GLaRef@CRAC2025: Should We Transform Coreference Resolution into a Text Generation Task?

Olga Seminck* and **Antoine Bourgois*** and **Yoann Dupont***
and **Mathieu Dehouck*** and **Marine Delaborde†**

* Lattice (CNRS UMR 8094 & ENS-PSL & Université Sorbonne Nouvelle), Montrouge, France
† LT2D (EA 7518, CY Cergy Paris Université), Cergy-Pontoise, France

olga.seminck@cnrs.fr, antoine.bourgois@protonmail.com,
yoann.dupont@sorbonne-nouvelle.fr, mathieu.dehouck@cnrs.fr,
marine.delaborde@cyu.fr

Your scientists were so preoccupied with whether they could, they didn't stop to think if they should.[1]

Abstract

We present the submissions of our team to the Unconstrained and LLM tracks of the Computational Models of Reference, Anaphora and Coreference (CRAC2025) shared task, where we ended respectively in the fifth and the first place, but nevertheless with similar scores: average CoNLL-F1 scores of 61.57 and 62.96 on the test set, but with very large differences in computational cost. Indeed, the classical pairwise resolution system submitted to the Unconstrained track obtained similar performance but with less than 10% of the computational cost. Reflecting on this fact, we point out problems that we ran into using generative AI to perform coreference resolution. We explain how the framework of text generation stands in the way of a reliable text-global coreference representation. Nonetheless, we realize there are many potential improvements of our LLM-system; we discuss them at the end of this article.

1 Coreference Resolution

Coreference resolution, the task of identifying and grouping textual linguistic expressions (mentions) that refer to the same entity, has been studied since the 1970s, beginning with rule-based systems for pronouns (Winograd, 1972; Hirst, 1981). The Message Understanding Conference (MUC) initiated a standardised framework for a coreference resolution shared task with the MUC-6 challenge (Grishman and Sundheim, 1995). Data-driven machine learning methods appeared with the availability of annotated corpora, initially in English. Subsequently, detection systems using statistical classifiers and pairs of mentions were developed (Soon et al., 2001), then mention-ranking systems like Denis and Baldridge (2008), usually in two stages: mention detection then coreference resolution. End-to-end global models later emerged and were evaluated in the CoNLL shared tasks (Pradhan et al., 2011, 2012). The arrival of deep neural models marked a turning point for the coreference resolution with models often inspired by Lee et al. (2017) later being replaced by BERT-based models (Joshi et al., 2019) and encoder-decoder architectures (Raffel et al., 2020), all contributing to improvements on benchmark datasets (Porada et al., 2024). In recent years, solutions based on seq2seq models (Zhang et al., 2023) and generative LLMs (Zhu et al., 2025) have also been proposed. These have been praised for their performance, while also revealing limitations (Gan et al., 2024); prompting reflection on the relevance of using such approaches for coreference resolution.

2 CRAC: Task Description and Corpora

The CRAC shared task 2025 is part of a series of annual challenges since 2016[2].

In 2024, the detection of zero mentions was added to the task[3] as were 4 new datasets (ancient Greek, Old Church Slavonic, Ancient Hebrew and English litBank) (Novák et al., 2024). CorPipe 24 (Straka, 2024), the winning entry in 2024, used a pretrained language encoder model with two variants: a two stages model (mentions detection then coreference resolution) and a single stage model.

Since 2025, the task corpus is based on CorefUD 1.3. (Novák et al., 2025) and contains 22 datasets for 17 languages, including for the first time ANCOR (Muzerelle et al., 2011), a French spoken language corpus. In addition to the Unconstrained track, a new LLM track was introduced this year, which

[1] Dr. Ian Malcolm - *Jurassic Park* (dir. S. Spielberg, 1993).

[2] https://corbon.nlp.ipipan.waw.pl/.

[3] With three possible starting points: coreference and zeros from scratch, coreference from scratch, refine the baseline.

focuses on using only large language models to resolve coreference, via prompting, fine-tuning, or in-context learning.

The Universal Anaphora corpus (which is the source corpus for the CRAC task) brings together independently created corpora in different languages. The different annotation schemes (when available) indicate that the concept of coreference can include various phenomena depending on each corpus. Indeed, some corpora contain annotations for all the referring expressions, while some others include selected expressions only, such as the English-LitBank corpus, which is annotated in coreferences only for a subset of entity types (Bamman, 2020). Despite efforts to standardise the format, some phenomena are represented differently in several languages. For example, zero mentions are generally represented by adding empty nodes to the UD trees, such as for the Spanish Ancora (Taulé et al., 2008). Yet, in the French Democrat corpus (Landragin, 2016), zero subjects are annotated on the verb[4].

3 System Descriptions and Results

Our team participated in both the Unconstrained and the LLM tracks submitting results for two entirely different systems. In this section, we describe the two approaches.

