Recall Them All: Long List Generation from Long Novels

Sneha Singhania
MPI for Informatics
ssinghan@mpi-inf.mpg.de

Simon Razniewski
ScaDS.AI & TUD
simon.razniewski@tu-dresden.de

Gerhard Weikum
MPI for Informatics
weikum@mpi-inf.mpg.de

Abstract

Language models can generate lists of salient literary characters for specific relations but struggle with long, complete lists spanning entire novels. This paper studies the non-standard setting of extracting complete entity lists from full-length books, such as identifying all 50+ friends of Harry Potter across the 7-volume book series. We construct a benchmark dataset with meticulously compiled ground-truth, posing it as a challenge for the research community. We present a first-cut method to tackle this task, based on RAG with LLMs. Our method introduces the novel contribution of harnessing IR-style pseudo-relevance feedback for effective passage retrieval from literary texts. Experimental results show that our approach clearly outperforms both LLM-only and standard RAG baselines, achieving higher recall while maintaining acceptable precision.

1 Introduction

Motivation and Problem. Analyzing literary texts often involves entity markup and the extraction of relations between characters (Piper et al., 2021; Bamman et al., 2024). For example, to discover narrative patterns in contemporary or historical fantasy stories, a tool should track character movements across locations and label them by role or sentiment (Wilkens et al., 2024). Similarly, cultural studies on gender roles in fiction across different epochs and regions (Kejriwal and Nagaraj, 2024) need labeling of character types and relationships.

To this end, tools for NER/NED and relational IE (RE for short) must be adapted to the specifics of literary language and narrative structure. In this paper, we focus on the task of RE: identifying subject-predicate-object (SPO) triples in fictional narratives, where S and O are named entities, and P is a binary relation such as parent, family, friend, or opponent.

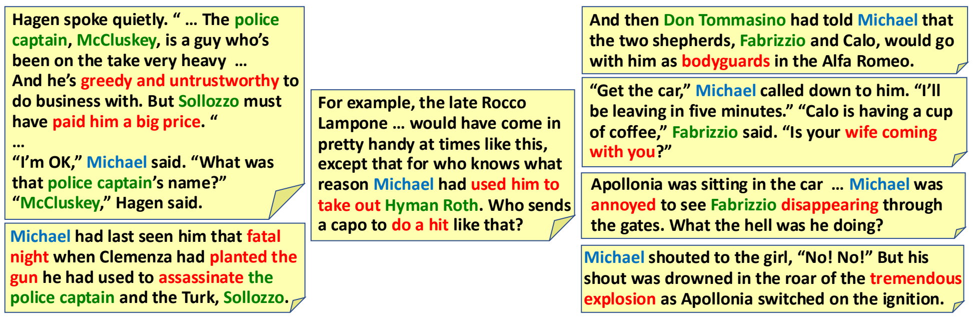
There is ample work on RE, based on deep neural networks (Han et al., 2020; Zhao et al., 2024). Recent methods employ LLMs for encoding input texts (Josifoski et al., 2022; Ma et al., 2023; Xu et al., 2024). These models are sequence-to-sequence taggers: given text T and target subject S, they identify candidate objects O appearing in T, tag cue words for relation P, and classify each SPO candidate as valid or invalid. The key limitation is that texts are short—often single paragraphs, commonly from Wikipedia. Thus, there are only a few O candidates, and the task reduces to classification: mapping SO candidates onto none, one, or more P.

RE methods perform well when the S and O entities are salient, the text T is short, and the language style can be learned upfront via training on Wikipedia or fine-tuning on a specific corpus. However, when the input spans an entire book, pre-training has limited value and fine-tuning is infeasible due to the lack of annotated data. Also, the desired outputs would include long-tailed O's that appear only a few times over hundreds of pages. In contrast to the $SO \rightarrow P$ approach of standard RE, we cast this underexplored task as $SP \rightarrow \{O\}$: given subject S and relation P, extract/generate a long—ideally complete—list of objects O that stand in relation P with S. This goal entails two major research questions:

RQ1: How well are LLMs performing on this challenge? How much value is added by running LLMs in RAG mode?

RQ2: How can the outcome of LLM/RAG methods be further enhanced? How can we boost recall without losing too much in precision?

To illustrate the problem, consider enemies/opponents of Michael Corleone in *The Godfather* books by Mario Puzo. Figure 1 shows book excerpts with cues about McCluskey, Sollozzo, Roth, Tommasino and Fabrizzio being in this list (which,



Hagen spoke quietly. " … The police captain, McCluskey, is a guy who's been on the take very heavy … And he's greedy and untrustworthy to do business with. But Sollozzo must have paid him a big price. "
…
"I'm OK," Michael said. "What was that police captain's name?" "McCluskey," Hagen said.

Michael had last seen him that fatal night when Clemenza had planted the gun he had used to assassinate the police captain and the Turk, Sollozzo.

