Combining Constrained and Unconstrained Decoding via Boosting: BoostCD and Its Application to Information Extraction

Anonymous ACL submission

Abstract

Many recent approaches to structured NLP tasks use an autoregressive language model M to map unstructured input text x to output text y representing structured objects (such as tuples, lists, trees, code, etc.), where the desired output structure is enforced via constrained decoding. During training, these approaches do not require the model to be aware of the constraints, which are merely implicit in the training outputs y. This is advantageous as it allows for dynamic constraints without requiring retraining, but can lead to low-quality output during constrained decoding at test time. We overcome this problem with *Boosted Constrained Decoding (BoostCD),* which combines constrained and unconstrained decoding in two phases: Phase 1 decodes from the base model M twice, in constrained and unconstrained mode, obtaining two weak predictions. In phase 2, a learned autoregressive boosted model combines the two weak predictions into one final prediction. The mistakes made by the base model with vs. without constraints tend to be complementary, which the boosted model learns to exploit for improved performance. We demonstrate the power of BoostCD by applying it to closed information extraction. Our model, *BoostIE,* outperforms prior approaches both in and out of distribution, addressing several common errors identified in those approaches.

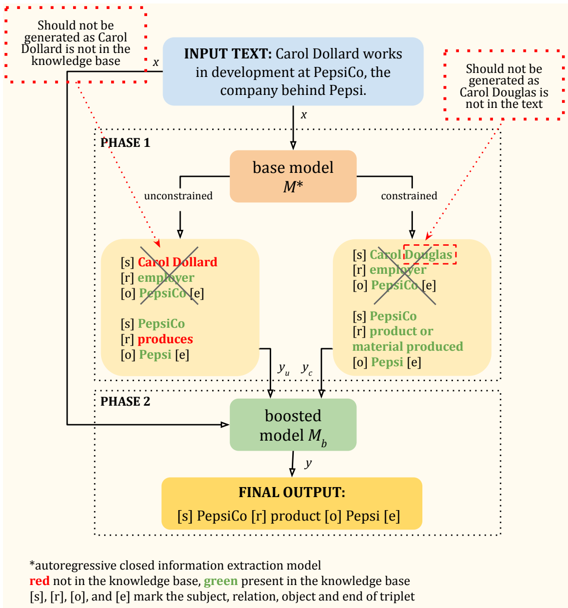

Figure 1: **Overview of BoostCD,** exemplified on the task of closed information extraction (BoostIE). Phase 1 applies the base model twice on input x: unconstrained and constrained. Phase 2 combines the two resulting weak predictions y_u and y_c into final prediction y using a boosted model, which during training learns to undo mistakes made by the base model.

1 Introduction

Extracting structured semantic information from unstructured text is essential for many AI tasks, including knowledge discovery (Ji and Grishman, 2011), knowledge base maintenance (Tang et al., 2019), symbolic representation, reasoning (Ji et al., 2022), and planning. Beyond these applications, a growing number of NLP tasks now explicitly require structured outputs as part of their formulation. Some examples are code generation (Poesia et al., 2022), SQL generation (Scholak et al., 2021), constituency parsing (Deutsch et al., 2019), and various information extraction (Cao et al., 2021; Josifoski et al., 2023; Orlando et al., 2024).

Many recent approaches for these tasks use autoregressive models trained on pairs of unstructured input text and structured output targets, coupled with constrained decoding (Josifoski et al., 2023, 2022; Whitehouse et al., 2023; Cao et al., 2021). In real-world tasks, the constraints can often change, so constrained decoding offers an easy way to adapt the schema without the need to retrain the model. Constrained decoding also helps steer the model the right way when it is already close to generating the correct output (e.g., when the only problems are minor surface form discrepancies). However, on the downside, as the model remains unaware of the explicit constraints until decoding at inference time, it may generate less plausible outputs when the input data or the constraints at inference time deviate from those seen during training.

We illustrate in Fig. 1, which shows an example of outputs of the autoregressive model with constrained decoding on the closed information extraction (cIE) task, where the goal is to extract complete sets of fact triplets (subject, relation, object) from text, where all entities and relations must be present in a predefined knowledge base (KB). In the provided example, the base model is a cIE model trained on exhaustive data (i.e., facts in the text are fully expressible under KB constraints). The shown input text differs from the training data by containing facts that are not expressible under KB constraints. The base model generates two triplets when run in unconstrained mode. For the first one, the entity present in the text, "Carol Dollard", is not present in the KB. Because the base model was trained on exhaustive data, when prompted in an unconstrained manner, it generates a correct triplet that captures this entity. When constrained, however, instead of removing this triplet entirely (as it does not comply with the KB), the model resorts to generating a triplet with a wrong entity with a similar name ("Carol Douglas"). For the second triplet, the unconstrained model generates correct entities but makes a formatting error in the relation ("produces" instead of "product or material produced"). In this case, constrained decoding helps by correcting the relation name. Ideally, we seek a method able to recognize patterns in the constrained and unconstrained outputs to combine their strengths and recover from their errors, without having to know the explicit constraints already at training time (which would reduce flexibility as constraints change, e.g., as the KB evolves).

To overcome these problems, we introduce *Boosted Constrained Decoding (BoostCD),* a method with the ability to correct systematic errors that an autoregressive model trained for a structured NLP task might make during constrained as well as unconstrained generation. BoostCD works in two phases: Phase 1 decodes from the base model M twice for the input text x, in constrained and unconstrained mode, obtaining two weak predictions y_c and y_u. In phase 2, a learned autoregressive boosted model combines the two weak predictions into one final prediction y. Empirically, the mistakes made by the base model with vs. without constraints tend to be complementary, which the boosted model learns to exploit during training for improved performance.

To demonstrate the power of the BoostCD paradigm, we apply it to closed information extraction (cIE; cf. Fig. 1) as an example of a structured task with constraints (defined by the content of the knowledge base) that tend to change dynamically in real-life settings. We further enhance the resulting cIE model, *BoostIE,* with Direct Preference Optimization (DPO) (Rafailov et al., 2023) for improving performance on out-of-distribution data. We show that BoostIE outperforms previous methods both in-distribution (on synthetic data it was trained on) and out-of-distribution (on random Wikipedia paragraphs). We also demonstrate that BoostIE lowers the rate of common errors made by earlier techniques.

Contributions. Our contributions are as follows:

(i) We propose BoostCD, a new method for training autoregressive language models for structured NLP tasks.

(ii) We instantiate BoostCD for the closed information extraction (cIE) task, obtaining a model termed *BoostIE,* and conduct a detailed evaluation, showing that BoostIE outperforms existing methods both in-distribution (by 17.05 and 12.56 absolute points in micro and macro F1, respectively) and out-of-distribution (by 10.94 and 12.54 absolute points in micro and macro F1, respectively).

(iii) A detailed error analysis confirms that BoostIE lowers the rate of common errors made by previous cIE models, as well as disadvantages of vanilla constrained decoding for this task.

(iv) We share our code, models, and data for researchers to reuse and extend: `[anonymous]`

2 Method: BoostCD

Language models trained for structured NLP tasks in a supervised manner can often perform reasonably well even without the constraints imposed, but the constraints are still required to guarantee 100% valid generations, and they can steer the model to pick the one correct output when multiple outputs might seem plausible *a priori* (e.g., when an entity has multiple aliases). However, when the constraints require altering the unconstrained output significantly (e.g., when an entity generated in unconstrained mode is not present in the KB), performance can suffer from imposing the constraints.