3.1 Unconstrained: Mention-Pair System

3.1.1 Architecture

The baseline system used in the Unconstrained track is a mention-pair based multi-stage coreference resolution system adapted from the existing *Propp* processing pipeline.[5]

As a first step, it extracts contextualized token embeddings using a frozen multilingual pretrained transformer encoder (`mt5-xl`[6]), applying overlapping sliding windows to capture maximum context and averaging embeddings across overlaps.

Mention spans are identified using stacked BiLSTM-CRF models trained to predict nested BIOES tags (Ratinov and Roth, 2009) at the sentence level. A separate BiLSTM model is used to identify head tokens for zero mentions.

Mentions are encoded using either the head token (for zero mentions) or the average of the first and last token embeddings (for multi-token spans). Mention-pair representations are the concatenation of the embeddings of two mentions with a rich set of linguistic and positional features, and are scored using a feedforward neural network.

To reduce complexity, the number of antecedent candidates is limited to 80 per mention. Clusters are formed using a highest-ranked-antecedent strategy and refined via transitive closure. Global decisions are improved through leveraging local high-confidence non-coreference links to avoid erroneous later merges.

3.1.2 Training and Computational Resources

Training our unconstrained system involves three main modules: mention detection, zero mention head detection, and coreference resolution. All components rely on word-level embeddings generated by the frozen encoder.[7]

- **Embedding Stage.** We use the `mt5-xl` model to extract contextualized embeddings for all tokens in the training and development datasets. The embedding model alone requires approximately 7.6 GiB of GPU memory. Processing all 12,187 documents (training + minidev) takes 55 minutes[8].

- **Mention Detection Stage.** The mention detection module is trained separately for each nesting level using the precomputed embeddings. The best models were obtained at epoch 23 (~4h46) for nested level 0 and epoch 21 (~4h32) for nested level 1, with a peak memory usage of 3.8 GiB.

- **Zero Mention Head Detection.** Trained similarly to the mention detection module, the best model was obtained after 24 epochs (~2h36), with a peak memory usage of 1.7 GiB.

- **Coreference Resolution.** The coreference resolution module is trained on all mention pairs using a batch size of 16,000 pairs per batch. The best model was obtained after 25 epochs (~3h57), with a peak memory usage of 1.8 GiB.

In the best-case scenario, the different modules are trained in parallel, so the total training time for the entire pipeline corresponds to the embedding

[4] A choice partly motivated by the annotation tool.
[5] https://github.com/lattice-8094/propp
[6] https://huggingface.co/google/mt5-xl

[7] More details about hyperparameters used for training each components can be found in the Appendix A.
[8] All experiments for the unconstrained track are performed on a single 48 GiB Nvidia RTX 6000 Ada Generation GPU.

time plus the duration of the longest individual module, resulting in a total of under 6 hours. Due to the size of the pretrained model used, the embedding step remains the most memory-intensive part of the pipeline and ultimately determines the minimum required GPU size (~8 GiB in our case).

Inference on the test set takes approximately 16 minutes, with peak GPU memory usage of 7.5 GiB. As with training, the embedding remains the main bottleneck, meaning that coreference resolution with this pipeline can be performed on any GPU capable of holding the embedding model.

3.1.3 Unconstrained Track Results

Despite its relatively simple design, our system achieves substantial improvements over the CRAC-2025 provided baseline (Table 1). On average, it yields a 5.56-point absolute gain in CoNLL F1-score across the test corpora. These gains are consistent across most languages, with particularly strong improvements observed on lower-resource corpora such as *grc_proiel* (+22.8), *hbo_ptnk* (+27.8), and *cu_proiel* (+12.8). This demonstrates the system's robustness and its capacity to generalize effectively across diverse linguistic settings.

Corpus	CRAC-coref	GLaRef
ca_ancora	68.01	**68.06**
cs_pcedt	56.94	**61.68**
cs_pdt	62.96	**66.59**
de_potsdamcc	55.70	**61.18**
en_gum	61.71	**61.86**
es_ancora	**70.52**	69.09
fr_democrat	54.99	**66.13**
hu_szegedkoref	54.54	**60.08**
lt_lcc	**65.35**	57.60
pl_pcc	66.55	**67.98**
ru_rucor	67.59	**71.45**
hu_korkor	43.17	**50.87**
no_bokmaalnarc	62.45	**67.09**
no_nynorsknarc	63.00	**66.28**
tr_itcc	**45.92**	44.28
cu_proiel	26.33	**39.10**
en_litbank	65.96	**69.96**
grc_proiel	28.54	**51.34**
hbo_ptnk	31.04	**58.80**
fr_ancor	63.77	**65.11**
hi_hdt	66.85	**69.51**
ko_ecmt	50.32	**60.57**
Average	**56.01**	**61.57**

Table 1: Test results for the Unconstrained track compared to the provided baseline (CRAC-coref).