For example, the late Rocco Lampone … would have come in pretty handy at times like this, except that for who knows what reason Michael had used him to take out Hyman Roth. Who sends a capo to do a hit like that?

And then Don Tommasino had told Michael that the two shepherds, Fabrizzio and Calo, would go with him as bodyguards in the Alfa Romeo.

"Get the car," Michael called down to him. "I'll be leaving in five minutes." "Calo is having a cup of coffee," Fabrizzio said. "Is your wife coming with you?"

Apollonia was sitting in the car … Michael was annoyed to see Fabrizzio disappearing through the gates. What the hell was he doing?

Michael shouted to the girl, "No! No!" But his shout was drowned in the roar of the tremendous explosion as Apollonia switched on the ignition.

Figure 1: Example for the problem of long lists from long narratives. For the subject "Michael Corleone", we aim to extract all 40 enemies/opponents, appearing in the books.

according to sources like fan wikis, has 40 people). We observe three cases: easy (left), hard (middle), and challenging (right). The easy cases are salient entities that are frequently mentioned—extracting them needs only one or two informative passages. The hard cases arise for entities that appear infrequently (like Hyman Roth, who is a minor figure in the books); here, the issue is finding the "needle in the haystack". Finally, the most challenging cases involve vague and terse cues for the predicate, requiring deeper inference over multiple, possibly scattered, passages—such as identifying Fabrizzio as the culprit behind the car bomb attack on Michael Corleone's wife.

Approach and Contributions. We devise a novel methodology to address this challenging task. Our method, called **L3X** (**L**M-based **L**ong **L**ist e**X**traction), operates in two stages:

Stage 1: Recall-oriented Generation. An LLM is prompted with a subject and predicate from a book, to generate a long list of candidate objects by various prompts, including RAG mode, with passages retrieved from the book text. In contrast to mainstream RAG, we retrieve a large number of passages (e.g., 500 for a given SP pair) and judiciously select the best ones for prompting.

Stage 2: Precision-oriented Scrutinization. Given a high-recall list of object candidates, we devise a classifier to corroborate or prune objects.

Since we tackle an unexplored task, we construct and release a new dataset for evaluation and as a resource for the NLP community. The data comprises 11 books or book series, with 16,000 pages total. It covers 8 relations of long-tailed nature (friends, opponents etc.). To use the copyrighted texts, purchasing the e-books is required.

Salient contributions of this work are: (1) the new task of extracting a long list of objects for a given subject and relation from book-length narrative texts; (2) L3X methodology for this task, based on retrieval-augmented LLMs and combining information-retrieval (IR) techniques with LLM generation; (3) experiments with a new benchmark, showing that L3X outperforms LLM-only and LLM-RAG baselines, with an in-depth analysis of strengths and limitations of different methods. The dataset, licensing details, code and experimental results are available at https://anonymous.4open.science/r/l3x-9E4A.

2 L3X Methodology

Figure 2 gives an overview of the L3X components and the data flow between them. The first four steps form the recall-oriented stage 1, the fifth step is for the precision-oriented stage 2. The following presents the full L3X pipeline. Baselines and L3X variants are derived from specific configurations (Sec 4) and given in experimental results (Sec 5).

2.1 Passage Retrieval

Long texts, like books, are chunked into short passages of 15 sentences, totaling up to 1000 characters. We create all overlapping passages (i.e., with shared sentences) to ensure that sentences with co-references stay connected to named entities in their proximity. Since books often contain extended direct speech, which may omit explicit speaker names, we enrich each passage with *mentions of people and locations* from the *preceding* 10 passages. This metadata annotation ensures that relevant named entity information from prior chunks remains accessible within the current passage.

On the large pool of enriched passages, indexed for efficient retrieval, we select the open-sourced and effective Contriever (Izacard et al., 2022), a BERT-based dense neural IR method fine-tuned on MS-MARCO dataset[1]. The query vector is con-

[1]https://github.com/facebookresearch/contriever

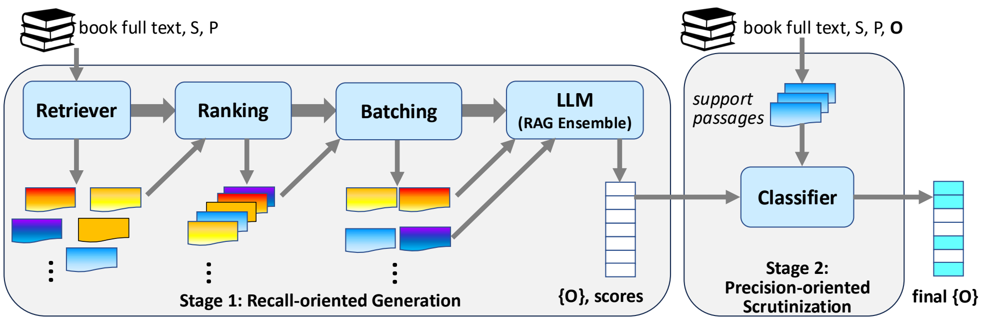


Figure 2: Overview of the L3X methodology.

structed from the SP pair; an example is: "enemies of Michael Corleone." Moreover, paraphrases of P and alias names are included, such as "opponents rivals Don Michael" for ensemble mode.

