion of outputs that conform to the data model. We can then filter out data models that perform poorly on these metrics. We will then ask our museologist collaborators to carry out a human-led evaluation of the database, judging the degree to which it captures the information encoded in their notes and answers their research questions.

Acknowledgments. This work has been funded by the UKRI-AHRC project "Museum Closure in the UK 2000–2025", Grant No. AH/X012816/1, Oct. 2023–Sept. 2025. We thank the project's external Advisory Board and all members of the project team.

Disclosure of Interests. The authors have no competing interests to declare.

References

1. Babaei Giglou, H., D'Souza, J., Auer, S.: LLMs4OL: Large language models for ontology learning. In: The Semantic Web – ISWC 2023: 22nd International Semantic Web Conference, Athens, Greece, November 6–10, 2023, Proceedings, Part I. p. 408–427. Springer-Verlag, Berlin, Heidelberg (2023), https://doi.org/10.1007/978-3-031-47240-4_22
2. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., 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., 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.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020), https://doi.org/10.48550/arXiv.2005.14165
3. Holtzman, A., Buys, J., Du, L., Forbes, M., Choi, Y.: The curious case of neural text degeneration. In: Proceedings of the Eighth International Conference on Learning Representations (2020), https://arxiv.org/abs/1904.09751
4. Huguet Cabot, P.L., Tedeschi, S., Ngonga Ngomo, A.C., Navigli, R.: RED$^{\text{fm}}$: a filtered and multilingual relation extraction dataset. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 4326–4343. Association for Computational Linguistics, Toronto, Canada (Jul 2023). https://doi.org/10.18653/v1/2023.acl-long.237
5. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2022), https://doi.org/10.48550/arXiv.2205.11916
6. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts (2023), https://arxiv.org/abs/2307.03172
7. Lo, A., Jiang, A.Q., Li, W., Jamnik, M.: End-to-end ontology learning with large language models (2024), https://arxiv.org/abs/2410.23584
8. London, K.C.: King's computational research, engineering and technology environment (create), https://doi.org/10.18742/rnvf-m076, retrieved June 5 2025
9. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (Mar 2001). https://doi.org/10.1145/375360.375365
10. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R.: Training language models to follow instructions with human feedback (2022), https://arxiv.org/abs/2203.02155
11. Papaluca, A., Krefl, D., Rodríguez Méndez, S., Lensky, A., Suominen, H.: Zero- and few-shots knowledge graph triplet extraction with large language models. In: Biswas, R., Kaffee, L.A., Agarwal, O., Minervini, P., Singh, S., de Melo, G. (eds.) Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024). pp. 12–23. Association for Computational Linguistics, Bangkok, Thailand (Aug 2024). https://doi.org/10.18653/v1/2024.kallm-1.2
12. Polat, F., Tiddi, I., Groth, P.: Testing prompt engineering methods for knowledge extraction from text. Semantic Web **16**(2) (2025). https://doi.org/10.3233/SW-243719

13. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III. p. 148–163. Springer-Verlag, Berlin, Heidelberg (2010), https://doi.org/10.1007/978-3-642-15939-8_10
14. Schaeffer, M., Sesboüé, M., Charbonnier, L., Delestre, N., Kotowicz, J.P., Zanni-Merk, C.: On the pertinence of LLMs for ontology learning. In: Proceedings of the 3rd International Workshop on Natural Language Processing for Knowledge Graph Creation (2024), https://ceur-ws.org/Vol-3874/paper1.pdf
15. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E.H., Le, Q.V., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2022), https://doi.org/10.48550/arXiv.2201.11903
16. Wright, G.A., Ballatore, A., Poulovassilis, A., Wood, P.T.: Modelling and visualising flows of objects from museums. ACM J. Comput. Cult. Herit. (2025), in press