Our system, adapted from the `Propp` architecture, follows a modular pipeline in which each stage depends on the previous one. This design introduces a key limitation: error propagation. The mention detection module plays a critical role, as errors at this stage directly affect downstream components such as mention pairing and clustering.

A notable challenge arises in datasets where singleton mentions (i.e., mentions not involved in any coreference chain) are not annotated. In such cases, the mention detector is trained only on spans that are part of coreference chains, resulting in an incomplete learning signal. This weakens its ability to identify valid mentions in general, particularly when the coreference resolution component depends entirely on the output of this detector.

This problem is further compounded by inconsistent annotation guidelines across datasets. As mentioned in Section 2, some corpora provide exhaustive mention annotations, while others are more selective. Such inconsistencies make it difficult for the system to generalize across languages and domains, and can lead to performance drops on datasets with different annotation guidelines.

3.2 LLM Track: Fine-tuning Gemma 3

For the LLM track, we developed two models based on fine-tuning of the `Gemma-3-12B-it` model using quantization and one single LoRA (Low-Rank Adaptation) (Biderman et al., 2024) adapter for all corpora. We proceeded to *peft* (parameter-efficient fine-tuning) with 4-bit NormalFloat quantization using QLora (Dettmers et al., 2024). The choice for the Gemma model was motivated by participation of members of our team in the shared task for Multilingual Grammatical Error Correction (MultiGEC-2025) (Masciolini et al., 2025), where they experienced particular problems with Llama 3 for under-resourced languages, in particular Icelandic and Slovene (Seminck et al., 2025). The task was won by a system build on Gemma 2 (Staruch, 2025), which is known to be a reliable multilingual model. Therefore, we decided to work with Gemma models for the current shared task.

We used the text2text-coref tool[9] provided by the CRAC organizers to transform the CoNLL data into a plain text format with in-text annotations and also to transform the system's output in plain-text back to CoNLL format. We proceeded to two distinct fine-tunings: a context-

[9] `https://github.com/ondfa/text2text-coref`

free model and a context-aware model. Our systems can be found on `https://github.com/lattice-8094/GLaRef-CRAC25-LLM-Track`.

3.2.1 Context-free Model

This model has the simplest design imaginable for coreference resolution using LLMs. We model the problem as just an annotation of coreference of the text: we give the whole text unannotated as an input, and the gold annotated text in the plain-text format as an output. The text is treated as a whole and there is no modelling of context. The prompt is given in (1). We experimented with different prompts, also leveraging `ChatGPT-4o` to enhance the prompt and give detailed instructions of the annotation schema. But in preliminary experiments, it turned out that a shorter prompt led to better performances and that the annotation schema can be learned implicitly during the fine-tuning of the model. Therefore, we kept a small prompt that is language agnostic.

(1) You are a linguist, expert in anaphora and coreference resolution. You have to annotate in the text which nouns, pronouns and other linguistic expressions refer to the same entity. Do only insert annotations. Do not insert extra linguistic material, nor punctuation markers and do not delete elements from the input texts.

Gemma 3 models can take up to 128K input tokens, so there is theoretically no problem of input length.

Our model was trained for 10 epochs, using batch size of one, for bigger batch sizes, the code threw an out of memory error. The training lasted about 3 days on two Nvidia RTX 6000 Ada Generation GPUs, featuring each 48 GB of memory capacity.

In Table 2, we can see that the results differ substantially across corpora. Whereas for some languages we observe scores above 70 points, for others the system's performance is poor. The main reason for this is the length of the texts per corpus. Despite the promise of handling up to 128K tokens of input, we soon realized that Gemma 3 was not capable of handling long texts properly, at least for this task, but it has been demonstrated for other tasks as well that output tends to degrade for longer texts, even if the maximum input length is respected (Levy et al., 2024; Liu et al., 2024). The system diverges from the original text when it is too long, for example by producing repetitive text (cycles), a well known problem of generative models (Fu et al., 2021; Ildiz et al., 2024). When the original text is not present anymore, it is impossible to gain points on in-text coreference resolution annotation. But what exactly a long text is depends on the language and the model's knowledge of the language. That has to do with the system's tokenizer. Tokens of under-resourced languages tend to be smaller than the ones of well-represented ones. This problem led us to the development of a second model.

3.2.2 Context-aware Model

The second fine-tuning splits the data into chunks of 8 sentences at a time. In the prompt, the most recent context (500 characters) that the model has already annotated is given, in order to preserve the coreference chains that were found earlier in the text.

If the chunk of sentences is the beginning of the text, the previous context is empty. In Example (2), we can see that the prompt is almost the same as the one of the Context-free Model.

(2) You are a linguist, expert in anaphora and coreference resolution. Based on the previous context, you have to annotate in the new sentence which nouns, pronouns and other linguistic expressions refer to the same entity.