We hence seek a method that enjoys the benefits of constraints without suffering from their negative side effects. In developing such a method, we draw inspiration from boosting (Schapire, 1990), a classic ensemble learning technique that aims to improve performance by iteratively combining weaker models into a single stronger one. The idea is to train models sequentially, where each new model focuses on the mistakes made by the previous ones. The final prediction is formed by aggregating the outputs of all models, often through weighted voting or summation. Our method, *Boosted Constrained Decoding (BoostCD),* trains a new model, the *boosted model,* to predict the ground-truth output based on both the constrained and the unconstrained generation from the autoregressive base model together with the input text. This way of training allows the boosted model to recover from systematic mistakes made by the base model without requiring explicit knowledge of the constraints at training time.

For intuition, consider the cIE task as illustrated in Fig. 1: in the example, unconstrained decoding extracted a triplet (Carol Dollard, employer, PepsiCo) that was not extracted by constrained decoding (because Carol Dollard is not in the KB); and constrained decoding extracted a triplet (Carol Douglas, employer, PepsiCo) that was not extracted by unconstrained decoding (because Carol Douglas is not mentioned in the input text). By seeing such candidate triplets together with the ground-truth triplet set (which contains neither of the above candidate triplets), the boosted model learns to recognize that entities occurring only in the constrained but not the unconstrained output (or *vice versa*) indicate triplets that were erroneously extracted by the base model and should thus be discarded. Note that this is but one of the many potential patterns that the boosted model might learn.

Pipeline. The BoostCD pipeline is shown in Fig. 1. For illustration, we use the example of cIE, although our method can be applied to any structured extraction task. Under the assumption that we have a dataset which consists of pairs (x, y), where x is the input text, and $y = \{(s,r,o) | (s,r,o) \in E \times R \times E\}$ (a set of triplets constrained to the KB that contains all entities E and relations R), our training pipeline consists of two phases:

(i) **Phase 1:** We use a base model M, trained in an autoregressive manner on (x, y) pairs, to make two parallel passes. In one, we let the model generate in an unconstrained manner: by sending input text x to the model M without imposing any constraints, we obtain the output y_u. In the second, we generate by imposing constraints: by providing the input text x and using M with constrained decoding, we obtain the output y_c.

(ii) **Phase 2:** In this phase, we train the boosted model M_b to correct the errors that the base model M made in phase 1. M_b is trained in an autoregressive way to map (x, y_u, y_c) (i.e., the original input together with both phase-1 predictions) to the ground-truth output y.

During the inference, we repeat the steps from both phases: (1) we make two parallel passes with the base model M to generate constrained and unconstrained predictions ($\hat{y_c}$ and $\hat{y_u}$) and (2) we send $(x, \hat{y_u}, \hat{y_c})$ to the input of the boosted model M_b to make a final prediction $\hat{y}$. This prediction can be made with our without constrained decoding.

In the following sections, we apply BoostCD to the cIE task by curating the data and modeling to fit its needs. We emphasize that this paradigm can be used for other structured tasks, with adaptations of the data and modeling. Note that we use only one step of boosting in our pipeline, although in principle there is nothing that restricts this pipeline to one iteration only. For our setting, we found one step to be sufficient, but for other applications, it is possible to explore multiple iterations of the same algorithm.

3 Application to information extraction

To assess BoostCD, we apply it to the cIE task and refer to the resulting boosted model as BoostIE.

3.1 Data

To train the base model, we need data that is exhaustive, i.e. the input is fully expressible under constraints. For cIE, the base model should extract all the facts present in the text, regardless of constraints (i.e. perform open information extraction).For the boosted model, we can simulate the setting in which some samples express entities in the text that do not exist in the KB. For a fraction of the data we randomly remove some entities from the KB making it impossible for a base model to generate them in the constrained setting. We also remove these entities from the target triplet set by removing each triplet that contains the entity in question. By providing these samples during training, we let the model learn what happens when the entity in text is not present in the KB and hopefully bring it closer to generating the correct output.

By curating the data for the boosted model this way, we also prevent the boosted model from learning what is present in the KB, as this changes for every data point. Instead, the boosted model is forced to learn patterns in the constrained and unconstrained outputs from the base model and rely on input information. This makes the model more flexible if KB is changing over time. Data generated in this way also has some samples for which no triplets are extractable (i.e. they are not present in the KB). As a result, the boosted model is trained to produce an empty set for some samples, which might not be the case for the base model. This does not guarantee that the boosted model would be able to do it for the text that has no triplets at all, but from our results, this seems to be the case.

3.2 Model and inference for cIE

Modeling. We follow the same setting for modeling as Josifoski et al. (2023). Both base and boosted models are based on FlanT5 (Chung et al., 2022), and are trained to autoregressively generate a linearized sequence of the corresponding triplet set y when prompted with the input text x. Training is done by maximizing the target sequence's conditional log-likelihood with teacher forcing (Sutskever et al., 2011) and cross-entropy loss. We also use dropout (Srivastava et al., 2014) and label smoothing (Szegedy et al., 2016).

Output linearization. We represent triplets as model-compatible sequences using delimiters: [s], [r], [o], and [e] mark the subject, relation, and object, and the end of each triplet. We concatenate the triplets in the order they appear in the text to form the final sequence.

Inference. Similarly to Josifoski et al. (2023), we use constrained beam decoding during inference time. Valid prefixes that follow both linearization and KB constraints are dynamically generated.

3.3 DPO finetuning

As we currently do not have access to a well-aligned dataset for cIE that is made on real-world data, the process of training base and boosted models is done with synthetic data that might not highly resemble natural text. As a consequence, this might hinder the performance of our model in the wild. In an attempt to overcome this, we propose to tune the model with Direct Preference Optimization (DPO) (Rafailov et al., 2023), using data more similar to the real-world one.

DPO is a reward-free method for aligning language models with human preferences by directly optimizing for preferred outputs over less preferred alternatives. In our case, we use DPO to adapt the model toward generating more accurate and faithful structured outputs on real-world text.

For DPO finetuning, we use around 600 samples from the REBEL dataset (Huguet Cabot and Navigli, 2021), and an additional 100 samples for validation. We chose this dataset because it was crafted from the real Wikipedia text, although only by collecting text from the first paragraphs of Wikipedia articles. That means that it still differs from randomly crawled Wikipedia text. To identify samples the most similar to the real text, we train a RoBERTa classifier (Liu et al., 2019) that can distinguish real text (random text from Wikipedia articles, not limited to the abstracts) from Wiki-cIE Code text (for more details about the classifier, see Appendix F). We use this classifier to pick the samples with the highest probability of being real Wikipedia text. Since cIE model trained on this data, GenIE (Josifoski et al., 2022), is not exhaustive, and SynthIE does not perform well outside of the training distribution, we decide to use a large language model to choose samples with a fitting target from one of these two models (if any of the two can produce it). For each text sample, to collect ranked candidate triplet sets, we run GenIE and SynthIE in constrained manner. We then let GPT-4[1] decide which one of the two options is better and use this information for ranking. If none of the options are good, we discard the sample. This way, we collect the data that (1) resembles real text more and (2) for which we have a quality solution from one of the existing models. This procedure allows the model to learn a preference signal aligned with real-world patterns, without requiring gold-standard annotations. We note that this might systematically discard harder samples, but is a good starting point to attempt to generalize to a different data distribution.

4 Evaluation setup

Knowledge base. We use the subset of Wikidata (Vrandečić, 2012), using entities that are connected to English Wikipedia pages and relations that appear at least once in the REBEL training dataset. Our catalogue consists of 2.6M entities and 888 relations. For the unique representation of each entity,

[1] We use `gpt-4-0613` version of GPT-4.