2.2 Passage Ranking

Default Ranking (def). The default passage ranking is given by the retriever scores. For a given SP pair, formulated as a natural language query, the dense retriever ranks top-d passages based on cosine similarity to the query vector.

Amplification (amp). For this novel *re-ranking* of passages, we employ the IR principle of *pseudo-relevance feedback* (Zhai, 2008). After extracting O lists from the initially selected passages (see Subsection 2.4), we assess the passage quality based on the number of distinct O's the passage yields. The best s passages are assumed to provide good cues about relation P in surface form (with hyper-parameter s). The averaged embedding vectors for these *high-yield passages* are the reference against which all retrieved passages are re-ranked. The *amp* technique works in two alternating steps and iterates them as follows:

1. For each SP pair, we consider the previously generated O values and the best s high-yield passages: those from which the LLM could extract the most objects.
2. All retrieved passages are re-ranked by the retriever's scoring model based on combining the original query (about SP) with the selected high-yield passages. The now highest-ranking passages go into the next batch of O extraction via LLM prompting, where each batch consists of b passages (e.g., $b = 4$).

For scoring, we utilize the retriever for computing cosine similarity of passages to a refined query: convex combination of the original query vector and the sum of top-s support passages' vectors: $\mathbf{E}(Q') = \alpha\mathbf{E}(Q) + (1-\alpha)\sum_{i=1}^{s}\mathbf{E}(S_i)$ with embeddings $\mathbf{E}(\,)$ and hyper-parameter α.

2.3 Passage Batching

For the high cost (or even infeasibility) of augmenting a large number of passages (e.g., all 500 retrieved ones) into an LLM, we group the passages into smaller batches of size b (a hyper-parameter; typical values being 2, 4, or 6), by default in descending ranking order. Alternatively, passages can be batched using one of two criteria:

- **Named Entity Overlap (neo):** passages with a large overlap in named entity mentions;
- **Passage Similarity (sim):** passages whose embeddings have a high cosine similarity.

For *neo*, we compute Jaccard similarity using min-hash sketches of entity sets, while *sim* uses embedding vectors computed by the retriever. Both strategies process a priority queue of passages as follows: for each rank r (starting with highest, r=1), find the b-1 most related passages from lower ranks (r'>r) to form a batch and prompt the LLM. Mark all the batch passages as "done" and proceed with the next lower rank (r'>r), which is not yet "done".

2.4 Prompt-based Object-List Generation

We append passages into the prompt for RAG-based list generation. As LLMs have limits on input context (and GPU memory demands increase with prompt length), we divide the top-k passages (ranked by retriever scores) into batches of b passages each (e.g., k=20, b=4 gives 5 batches). The O values generated from batch-wise processing are combined by their union for high recall.

Prompts can be zero-shot or few-shot, with the latter including a small set of demonstration examples for in-context inference. The examples explicitly mention SP appearing in books disjoint from the dataset, along with their complete O lists. In **single-prompt** mode, the LLM uses only the best of these formulations. In **ensemble** mode, for each relation, we manually prepare five prompt templates, and repeat the LLM-based generation with all templates. The final O list is the union of the O values generated across all runs. We focus on the **few-ensemble** setting for the main results.

2.5 Classifier to Enhance Precision

In the precision-oriented scrutinization stage, we leverage the fact that, unlike in the first stage, we have lists of candidate objects. This allows us to identify the passages from which the corresponding SPO triples were extracted or where they appear.

Scoring of O Candidates. Each LLM call returns a list with a score for the entire list, no scores for individual objects. However, with batch-wise LLM calls and the ensemble with different prompts, we can derive a total score for each O candidate (for a given SP), by a weighted occurrence frequency:

$$\text{score}(O) = \sum_{\text{batch}_i} \exp\left(\text{score}_{\text{M}}(L_i)\right) \times \mathbf{I}_i(O)$$

where $\mathbf{I}_i(O)$ is an indicator variable set to 1 if O occurs in the output list L_i for the ith batch of passages, and zero otherwise. score_{M} is the LLM log probability. This can then be used for direct pruning by thresholding on scores.

Default Thresholding (thr). The simplest scrutinizing technique is to prune O candidates below a a specified cut-off point in the ranked list of per-O scores. As the score distribution is often skewed, we do not truncate by score value, but set the cut-off point to be the t^{th} quantile of the cumulative score distribution, with the default setting t=0.8.