Previous context: {gold_previous_context}

Do only insert annotations. Do not insert extra linguistic material, nor punctuation markers and do not delete elements from the input texts.

Before deciding to train a model with this context size, we experimented by giving it the entire context annotated thus far. It led to a disastrously bad result. Inspecting manually the output, it seems that the LLM does not 'understand' prompts that are too long. If there is already a long context that has been annotated, the LLM can no more make sense even of the task. We thus strictly restrained the given context to 500 characters (we choose characters in order to keep a similar context length across different languages in the corpora as token length is highly variable).

This model was trained on the same hardware as the Context-free model, but only on 3 epochs (mostly motivated by limited time and an increased number of training examples dues to cutting up long texts into chunks of 8 sentences). Training lasted about two days.

First, we tested the context-aware model by preprocessing the development and test datasets the same way as the training data (chunks of 8 sentences and a context of 500 characters). Again, the results can be found in Table 2.

What we first observe is that there are some 'FAIL' results. There are two types of FAIL:

(a) The system cannot predict the corpus due to 'Torch Dynamo Hit Recompile Limit' Errors.

(b) The system has produced output that is incompatible with the text2text-coref toolkit, which prevents it from producing a CoNLL file from a plain-text output of the model.

The first problem is caused by the on the fly construction of data to predict, which leads to recompilation of the NN graph. As every chunk is accompanied by the most recent annotated context, the model has to base each prompt on its previous output. This leads to prediction data that is unstable and incompatible with the Torch library (or at least disfavored by it). Even though we found after the deadline of this shared task that there is a parameter that can be changed to enlarge the capacity of the prompt cache (which would increase the tolerance of the system to changing the prompt), it would have slowed down the system even more, meaning that prediction times would even be higher than the 1,5 days it takes the system already to predict the test set. Another option to solve this problem in the future could be to create fixed-sized prompts at the subword level, using pruning or padding when necessary, to avoid recompilation.

The second problem can undoubtedly be solved by working on the transformation scripts. We solved a small part by searching for and deleting invalid hash-tag sequences. For example, in *en_gum*, the model often generated sequences of "##", which causes errors when executing the text2text-coref tool. Unfortunately, we did not have enough time to address all the text2text-coref related issues and hence, there are some corpora that we did not manage to predict. But in the end, our context-aware approach seems to solve the problem of long texts. The performance increases significantly for the majority corpora that we managed to process.

For some corpora on which the context-free model obtained good results, the context-aware model did not manage to improve the scores (for example ca_ancora or es_ancora). We noticed that these corpora feature rather short texts and our conclusion was that the 500 characters context given in the prompt might be too short. We therefore wanted to develop a new model that had a larger context. We also wanted to address the problem of the torch dynamo recompilation limit by making less requests by enlarging the chunks.

As time fell short, we decided to use the context-aware model trained on contexts of 500 characters and chunks of 8 sentences but with different prediction parameters without retraining. We predicted chunks of 10 sentences giving 700 characters of context. The results can be found in Table 2.

We see that for most corpora, this run yielded the best results and we got a number of FAILs that is much lower. However, there are corpora performing best in the 8sent_500char setting and even two corpora where the context-free model is the best. This indicates that the trade-off between smaller texts to predict (thanks to chunking) and having only access to the most recent context is different for each corpus, depending on the LLMs knowledge of the language, and the size of the texts. It seems that each corpus would have its own optimal parameters.

Our final submission, combined best scores of all the LLM-predictions, leading to an average score of 62.96.

4 Discussion

Even though our LLM-approach yielded the highest scores in the LLM track (with only one point ahead of the second best submission), performances of systems in the unconstrained track cannot be ignored. Indeed, when we just compare our two submissions (mention-pair and Gemma 3 fine-tuning), we have to conclude that performance is very similar. And that is without taking into account the fact that the winner of the unconstrained track, the corpipe-ensemble system, largely encompasses our endeavours with an average score of 75.84. So, an important question that needs to be asked is: is it worth the trouble to use LLMs for coreference resolution? After all, their use is very costly in computation resources. For example, the training time for our two submissions differs significantly: only 6 hours for the classic model versus 2 or 3 days for the LLM-based system. The gap is even more striking at inference time, where the unconstrained system requires approximately 16 minutes to process the test-set, compared to about a day and half for the

