Recall Them All: Retrieval-Augmented Language Models for Long Object List Extraction from Long Documents

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

Relation extraction methods from text often prioritize high precision but at the expense of recall. However, high recall is crucial for populating long lists of object entities that stand in a specific relation with a given subject. In long texts, cues for relevant objects can be spread across many passages, posing a challenge for extracting long lists. We present the L3X method which tackles this problem in two stages: (1) recall-oriented generation using a large language model with judicious techniques for retrieval augmentation, and (2) precision-oriented scrutinization to validate or prune candidates.

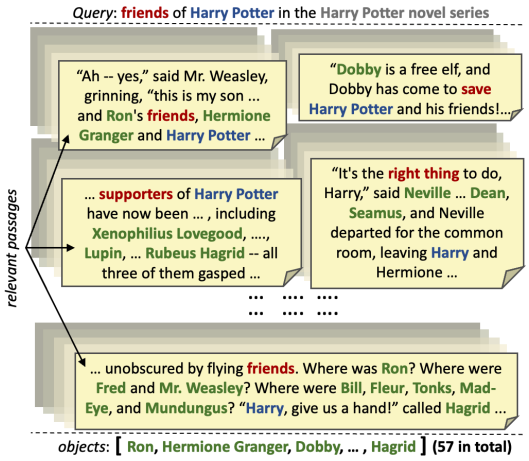

Figure 1: Example for extracting long lists from long texts. For the subject "Harry Potter", we aim to extract all 57 friends, appearing throughout the book series.

1 Introduction

Motivation and Problem. Information extraction (IE, for short) is the methodology for distilling structured information out of unstructured texts. Specifically, relation extraction aims to yield subject-predicate-object (*SPO*) triples where S and O are named entities that stand in a certain relation P. State-of-the-art methods are based on neural learning, and perform well in terms of precision but with limited recall (Han et al., 2020). Recently, large language models (LLM) have been studied for this task, with emphasis on long-tail facts, yet they exhibit similar deficits in recall (Kandpal et al., 2023; Veseli et al., 2023; Sun et al., 2023). Moreover, most methods are designed to operate only on single passages, as classifiers or sequence taggers.

This work addresses the underexplored and unsolved problem that IE faces with two *"longs"*: extracting a **long list** of object entities that stand in a certain relation to a subject, appearing in **long text**, such as entire books or websites with many pages. Examples for this open challenge would be extracting a complete list of (nearly) all acquisitions and subsidiaries of Alphabet Inc., identifying all artists who have covered Bob Dylan songs, or finding all friends of Harry Potter in the Harry Potter book series. Figure 1 illustrates this task.

The most challenging cases arise when a relation is expressed merely by soft cues in a few passages, scattered across an entire book. For instance, consider Harry's friend Neville: Figure 1 shows a weak cue—very unwieldy for established IE methods. Additional similar weak cues are spread throughout the books, and only by aggregating all of them can we confidently infer this friendship. This calls for novel methods regarding (i) retrieving informative passages, and (ii) performing inference over multiple passages.

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 the subject and relation at hand, and tasked to generate a full list of objects through various prompt formulations. In addition, we use information retrieval (IR) methods to find promising candidate passages from long texts and feed

them into the LLM prompts. In contrast to prior works on retrieval-augmented LLMs, we retrieve a large number of such passages (e.g., 500 for a given SP pair) and judiciously select the best ones for prompting. Moreover, our method iteratively re-ranks the passages and re-prompts the LLM, to improve recall of initial generation of objects.

Stage 2: Precision-oriented Scrutinization. Given a high-recall list of object candidates from Stage 1, the second stage uses conservative techniques to corroborate or prune objects. We employ novel techniques to identify high-confidence objects and their best support passages, and use them to re-assess lower-confidence candidates.

Since we are solving a new task, we curated two datasets, covering fiction books and web documents, respectively. The books dataset, which is our primary target, consists of 11 books or book series, with a total of 16,000 pages. It addresses 8 relations of long-tailed nature (incl. friends, opponents, placeHasPerson etc.). The second dataset comprises ca. 10 million web documents sampled from the C4 corpus (Dodge et al., 2021), focusing on 3 long-tailed factual relations (hasCEO, hasSubsidiary, and isMemberOf). Here, for each SP pair, we need to tap into many thousands of pages, which can be conceptualized as a single long text.

Due to the inherent trade-off between precision and recall, neither metric alone is suitable for our task, and F1 would merely be a generic compromise. The task instead requires maximizing recall, for an effect on knowledge graph (KG) population, with sufficiently high precision to keep downstream curation efforts manageable. Therefore, the metric that we aim to optimize is **Recall@PrecisionX (R@P$\mathtt{x}$)**, where $\mathtt{x}$ is the minimum precision target to be achieved (e.g., $\mathtt{x}$ being 50% or, ideally, 80%). In experiments with Llama3.1-instruct-70B (Meta, 2024) as underlying LLM, we reach 80-85% recall using our passage re-ranking and batching technique and ca. 50% R@P50 and 37% R@P80 through our scrutinization process.

The salient contributions of this work are: (1) the new task of extracting a long list of objects for a given subject and relation from long documents; (2) a methodology for this task, based on retrieval-augmented LLMs and combining IR techniques with LLM generation; (3) experiments with new benchmarks, showing that L3X outperforms LLM-only baselines that rely on parametric memory from their pre-training, and providing an in-depth analysis of strengths and limitations of different methods. The new datasets, code, and additional experimental results, will be open-sourced upon publication.

2 Related Work

Relation Extraction. A common task in IE is to extract the relation P that holds between two given entities, subject (S) and object (O), where P comes from a pre-specified set of possible predicates. State-of-the-art methods (e.g., (Han et al., 2020; Wang et al., 2020; Cabot and Navigli, 2021; Xie et al., 2022; Josifoski et al., 2022; Ma et al., 2023)) typically operate on single passages, as input to a multi-label classifier or sequence tagger.

Recent works (Zhao et al., 2024; Xu et al., 2024) have advanced the scope of the extractors' inputs under the theme of "long-distance IE", going beyond single sentences/passages. However, techniques like graph neural networks or LLM-powered generative IE are geared for short news or chats, and cannot cope with book-length texts. In the popular document-level benchmark DocRED (Yao et al., 2019), inputs are single paragraphs from Wikipedia. Aggregating cues from many passages (as required, e.g., for determining that Neville is Harry's friend) is out of scope. Moreover, these prior works assume texts for extraction are given upfront, or retrieved by matching S and O in proximity. In contrast, our long-list task takes S and P as input and seeks to generate previously unseen O as output. This changes the goal from high-precision classification to high-recall extraction.