Support Passages as Evidence. While stage 1 starts with SP only, stage 2 has O candidates. This allows us to *search the full book* for snippets that contain cues for the entire SPO triple. For each SPO, we retrieve the top-p passages, termed *support passages*. These passages differ from the high-yield passages used by *amp* in stage 1, as they are retrieved afresh for each SPO. For retrieval, we generate passage embeddings using the retriever's text-to-vector model. The vectors are compared against embeddings of the concatenated SPO strings, including SO alias names and paraphrases of P, using cosine similarity.

Predicate Classifier (pred). The collection of support passages, for all SO with the same relation P, can be used to learn an embedding for P cues, sort of a "mini-LM" for P. The intuition is that support passages with indicative phrases, such as "life-or-death combat with", "deeply hates" or "I will destroy you" (in direct speech), can collectively encode a better signal for P. To construct the classifier, we perform the following steps:

1. For each O, retrieve top-p support passages, and encode them into embedding vectors.
2. Identify the top-ranked O values with score(O) above a threshold ω.
3. Using the top-ranked O, combine the per-O passage vectors by a weighted sum, with score(O) as weights, to obtain a single P-vector.
4. Each SO pair under scrutiny (O below the threshold ω) is tested by comparing the vector of the top-p support passages for this SPO candidate against the P-vector computed using steps 1 to 3.
5. Accept an SO pair if the cosine similarity between the embeddings is above threshold θ.

We construct a *pred* classifier for each SP pair, in a completely self-supervised manner. All classifiers share hyper-parameters ω, p and θ; these are tuned via withheld train/dev data with SPO ground-truth, but without any supervised passage labels.

3 Dataset

Extracting long O lists from long books is a new task, with no suitable datasets available. We constructed a new dataset of books and ground-truth O lists for SP pairs. We selected eleven book series[2], discussed on community websites[3]. These fan sites feature lists and infoboxes from which we derived SPO ground-truth with high confidence (with manual curation). Book length goes up to 10K passages in epic series like A Song of Ice and Fire.

Since entities often appear under multiple surface forms, we manually constructed an alias name dictionary. On a per-book basis, we ensured that certain first names, last names, or nicknames were uniquely identifiable, e.g., "Daenerys" is unique but "Targaryen" is ambiguous. So for this entity, aliases include "Dany", "Daenerys Targaryen", "Daenerys Stormborn", but not "Targaryen". LLM outputs like "Targaryen" alone are counted as false.

Our dataset comprises 764 distinct SP pairs for 8 relations. In total, it covers ca. 5,300 entities, referenced under ca. 12,000 alias names. While the S entities are prominent book characters, their associated O lists are long and dominated by rarely men-

[2]A Song of Ice and Fire Series, Godfather Series, Harry Potter Series, Outlander Series, Little Women, Malibu Rising, Pride and Prejudice, Steve Jobs, The Girl with the Dragon Tattoo, Wuthering Heights, The Void Trilogy

[3]www.cliffsnotes.com, www.bookcompanion.com, www.fandom.com

tioned, long-tail entities. To highlight the gap with standard RE, we examined the Wikidata knowledge graph (KG) for triples involving the 30 Harry Potter characters used as target S. While the KG includes most of our predicates, it lacks substance beyond metadata (e.g., featured-in-media, library IDs). It is also extremely sparse: it lists only 2 of Harry's enemies, compared to 50+ in our ground truth—a trend consistent across other subjects and relations.

Relation Difficulty. The chosen 8 predicates include 3 *easy relations* with a limited number of O values (parent, child, and sibling) and 5 *hard relations* with long O lists (family, friend, opponent, placeHasPerson (i.e., people being at a place), and hasMember (i.e., members of orgs. or events).

4 Experimental Setup

Evaluation Metrics. By the design rationale of L3X, we use different metrics for stage 1 and stage 2. For the recall-oriented stage 1, the obvious measure of interest is **Recall**: the fraction of ground-truth object (O) values correctly generated. For stage 2, neither precision nor recall alone reflect our objective, and F1 would merely be a generic compromise. Instead, we aim to achieve high recall while keeping precision at an acceptable level. Therefore, our key metric—computed from the final ranked lists— is **Recall@PrecisionX (R@Px)**, where x is the precision to be guaranteed (e.g., x being 50% or, ideally, 80%). R@Px metric reflects the need for high-coverage outputs worthwhile for downstream applications such as tool-supported literature analysis, while avoiding too many errors as these entail manual curation. For both stages, we also report *precision* values and the *precision-recall area under the curve*, *AUC*.

All reported numbers are *macro-averaged percentage* scores, computed in three steps. For each SP pair, we first compute the precision and recall of the generated O list against the ground-truth. These are then averaged across all SP pairs for each P. Finally, the results are averaged across all relations.