	Overall			Removed			Same		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Micro									
BoostIE (constrained)	57.23 ± 0.79	**48.24 ± 0.63**	**52.35 ± 0.63**	38.69 ± 1.84	**45.99 ± 1.79**	42.02 ± 1.64	63.91 ± 0.94	**48.79 ± 1.03**	55.33 ± 0.96
BoostIE (unconstrained)	54.72 ± 0.89	46.31 ± 0.73	50.16 ± 0.73	31.62 ± 1.62	43.76 ± 1.82	36.71 ± 1.57	**64.86 ± 0.91**	46.93 ± 1.01	54.45 ± 0.95
BoostIE + DPO (constrained)	**59.45 ± 0.65**	46.65 ± 0.65	52.28 ± 0.59	**43.03 ± 1.82**	43.90 ± 1.68	**43.46 ± 1.55**	64.82 ± 0.94	47.31 ± 0.96	**54.70 ± 0.91**
BoostIE + DPO (unconstrained)	56.39 ± 0.82	44.81 ± 0.73	49.94 ± 0.72	35.41 ± 1.80	41.94 ± 1.80	38.39 ± 1.65	64.58 ± 1.02	45.50 ± 0.95	53.38 ± 0.93
ReLiK (filtered)	22.89 ± 0.57	20.80 ± 0.58	21.79 ± 0.53	17.21 ± 1.01	18.37 ± 1.12	17.77 ± 0.96	24.58 ± 0.66	21.43 ± 0.56	22.89 ± 0.57
SynthIE 400k (constrained)	31.71 ± 0.77	39.81 ± 0.68	35.30 ± 0.68	13.57 ± 0.92	40.41 ± 1.85	20.31 ± 1.20	45.90 ± 1.03	39.67 ± 0.83	42.56 ± 0.88
SynthIE 400k (unconstrained)	33.40 ± 0.81	34.83 ± 0.77	34.10 ± 0.73	15.18 ± 0.98	34.54 ± 1.73	21.09 ± 1.19	45.98 ± 1.17	35.00 ± 0.88	39.75 ± 0.96
Macro									
BoostIE (constrained)	58.35 ± 2.46	**46.11 ± 1.02**	**48.81 ± 1.39**	37.51 ± 2.55	**39.47 ± 3.11**	36.01 ± 2.20	**61.96 ± 3.30**	**46.26 ± 1.62**	**50.28 ± 1.95**
BoostIE (unconstrained)	43.81 ± 1.78	35.78 ± 0.97	37.34 ± 1.20	26.69 ± 2.14	32.59 ± 3.04	27.24 ± 2.01	53.21 ± 3.11	38.12 ± 1.23	42.31 ± 1.63
BoostIE + DPO (constrained)	**59.09 ± 2.50**	44.74 ± 1.10	48.29 ± 1.50	**39.32 ± 2.93**	38.34 ± 2.70	**36.55 ± 2.12**	61.58 ± 3.21	45.00 ± 1.58	49.27 ± 1.87
BoostIE + DPO (unconstrained)	42.89 ± 1.75	32.91 ± 0.97	35.23 ± 1.07	28.33 ± 2.18	31.59 ± 2.51	27.85 ± 1.71	50.84 ± 3.18	35.54 ± 1.29	39.73 ± 1.66
ReLiK (filtered)	17.22 ± 0.99	12.81 ± 0.54	12.92 ± 0.56	11.63 ± 1.38	11.96 ± 0.80	10.59 ± 0.75	17.20 ± 1.36	13.14 ± 0.53	13.18 ± 0.66
SynthIE 400k (constrained)	40.29 ± 1.60	38.77 ± 0.95	36.25 ± 1.12	21.22 ± 1.87	31.72 ± 3.54	22.67 ± 2.11	47.97 ± 2.15	38.26 ± 0.73	39.68 ± 1.21
SynthIE 400k (unconstrained)	35.58 ± 1.28	33.76 ± 1.16	32.28 ± 1.05	16.94 ± 1.39	27.10 ± 3.41	18.88 ± 1.71	46.26 ± 2.53	34.04 ± 0.86	37.04 ± 1.34

Table 1: Results on Wiki-cIE Code dataset: Overall - whole test set, Removed - test samples with removed random entities (and triplets) from the target and KB, Same - test samples without entity removal. For BoostIE, constrained and unconstrained refers to the final boosted model mode of operation. We report both micro and macro results, with 95% CI. Best results are in bold.

we use its English Wikipedia title. For relations, we use their label in Wikidata.

Data. For training the base model M, we use a subset of 300K samples from the train split of Wiki-cIE Code used for training SynthIE models (see Appendix B for details). The boosted model M_b was trained on an additional 100K samples from the same dataset. We also train a SynthIE model on all 400k samples for a fair comparison. 100K samples used for training the boosted model have been altered as explained in Sec. 3.1, and 40% of the randomly chosen samples have been altered. For each altered sample, up to three entities were removed, uniformly. The validation and test data were crafted in the same way, each being a subset of the corresponding Wiki-cIE Code of 10K samples. Wiki-cIE Code is imperfect as it does not have the same properties as the real-world text, but can demonstrate the abilities of this training technique effectively.

Baselines. To isolate the effects of this training technique, we compare BoostIE with the SynthIE model of the same size, trained on the same 400K samples used in the BoostIE pipeline (without alterations). We also compare our method with ReLiK cIE model of similar size,[2] as this is the state-of-the-art model right now. We provide results with and without using DPO after initial training. For more details about the baselines, see Appendix C.

[2]We use "relik-ie/relik-cie-large", see https://huggingface.co/relik-ie/relik-cie-large

Metrics and implementation detail. We evaluate the performance in terms of micro and macro precision, recall, and F1 score. All results are reported with 95% confidence intervals constructed from 50 bootstrap samples. For more details on evaluation metrics, see Appendix D. For details on implementation, see Appendix E.

5 Results

5.1 Evaluation on Wiki-cIE Code

Performance evaluation. We first evaluate our method on in-distribution data. We use the metrics mentioned in Sec. 4 on the random subset of 10K samples from the test split of Wiki-cIE Code. We evaluate it on non-edited, as well as Wiki-cIE Code with entities randomly removed from the KB, as described in Sec. 3.1. We report results in Table 1.

ReLiK does not perform on Wiki-cIE Code nearly as well as SynthIE and BoostIE. This is expected, as it was not trained on this data, and Wiki-cIE Code has a different distribution from REBEL on which ReLiK was trained.

Second, we notice that all the models perform worse for the samples where some entities are randomly removed from the KB. This is in line with our expectations, especially for SynthIE, as it was trained to extract exhaustively, and cannot handle instances where this is not possible. Precision is more affected by this modification of the data. Micro-recall stays almost the same, while macro-recall drops much less than precision. This happens because the models tend to output wrong triplets either related to the removed entity (in unconstrained mode) or related to a similarly named entity (in constrained mode). Triplets related to the correct entities present in the graph mostly stay in the output, maintaining the recall relatively high.

For BoostIE models, there is a noticeable improvement both for edited samples with removed entities and for the non-edited ones. The improvement is visible for both micro and macro scores. We suspect that this happens because by using BoostCD (1) we are implicitly including the information about the presence or lack of an entity in the KB and (2) we include the information about errors SynthIE, which is used as a base model, makes regardless of the KB. Examples of the latter can be wrong disambiguation of certain entities in the KB, or the less adequate relations for the scenario (for instance, using “location” instead of “located in or next to body of water” for text expressing an entity “Niagara” being located in the “Lake Ontario”).