OpenIE. OpenIE (Mausam, 2016; Stanovsky et al., 2018; Kolluru et al., 2022) is a variant where S, P and O are simply surface phrases without linkage to a knowledge base. While OpenIE may provide broader coverage across different relations, it is unsuitable for populating lists of crisp object entities for a given relation. Even when powered by distant supervision with (S,O) pairs, it remains limited to extraction from single sentences or short passages (e.g., (Smirnova and Cudré-Mauroux, 2019)).

LLMs as Knowledge Bases. Petroni et al. (2019) showed that LLM prompts can generate facts of knowledge-base style. The approach has been expanded and refined in various ways (e.g., (Jiang et al., 2020; Shin et al., 2020; Qin and Eisner, 2021; Chen et al., 2022)). These aim at precision, disregarding recall and the long tail. Recent studies indicate that LLMs have major problems in dealing

with long tail facts (Veseli et al., 2023; Sun et al., 2023; Singhania et al., 2023; Kandpal et al., 2023).

Retrieval-Augmented Generation. For better LLM generations, relevant text snippets can be retrieved and fed into in-context prompts, through the popularly known RAG paradigm (Lewis et al., 2020; Guu et al., 2020). The surveys (Cai et al., 2022; Asai et al., 2023; Wang et al., 2023; Gao et al., 2023) discuss RAG architectures for improving overall task accuracy.

Evidence and Factuality. LLMs can be harnessed to assess the factuality of statements (e.g., (Manakul et al., 2023; Min et al., 2023; Chern et al., 2023; Wang et al., 2023)). These techniques leverage external sources, such as Wikipedia articles, which is infeasible in our setting, where the focus is on long-tail entities within long (fictional) books.

IE from Books. Prior works by Bamman et al. (2019); Stammbach et al. (2022); Chang et al. (2023) pursue LLM-supported IE about characters from fiction books. However, these methods focus on generating a single name from a single passage.

3 Recall-oriented Generation

Our first, recall-oriented stage comprises several steps; Figure 2 (left side) provides an overview of the flow between the components.

1. **Retrieval** of a large pool of passages from the long text, using a dense retriever by searching with S and a set of paraphrases of P.
2. **Re-ranking** passages by various criteria. We present two techniques to prioritize passages based on (i) **num:** number of *named-entity mentions* in a passage, to leverage co-occurrences of multiple O values for the same predicate (e.g., a passage about several friends), and (ii) **amp:** pseudo-relevance feedback (Zhai, 2008) to *amplify* signals from best passages to refine the prompt for the next round.
3. **Batching** passages with (i) **neo:** similar entities (including their aliases) identified through named entity overlap, and (ii) **sim:** similar narratives via embeddings, to provide semantically coherent inputs to the LLM.
4. **Prompting** the LLM in *retrieval-augmented mode* using the top-ranked passages. The prompt explicitly includes the book title, S and P. For recall, this is an *ensemble* over different choices of retrieved passage (step 1.), and the output of this stage is the union of all objects generated by the LLM.

The first three steps are optional, enabling simpler configurations. Running only step 4 (without retrieved passages) results in an LLM-only/no-RAG variant, serving as a **direct prompting** baseline. Running only steps 1 and 4 produces a simplified variant of L3X, referred to as **def** (for default configuration), where passages are ranked by retriever scores and batched in the same order. Each of these steps is elaborated below. As Figure 2 shows, some of the steps can also be iterated; Section 3.2 discusses this for the *amp* technique.

3.1 Passage Retrieval

Long texts, like entire books, are chunked into short passages of 15 sentences, totaling up to 1000 characters. We construct all overlapping passages (i.e., sharing sentences), to ensure that sentences with coreferences stay implicitly connected to named entities in their proximity. Since books contain long pieces of direct speech, which may not mention all speakers' names explicitly, we further enrich each passage with *mentions of people and locations* from the *preceding* 10 passages by default. This metadata annotation ensures that useful named entity information from prior chunks is available within the current passage. All person-person relations and the placeHasPerson relation can potentially benefit from this form of contextualization.

On the large pool of enriched passages, indexed for efficient retrieval, we experimented with several retrieval models, including BM25 (Robertson and Zaragoza, 2009), Contriever (Izacard et al., 2022), and techniques based on OpenAI embeddings. Among the best-performing, we selected Contriever, a BERT-based dense neural IR method fine-tuned on MS-MARCO (Izacard et al., 2022), with the added benefit of being open-source[1] and deployable under our full control. For brevity, we report only experiments with Contriever.

3.2 Passage Ranking

The default passage ranking is directly derived from retriever scores. Additionally, we propose two *re-ranking heuristics* to enhance the richness and aptness of top-k passages.

Default Ranking (def): For a given SP pair, formulated as a natural language query, the dense

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

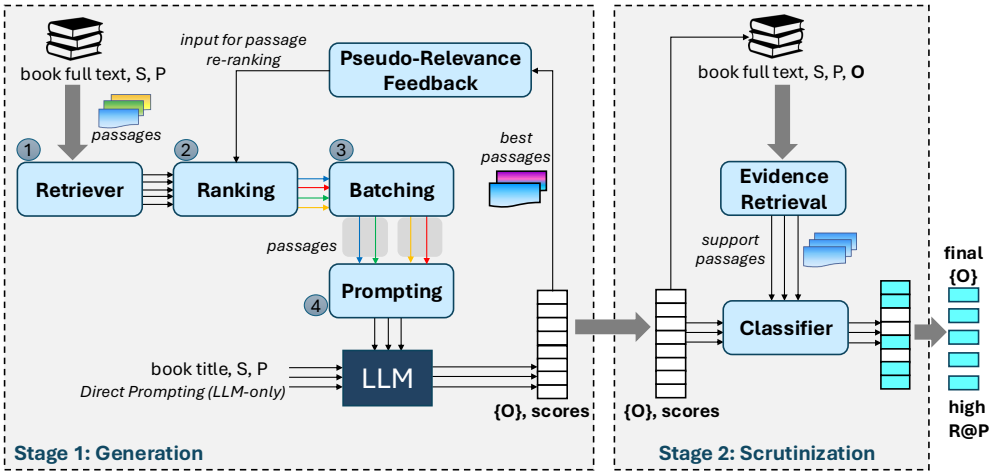

Figure 2: Overview of the L3X methodology.

retriever ranks top-d passages based on cosine similarity to the query vector.