System Configurations and Baselines. The L3X methodology comes with options for components and configurations. For our main experiments, we operate in the few-ensemble prompt setting and focus on the following choices:

- **LLM-only**: directly prompting the LLM without passages (Subsection 2.4). For stage 2, we apply *thr* pruning with t=0.8.
- **RAG:** restricting L3X to the Retriever (Subsection 2.2) and LLM prompting, leads to standard RAG, with *def* passage ranking at stage 1 and *thr* (t=0.8) for stage 2.
- **L3X-amp-thr:** a configuration with *amp* for re-ranking, and *thr* (t=0.8) for pruning.
- **L3X-amp-pred:** a configuration with *amp* for re-ranking, and the *pred* classifier in two variants: *pred(g)* with globally tuned hyper-parameters and *pred(p)* with predicate-specific hyper-parameter values (see below).
- **L3X-amp-neo:** a configuration with *neo* batching and a pruning classifier (*thr* or *pred*).
- **L3X-amp-sim:** a configuration with *sim* batching and a pruning classifier (*thr* or *pred*).

Hyper-Parameters. L3X includes several tunable hyper-parameters; Optimal values are identified using withheld train/dev data. To this end, we split the entire dataset into three folds (30:20:50), via stratified sampling over books and SP pairs, ensuring equal representation of varying O-list lengths. For each S in train/dev, the complete O list is taken in the ground-truth to avoid information leakage into the test set. Hyper-parameters are tuned via grid search, maximizing the recall metric in stage 1 and R@P50 metric in stage 2. This is done in two modes: a single *global* value per hyper-parameter, or *per-predicate* values, specific to each P.

5 Results

We present the main findings on the long list generation task.

5.1 RQ1: Performance of LLMs and RAG

Table 1 reports macro-averaged results for the LLM-only setting with three widely used models (GPT-3.5[4], Llama3.1-8B, Llama3.1-70B[5]) and RAG results with the best of these (Llama3.1-70B). All are in few-shot ensemble mode, and all use *thr* (t=0.8) for stage 2.

LLM-only performance is poor, achieving less than 50% recall after stage 1, with mediocre precision. Llama-70B and GPT-3.5 perform comparably, while Llama-8B substantially lags behind. In RAG mode (with *def* ranking of passages), results improve: 84% recall after stage 1, but precision stays low even after *thr*-based scrutinization. The best R@P50 number is 40.2%. As a reference, we estimate an oracle upper bound of 88% by counting

[4] platform.openai.com/docs/models/gpt-3.5-turbo

[5] huggingface.co/meta-llama/Llama-3.1-8B, huggingface.co/meta-llama/Llama-3.1-70B

the distinct O values from ground truth that appear in at least one of the retrieved top-500 passages.

The insight here is that LLMs can recall only a fraction of O's from pre-training, and add many false positives. Equipped with book passages, the recall is improved, but false positives remain a major challenge for this very difficult task.

5.2 RQ2: Added Value of L3X Configurations

Table 2 compares different L3X configurations, contrasting them with Llama3.1-70B model in RAG mode. Adding smart re-ranking (*amp*) and batching to RAG pays off very well, and the sophisticated classifier (*pred*) also enhances scrutiny. After stage 1, the recall by L3X variants is similar to the RAG, but we observe a notable improvement in AUC, reaching 27.5%. This signals a higher concentration of true positives among the top-ranked O values—an important asset for stage 2.

The L3X *amp* method for iterative re-ranking achieves the biggest boost over the RAG baseline: moving R@P50 close to 50% and R@P80 to around 36%. Combining it with one of the two batching techniques does not add value, as *amp* by design is already judicious in picking its batches. Replacing the *thr* pruning with the sophisticated *pred* classifier further enhances the performance a bit. Again, drill-down by predicate shows higher gains for some of the hard P, indicating potential for more. The influence of hyper-parameter tuning for *pred* is discussed below.

The bottom line is that L3X *amp* adds substantial benefits over LLM-only and standard RAG methods, highlighting the crucial role of judicious passage ranking. The final R@P results—reflecting the benefit/cost ratio for downstream usage—are promising, but still fall short of being fully satisfying. This emphasizes the challenging nature of the new task explored in this paper.

Hyper-parameter Tuning for pred. The *pred* classifier has three hyper-parameters. Setting their values by global grid search with train/dev data in a self-supervised manner leads to the best results, with ω=20, p=5, and θ=0.75. As various P exhibit different characteristics, we would expect further gains with per-predicate grid search. Indeed, this led to rather different predicate-specific values, e.g., for Sibling, the best values are ω=10, p=2, θ=0.9, but for Friend we get ω=50, p=1, θ=0.55. Nevertheless, *pred(p)* did not achieve significant improvements over the globally tuned variant *pred(g)*. We attribute this to the fact that the simpler configurations are already close to the best possible outputs given the inherent difficulty of the task.

Comparison to Other Classifiers. We explored two alternative approaches for stage 2 scrutinization. First, we used the LLM itself to *elicit its own confidence* (Wang et al., 2023). For each SPO, we included the support passages along with all named entities into the prompt for in-context inference: "Given this information, is [SPO] a correct statement?". This approach performed poorly. For example, with the L3X *amp*, it has 46.6% precision, 44.7% recall, 19.0% AUC, 31.7% R@P50 and 20.4% R@P80.