Corpus	C-F	8s_500c	10s_700c
ca_ancora	71.83	70.44	**73.45**
cs_pcedt	53.39	64.47	**65.12**
cs_pdt	70.13	FAIL-a	**71.33**
cu_proiel	8.92	57.22	**58.25**
de_potsdamcc	58.75	FAIL-b	**59.60**
en_gum	44.34	FAIL-a	**58.73**
en_litbank	44.00	64.70	**69.01**
es_ancora	**74.43**	71.72	72.61
fr_ancor	14.40	64.73	**66.74**
fr_democrat	16.85	**60.43**	FAIL-a
grc_proiel	13.68	**65.75**	65.16
hi_hdtb	56.36	51.64	**52.74**
hbo_ptnk	1.00	FAIL-b	**43.96**
hu_korkor	46.39	**52.53**	52.46
hu_szegedkoref	56.42	56.41	**59.82**
ko_ecmt	60.52	61.09	**63.04**
lt_lcc	56.38	**62.55**	62.28
no_bokmaalnarc	57.40	64.14	**64.74**
no_nynorsknarc	**61.63**	61.60	FAIL-a
pl_pcc	70.81	72.21	**72.55**
ru_rucor	65.40	68.26	**68.79**
tr_itcc	6.08	51.92	**56.23**
Average	**47.85**	**58.91**	**62.67**

Table 2: CoNLL F1-scores of the LLM track on the test set. C-F: Context-free. Xs_Yc: X sentences, Y characters. FAIL-a: Torch Dynamo Recompilation Limit Error. FAIL-b: Text2text-coref Tool Error.

LLM-based approach. This is a substantial difference for a performance that remains comparable to that of a traditional mention-pair system.

There is a lot of room for improvement in the design of our context-aware model. In the first place by optimizing the context size, the length of the chunks, pre-treatment of prompts to avoid recompilation problems, and the machine learning parameters —which would undoubtedly allow us to gain a number of extra points in performance— and in the second place by design modifications which we will discuss broadly in Section 5. But according to us, one of the core problems of using LLMs for coreference resolution is that it asks to transform coreference resolution into a text generation task. In the remainder of this section we will explain what are the fundamental problems of doing so.

When used for coreference resolution in the plain-text format, LLMs are optimized to perform annotation. So in fact, our context-aware model handles coreference as an annotation problem, that should be handled as a text generation problem. Although using LLMs for annotation tasks is commonly done (Tan et al., 2024), conceptually it has important consequences when dealing with coreference.

Firstly, it defines coreference resolution necessarily as an incremental task: chunks are annotated in the order of the text and this leads inevitably in making only local decisions. Even if, from a cognitive point of view of coreference resolution, it seems reasonable to treat coreference as incremental (Seminck, 2018), many coreference systems are in fact not incremental, for example our pair-wise system performs resolution based on highest scoring mention-pair clustering, instead of incremental clustering in the order of the text.

As a result, it takes away the abstract representation of coreference chains, by providing only local annotations on word levels in text. The text-global modeling of coreference is at best only implicitly present, but in the setting of our context-aware model, more likely, absent. This led to annoying mistakes in long text. What can happen is that when a context is presented with entities numbered for example ‘51’, ‘67’ and ‘98’, the system will use lower numbers, starting again from ‘1’ to annotate new coreference chains. Although we could imagine simple ways to prevent this behaviour (for example by explicitly stating in the prompt that it is forbidden to restart numbering from ‘1’), it would be interesting to think about a way to make the system aware of the coreference annotation of the entire text, without giving the entire annotated preceding context.

Lastly, we would like to point out the problem of transforming coreference resolution into a text generation problem. The objects we have to deal with are necessarily string variables and only string variables. Of course, this could be seen as a general problem for using generative AI for any scientific problem. Coreference resolution is particularly impacted by the previous problem: how to represent a global and abstract presentation of the coreference chains using only a single string variable?

Even though the learning power of LLMs is impressive and one can try to insert abstract representations into the prompt to be handled, the way the LLM treats this information is a black box. For the LLM this information is part of the string, just as both the original text and the text annotations: there is no actual distinction between these things. There is no guarantee that from the output, the original text, readable annotations and abstract global coreference chain annotations can be recovered. Of course, performance could be increased by enhancing the post-hoc scripts that parse and align the original text with the LLM-output by foreseeing more unwanted scenarios and creating patch-work solutions for them. But it does not change the problem fundamentally, we still have no guarantee of stability of our research objects.

Moreover, the larger the amount of additional information we may want to inject, treat with the LLM and then recover from the output, the lower the chances we actually succeed, as the probability of mixing up information increases. The LLM framework puts us out of control of the objects we want to calculate and manipulate. This is true for many uses of LLMs, stretching far beyond the problem of coreference. But we have to reflect on the question whether we can and want to accept it.

5 Future Work

Despite our conclusion that generative large language models are not easy to use to model coreference, participating in the shared task has given us a lot of ideas about how we could enhance our contribution next year. Even though we are not convinced that putting into practice these solutions would take away our reservations about the unsuitedness of text generation for coreference resolution, we are confident that they will enable us to increase significantly our scores. We will discuss these ideas and hope that we (or other teams and researchers) could benefit from them when developing new systems.