Entity Mention Frequency (num): *re-ordering passages by frequency of named-entity mentions.* We detect mentions of entities (without disambiguation) of the proper type (usually person, place, or org) using spaCy and a hand-crafted dictionary of alias names for S and O, and paraphrases for P (incl. both nominal and verbal phrases). Top-m passages with higher counts of mentions are prioritized, as they could potentially yield multiple O candidates.

Amplification (amp): *selecting support passages and re-ranking passages by pseudo-relevance feedback.* After extracting object lists from the initially selected passages, we assess the passage quality based on the no. of objects they yield. The best passages, termed *support passages*, are assumed to provide good cues about predicate P in surface form. Following the rationale of pseudo-relevance feedback (Zhai, 2008), these support passages are fed back to the retriever for refined scoring, based on embedding similarity between the candidate pool and the support passages.

The *amp* technique works in two steps and iterates them as follows:

1. For each SP pair, we consider the previously generated O values and the best s support passages: those from which the LLM could extract the most objects (optionally weighted by extraction scores).
2. All passages in the current pool are re-ranked by the retriever's scoring model based on combining the original query (about S and P) with the selected support passages. The now highest-ranking passages are then used for the next round of O extraction by prompting the LLM.

Steps 1 and 2 are iterated in an alternating manner. For scoring by the embedding-based (dense) retriever, the refined query is a convex combination of the original query embedding and the sum of the top-s support passages' vectors:

$$\mathbf{E}(Q') = \alpha\mathbf{E}(Q) + (1-\alpha)\sum_{i=1}^{s}\mathbf{E}(S_i)$$

with embedding function **E( )** and hyper-parameter α. Algorithm 1 gives pseudo-code for *amp*.

3.3 Passage Batching

To feed passages into the LLM, the default approach combines successive ranks into small batches, as determined by the (re-)ranker. Alternatively, we can group or batch passages (Fan et al., 2024) based on coherent story structures to aid the LLM in extracting O values. We devise two criteria for this purpose and batch:

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

For neo, we compute Jaccard similarity using minhash sketches of named entities, while sim uses the OpenAI text-embedding model. Both techniques process a priority queue of passages as follows: for each rank r (starting with the highest rank, 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".

Algorithm 1: Iterative Extraction with Pseudo-Relevance Feedback (amp Method).

Input: $\mathcal{C}$: candidate pool of retrieved passages; Q: retriever query in natural language with SP mentions; k: max no. of passages for prompting LLM; b: batch size; s: no. of support passages; α: feedback weight for query reformulation; E: retriever's embedding function

Output: List of object values $\mathcal{O}$

Initialize: $\mathcal{O} \leftarrow \emptyset$; $q \leftarrow \text{E}(Q)$; //embedding vector

for $i \leftarrow 1$ *to* $\lceil k/b \rceil$ **do**

- $\mathcal{K} \leftarrow \text{Retriever}(\mathcal{C}, q)$; //ranking $\mathcal{C}$ for top-k passages
- $p_b \leftarrow \text{Batching}(\mathcal{K})$; //b passages by def or by neo/sim (Section 3.3)
- $\mathcal{O}_b \leftarrow \text{LLM}(p_b)$; //extracting objects by prompting LLM with passages p_b
- $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{O}_b$;
- $\mathcal{S}_b \leftarrow \emptyset$;
- **foreach** *passage* $p \in p_b$ **do**
  - **if** $o \in \mathcal{O}_b$ *appears in* p **then**
    - $\mathcal{S}_b \leftarrow \mathcal{S}_b \cup p$; //finding support passages from passages p_b
- $\mathcal{S} \leftarrow \text{Top}(\mathcal{S}_b)$; //selecting top-s passages by #objects
- $q \leftarrow \alpha \cdot q + (1-\alpha) \cdot \frac{1}{s} \sum_{p \in \mathcal{S}} \mathbf{E}(p)$; //convex combination to rerank $\mathcal{C}$

return $\mathcal{O}$

3.4 Prompt-based Object Generation

The retrieved top-k passages mention the subject in some form (e.g., first name, last name, or alias) and may contain other named entities. We append the passages into the prompt context for retrieval-augmented list generation (Liu et al., 2023; Gao et al., 2023; Zhao et al., 2023). As LLMs have limits on input context (and GPU memory demands increase with input 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 optionally include a small set of demonstration examples for in-context inference. These examples explicitly mention SP appearing in books disjoint from the dataset, along with their complete O lists, aiding the instruction-tuned LLM in object list generation. We refer to this mode as **few-shot** prompting, while the basic mode without examples is referred to as **zero-shot**. Table 1 shows an example for the few-shot prompt formulations.

In **single-prompt** mode, the LLM uses only the best of these formulations (i.e., considered most natural by humans). In **ensemble** mode, for each relation, we manually prepare five prompt templates for direct prompting, and five retriever query templates for L3X-RAG, and repeat all LLM-based extraction tasks with all templates. The final O is the union of the O values generated across all runs.

Of the four configurations (zero-single, zero-ensemble, few-single, few-ensemble), we report main results for the **few-ensemble** setting, with the other configurations evaluated in ablation studies.

System:- You are a knowledge base. Generate the complete list of names (Objects) who are parents, including step parents, of the specified person in the given book. List the names one after the other, separated by commas.
Few-Shot examples:
Input: Book: A Promised Land, Subject: Barack Obama, Relation: parent Output: [Barack Obama Senior, Stanley Ann Dunham]
Input: Book: The Fellowship of the Ring, Subject: Frodo Baggins, Relation: parent Output: [Drogo Baggins, Primula Brandybuck]
User:- Use the attached passages from the book. Book:{B}, Subject:{S}, Relation:{P}, Passages: {T}

Table 1: Example of prompt template for Parent relation (placeholders in curly brackets).

4 Precision-oriented Scrutinization

To scrutinize the candidate objects O for a given SP and eliminate false positives, we devise several techniques. The key idea is to identify passages that clearly reflect SPO triples, and use these *support passages* to rank and prune O values, and also learn embeddings for the P predicates. Figure 2 (right side) gives a pictorial overview. From the first stage, each batch of passages yields an LLM-generated score for the output list of O values. The total score for this O (for a given SP) can be computed as a weighted occurrence frequency:

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

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$_{\text{LLM}}$ is the log probability. This scoring serves as a simple baseline for pruning doubtful O values.