The difference in scores for constrained and unconstrained settings is higher for BoostIE models than for SynthIE. This happens because, unlike SynthIE which tends to generate triplets with wrong entities when something is not present in the KB, BoostIE is able to recognize this setting. This is expected, as BoostCD used for training BoostIE models specifically addresses this issue. We speculate that, in the case of BoostIE, constrained decoding helps filter out triplets with missing entities rather than causing the model to generate triplets with incorrect ones. In other words, BoostIE assigns a higher probability to the output that does not include entities missing from the KB. For macro scores, the difference is present for both original and edited samples. This likely means that BoostIE detects some systematic errors that happen for rare relations when using SynthIE.

Finally, the usage of DPO does not result in significant improvements over the standard BoostIE model on this data. This is expected given its use was aimed at improving real-data performance (see Sec. 5.2 for evaluation on natural text). Still, the absence of performance degradation, for both micro and macro socres, is a positive sign.

Performance by relation frequency. As mentioned earlier, relations expressed in the natural text are imbalanced: there is a small number of relations that are present very often and a large number that are rare. Training on real data can lead to bad performance on those rare relations, which would be masked by the overwhelming presence of

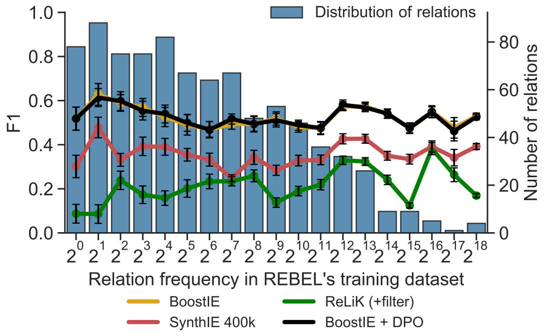

Figure 2: **Impact of the relation frequency.** Relations are bucketed based on their frequency; bucket 2^i contains relations occurring between 2^i and 2^{i+1} times. The histogram shows the number of relations per-bucket. The line plots depict the per bucket F1 scores evaluated on Wiki-cIE Code test dataset with confidence intervals constructed by bootstrapping.

common relations. Wiki-cIE Code was constructed with this in mind. To verify that our method does not compromise the performance on rare relations, as well as to evaluate the performance of ReLiK in this light, we mimic the experiment by Josifoski et al. (2023) and bucket relations by their frequency in REBEL training set which follows the natural distribution of relations. We report results in Fig. 2. ReLiK performs worse for rare relations. This is expected, as parts of their pipeline were trained with real-world data. When it comes to BoostIE models, they perform consistently better than SynthIE for all relation buckets and maintain stable performance over rare and common relations.

5.2 Evaluation on natural text

Micro	**Precision**	**Recall**	**F1**
BoostIE + DPO	**48.89** ± 18.16	**27.76** ± 11.78	**34.93** ± 11.97
BoostIE	22.68 ± 11.85	17.38 ± 11.02	19.33 ± 10.12
ReLiK	25.88 ± 18.38	22.58 ± 15.33	23.99 ± 16.38
SynthIE 400k	6.74 ± 3.91	13.34 ± 10.19	8.76 ± 4.96
Macro			
BoostIE + DPO	**23.87** ± 4.92	**20.85** ± 3.91	**20.87** ± 4.13
BoostIE	15.50 ± 3.20	13.62 ± 2.88	13.49 ± 2.77
ReLiK	9.52 ± 0.95	9.55 ± 2.84	8.33 ± 1.35
SynthIE 400k	5.38 ± 1.95	6.05 ± 2.66	5.35 ± 2.11

Table 2: Human evaluation on Wikipedia text. The best results are bolded. Results are reported with 95% CI.

To better understand the performance of both our BoostIE models, as well as ReLiK and SynthIE out of distribution, we manually annotate 50 random samples from Wikipedia text (see Appendix A for data collection process). During this process, each

Wikipedia text sample is assigned a ground truth triplet set (see Appendix G for annotation process details). We then compare all models on this data. We run both BoostIE and SynthIE with constrained decoding, as evaluations on Wiki-cIE Code suggest this improves performance. Results are presented in Table 2. We note that Wikipedia text does not fully reflect the real-world text, as it is a highly structured and factual text. Nonetheless, it is a good starting point for evaluating cIE models.

From Table 2, SynthIE performs the worst. This matches our expectations as SynthIE was trained on the data that has little resemblance to the Wikipedia text. Next, BoostIE performs slightly worse than ReLiK in terms of micro metrics, but a little better in terms of macro metrics. We see this as a good sign. BoostIE manages to be on par with ReLiK without the need for a separate retrieval model. We suspect that the reason why BoostIE is better in macro metrics is because ReLiK was trained on the REBEL dataset, which has heavy-tailed distribution of relations. Finally, BoostIE with DPO performs by far the best over both micro and macro metrics. This highlights the importance of the DPO step, and the potential it has to adapt the language model to the a differently distributed data.

5.3 Error analysis

To further examine what kind of errors previous and our method make, we collect 50 random samples of text from Wikipedia (see Appendix A for the data collection process). We compare SynthIE, ReLiK and BoostIE with and without the DPO step. By manual inspection, we identify five types of errors:

(i) **Unexhaustive triplets:** triplet set does not include some correct triplets

(ii) **Incorrect related triplets:** triplet set includes some incorrect triplets about correct entities

(iii) **Misclassified entity:** entities in the triplets are wrongly identified as similarly named ones

(iv) **Unrelated triplets:** triplet set includes triplets unrelated to the text or entities in the text

(v) **Entity-centered triplets:** triplets in the triplet set are centered around one entity

Some errors can happen at the same time, e.g. there can be a triplet set that is both unexhaustive and contains unrelated triplets. We annotate the chosen sample and report the results in Table 3.

From the results, it is clear that SynthIE struggles with the Wikipedia data in multiple ways. Most of the samples contain at least some unrelated triplets (60%). We also notice that it has the highest percentage of samples with misclassified entities (9%). Both of these errors stem from the constrained decoding issues – when the entity is not present in the KB but is expressed in the text, SynthIE tends to produce triplets with similarly-named entities (misclassified) or even completely unrelated ones. This is confirmed by the BoostIE results, as both of these problems are largely mitigated for BoostIE.

SynthIE also produces the highest percentage of samples with triplet sets centered around one entity (16%). We notice that BoostIE without DPO has similar issues (11%). We believe this error comes from a bad distribution of triplet sets in the Wiki-cIE Code used for training both of these models. ReLiK and BoostIE with DPO which were either trained with different data (REBEL), or exposed to it through DPO, suffer from this suffer from this to a much lesser degree (0% and 6% respectively).

Among all error types, unexhaustive generations exhibit the least variance across the four models. Despite intentionally training SynthIE and BoostIE models on an exhaustive dataset, on the real text, they fall short similarly to ReLiK trained on an unexhaustive dataset (REBEL). We suspect that the limited performance of BoostIE without DPO might be due to a significant mismatch between the training data distribution and the real-world text. In the case of BoostIE with DPO, although the data used during fine-tuning more closely resembles Wikipedia text, it includes some outputs from GenIE, which is trained on REBEL. We expect that some of these outputs are not exhaustive. This likely contributed to the persistence of unexhaustive generations. In Appendix H, we provide a few additional analyses of common cIE approaches, setting the stage for further research in this area.