5.1 Improving Modelling of Coreference

Currently, in the context-aware model, as texts are split into chunks, the model never has access to the entire representation of coreference, as it is only implicitly present as the previously annotated most recent context. We could try to enhance the model by making it explicitly state all the clusters constructed so far and feed it as additional information into the prompt. Then, after annotation, extract the newly formed clusters and re-build the global coreference annotation. We expect this to help against restarting numbering coreference clusters from '1', but foresee the possibility that this representation might be unstable across the text, as it could be corrupted during text generation.

A second idea to improve the global representation of coreference resolution is to model a text in the memory of a chat conversation where each chunk is user-turn followed by a model's response. Although correctly memorizing very long conversations is still a challenge for LLMs (Maharana et al., 2024), we would like to test their abilities to keep track of global coreference chains using the memory of the chat conversation.

5.2 Task-Specific Loss Function

The fine-tuning we performed for the LLM track currently relies on the standard cross-entropy loss used in language modeling, as implemented in the `gemma-3-12b-it` model. However, this loss function is not well aligned with the specific needs of coreference resolution; while maintaining overall textual fidelity is important, assigning correct coreference identifiers is absolutely critical.

In standard text generation, two outputs such as `[e111]` and `[e112]` are nearly indistinguishable in terms of loss. The model is only minimally penalized for generating a slightly incorrect entity ID, even though such mistakes can drastically impact the coreference resolution.

One direction for future work would be to implement a task-specific loss function. After generating a batch of annotated text, we could compute a batch-level coreference evaluation metric (e.g. CoNLL F1-score). Though technically challenging, it could make LLM fine-tuning more sensitive to the actual goals of coreference resolution.

5.3 Improving the Input Format

The current plain-text format provided by the CRAC shared task uses a custom inline annotation style to mark entity spans and coreference chains. For example:

> Down the|**[e1** Rabbit-Hole|**e1]** Alice|**[e2]** was beginning to get very tired of sitting by her|**[e2],[e3** sister|**e3]** on the|**[e4** bank|**e4]** , and of having nothing to do : once or twice she|**[e2]** had peeped into the book her|**[e2],[e3** sister|**e3]** was reading

We propose exploring alternative tagging schemes better suited to LLMs, such as formats inspired by markup languages like HTML or XML. These clearly mark span boundaries with readable, nested tags, explicitly marking start and end of each span (<entity_start> </entity_end>):

> Down **<e1>**the Rabbit-Hole**</e1>** **<e2>**Alice**</e2>** was beginning to get very tired of sitting by **<e3><e2>**her**</e2>** sister**</e3>** on **<e4>**the bank**</e4>** , and of having nothing to do : once or twice **<e2>**she**</e2>** had peeped into the book **<e3><e2>**her**</e2>** sister**</e3>** was reading

Such a structure might be easier to tokenize and interpret by LLMs and may lead to better generalization and consistency in generation-based settings. Adopting this alternative would require adapting the conversion scripts from CoNLL-U to plain text, and from LLM outputs back to CoNLL-U. We believe this modification could help bridge the gap between coreference annotation conventions and LLM-friendly input formats, potentially improving model performance.

We could also try, together with the newly developed task-specific loss function, to fine-tune directly on the CoNLL-U format. This would limit error propagation caused by the transformation scripts.

5.4 LLM-based Pair-Wise Resolver

To limit the undesirable effects of text generation (loss of control on our study objects), we could split the coreference resolution task into sub-problems and come back to a pair-wise resolution system using LLMs. We would first use an LLM for mention detection, and then another for pair-wise classification, where pairs of mentions are classified as coreferent or not, fine-tuning the LLM to produce a binary response.

While this system would undoubtedly be computationally extremely heavy, as it asks for tens of thousands of calls to the LLM in order to perform pair-wise resolution, it would be an interesting experiment to see whether performance on the mention-detection and the pair-wise resolution increases with respect to classical systems, such as our mention-pair system. According to the results, we could also consider to replace a given module by an LLM-based system. If the LLM results are high but very costly computationally, we could also use it only for the more difficult cases of resolution. The current pair-wise system outputs confidence scores for its calculations, we could use the LLM-based system only for low confidence scores.

5.5 Student Training of LLM with Oracles

We only have access to gold data in order to fine-tune the coreference resolution systems. But, the incremental setting imposed by the LLM puts us in a situation where error propagation can be an issue. Therefore, we could want to teach the LLM to resolve coreference based on its previous predictions even if they contain errors. However, learning to predict the gold annotation given what has already been predicted (the context) can actually be detrimental. For example, if due to early errors, two chains have seen their indices swapped in the context, trying to predict the original gold indices is actually incoherent. To remedy this, we would need to relabel the current chunk to replace gold tags, given what has already been predicted in the context. This is computationally very expensive, likely NP-hard, given that the coreference metrics consider the annotation of the whole text. We consider to train oracles to predict good relabeling of the gold data at a reasonable cost, inspired by works done in syntactic parsing where oracles are trained to predict sequences of transitions of a system that reconstruct a parse tree (Coavoux and Crabbé, 2016; Shen et al., 2021).