4.1 Evidence Retrieval

While stage 1 needs to start the retrieval with S and P only, stage 2 has O candidates at its disposal. This allows us to search the entire book for textual snippets that explicitly indicate SPO triples. For each SPO candidate, we retrieve the top-s passages, termed the *support passages* for SPO. Note that these are different from the support passages used by the *amp* method in stage 1, as we now retrieve from scratch from the entire book.

To retrieve the support passages, we use the OpenAI text-embedding model to generate passage embeddings. These are compared against embeddings of the concatenated SPO strings, including SO alias names and paraphrases of P, using cosine similarity.

4.2 Classifiers

We devise several classifiers to scrutinize O values.

Score-based Thresholding (thr): As a baseline without support passages, the O candidates for a given SP are ranked using $\texttt{score}(O)$. We accept those that fall within the t^{th} quantile (e.g., $t = 0.8$) of the cumulative score distribution.

Confidence Elicitation (conf): We prompt the LLM again to assess its confidence in the generated O values. For each *SPO*, top-p support passages in their enriched form (with all named entities incl. S and O) are included into the LLM prompt for in-context inference: "Given this information, is *SPO* a correct statement?". The *conf* classifier accepts an O candidate if the LLM gives a "yes" reply. This approach differs from the passage-based extraction of the recall-oriented stage, as the support passages are retrieved individually for each O-candidate.

Predicate-specific Classifier (pred): The collection of support passages, for all SO with the same predicate P, can be utilized 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, we retrieve top-p support passages, and encode them into embedding vectors.
2. We identify the top-ranked O values with $\texttt{score}(O)$ above a threshold ω.
3. Using the top-ranked O, we combine the per-O passage vectors by a weighted sum, with $\texttt{score}(O)$ as weights, to obtain a single P-vector (classifier).
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. The classifier accepts a low-ranked SO if the cosine similarity between the embeddings is above a threshold θ.

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

Discriminative Classifier (dis): Another way of harnessing the SPO support passages is to train a discriminative classifier, again in a self-supervised manner. We consider the ranked list of O values for a given SP and pick:

- the top-q high-scoring O candidates
- the bottom-r low-scoring O candidates

with q and r as hyper-parameters. For each top-q and bottom-r candidate O, we retrieve top-p support passages, forming one passage pool for the high-scoring Os and another pool for the low-scoring Os. In each of these pools, the passages are cast into embeddings, and weighted averaged with $\texttt{score}(O)$ to form SP_{high} and SP_{low} vectors.

Finally, each candidate O for a given SP is classified by whether its own support-passage vector is closer to the SP_{high} or the SP_{low} vector, in terms of cosine distance, leading to acceptance or rejection, respectively.

5 Experimental Setup

5.1 Datasets

We make use of *fiction books* as a most representative, primary target for experimental studies. A second dataset, on *web contents with business-oriented relations*, exhibits different characteristics, and adds diversity to the experiments. The results go into more depth and variety on the books data, and are shorter on the complementary web data.

Books Data. The task of extracting long O lists from long texts is novel, with no suitable benchmark datasets available. Therefore, we constructed a new dataset of books and corresponding ground-truth O lists associated with SP pairs.

We selected eleven popular novels and entire book series[2], enthusiastically discussed on community websites[3]. These fan communities feature extensive lists and infoboxes from which we derived SPO ground-truth with high confidence. As detailed in Sec 3.1, the total no. of passages per book varies, ranging from ca. 10,000 passages in epic book series like A Song of Ice and Fire to ca. 700 passages in shorter books like Malibu Rising.

Since entities often appear under multiple surface forms, we manually constructed an entity name dictionary grouping alias names for each distinct entity. On a per-book basis, we ensured that certain first names, last names, or nicknames were uniquely identifiable. For example, "Daenerys" is unique, whereas "Targaryen" is ambiguous. So for this entity, aliases include "Daenerys", "Dany", "Daenerys Targaryen", "Daenerys Stormborn", but not "Targaryen". LLM outputs like "Targaryen" alone are thus counted as false. This construction was aided by additional community sources[4].

This dataset comprises 764 distinct SP pairs for 8 predicates. In total, it covers ca. 5300 entities that appear under ca. 12,000 alias names. The S entities are typically prominent characters in the books, but they are associated with long O lists mostly consisting of rarely mentioned long-tail entities.

Relation Difficulty. The chosen 8 predicates has 3 *easier relations* with a relatively limited no. of O values (parent, child, and sibling) and 5 *harder relations* with potentially long O lists (family, friend, opponent, placeHasPerson—i.e., people being at a place, and hasMember—i.e., members of org. or events). Table 2 gives statistics for our dataset.

Web Data. To demonstrate the generalizability of L3X, we constructed a second dataset with partly similar and partly complementary characteristics. The data is derived from the large common crawl of web pages (C4 corpus) (Dodge et al., 2021), and we aim to extract long object lists for three business/biography relations: CEOs of companies (incl. past ones), subsidiaries of companies, and organizations that a famous person is part of (e.g., companies, societies, charities, schools at different levels). The dataset has 100 subjects for each P, covering ca. 6400 entities appearing under ca. 24,000 alias names. Table 3 gives per-relation statistics.

Relation	Type	#S	#O per S	
			range	μ (σ)
parent	pers→pers	85	1–4	1.9 (0.6)
child	pers→pers	48	1–9	3.3 (2.4)
sibling	pers→pers	65	1–8	3.0 (1.8)
family	pers→pers	81	1–47	12.1 (9.8)
friend	pers→pers	99	1–85	11.1 (16.5)
opponent	pers→pers	88	1–60	8.9 (11.2)
placeHasPer	loc→pers	189	1–92	6.7 (12.6)
hasMember	org→pers	109	1–142	11.6 (20.5)

Table 2: Books Dataset Statistics. **#S** denotes the no. of unique subjects and **#O** is the no. of objects per subject.

Relation	Type	#S	#O per S	
			range	μ (σ)
hasCEO	org→pers	100	2-43	8.5 (5.2)
isMemberOf	pers→org	100	3-74	20.1 (14.7)
hasSubsidiary	org→org	100	2-308	71.8 (52.0)

Table 3: Web Dataset Statistics.

5.2 System Configurations

The presented methodology comes with many options for its different components, with the most important choices occurring for ranking, batching and scrutinization. In our experiments, we focus on configurations for these three components, labeling them accordingly, e.g., as *num/sim/thr* or *amp/neo/pred*. The *thr* classifier with default setting $t = 80$ is abbreviated as *thr80*.

Hyper-Parameters. The L3X framework comes with tunable hyper-parameters; Table 4 lists the default values for most experiments. We widely varied these settings, reporting notable cases in Section 6.4 on sensitivity studies.