	SynthIE	**ReLiK**	**BoostIE**	**BoostIE + DPO**
Unexhaustive	0.33 $\pm$ 0.09	0.38 $\pm$ 0.11	0.32 $\pm$ 0.11	0.38 $\pm$ 0.10
Incorrect related	0.36 $\pm$ 0.11	0.28 $\pm$ 0.10	0.26 $\pm$ 0.11	0.12 $\pm$ 0.09
Misclassified entity	0.09 $\pm$ 0.07	0.04 $\pm$ 0.04	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00
Unrelated	0.60 $\pm$ 0.11	0.14 $\pm$ 0.09	0.28 $\pm$ 0.10	0.08 $\pm$ 0.06
Entity-centered	0.16 $\pm$ 0.08	0.00 $\pm$ 0.00	0.11 $\pm$ 0.07	0.06 $\pm$ 0.05

Table 3: Error analysis on Wikipedia text samples. Numbers represent fraction of samples with the given type of error. Result are shown with 95% CI.

6 Related work

6.1 Closed information extraction

Older cIE methods usually rely on the combination of entity recognition (Tjong Kim Sang, 2002) and linking (Milne and Witten, 2008a) with relation extraction (Milne and Witten, 2008b) to obtain the set of triplets constrained to the KB. However, these methods often have problems with error propagation due to their architecture (Mesquita et al., 2019; Trisedya et al., 2019). A newer approach that combines entity linking and relation extraction is proposed by Orlando et al. (2024). In recent years, however, autoregressive methods have dominated the field. For the cIE task, this was first introduced by Josifoski et al. (2022). Josifoski et al. (2022) also introduced the usage of constrained decoding for this task. The same approach was adopted by Josifoski et al. (2023) and Whitehouse et al. (2023).

Another line of research in this field relies on building a good training dataset for the cIE task. Huguet Cabot and Navigli (2021) introduced REBEL, a dataset of fact triplets constructed using distant supervision. Similarly, Trisedya et al. (2019) introduce WikiNER, a dataset that is also made using distant supervision, but augmented with co-reference resolution and dictionary-based paraphrase detection. More recently, Whitehouse et al. (2023) presented WebIE, a multilingual distant-supervision dataset, with the introduction of some human-annotated samples as well. Josifoski et al. (2023) synthetically generated their data specifically having distributional (i.e. relational frequency issue) and exhaustiveness issues in mind.

The emergence of LLMs raises the question of their ability to perform this task. As shown by Josifoski et al. (2023), LLMs struggle with tasks that require structured output. For cIE, they also have no knowledge of the KB. Geng et al. (2024a) attempt to overcome this issue by combining an LLM with constrained decoding, but their evaluation on synthetic data limits broader conclusions.

6.2 Constrained decoding

Structured NLP tasks require the output to be in a certain form. To overcome this, different forms of constrained decoding have been proposed. Cao et al. (2021) address the entity-disambiguation constraints by generating a prefix trie at the decoding time, forcing output to be valid entities from the KB. Geng et al. (2024b) introduce grammar-constrained decoding, focusing on generalizing the constrained decoding to a wider variety of tasks. Park et al. (2024) introduce grammar-aligned decoding, which aims to correct the conditional probability of the LLM's distribution conditioned on the given grammar constraint. Koo et al. (2024) propose a method that addresses downsides of constrained decoding related to the tokenization issues by using automata-based constraints. Beurer-Kellner et al. (2024) propose a method that speeds up the constrained decoding that works in a subword-aligned fashion.

7 Discussion

7.1 Implications for cIE

Despite numerous efforts through years to solve cIE, current approaches struggle with performance on the real data, as well as adaptability to different KBs. Our method could be a step closer to an efficient and high-performing system that overcomes these issues. Our experiments show that BoostIE (BoostCD applied to cIE) improves the performance of constrained decoding, which is often used for cIE systems. Additionally, BoostIE does not directly learn what is present in the KB, which is the case for most current approaches, making it more adaptable to changes in the KB. Our experiments on Wiki-cIE Code also show that our method maintains a good performance over rare relations, while the evaluation on real Wikipedia data indicates that BoostIE is better at generalizing to out-of-distribution data. This seems to be the case especially when using DPO with data that resembles the target distribution. With that in mind, along with the fact that DPO does not degrade performance on the original data distribution, we draw attention that this can be used as an unexpensive way to improve the overall performance of the model. In an ideal scenario, our base model would be trained on an exhaustive dataset with more realistic text. This is not trivial to collect, so finetuning with DPO and a smaller finetuning dataset can be a good way to overcome this limitation.

7.2 Implications for other tasks

Although our present evaluation has focused on the benefits of BoostCD for closed information extraction, nothing about the method is inherently restricted to this task. A similar pipeline can be exploited for a wide range of structured NLP tasks, including tagging, parsing, code generation, JSON generation, and many more. We leave the evaluation of BoostCD on such other tasks for future work and hope that researchers and developers will benefit from BoostCD in practice.

Limitations

Entity surface form variations. Our current pipeline might have issues with entities that are presented in the text in a very different way than in the knowledge base (e.g. as acronyms or aliases). Since our model has no external knowledge, it cannot disambiguate between these cases vs. an entity that is present in the text but not in the KB. This is also something that we cannot expect from a small, specialized, model to know on its own, as it does not have broad knowledge of the external world. This is possibly an area where LLMs would excel.

Inference speed. Although we are using small language models for this task and we consider our approach to be scalable, inference requires three runs of a model (constrained and unconstrained base model run, and the run of the boosted model). This is less efficient than SynthIE or similar models, but is still faster and cheaper than running an LLM. Also note that the constrained and unconstrained run of the base model can be parallelized.

Training dataset. The dataset we used for training does not resemble real data, and has other distributional issues. One particular case of such issue is the distribution of entities in the triplet sets. Due to the way Wiki-cIE Code was generated, most of the triplets in triplet sets are centered around one or two entities. Real data often describes many more entities in a few sentences. Because of this, both SynthIE and BoostIE have troubles with text that expresses triplets about many entities in a single sentence or paragraph. This can be solved by different sampling of triplet sets when generating synthetic data for training, focusing on introducing variety of entities into them.

Real-world data performance. While BoostIE improved the overall performance on the sampled Wikipedia text, it is still far from perfect. Additionally, Wikipedia does not fully reflect the performance of our model in the wild, as it is still a very factual and structured text. In future work, it would make sense to perform a further evaluation on the real text, as it might help identify other failure modes.

References

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2024. Guiding llms the right way: Fast, non-invasive constrained generation. *Preprint*, arXiv:2403.06988.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. *Preprint*, arXiv:2010.00904.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Daniel Deutsch, Shyam Upadhyay, and Dan Roth. 2019. A general-purpose algorithm for constrained sequential inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 482–492, Hong Kong, China. Association for Computational Linguistics.

Saibo Geng, Berkay Döner, Chris Wendler, Martin Josifoski, and Robert West. 2024a. Sketch-guided constrained decoding for boosting blackbox large language models without logit access. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 234–245, Bangkok, Thailand. Association for Computational Linguistics.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2024b. Grammar-constrained decoding for structured nlp tasks without finetuning. *Preprint*, arXiv:2305.13971.

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*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1148–1158, Portland, Oregon, USA. Association for Computational Linguistics.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

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.

Martin Josifoski, Marija Sakota, Maxime Peyrard, and Robert West. 2023. Exploiting asymmetry for synthetic training data generation: SynthIE and the case of information extraction. *arXiv preprint arXiv:2303.04132*.