Second, we evaluated *standard relational IE methods* that classify an SO pair to a given predicate P. We fine-tuned two state-of-the-art models—GenIE (Josifoski et al., 2022) and DREEAM (Ma et al., 2023)—on our train/dev folds. However, both models performed very poorly, with recall below 5% and precision no higher than 10%. This highlights the difficulty of our task: these models were trained on Wikipedia-style text, very different from long and complex fiction.

6 Discussion

6.1 Drill-Down and Sensitivity

Predicate Drill-Down. While results are macro-averaged over all relations, some predicates are easier than others (see Section 3). We analyzed performance per predicate using the best configurations after stage 1 and 2. Stage 1 recall is fairly consistent across predicates (75-90%), but stage 2 R@P numbers vary widely:"easy" relations with short, well-defined lists perform well, while "hard" relations—those with longer lists and vaguer cues—show a significant drop. As expected, Opponent is the most difficult predicate, where even our best method reaches only ca. 32% of R@P50. Full per-predicate scores after both stages is in Table 3.

Entity Popularity. We further analyzed performance by splitting ground-truth O entities in the test set into *head* and *tail* groups, based on their frequency in the book. Entities above the 75^{th} percentile were labeled as head, the rest as tail. This results in four combinations: (easy P, head), (easy P, tail), (hard P, head), and (hard P, tail). We observe that L3X-amp consistently outperforms standard RAG across all four cases. However, in the most challenging setting—hard P with tail O—performance drops sharply.

LLM	Config	Stage 1			Stage 2 (thr, t=0.8)				
		P	R	AUC	P	R	AUC	R@P50	R@P80
GPT-3.5	LLM-only	43.6	43.9	21.3	**41.5**	31.4	17.4	25.4	20.2
Llama-8B	LLM-only	21.2	31.9	11.5	21.2	27.8	10.5	16.9	12.4
Llama-70B	LLM-only	34.1	47.7	20.5	37.0	39.5	20.5	31.4	21.9
Llama-70B	RAG	12.0	84.3	22.9	14.6	**82.8**	**22.7**	**40.2**	**26.1**

Table 1: Results for LLM-only and RAG in few-shot ensemble mode.

L3X Config	Stage 1			Stage 2				
	P	R	AUC	P	R	AUC	R@P50	R@P80
RAG-thr	12.0	84.3	22.9	14.6	82.8	22.7	40.2	26.1
amp-thr	13.7	83.6	27.5	16.0	81.0	27.4	48.6	35.9
amp-neo-thr	14.1	83.4	27.1	16.7	**81.6**	26.9	47.7	35.4
amp-sim-thr	14.1	83.4	27.1	16.0	80.5	26.3	47.0	33.8
amp-pred(p)	13.7	83.6	27.5	**23.5**	77.3	28.0	48.7	36.2
amp-pred(g)	13.7	83.6	27.5	20.4	80.4	**28.1**	**49.7**	**36.5**
amp-neo-pred(p)	14.1	83.4	27.1	22.1	76.4	27.4	48.0	35.4
amp-neo-pred(g)	14.1	83.4	27.1	19.8	79.9	27.6	48.7	35.7

Table 2: Results for L3X configurations with Llama-70B in few-shot ensemble mode.

Relation	Stage 1						Stage 2							
	RAG			L3X-amp			RAG-pred(g)				L3X-amp-pred(g)			
	P	R	AUC	P	R	AUC	P	R	AUC	R@P50	P	R	AUC	R@P50
parent	25.9	75.6	25.5	29.8	76.2	26.0	38.2	71.4	27.3	57.7	42.7	76.2	27.9	61.3
children	20.7	86.5	27.6	19.6	82.5	32.1	33.7	83.9	28.1	60.9	30.7	82.0	36.5	72.3
sibling	26.9	87.2	34.9	36.6	86.2	47.7	38.2	85.8	38.0	65.4	45.3	85.2	47.7	79.4
avg. Easy P	24.5	83.1	29.3	28.7	81.6	35.3	36.7	80.4	31.1	61.3	39.6	81.1	37.3	71.0
family	5.5	79.8	25.2	5.1	79.8	33.3	11.5	77.2	25.2	34.0	12.4	76.3	33.1	44.8
friend	7.1	85.4	20.1	7.5	85.5	24.1	11.7	80.3	19.7	27.1	11.7	80.3	23.8	35.5
opponent	4.1	80.8	17.7	4.4	81.1	18.9	7.2	73.6	17.6	29.3	8.6	74.3	18.9	32.4
hasMember	2.6	89.0	16.5	2.8	86.6	20.8	5.5	82.1	16.5	25.7	5.5	83.4	20.7	32.5
placeHasPer	3.0	89.8	15.4	3.4	90.7	16.8	5.6	82.0	14.7	30.3	6.0	85.3	16.5	39.6
avg. Hard P	4.5	85.0	19.0	4.7	84.7	22.8	8.3	79.0	18.7	29.3	8.8	79.9	22.6	37.0
avg. All P	12.0	**84.3**	22.9	13.7	83.6	27.5	18.9	79.5	23.4	41.3	**20.4**	80.4	**28.1**	**49.7**

Table 3: Drill-Down Recall Results by Predicate for Stage 1 and Stage 2.