The best settings were identified using withheld train/dev data. To this end, we split the entire dataset into two folds (50:50), through stratified sampling on books and SP pairs, ensuring equal representation of varying O-list lengths in both folds. For each subject in train/dev, the complete O list is taken from the ground truth, to prevent information leakage into the test set.

[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

[4] incl. potterdb.com for Harry Potter, www.reddit.com/r/asoiaf for Song of Ice and Fire, and others

The best hyper-parameter values are determined through grid search, maximizing the recall metric in Stage 1 and the R@P50 metric in Stage 2 (or alternatively AUC). This is done for two modes: a single *global* value for a hyper-parameter, or *per-predicate* values, specific for each P.

Evaluation Metrics. To obtain insights into the precision-recall trade-off and to assess the end-to-end goal of populating a knowledge base with high-quality SPO triples, the primary metric of interest is *Recall@Precision (R@P)*, focusing on the most interesting cases of R@P50 and R@P80 (50% and 80% correctness). Additionally, we report precision and recall, both before and after scrutinization. To further reflect on the inherent trade-off between precision and recall, we also compute the *area under the curve (AUC)* of the precision-recall curve. All reported numbers are *macro-averaged percentage* scores, computed as follows:

1. For each SP pair, we consider the resulting O list and compute the list's precision and recall.
2. We average the numbers over SP pairs for P.
3. We average the numbers over predicates P.

Note that there is only a mild imbalance between different predicates and our calculation is based on the respective number of SP pairs rather than total O counts. As a secondary metric we also compute *micro-averages* over all SP pairs and their O lists, regardless of the predicate, and report the findings.

Ground-Truth Variants. As detailed in Section 5.1, the ground-truth O lists are derived from online sources with high quality control. Inevitably, such lists are often incomplete, particularly for books with hundreds of minor characters. Therefore, we also evaluate by *pooling-based ground truth*, where the true positives within the union of all O values returned by all the methods is the complete ground truth.

6 Experimental Results

We first present our findings on the books dataset, as this is the primary representative of our setting—see Subsections 6.1 and 6.2. Results for the web dataset are given in Subsection 6.3. Due to space constraints, we restrict the presented experiments to the most notable configurations, with additional discussions on hyper-parameter sensitivity and error cases in Subsections 6.4 and 6.5. In all tables, the best value for a metric is in boldface, and row-wise best values are shaded.

Parameter	Default
l: passage length (#char)	1000
passage overlap (#char)	200
d: # retrieved passages	500
k: top-k passages	40
b: # passages per batch	2
m: # passages in *num*	50
s: # support passages in *amp*	2
α: feedback weight for *amp*	0.7
t: percentile retained for *thr*	0.8
p: top-p support passages for *conf*	2
ω: cut-off score for O values in *pred*	50
p: top-p passages for *pred* and *dis*	5
θ: acceptance bound for *pred*	0.85
q: top-q O candidates for *dis*	50
r: bottom-r O candidates for *dis*	50

Table 4: Hyper-Parameters for L3X

6.1 Main Findings for Books Data

6.1.1 Stage 1: Recall-oriented Extractor.

Table 5 reports macro-averaged results for different configs of LLM-only and L3X-RAG generations. The table shows both Stage 1 and Stage 2 results, but for Stage 2 all methods employ the basic *thr* classifier with default hyper-parameter t=0.8.

The results clearly show the superiority of L3X over LLM-only, with major gains in recall (by Stage 1) and substantial improvements for R@P in Stage 2, reaching almost 50% for R@P50. We make the following key observations:

Baselines: LLM-only methods perform poorly, even as a few-shot ensemble. Stage 1 recall saturates near 50%, with AUC reaching ca. 20%.

L3X at Stage 1: All L3X configurations greatly improve recall, up to almost 85%. To calibrate these numbers, we also determined an oracle-based upper bound: given all retrieved passages (top-d=500), how many ground-truth objects do actually appear in at least one of these passages. For the books dataset, this oracle suggests a recall of 88%. So our results are very close to what can be spotted at all, with a reasonable retriever budget.

The default configuration *def* already achieves competitive performance, and the best option for recall is the *num* method with entity-count-based re-ranking. However, in terms of AUC, the method that shines most is the iterative *amp*, leading to an AUC value of 27.5%.

Best L3X Config at Stage 2: The strong AUC of *amp* is a strong starting point for the *thr* classifier at Stage 2, where it achieves the best R@P values, with almost 50% for R@P50—with a large margin over the second-best method.

	Method	Stage 1			Stage 2 (thr80)				
		P	R	AUC	P	R	AUC	R@P50	R@P80
LLM-only	zero-single	41.9	38.7	15.8	42.6	28.6	15.0	19.6	15.9
	few-single	38.5	43.4	18.2	39.1	31.9	16.9	23.7	17.6
	zero-ensemble	32.1	49.6	20.2	34.2	41.9	19.8	29.4	21.3
	few-ensemble	34.1	47.7	20.5	37.0	39.5	20.5	31.4	21.9
L3X (few-ensemble)	def	12.0	84.3	22.9	14.6	82.8	22.7	40.2	26.1
	+num	11.8	**85.2**	21.8	14.8	**83.1**	22.7	39.7	24.8
	+amp	13.7	83.6	**27.5**	16.0	81.0	**27.4**	**48.6**	**35.9**
	+neo	12.6	83.8	22.4	15.5	81.6	23.0	39.1	25.1
	+sim	12.5	84.2	22.2	15.3	82.5	23.2	39.4	23.5
	+num +neo	12.9	85.0	22.3	15.2	81.9	22.8	38.0	25.0
	+amp +neo	**14.1**	83.4	27.1	**16.7**	81.6	26.9	47.7	35.4
	+num +sim	12.1	83.8	21.8	14.7	81.5	22.4	38.0	24.8
	+amp +sim	**14.1**	83.4	27.1	16.0	80.5	26.3	47.0	33.8

Table 5: Results (%) for L3X Stage 1 Configurations on Books Data (ranking:{num, amp}; batching:{neo, sim}; top-k=40; default thresholding ($t = 0.8$) for Stage 2)

Influence of Batching: When combined with batching via *neo* or *sim*, L3X improves in precision, but loses recall, and eventually stays inferior to *amp* alone. While unexpected, it is not counter-intuitive: *amp* operates iteratively, and judiciously picks its batch of passages in each round. So batching does help, but the big mileage already comes from the amplification of support passages.