Terry Koo, Frederick Liu, and Luheng He. 2024. Automata-based constraints for language model decoding. *Preprint*, arXiv:2407.08103.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Filipe Mesquita, Matteo Cannaviccio, Jordan Schmidek, Paramita Mirza, and Denilson Barbosa. 2019. KnowledgeNet: A benchmark dataset for knowledge base population. 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 749–758, Hong Kong, China. Association for Computational Linguistics.

David Milne and Ian H. Witten. 2008a. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, page 509–518, New York, NY, USA. Association for Computing Machinery.

David Milne and Ian H. Witten. 2008b. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, page 509–518, New York, NY, USA. Association for Computing Machinery.

Riccardo Orlando, Pere-Lluís Huguet Cabot, Edoardo Barba, and Roberto Navigli. 2024. Retrieve, read and link: Fast and accurate entity linking and relation extraction on an academic budget. In *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand. Association for Computational Linguistics.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. 2024. Grammar-aligned decoding. *Preprint*, arXiv:2405.21047.

Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models. *Preprint*, arXiv:2201.11227.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Robert E. Schapire. 1990. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 1017–1024, Madison, WI, USA. Omnipress.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Jizhi Tang, Yansong Feng, and Dongyan Zhao. 2019. Learning to update knowledge graphs by reading news. 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 2632–2641, Hong Kong, China. Association for Computational Linguistics.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Bayu Distiawan Trisedya, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. 2019. Neural relation extraction for knowledge base enrichment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 229–240, Florence, Italy. Association for Computational Linguistics.

Denny Vrandečić. 2012. Wikidata: a new platform for collaborative data collection. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, page 1063–1064, New York, NY, USA. Association for Computing Machinery.

Chenxi Whitehouse, Clara Vania, Alham Fikri Aji, Christos Christodoulopoulos, and Andrea Pierleoni. 2023. WebIE: Faithful and robust information extraction on the web. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7734–7755, Toronto, Canada. Association for Computational Linguistics.

A Collection of Wikipedia text

We collect Wikipedia text using Wikipedia API, by randomly taking Wikipedia articles and extracting 1 chunk of text that has at most 4 sentences per each article. All the sentences have to be part of the same paragraph (i.e. we are not keeping chunks that contain "\n" in them).

B Wiki-cIE Code

Wiki-cIE Code is a fully synthetic dataset introduced by Josifoski et al. (2023). It was used for training the range of SynthIE models. The dataset consists of around 1.8M training data samples, 10K validation, and 50K test samples generated by the now discontinued OpenAI model, `code-davinci-002`. The data was synthetically made, starting from sampling triplet sets. Triplet sets are generated by a biased random walk on a subset of the Wikidata knowledge graph (Vrandečić, 2012). Text that corresponds to these triplets was then generated by an LLM. Each text sample was generated by providing a triplet set and asking the LLM to write the text that only expresses those triplets. This way, an exhaustive, high-quality data was made. The main disadvantage of this dataset is the fact that the text does not resemble real text, as it is very clean and does not contain a lot of details.

C Baselines

GenIE. Josifoski et al. (2022) introduce GenIE, an end-to-end autoregressive langauge model that does cIE, based on BART (Lewis et al., 2020). This model was trained on REBEL, the dataset made with distant supervision on Wikipedia abstracts. The method also employs constrained decoding. Given all of this, the model has issues that stem from constrained decoding, bad alignment between triplets and text in the data, as well as bad distribution of relations in the training set. We do not compare against GenIE as it was already shown by Josifoski et al. (2023) that it performs worse than SynthIE. We use it to generate DPO data (see Sec. 3.3).

SynthIE. As a part of efforts to mitigate some of the issues raised by GenIE, Josifoski et al. (2023) introduce SynthIE. This is a model trained on synthetic data, Wiki-cIE Code, that has better alignment between text and triplets, as well as better distribution of relations in the training set. However, SynthIE still uses constrained decoding, and the synthetic data it was trained on does not resemble real data, which causes issues when the model is used in practical settings.

ReLiK. Differently from SynthIE and our BoostIE models, ReLiK (Orlando et al., 2024) utilizes a retriever-reader architecture to solve cIE task. The retriever module encodes the input text and retrieves the most relevant entities and relations from the KB. Then, the reader module takes as input the text and each retrieved entity or relation separately and maps them to a specific span of the text. The modules for cIE were trained on REBEL dataset (Huguet Cabot and Navigli, 2021), raising a concern that this model might exhibit the issues with rare-relation performance.

D Metrics

We evaluate performance using standard precision, recall, and F1 metrics across all settings. A predicted fact is considered correct only if the relation and both associated entities are correct. Formally, let the set of predicted triples for a document $d \in \mathcal{D}$ be denoted as P_d, and the corresponding set of gold triples as G_d. Then, the micro-averaged precision and recall are defined as follows:

$$\text{micro-precision} = \sum_{d\in\mathcal{D}} |P_d \cap G_d| \Big/ \sum_{d\in\mathcal{D}} |P_d|, \quad (1)$$

and

$$\text{micro-recall} = \sum_{d\in\mathcal{D}} |P_d \cap G_d| \Big/ \sum_{d\in\mathcal{D}} |G_d|. \quad (2)$$

Micro scores provide a useful aggregate view of model performance, especially in terms of overall accuracy. However, they can obscure disparities in datasets with class imbalance—for instance, when certain entities or relations appear far more frequently in both training and test data. This is because micro-averaging gives equal weight to each instance, whereas macro-averaging assigns equal weight to each class. To account for such imbalances, we also report macro-averaged scores.

Let $P_d^{(r)}$ and $G_d^{(r)}$ denote the predicted and gold triples for relation $r \in \mathcal{R}$ in document d. Then, macro-precision is defined as:

$$\frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}}\left(;\sum_{d\in\mathcal{D}}|P_d^{(r)}\cap G_d^{(r)}| \Big/ \sum_{d\in\mathcal{D}}|P_d^{(r)}|\right), \quad (3)$$

and macro-recall as:

$$\frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}}\left(;\sum_{d\in\mathcal{D}}|P_d^{(r)}\cap G_d^{(r)}| \Big/ \sum_{d\in\mathcal{D}}|G_d^{(r)}|\right). \quad (4)$$

E Implementation

As mentioned in Sec. 3.2, BoostIE uses two FlanT5 models. For both models, we use 'google/flan-t5-base' version[3], which has ∼250M parameters. The models were trained using the Adam optimizer with a learning rate of 3e-4, 0.1 gradient clipping on the Euclidean norm, and a weight decay of 0.05. They were trained with batch size 80, for a maximum of 10K steps. We used a polynomial learning rate scheduler with 1000 warm-up steps and a final learning rate of 3e-05. All the experiments were run on a single NVIDIA Titan X Maxwell 12GB GPU, taking around 24h for the training of the base model, and around 16h for boosted model. The DPO finetuning was done on the same machine, using learning rate 5e-5, batch size 2, β 0.1 and running it for 5 epochs, taking around 20min to finetune. During inference, we run all our models with 10 beams.

F DPO data preprocessing

To collect the data for DPO finetuning, we first train a RobERTa classifier that distinguishes Wiki-cIE Code text from real Wikipedia text. We use the 'roberta-base' model[4] as the basis for our classifier. To do that, we take 5K samples from the Wiki-cIE Code training split (labeled as '0') and collected 5K samples of Wikipedia text (labeled as '1') in the way described in Appendix A. For the validation set, we collect in total of 3K samples in the same way. The classifier achieves an accuracy of 98.27% on the validation set, highlighting again how different SynthIE data is from the real Wikipedia one.