6 Conclusion

We fine-tuned the `Gemma-3-12B-it` model to perform coreference resolution in the LLM track of the CRAC shared task and ended first. We found that our approach was adaptable to all the languages of the shared task, but that the systems were computationally very costly, especially compared to our pair-wise coreference resolution system submitted in the Unconstrained track of the CRAC shared task. Analyzing our results, we come to the conclusion that it is not obvious use generative LLMs for coreference resolution. Coreference resolution being a global discourse phenomenon, it is difficult to model it as a text generation task. Notwithstanding this fundamental problem, our work can be seen as one of the first attempts to fit the problem resolution task in the framework of LLMs and provides a rich ground for reflection on multiple areas of improvement for future work.

Acknowledgements

This research was funded in part by PRAIRIE-PSAI (Paris Artificial intelligence Research institute–Paris School of Artificial Intelligence, reference ANR-22-CMAS-0007).

References

David Bamman. 2020. Litbank: Born-literary natural language processing. *Computational Humanites, Debates in Digital Humanities (2020, preprint)*.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less. *Preprint*, arXiv:2405.09673.

Maximin Coavoux and Benoît Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 172–182, Berlin, Germany. Association for Computational Linguistics.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pages 660–669.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. 2021. A theoretical analysis of the repetition problem in text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12848–12856. Association for the Advancement of Artificial Intelligence (AAAI).

Yujian Gan, Massimo Poesio, and Juntao Yu. 2024. Assessing the capabilities of large language models in coreference: An evaluation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1645–1665, Torino, Italia. ELRA and ICCL.

Ralph Grishman and Beth Sundheim. 1995. Design of the muc-6 evaluation. In *Proceedings of the 6th conference on Message understanding*, pages 1–11. Association for Computational Linguistics.

Graeme Hirst. 1981. *Anaphora in natural language understanding: a survey*. Springer.

M Emrullah Ildiz, Yixiao Huang, Yingcong Li, Ankit Singh Rawat, and Samet Oymak. 2024. From self-attention to markov models: unveiling the dynamics of generative transformers. In *Proceedings of the 41st International Conference on Machine Learning*, pages 20955–20982.

Mandar Joshi, Omer Levy, Daniel S Weld, and Luke Zettlemoyer. 2019. Bert for coreference resolution: Baselines and analysis. *arXiv preprint arXiv:1908.09091*.

Frédéric Landragin. 2016. Description, modélisation et détection automatique des chaînes de référence (democrat). *Bulletin de l'Association Française pour l'Intelligence Artificielle*, (92):11–15.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15339–15353, Bangkok, Thailand. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, Bangkok, Thailand. Association for Computational Linguistics.

Arianna Masciolini, Andrew Caines, Orphée De Clercq, Joni Kruijsbergen, Murathan Kurfalı, Ricardo Muñoz Sánchez, Elena Volodina, and Robert Östling. 2025. The MultiGEC-2025 shared task on multilingual grammatical error correction at NLP4CALL. In *Proceedings of the 14th Workshop on Natural Language Processing for Computer Assisted Language Learning*, pages 1–33, Tallinn, Estonia. University of Tartu Library.

Judith Muzerelle, Anaïs Lefeuvre, Jean-Yves Antoine, Emmanuel Schang, Denis Maurel, Jeanne Villaneau, and Iris Eshkol. 2011. Ancor, premier corpus de français parlé d'envergure annoté en coréférence et distribué librement. In *TALN'2013, 20e conférence sur le Traitement Automatique des Langues Naturelles*, pages 555–563.

Michal Novák, Barbora Dohnalová, Miloslav Konopík, Anna Nedoluzhko, Martin Popel, Ondřej Pražák, Jakub Sido, Milan Straka, Zdeněk Žabokrtskỳ, and Daniel Zeman. 2024. Findings of the third shared task on multilingual coreference resolution. *arXiv preprint arXiv:2410.15949*.

Michal Novák, Martin Popel, Daniel Zeman, Zdeněk Žabokrtský, Anna Nedoluzhko, Kutay Acar, David Bamman, Peter Bourgonje, Silvie Cinková, Hanne Eckhoff, Gülşen Cebiroğlu Eryiğit, Jan Hajič, Christian Hardmeier, Dag Haug, Tollef Jørgensen, Andre Kåsen, Pauline Krielke, Frédéric Landragin, Ekaterina Lapshinova-Koltunski, and 23 others. 2025. Coreference in universal dependencies 1.3 (CorefUD 1.3). LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL).

Ian Porada, Xiyuan Zou, and Jackie Chi Kit Cheung. 2024. A controlled reevaluation of coreference resolution models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 256–263, Torino, Italia. ELRA and ICCL.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint conference on EMNLP and CoNLL-shared task*, pages 1–40.