Micro-averaging: With micro-averaging rather than macro-averages, the relative gains/losses across configs hardly change. The best results are still with *amp*, reaching 84.7% recall and 24.9% AUC in Stage 1, and 44.7% R@P50 and 31.6% R@P80 in Stage 2.

Pooling-based Ground Truth: In this evaluation mode, stage-1 recall by *def* and *amp* increases by ca. 10 points, reaching 93% and 92%, respectively. This is intuitive, as pooled ground truth is a subset of the fully hand-crafted ground truth. The numerical gains carry over to Stage 2: with *thr80*, *amp* goes up to 56.5% R@P50 and 41.4% R@P80.

Influence of LLM Pre-Training Most books in our data are well discussed in online media (incl. movie/TV adaptations). The LLM implicitly taps into this contents by its parametric memory. To assess the influence of LLM pre-training, we evaluate LLM-only vs. L3X *amp* for one of our books, the Void Trilogy. This book series is barely covered on the Web; we invested great effort for compiling its ground truth. The results show that LLM-only fails completely on this case: 12% recall with poor precision of ca. 5%, whereas our RAG-based *amp* gets 82% recall after Stage 1, and 34.1% R@P50 and 38.3% R@P80 with default *thr80* in Stage 2.

6.1.2 Stage 2: Precision-oriented Scrutinizer.

In the following, the presentation is restricted to a small subset of the best performing stage-1 configurations, namely, *amp* and *amp/neo*, plus the default L3X-RAG *def* for contrast. Table 6 shows the stage-2 results with different classifiers for scrutinization. We highlight the following key findings:

Best Configurations pred and dis: The basic *thr* technique already works fairly well, especially with tuning of its hyper-parameter t. The more sophisticated classifiers *pred* and *dis* still have an added benefit, improving the final R@P values by another 1%, going up to 49.7% for R@P50.

Hyper-parameter Tuning for pred: The *pred* method has three hyper-parameters. Setting their values by global grid search with train/dev data leads to the best results, with ω=20, p=5, and θ=0.75. As the various P exhibit different characteristics, we would expected even further gains with the per-predicate grid search. Indeed, this led to rather different predicate-specifc values. For example, for Sibling, the best values are ω=10, p=2, θ=0.9, whereas for Friend we have ω=50, p=1, θ=0.55. This makes sense, as Sibling lists are much shorter, and long Friend lists have much noisier support passages. So the setting is tighter for Sibling, and has more liberal θ for Friend but only 1 support passage per O to tame noise. Nevertheless, the *pred* p and *dis* p techniques did not achieve significant improvements over the globally tuned variants *pred* g and *dis* g. We attribute this result to the fact that the simpler configurations are already close to the best possible outputs (due to the difficulty of the task).

L3X		P	AUC	R@P50	R@P80	R@P90
def	thr g	14.6	22.7	40.2	26.1	24.1
	thr p	16.5	22.1	39.5	25.9	23.9
	conf	43.6	18.2	28.4	18.4	17.8
	pred g	18.9	23.4	41.3	26.6	24.8
	pred p	21.5	22.9	40.7	26.5	24.7
	dis g	20.2	23.1	41.2	26.6	24.8
	dis p	21.4	21.8	40.1	26.2	24.7
amp	thr g	16.4	27.3	48.5	35.8	32.9
	thr p	17.5	27.2	48.5	35.8	32.9
	conf	**46.6**	19.0	31.7	20.4	19.3
	pred g	20.4	**28.1**	**49.7**	**36.5**	**33.3**
	pred p	23.5	28.0	48.7	36.2	33.0
	dis g	21.5	27.9	**49.7**	**36.5**	**33.3**
	dis p	21.2	27.5	49.6	36.4	**33.3**
amp/neo	thr g	16.4	26.9	47.7	35.4	33.1
	thr p	17.4	26.7	47.5	35.3	33.0
	conf	45.4	18.7	30.7	20.1	19.6
	pred g	19.8	27.6	48.7	35.7	33.1
	pred p	22.1	27.4	48.0	35.4	32.9
	dis g	20.7	27.4	48.7	35.7	33.1
	dis p	20.9	26.2	48.0	35.4	32.8

Table 6: Results (%) for L3X Stage 2 Configurations (classifiers: {thr, conf, pred, dis}, g refers to global grid search and p is per-predicate grid search).

Confidence Elicitation from LLM: *conf* performs poorly, as the LLM becomes rather conservative when fed with support passages about SPO candidates, rejecting too many valid entries.

Comparison to Relational IE. As Stage 2 processes full SPO triples, we can apply relational IE for classifying whether an SO pair satisfies P. We considered two state-of-the-art methods, GenIE (Josifoski et al., 2022) and DREEAM (Ma et al., 2023), and tuned them on our predicates with the train/dev fold. Still, both performed very poorly, reaching less than 5% recall and precision 10% at best. Clearly, our task is outside their comfort zone, as passages from fiction books are very different from Wikipedia paragraphs for which these methods were originally designed. This underlines the uniqueness and challenging nature of the proposed long-lists-from-long-documents task.

6.2 Analysis by Drill-Down

The reported results are macro-averaged over all relations. However, some relations P are easier to deal with than others (see Sec 5). Table 7 shows results with drill-down by P, comparing the best configs. after Stage 1 and Stage 2, respectively.

For stage-1 recall, we achieve similar performance across predicates, all between ca. 75% and 90% recall. For stage-2 R@P results, we clearly see the big gap between "easy" relations with crips lists of a few objects, and "hard" relations with long lists and more loosely defined cues in the text passages. Not surprisingly, the Opponent relation is the hardest case, where even our best method reaches only ca. 30% for R@P50. This calls for more research on this challenging task.

Entity Popularity. For drill-down analysis, we partition the test set ground-truth O entities into *head* and *tail* groups. Based on the frequency of each unique O in a book, we define the head as entities appearing above the 75th percentile and the tail as those below it. Table 8 shows these results on four combinations (easy P, head), (easy P, tail), (hard P, head), and (hard P, tail). We observe that *amp* still outperforms *def* for all four cases. However, for the most challenging case of hard P and tail O, the absolute numbers degrade strongly. This highlights the complexity of the task and the need for further research.

6.3 Findings for Web Data

Table 9 shows results for the Web dataset, with different stage-1 configurations, and default *thr80* for scrutinization. By and large, the results reconfirm our key findings with the Books data: LLM-only methods are far inferior; all L3X-RAG methods boost recall; the *amp* has the best AUC after Stage 1 and is the overall winner for R@P after Stage 2. The oracle-based upper bound here is 65% for recall from top-d=500 passages.