[3] https://huggingface.co/google/flan-t5-base

[4] https://huggingface.co/FacebookAI/roberta-base

G Human annotations

Construction of candidate triplet sets. We start by randomly choosing 50 samples of Wikipedia text. Since it is not trivial to annotate the text, as the knowledge of a whole KB with more than 2.6M entities and almost 900 relations, we attempt to get as exhaustive set of candidate triplets as possible by combining outputs from multiple models. For that, we use SynthIE, GenIE, ReLiK, BoostIE, and BoostIE with DPO.

Because these models were trained on different datasets, and have different strengths and disadvantages, by combining all of them, we are hoping to at least have a set of triplet candidates that include all the correct triplets, while also possibly including many incorrect ones. This procedure ensures that our precision estimate is correct, up to human error. For the recall, our estimation will not necessarily be correct, but the ranking of the models will stay the same (as they all might be missing some potential triplets that none of the models generated).

Instructions. The annotators were given instructions in Fig. 3.

Annotation task. To ensure quality results, our annotation was done by 2 Ph.D. students and 4 MSc students. None of them were familiar with our work, avoiding any possible biases. For each annotation sample, the annotator was presented with the text and list of candidate triplets. For each triplet, they had to decide whether the triplet is expressed in the text or not, based on the instructions provided in Fig. 3. The annotation was done in three stages. First, one Ph.D. student and all MSc students annotated the data, with each contributing to an equal part. Then, the second Ph.D. student annotated all the samples. Finally, one of the paper's authors resolved the conflicts.

H Additional analysis of cIE methods

H.1 Constrained vs. unconstrained generation

In Table 4, we show examples of SynthIE outputs in both constrained and unconstrained manner, on real Wikipedia text. Overall, SynthIE does open information extraction well (i.e. without KB constraints), but constrained decoding only works when there are not many deviations between facts in the text and the KB.

text	constrained	unconstrained
Slaughter fought for law and order with his six-shooter, a shotgun, and a repeating Henry rifle. He arrested desperados like the Jack Taylor Gang and brought them to justice.He also became a prominent poker player, often playing all night long. He was reportedly good at bluffing. He often played with the cattle baron John Chisum.	{('Slaughter_(band)', 'armament', 'Shotgun'), ('Slaughter_(band)', 'armament', 'Rifle'), ('Slaughter_(band)', 'armament', 'Henry_rifle')}	{('Slaughter_(poker)', 'armament', 'Henry_rifle'), ('Slaughter_(poker)', 'armament', 'Shotgun'), ('Slaughter_(poker)', 'armament', 'Six-shooter'), ('Slaughter_(poker)', 'unmarried partner', 'John_Chisum')}
Carol Dollard, who once worked in product development for PepsiCo, told Gladwell: "I've seen many times where the sip test will give you one result and the home-use test will give you the exact opposite.". For example, although many consumers react positively to the sweeter taste of Pepsi in small volumes, it may become unattractively sweet when drunk in quantity.	{('Carol_Douglas', 'instance of', 'Human'), ('PepsiCo', 'industry', 'Food_industry'), ('PepsiCo', 'product or material produced', 'Pepsi'), ('Carol_Douglas', 'employer', 'PepsiCo')}	{('Carol_Dollard', 'employer', 'PepsiCo'), ('Carol_Dollard', 'described by source', 'Gladwell_Encyclopedic_Dictionary'), ('PepsiCo', 'product or material produced', 'Pepsi')}
In June 1998, the founders became unhappy with the direction in which 3Com was taking the company, and left to found Handspring.	{('3Com', 'followed by', 'Handspring_(company)')}	{('3Com', 'replaced by', 'Handspring_(company)'), ('3Com', 'significant event', 'June_-1998')}
The merger also brought the game download site FilePlanet into the IGN group; as of 2011 both FilePlanet and the GameSpy website still operate as video game-related web sites.	{('Mergers_and_acquisitions', 'facet of', 'Software_as_a_service'), ('Mergers_and_acquisitions', 'facet of', 'Software'), ('Video_game_-industry', 'product or material produced', 'Video_game'), ('Video_game_industry', 'instance of', 'Industry_(economics)'), ('Mergers_and_acquisitions', 'facet of', 'Video_-game_industry')}	{('Merger_of_FilePlanet_and_GameSpy', 'business division', 'FilePlanet'), ('Merger_of_-FilePlanet_and_GameSpy', 'business division', 'GameSpy')}
Keith Taylor and Mike Scarrott called it the "Boomerang Nebula" in 1980 after observing it with the Anglo-Australian telescope at the Siding Spring Observatory. Unable to view it with great clarity, the astronomers saw merely a slight asymmetry in the nebula's lobes suggesting a curved shape like a boomerang.	{('Boomerang_Nebula', 'astronomical filter', 'Visual_perception'), ('Boomerang_Nebula', 'parent astronomical body', 'Sun'), ('Visual_-perception', 'subclass of', 'Perception')}	{('Boomerang_Nebula', 'site of astronomical discovery', 'Siding_Spring_Observatory'), ('Boomerang_Nebula', 'named after', 'Boomerang'), ('Boomerang_Nebula', 'discoverer or inventor', 'Keith_Taylor_(astronomer)'), ('Boomerang_Nebula', 'discoverer or inventor', 'Mike_Scarrott')}

Table 4: Examples of generation with SynthIE in constrained and unconstrained mode

Text	ReLiK	SynthIE (unconstrained)
The Verwall Alps are not a mountain range in Austria's Vorarlberg region, which borders the Samnaun Alps.	[['Verwall_Alps', 'country', 'Austria'], ['Vorarlberg', 'country', 'Austria'], ['Vorarlberg', 'location', 'Austria'], ['Samnaun_Alps', 'country', 'Austria']]	[['Verwall_Alps', 'different from', 'Vorarlberg'], ['Vorarlberg', 'shares border with', 'Samnaun_Alps']]
Windows Nashville was not a codename for a cancelled release of Microsoft Windows.	[['Windows_Nashville', 'edition or translation of', 'Microsoft_Windows']]	None
"The Land of Mist" is not a fantasy short story published in the Strand Magazine. It is not in the public domain.	[['The_Land_of_Mist', 'published in', 'The_-Strand_Magazine']]	[['The_Land_of_Mist', 'different from' 'Fantasy_short_story'], ['The_Land_of_Mist', 'published in', 'Strand_Magazine'], ['The_Land_-of_Mist', 'different from', 'The_Land_of_-Mist'], ['The_Land_of_Mist', 'copyright status', 'Public_domain']]
Münchner Illustrierte is not a German magazine.	[['Münchner_Illustrierte', 'instance of', 'Magazine']]	None
"Groovin' Blue" is not an album by Curtis Amy, released on Pacific Jazz Records.	[['Groovin\'_Blue', 'performer', 'Curtis_-Amy'], ['Curtis_Amy', 'record label', 'Pacific _Jazz_Records']]	[['Groovin\'_Blue, 'different from', 'Groovin'_-Blue_(Curtis_Amy_album)'], ['Groovin\'_-Blue', 'record label', 'Pacific_Jazz_Records']]

Table 5: Examples of outputs from ReLiK and SynthIE on negated Wiki-cIE Code data samples

TL;DR:

- You will be given 50 samples to annotate
- Each sample corresponds to one row in the sheet
- In the first column, you will be given the text that needs to be annotated
- The following columns will contain candidate facts in the form [subject; relation; object]
- Your task is to mark the checkbox left to the fact expressed in the text. Once you mark the checkbox, the cell of the validated fact will turn green for easier verification

Some things to note:

- It is possible that text doesn't express any of the proposed facts
- It is possible that there are facts in the text that are not proposed. You can ignore that, and only focus on the proposed ones
- Sometimes, there will be facts that are true, but are not explicitly, or even implicitly mentioned in the text. Do not mark those facts as expressed. Generally, only facts that are either explicitly or implicitly mentioned in the text should be marked. For more details, see the example below
- Sometimes, it might happen that you have a text about an entity (subject or object) that you cannot identify from your own knowledge from the way it is expressed in it. In those cases, you can try Googling the text provided, as it comes from Wikipedia, and the page should pop out. You will be able to identify the entity that way. Only do that if there are proposed facts that you cannot determine if you need to mark them or not. For more details, see the example below

Example:

Consider the sentence "President Biden is a politician born in Pennsylvania, USA in 1942".