Sameer Pradhan, Lance Ramshaw, Mitch Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the fifteenth conference on computational natural language learning: shared task*, pages 1–27.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Olga Seminck. 2018. *Cognitive computational models of pronoun resolution*. Ph.D. thesis, Université Sorbonne Paris Cité.

Olga Seminck, Yoann Dupont, Mathieu Dehouck, Qi Wang, Noé Durandard, and Margo Novikov. 2025. Lattice @MultiGEC-2025: A spitful multilingual language error correction system using LLaMA. In *Proceedings of the 14th Workshop on Natural Language Processing for Computer Assisted Language Learning*, pages 34–41, Tallinn, Estonia. University of Tartu Library.

Yikang Shen, Shawn Tan, Alessandro Sordoni, Siva Reddy, and Aaron Courville. 2021. Explicitly modeling syntax in language models with incremental parsing and a dynamic oracle. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1660–1672, Online. Association for Computational Linguistics.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

Ryszard Staruch. 2025. UAM-CSI at MultiGEC-2025: Parameter-efficient LLM fine-tuning for multilingual grammatical error correction. In *Proceedings of the 14th Workshop on Natural Language Processing for Computer Assisted Language Learning*, pages 42–49, Tallinn, Estonia. University of Tartu Library.

Milan Straka. 2024. Corpipe at crac 2024: Predicting zero mentions from raw text. *arXiv preprint arXiv:2410.02756*.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 930–957, Miami, Florida, USA. Association for Computational Linguistics.

Mariona Taulé, Maria Antònia Martí, and Marta Recasens. 2008. Ancora: Multilevel annotated corpora for catalan and spanish. In *Lrec*, volume 2008, pages 96–101.

Terry Winograd. 1972. Understanding natural language. *Cognitive psychology*, 3(1):1–191.

Wenzheng Zhang, Sam Wiseman, and Karl Stratos. 2023. Seq2seq is all you need for coreference resolution. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11493–11504, Singapore. Association for Computational Linguistics.

Lixing Zhu, Jun Wang, and Yulan He. 2025. Llm-Link: Dual LLMs for dynamic entity linking on long narratives with collaborative memorisation and prompt optimisation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 11334–11347, Abu Dhabi, UAE. Association for Computational Linguistics.

A Unconstrained Track Model Architecture and Hyperparameters

A.1 Mention Detection Model

A.1.1 Architecture

- **Locked Dropout** (0.5) applied to embeddings for regularization.
- **Projection Layer**: Highway network mapping $1024 \rightarrow 2048$ dimensions.
- **BiLSTM Layer**: Single bidirectional LSTM (256 hidden units per direction).
- **Linear Layer**: Maps 512-dimensional BiLSTM outputs to BIOES label scores.
- **CRF Layer**: Enforces structured consistency in predictions.

A.1.2 Training Parameters

- **Data Splitting**: 85%/15% train-validation split.
- **Batch Size**: 16 sentences per batch.
- **Optimization**: Adam optimizer (lr $= 1.4 \times 10^{-4}$, weight decay = 10^{-5}).
- **Learning Rate Scheduling**: ReduceLROnPlateau (factor = 0.5, patience = 2).
- **Average Training Epochs**: 22.
- **Hardware**: Trained on a single 48 GiB Nvidia RTX 6000 Ada Generation GPU.

A.2 Coreference Resolution Model

A.2.1 Architecture

- **Model Input**: 2,063-dimensional vector, composed of concatenated:
  - **CamemBERT embeddings**: Maximum context embeddings for both mentions (2 × 1,024 = 2,048 dimensions).
  - **Mention Features** (15 dimensions):
    - Mention length.
    - Position of the mention's start token in the sentence.
    - Dependency relation of the mention's head (one-hot encoded).
  - **Mention Pair Features** (8 dimensions):
    - Distance between mention IDs.
    - Distance between start and end tokens of mentions.
    - Sentence and paragraph distance.
    - Difference in nesting levels.
    - Ratio of shared tokens between mentions.
    - Exact text match (binary).
    - Exact match of mention heads (binary).
    - Match of syntactic heads (binary).
- **Hidden Layers**:
  - Three fully connected layers.
  - 1,900 hidden units per layer with ReLU activation.
  - Dropout rate of 0.6 for regularization.
- **Final Layer**:
  - Linear layer mapping from 1,900 dimensions to a single scalar score.
  - Output: Continuous value between 0 (not coreferent) and 1 (coreferent).

A.3 Model Training

- **Data Splitting**: 85%/15% train-validation split.
- **Batch Size**: 16,000 mention-pairs per batch.
- **Optimization**: Adam optimizer (**lr** $= 4 \times 10^{-4}$, weight decay = 10^{-5}).
- **Antecedent Candidates**: 80 maximum.
- **Antecedent Candidates**:
- **Hardware**: Trained on a single 48 GiB Nvidia RTX 6000 Ada Generation GPU.