Role of LLM's Parametric Memory. To assess the influence of pre-training, we compared LLM-only to L3X-amp on the Void Trilogy—a series with minimal Web coverage, for which we invested great effort in compiling ground truth. Results show that LLM-only fails completely on this case: 12% recall and just 5% precision, whereas L3X-amp gets 82% recall after stage 1, and 34.1% R@P50 and 38.3% R@P80 with *thr* in stage 2.

Sensitivity of Hyper-Parameters. We conducted extensive experiments to assess the sensitivity of hyper-parameters, specifically the no. of top-k retrieved passages and batch size b. We observe that increasing k improves recall, but with diminishing returns and higher LLM cost. Batch size b matters more when k is large—larger batches boost recall by providing more context, but also increases prompt length and cost. Reducing the number of retriever query reformulations hurts both recall and R@P, highlighting the value of query diversity.

6.2 Error Cases.

We observed recurring error types and discuss three of the most notable cases.

Hallucinations. LLM calls often return huge lists of O's, including names that do not occur in the respective books. Even in RAG mode, the LLM does not necessarily restrict its outputs to entities present in the input passages—a case of *unfaithful* generation. To quantify the effect, we compute the no. of generated O's that do not appear in the respective book, normalized by the total no. of generated O values. Hallucination rates after stage 1 were: LLM-only: 55.3%, RAG: 51.7%, L3X-amp: 40.7%, L3X-amp-neo: 38.1%. Hallucinations include made-up names and non-entity phrases. This underlines the importance of stage 2 scrutinization.

Confusing Predicates. LLMs generate valid O values that are related to subject S, but under the wrong relation P. A notable case is when O belongs to a different ground-truth predicate Q ($\neq P$) (e.g., Dumbledore appearing as Harry Potter's parent instead of friend), We computed the #P$\times$#P confusion matrix, counting Os generated under P when their true relation is Q. With L3X-amp-pred(g), we observed a 60:30:10 ratio: correct TPs, predicate-confused TPs, and false positives (FPs). This shows that most SO pairs are reasonable, but predicate accuracy at high recall remains a challenge.

Missing True Positives in the Low Ranks. The majority of TPs are at high ranks, followed by a long tail of mostly FPs but sprinkled with TPs at lower ranks. To assess how well stage 2 recovers *low-ranked* TPs, we use the R@P50 cut-off rank to count the missing TPs below this threshold—those misclassified as false negatives. Even with our best methods, about 16% of all the ground-truth O values fall into this low-rank, missed-TP category.

7 Related Work

Relation Extraction. A core task in IE is extracting the relation P between two entities, subject S and object O, where P comes from a predefined set of predicates. State-of-the-art methods (Han et al., 2020; Wang et al., 2020; Cabot and Navigli, 2021; Josifoski et al., 2022; Ma et al., 2023) typically operate on single passages using a multi-label classifier or sequence tagger. Recent works (Zhao et al., 2024; Xu et al., 2024) have advanced the scope of the extractors' input under the theme of "long-distance IE", extending beyond single sentences or passages. However, techniques like graph neural networks or LLM-powered generative IE are geared for short news or encyclopedic texts, and cannot cope with book-length texts. Even the popular document-level benchmarks, DocRED (Yao et al., 2019) and REBEL (Cabot and Navigli, 2021), limit inputs to single Wikipedia paragraphs.

Retrieval-Augmented Generation. LLMs excel in QA and IE tasks by drawing from their extensive parametric knowledge (pretrained over massive contents), particularly with few-shot in-context inference (Zhao et al., 2023; Minaee et al., 2024). However, they are still susceptible to hallucinations, especially for long-tail entities and facts (Ji et al., 2022). To improve the overall task accuracy, recent work has focused on integrating relevant text snippets into in-context prompts through the RAG paradigm (Lewis et al., 2020; Guu et al., 2020; Cai et al., 2022; Asai et al., 2023; Wang et al., 2023; Gao et al., 2023). However, the effectiveness of RAG crucially depends on the retriever policy, and the case of long novels has not been studied so far.

Information Extraction from Books. Prior works (Bamman et al., 2019; Stammbach et al., 2022; Chang et al., 2023) pursue LLM-supported IE about characters from fiction books. But these methods focus on NER-like generation of single names from single passages. Bamman et al. (2024) considers usage of LLMs for cultural analytics, and Piper and Bagga (2024) shows LLMs for characterizing and annotating narrative texts. None of these works addresses full-fledged RE from entire books.