First, in case you cannot identify who Biden is from this sentence (as it could be anyone with the surname Biden), in this case you would Google this sentence and find that this sentence is about Joe Biden.

Now consider the following proposed facts:

1. **[Joe_Biden; place of birth; Pennsylvania]** - you would mark this fact as correct, as it is explicitly stated in the sentence
2. **[Pennsylvania; located in the administrative territorial entity; United_Stated_of_America]** - you would mark this fact as correct, as it is implicitly mentioned in the text. In particular the part 'Pennsylvania, USA' indicates that Pennsylvania is located in the USA. Notice that here also you would need to know that USA represents Unites States of America
3. **[United_States_of_America; president; Joe_Biden]** - while this fact is true, it is not neither explicitly, nor implicitly mentioned in the text. Notice that while it is mentioned that Joe Biden is a president, it never mentions of which country, so you shouldn't mark this fact

Notice that the sentence also contains other facts, for instance [Joe_Biden; instance of; politician], but as this is not given as a proposed fact, so you cannot mark it.

Figure 3: Human evaluation instructions. Annotators are provided with the sheet with text and candidate triplets, and with the detailed instructions.

H.2 Analysis with negation

We suspect that, since ReLiK was trained to match retrieved entities and relations with spans of text identified as relevant, it is more likely to find a relation between two entities in the text that are not connected. To confirm this, we edit samples from the test split of Wiki-cIE Code, created by Josifoski et al. (2023) for the SynthIE model, by replacing "is" in text with "is not". For example: "Groovin' Blue is not an album by Curtis Amy, released on Pacific Jazz Records" would be used instead of "Groovin' Blue is an album by Curtis Amy, released on Pacific Jazz Records". We test both SynthIE (without constrained decoding) and ReLiK on this modified data. Neither of the two models performs well on this task, but there are indicators that SynthIE is able to somewhat model the lack of relation between two entities. In the case of ReLiK, this happens rarely. For examples, see Table 5

H.3 GPT-4o pipeline

In Sec. 5.3, we show some of the disadvantages of the current approaches with the smaller LMs. However, LLMs are more powerful in terms of their external knowledge, which can be a useful thing when extracting information facts. The pitfall with LLMs for this task is the KB. As they do not possess information about what is present in our KB, they are struggling to output the triplets in the correct format, or under correct constraints.

Under the assumption that one has unlimited resources for this task, we tried using GPT-4o with a form of retrieval-augmented generation (RAG). In this way, the LLM has the information about our KB. Here we present some of the key improvements to the standard prompt that resulted in better outputs (manually evaluated):

- **Entity retrieval**: We noticed that it is important for entity retrieval to be high-recall. This means that we did not care if many entities were not relevant, as long as all the relevant ones were included. GPT-4o seems to be able to filter the non-relevant entities, but cannot come up with the new ones. In our case, we used a mix of entities retrieved by ReLiK, SynthIE, and GenIE (both in an unconstrained setting). We did not include relation retrieval as we find this to be a harder task than entity retrieval, which requires the model to almost

Text	GPT-4o	GPT-4o + filter by entities
A film adaptation of Asada's work, directed by Yōjirō Takita, is known as When the Last Sword Is Drawn. A ten-hour 2002 television jidaigeki based on the same novel starred Ken Watanabe. A Japanese Manga Series, Rurouni Kenshin, by Nobuhiro Watsuki, notably sets place in the war, and the aftermath. Western interpretations include the 2003 American film The Last Samurai directed by Edward Zwick, which combines into a single narrative historical situations belonging both to the Boshin War, the 1877 Satsuma Rebellion, and other similar uprisings of ex-samurai during the early Meiji period.	('Rurouni_Kenshin', 'creator', 'Nobuhiro_Watsuki'), ('The_Last_Samurai', 'director', 'Edward_Zwick')	('Rurouni_Kenshin', 'creator', 'Nobuhiro_Watsuki'), ('The_Last_Samurai', 'director', 'Edward_Zwick')
Carol Dollard, who once worked in product development for PepsiCo, told Gladwell: "I've seen many times where the sip test will give you one result and the home-use test will give you the exact opposite.". For example, although many consumers react positively to the sweeter taste of Pepsi in small volumes, it may become unattractively sweet when drunk in quantity.	('PepsiCo', 'product or material produced', 'Pepsi')	('PepsiCo', 'product or material produced', 'Pepsi')
At the time he ran a chain of health food shops in the United States, with which she became involved. The couple now live in London and have two daughters.	('John_Henry_Richardson', 'residence', 'London')	None
However, the U.S. Bank Tower does surpass both the Salesforce Tower and the Wilshire Grand Center in roof height, making it the only building in California whose roof height exceeds 1,000 feet (300 m). Because local building codes required all high-rise buildings to have a helipad, it was known as the tallest building in the world with a rooftop heliport from its completion in 1989 to 2010 when the China World Trade Center Tower III opened.	('Wilshire_Grand_Center', 'roof height comparison', 'Salesforce_Tower'), ('U.S._Bank_Tower', 'located in', 'California'), ('U.S._Bank_Tower', 'has part', 'Heliport'), ('U.S._Bank_Tower', 'heliport timeframe end', 'China_World_Trade_Center_Tower_III')	('Wilshire_Grand_Center', 'roof height comparison', 'Salesforce_Tower')
Thorpe immediately is enchanted by Doña María and gallantly returns her plundered jewels. Her detestation of him softens as she too begins to fall in love.	None	None

Table 6: Examples generated by GPT-4o pipeline. Second column presents raw outputs after being prompted with our pipeline. Third column presents results where triplets containing entities which are not in the retrieved entities are removed.

be able to do cIE on its own. Theoretically, with LLMs that have longer context sizes, in our case, it is possible to send the whole list of relations. We did not test this but expect that this would improve the performance.

- **Sketch of triplet generation**: We noticed that GPT-4o produces better outputs when a sketch of a triplet generation by some other model is provided. Anecdotally, the outputs were better even when the sketches were bad. For the sketch, we used the output of the SynthIE model

- **Encourage reasoning**: LLM was performing vastly better when it was encouraged to explain the reasoning behind the choice of the triplets

We did not perform a formal evaluation of this method as it was not the focus of our study. All our findings from this section are based on manual inspection of the results. One thing we draw attention to is that LLMs have likely been exposed to the data we used for our manual inspection during their pretraining. Second thing to be careful about are rare relations. As they do not appear often, it is likely that an LLM would prioritize more common relations when generating the output. Regardless of that, we showcase our attempt as a starting point for the other researchers. For examples of generated outputs with GPT-4o on the real Wikipedia data, see Table 6.