A significant difference to the Books data, however, is that all absolute values are substantially lower here (incl. the oracle). For example, the best R@P50 values (by *amp*) are around 30%, compared to almost 50% for the books experiment. The explanation clearly is that the dataset itself is even more challenging: the ground-truth lists of objects are even longer, with even more long-tail entities, and they are spread across a very large number of web pages—a situation as if all pages (about the same S) were concatenated into a very long, highly incoherent document).

Table 10 compares stage-2 classifiers, along with drill-down by the three predicates. The trends align with those observed in the books dataset, with *amp/pred g* achieving the highest scores. While performance is strong on the relatively easier CEO relation (ca. 65% R@P50), it struggles on the highly challenging hasSubsidiary relation.

L3X	stage 1: recall			def/pred-g			amp/pred-g			amp/neo/pred-g		
	def	amp	amp/neo	AUC	R@P50	R@P80	AUC	R@P50	R@P80	AUC	R@P50	R@P80
parent	75.6	76.2	73.8	27.3	57.7	47.0	27.9	61.3	52.4	27.9	61.3	53.6
children	86.5	82.5	84.6	28.1	60.9	43.8	36.5	72.3	61.0	34.4	66.1	55.2
sibling	87.2	86.2	89.3	38.0	65.4	50.4	47.7	79.4	67.7	46.3	77.9	66.0
avg. Easy P	83.1	81.6	82.6	31.1	61.3	47.1	37.3	71.0	60.3	36.2	68.4	58.3
family	79.8	79.8	78.5	25.2	34.0	15.0	33.1	44.8	33.2	32.9	46.2	32.1
friend	85.4	85.5	85.2	19.7	27.1	13.1	23.8	35.5	17.0	23.2	35.5	18.6
opponent	80.8	81.1	80.1	17.6	29.3	14.5	18.9	32.4	14.6	18.8	30.9	14.2
hasMember	89.0	86.6	86.1	16.5	25.7	14.0	20.7	32.5	20.9	21.1	36.5	20.9
placeHasPer	89.8	90.7	89.4	14.7	30.3	15.3	16.5	39.6	25.0	16.1	35.6	24.8
avg. Hard P	85.0	84.7	83.9	18.7	29.3	14.4	22.6	37.0	22.1	22.4	36.9	22.1
avg. All P	**84.3**	83.6	83.4	23.4	41.3	26.6	**28.1**	**49.7**	**36.5**	27.6	48.7	35.7

Table 7: Drill-Down Results by Predicate for Books Dataset

L3X		Stage 1			Stage 2 (thr80)				
		P	R	AUC	P	R	AUC	R@50	R@80
def	EH	18.0	90.3	26.3	22.2	89.3	26.4	65.9	46.1
	ET	16.7	79.3	15.3	19.9	74.9	16.8	35.7	26.0
amp	EH	23.4	87.2	32.1	26.1	83.0	31.5	74.0	55.1
	ET	17.3	75.9	18.9	21.1	73.1	20.2	45.5	35.0
def	HH	2.4	92.1	17.3	3.0	91.6	17.4	32.4	16.6
	HT	3.0	75.3	8.5	3.6	73.7	8.3	8.0	3.1
amp	HH	2.5	91.9	20.6	3.2	91.1	20.4	40.5	24.7
	HT	3.1	74.9	10.4	3.5	72.1	10.4	13.1	4.1

Table 8: Results on Head-vs-Tail. EH: easy P+head, ET: easy P+tail, HH: hard P+head and HT: head P+tail.

Method	Stage 1			Stage 2 (thr80)				
	P	R	AUC	P	R	AUC	R@50	R@80
zero-ens	25.0	43.5	19.0	28.1	39.7	17.3	25.9	15.5
few-ens	28.3	41.9	18.5	31.8	37.8	16.9	25.2	15.2
def	4.1	**70.6**	18.0	4.9	**67.9**	17.8	21.5	7.2
num	4.4	70.3	17.9	5.2	67.9	17.4	20.8	8.3
amp	13.4	60.5	**23.5**	15.7	57.8	**23.3**	**31.9**	**18.3**
amp/neo	18.5	51.8	20.5	**21.0**	48.8	19.5	29.3	13.6

Table 9: Results on Web Data. Stage-1 with ranking:{num, amp}; batching:{neo}; k=40; Stage-2: thr80.

6.4 Sensitivity of Hyper-Parameter Settings

We performed extensive experiments on varying hyper-parameter settings. We report on the sensitivity of the two most important Stage 1 choices: k (no. top-k passages) and b (no. passages per batch). Figure 3 shows *R@P50* numbers for *amp/neo* by varying k from 5 to 40 and b from 2 to 5. We observe that increasing k is very useful, but has diminishing returns beyond a certain value. Note that larger k also increase the cost for invoking the LLM more often. For small k, there is no benefit from larger batches, but this changes for large k, where bigger batches keep improving recall. Note that this also increases LLM costs, as we pass more tokens into the inference.

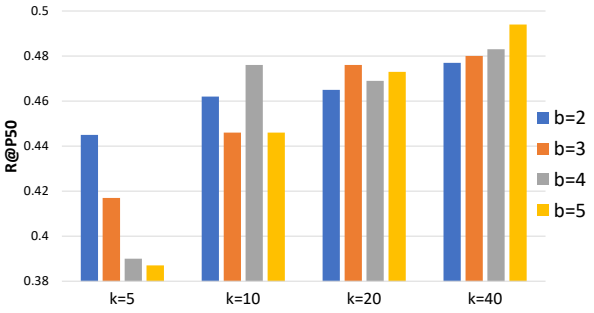

Figure 3: Varying hyper-parameters k, b for *amp/neo*.

All L3X-RAG configs use five reformulations for retriever queries. Table 11 presents the change in results with fewer query variants. Both recall and R@P values drop with less queries, showing the vital role of diversified formulations.

6.5 Error Discussion

We observed a variety of typical error cases, and discuss three of the most notable ones.