8 Conclusion

We introduced the task of extracting long lists of objects from long documents, and proposed the L3X methodology, comprising LLM prompting, retrieval augmentation, passage re-ranking and batching, and classifier-based pruning. Extensive experiments demonstrate that L3X significantly outperforms baselines in both recall and R@P. Our best performing L3X configuration *amp-pred(g)*, leveraging pseudo-relevance feedback and a tuned classifier, achieves remarkable performance of ca. 85% recall and ca. 37% R@P80 on full-length books. However, drill-down analyses by relation and entity popularity reveal substantial gaps in the hard cases. This highlights the core challenge of our task: while scattered textual cues across long books may be intuitive for humans, they remain difficult for AI systems, including LLMs, to reliably detect and extract.

References

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46, Toronto, Canada. Association for Computational Linguistics. Tutorial materials at "http://acl2023-retrieval-lm.github.io/".

David Bamman, Kent K Chang, Li Lucy, and Naitian Zhou. 2024. On classification with large language models in cultural analytics. *Computational Humanities Research Conference (CHR)*.

David Bamman, Sejal Popat, and Sheng Shen. 2019. An annotated dataset of literary entities. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2138–2144, Minneapolis, Minnesota. Association for Computational Linguistics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 2370–2381. Association for Computational Linguistics.

Deng Cai, Yan Wang, Lemao Liu, and Shuming Shi. 2022. Recent advances in retrieval-augmented text generation. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 3417–3419. ACM. Tutorial materials at "https://jcyk.github.io/RetGenTutorial/".

Kent Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. 2023. Speak, memory: An archaeology of books known to ChatGPT/GPT-4. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7312–7327, Singapore. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Xu Han, Tianyu Gao, Yankai Lin, Hao Peng, Yaoliang Yang, Chaojun Xiao, Zhiyuan Liu, Peng Li, Jie Zhou, and Maosong Sun. 2020. More data, more relations, more context and more openness: A review and outlook for relation extraction. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, AACL/IJCNLP 2020, Suzhou, China, December 4-7, 2020*, pages 745–758. Association for Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Wenliang Dai, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55:1 – 38.

Martin Josifoski, Nicola De Cao, Maxime Peyrard, Fabio Petroni, and Robert West. 2022. GenIE: Generative information extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4626–4643, Seattle, United States. Association for Computational Linguistics.

Mayank Kejriwal and Akarsh Nagaraj. 2024. Quantifying gender disparity in pre-modern english literature using natural language processing. *Journal of Data Science*, 22(1).

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Youmi Ma, An Wang, and Naoaki Okazaki. 2023. DREEAM: Guiding attention with evidence for improving document-level relation extraction. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1971–1983, Dubrovnik, Croatia. Association for Computational Linguistics.

Shervin Minaee, Tomás Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *CoRR*, abs/2402.06196.

Andrew Piper and Sunyam Bagga. 2024. Using large language models for understanding narrative discourse. In *Proceedings of the 6th Workshop on Narrative Understanding*, pages 37–46.

Andrew Piper, Richard Jean So, and David Bamman. 2021. Narrative theory for computational narrative understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana,* *Dominican Republic, 7-11 November, 2021*, pages 298–311. Association for Computational Linguistics.

Dominik Stammbach, Maria Antoniak, and Elliott Ash. 2022. Heroes, villains, and victims, and GPT-3: Automated extraction of character roles without training data. In *Proceedings of the 4th Workshop of Narrative Understanding (WNU2022)*, pages 47–56, Seattle, United States. Association for Computational Linguistics.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Jiayang Cheng, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, Yidong Wang, Linyi Yang, Jindong Wang, Xing Xie, Zheng Zhang, and Yue Zhang. 2023. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *CoRR*, abs/2310.07521.

Difeng Wang, Wei Hu, Ermei Cao, and Weijian Sun. 2020. Global-to-local neural networks for document-level relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3711–3721. Association for Computational Linguistics.

Matthew Wilkens, Elizabeth F Evans, Sandeep Soni, David Bamman, and Andrew Piper. 2024. Small worlds: Measuring the mobility of characters in english-language fiction. *Journal of Computational Literary Studies*, 3(1).

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024. Large language models for generative information extraction: a survey. *Frontiers Comput. Sci.*, 18(6):186357.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. Docred: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 764–777. Association for Computational Linguistics.

ChengXiang Zhai. 2008. Statistical language models for information retrieval: A critical review. *Found. Trends Inf. Retr.*, 2(3):137–213.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Xiaoyan Zhao, Yang Deng, Min Yang, Lingzhi Wang, Rui Zhang, Hong Cheng, Wai Lam, Ying Shen, and Ruifeng Xu. 2024. A comprehensive survey on relation extraction: Recent advances and new frontiers. *ACM Comput. Surv.*, 56(11):293:1–293:39.