Hallucinations. The LLM calls often return extremely long lists of objects, including names that do not occur in the respective books at all. In RAG mode, the LLM does not necessarily restrict its outputs to names occurring in the input passages. These are *unfaithful* generations, but for recall, our main target, producing names from parametric memory is an advantage. To quantify the issue of hallucinated O values, we compute the no. of generated objects that do not appear in the respective book:

$$|\cup_{SP} \{\text{generated O for SP} \mid O \notin \text{book}\}|$$

normalized by the total no. of generated O values. We observed the following hallucination rates after Stage 1: LLM-only: 55.3%, def: 51.7%, amp:

	def/thr80			amp/thr80			amp/pred(g)			amp/neo/pred(g)		
	AUC	R@P50	R@P80	AUC	R@P50	R@P80	AUC	R@P50	R@P80	AUC	R@P50	R@P80
hasCEO	33.5	51.0	19.5	41.1	62.0	42.6	44.7	64.7	44.4	42.9	63.2	42.9
isMemberOf	12.1	10.0	1.2	15.9	21.8	8.3	16.4	23.2	9.1	15.6	22.2	9.9
hasSubsidiary	7.8	3.4	0.9	12.8	12.0	4.1	13.1	13.0	5.9	12.9	12.3	4.0
macro-avg.	17.8	21.5	7.2	23.3	31.9	18.3	**24.7**	**33.6**	**19.8**	23.8	32.6	18.9

Table 10: Results (%) for Web Data with Drill-Down Results by Predicate.

L3X	**Stage 1**			**Stage 2 (thr80)**				
amp	P	R	AUC	P	R	AUC	R@50	R@80
#q=5	13.7	83.6	27.5	16.0	81.0	27.4	48.6	35.9
#q=3	15.2	81.1	27.2	18.1	78.5	27.5	48.2	35.4
#q=1	20.5	76.2	26.1	23.4	71.1	26.5	44.0	31.5

Table 11: Varying no. queries for retriever with *amp*.

40.7%, amp/neo: 38.1%. Erroneous objects include made-up names and non-entity phrases, such as "X's sister", where the LLM appends a phrase related to the predicate instead of an O name. This underlines that stage-2 scrutinizing is essential.

Confusing Predicates. Another common case is that the LLM generates valid O values that are not in the proper relation P with subject S. The most interesting situation here is when that incorrect O is in relation with S for another predicate Q ($\neq$ P) (e.g., Dumbledore appearing among Harry Potter's parents instead of being a friend).

To quantify, we compute a #P$\times$#P confusion matrix, with counts of generated O for P when ground truth is Q. For our best method, *amp/pred(g)*, we observed a ratio of ca. 60:30:10 of accepted true positives (TP), predicate-confused TPs, and accepted false positives (FP). This suggests that merely extracting the right SO pairs is not the problem (only 10% completely FPs), but getting the predicate correct is the big issue here. The most salient predicate pairs of confusion are (Friend,Family) and (Friend,Opponent). This may be surprising on first glance, but it is the sophistication and subtlety of fictional literature that makes this a daunting case.

Missing True Positives in the Low Ranks. Majority of TPs are in the higher ranks. These are followed by a long tail, with mostly FPs but sprinkled with TPs at lower ranks. To assess how well Stage 2 recovers *low-ranked* TPs, we identify the ranking cut-off for R@P50 and count the missing TPs below this threshold—i.e., those misclassified as false negatives. Even with our best methods, ca. 16% of all the ground-truth O values fall into the category of low-rank missing TPs.

7 Conclusion

We introduced the task of extracting long lists of objects from long documents, and developed the L3X methodology, comprising LLM prompting, retrieval augmentation, passage ranking and batching, and classifiers to scrutinize candidates and prune false positives. Extensive experiments with a range of L3X configurations over two datasets provide key insights. First, L3X greatly outperforms LLM-only extraction in recall and R@P. Second, one of our methods, *amp/pred* with pseudo-relevance feedback and a classifier with tuned hyper-parameters, achieves remarkable performance of ca. 85% recall and ca. 37% R@P80 on full-length books. However, drill-down analyses by predicate and head-vs-tail entities show that for the hardest cases, there is substantial room for improvement. Third, this underlines the biggest challenge of our task: passages scattered across long books give cues to a smart human, but are still very hard to pinpoint and extract for AI systems (incl. LLMs).

8 Limitations

This work is based on Llama3.1-70B. We also ran Stage 1 studies using other LLMs. With a 10x smaller Llama-8B model, recall fell below 50% (compared to ca. 80% with the 70B model). With GPT-3.5, preliminary experiments showed results similar to Llama-70B. Still, a broader comparison of different LLMs for our task would be desirable.

In the absence of suitable datasets, we constructed a new benchmark resource—limited in scale, though. Expanding the dataset would be useful , however, scaling up the benchmark for books poses a major challenge. On one hand, for popular books, data for ground truth construction is readily available, but these books are likely well captured in the model's parametric memory from pre-training. On the other hand, we could consider books in the long tail of popularity or brand-new books, but this incurs high cost to obtain ground truth, requiring end-to-end reading and great diligence for annotations.

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, 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.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 2778–2788. ACM.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. 2023. Factool: Factuality detection in generative AI - A tool augmented framework for multi-task and multi-domain scenarios. *CoRR*, abs/2307.13528.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Meihao Fan, Xiaoyue Han, Ju Fan, Chengliang Chai, Nan Tang, Guoliang Li, and Xiaoyong Du. 2024. Cost-effective in-context learning for entity resolution: A design space exploration. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*, pages 3696–3709. IEEE.

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*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438.

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.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.

Keshav Kolluru, Muqeeth Mohammed, Shubham Mittal, Soumen Chakrabarti, and Mausam. 2022. Alignment-augmented consistent translation for multilingual open information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2502–2517. Association for Computational Linguistics.

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.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35.

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.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9004–9017. Association for Computational Linguistics.

Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4074–4077. IJCAI/AAAI Press.

Llama Team AI @ Meta. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 12076–12100. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft

prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5203–5212. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Sneha Singhania, Simon Razniewski, and Gerhard Weikum. 2023. Extracting multi-valued relations from language models. In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pages 139–154, Toronto, Canada. Association for Computational Linguistics.

Alisa Smirnova and Philippe Cudré-Mauroux. 2019. Relation extraction using distant supervision: A survey. *ACM Comput. Surv.*, 51(5):106:1–106:35.

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.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 885–895. Association for Computational Linguistics.

Kai Sun, Yifan Ethan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2023. Head-to-tail: How knowledgeable are large language models (llm)? A.K.A. will llms replace knowledge graphs? *CoRR*, abs/2308.10168.

Blerta Veseli, Simon Razniewski, Jan-Christoph Kalo, and Gerhard Weikum. 2023. Evaluating the knowledge base completion potential of GPT. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 6432–6443. 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.

Yiqing Xie, Jiaming Shen, Sha Li, Yuning Mao, and Jiawei Han. 2022. Eider: Empowering document-level relation extraction with efficient evidence extraction and inference-stage fusion. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 257–268. Association for Computational Linguistics.

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.