Efficient Uncertainty Estimation for LLM-based Entity Linking in Tabular Data

Carlo Alberto Bono[1], Federico Belotti[2] and Matteo Palmonari[2]

[1] *Politecnico di Milano, DEIB, Via Ponzio 34/5, Milano, 20133, Italy*
[2] *Università degli Studi di Milano Bicocca, DISCo, Viale Sarca, 336, 20126, Milano*

Abstract
Linking textual values in tabular data to their corresponding entities in a Knowledge Base is a core task across a variety of data integration and enrichment applications. Although Large Language Models (LLMs) have shown State-of-The-Art performance in Entity Linking (EL) tasks, their deployment in real-world scenarios requires not only accurate predictions but also reliable uncertainty estimates, which require resource-demanding multi-shot inference, posing serious limits to their actual applicability. As a more efficient alternative, we investigate a self-supervised approach for estimating uncertainty from single-shot LLM outputs using token-level features, reducing the need for multiple generations. Evaluation is performed on an EL task on tabular data across multiple LLMs, showing that the resulting uncertainty estimates are highly effective in detecting low-accuracy outputs. This is achieved at a fraction of the computational cost, ultimately supporting a cost-effective integration of uncertainty measures into LLM-based EL workflows. The method offers a practical way to incorporate uncertainty estimation into EL workflows with limited computational overhead.

Keywords
Entity Linking, Uncertainty, LLMs

1. Introduction

Resolving ambiguity in tabular data by linking cell values to entity identifiers in a knowledge base, e.g., a Knowledge Graph (KG) such as Wikidata, is a fundamental integration challenge that impacts many knowledge-driven applications. The task is denoted by different terms: *Entity Linking* (EL) is borrowed from Natural Language Processing (NLP) [1, 2], *Entity Reconciliation* designates entity matching across data sources[1], and *Cell Entity Annotation* (CEA) concerns the annotation of cell values in tables [3]. The latter is part of the broader objective of understanding tabular data by matching it with a background KG, sometimes referred to as Semantic Table Interpretation (STI) [4].

Large Language Models (LLMs), which have achieved remarkable performance across a variety of NLP tasks, have also been proposed in the context of tabular data understanding [5, 6] and EL [7, 8, 9, 3], where an LLM is asked to link values to the correct entity from a set of candidates retrieved by dedicated components. Evidence suggests that medium-sized LLMs fine-tuned on the task and general-purpose top-tier LLMs achieve state-of-the-art results and generalization capabilities on different benchmarks [3].

However, LLM-based approaches to EL usually return only the label of the selected candidate, without providing explicit insights into the uncertainty associated with the model's decision. This limitation is exacerbated by the non-determinism in the generation process, since multiple runs on the same input may return different answers. In EL and similar matching and classification tasks, previous approaches for uncertainty estimation, including those using language models in combination with classifiers, e.g., TURL [10], associate the link with some confidence score. Confidence scores are explicitly included in the Reconciliation API[2] proposed by the W3C Entity Reconciliation Community Group. Since a human-in-the-loop process is frequently used to improve the quality of links in quality-critical applications [11, 12], the presence

OM 2025: The 20th International Workshop on Ontology Matching collocated with the 24th International Semantic Web Conference (ISWC 2025), November 2nd, 2025, Nara, Japan
carlo.bono@polimi.it (C. A. Bono)
0000-0002-5734-1274 (C. A. Bono); 0009-0008-0140-3318 (F. Belotti); 0000-0002-1801-5118 (M. Palmonari)

[1] https://www.w3.org/groups/cg/reconciliation/
[2] https://www.w3.org/community/reports/reconciliation/CG-FINAL-specs-0.2-20230410/

of confidence scores enables the identification of potential errors by prioritizing links that humans could review to improve the quality of the results. Understanding and quantifying the uncertainty of the LLM output is essential to ensure the robustness and trustworthiness of the model output, to highlight where the model is fragile or ungrounded, and to direct human intervention in order to maximize its efficacy.

An established way of measuring uncertainty in LLMs is to determine the likelihood of an answer being consistent over multiple independent generations [13, 14, 15]. In the context of LLMs, this introduces significant computational overhead, as the generation time scales linearly with the number of tokens in both the prompt and the output, even with caching techniques [16]. Additionally, if we consider that EL may be applied to large datasets, this approach may be unsustainable. To mitigate the resource demands of large-scale EL scenarios, we propose an efficient approach to estimate uncertainty in LLMs without relying on multiple generations during inference. More precisely, we propose an efficient self-supervised method that learns to estimate the uncertainty observed over multiple generations of an LLM using observables from a single generation. The method leverages token-level features, derived from the probability distribution over the output vocabulary, to train a lightweight regression model targeting the "true" observed uncertainty.

Although in this paper we focus on EL on tabular data, we believe that our method could be applied to different closed-form tasks that can benefit from uncertainty quantification.

This paper makes the following contributions:

- Formalizes uncertainty-aware EL on tabular data and proposes a self-supervised regressor that learns to approximate multi-shot uncertainty from single-shot token-level features;
- Introduces a lightweight, model-agnostic feature set from output-layer probabilities and optional intermediate-layer signals, requiring no task labels;
- Evaluates the approach across several instruction-tuned LLMs, showing how uncertainty-awareness can effectively lead to the detection of low-accuracy outputs, which can then be corrected to improve accuracy under a constrained review budget.

2. Related Work

Uncertainty in EL for Tabular Data. Associating confidence scores in matching tasks, including EL for tabular data, is a well-established practice [17, 18, 4], where scoring a list of retrieved candidates is a typical intermediate step. This is also true in approaches combining a pretrained language model with classifiers, such as TURL [10] and UNICORN [19]. While scores computed in pre-LLMs methods somehow support confidence estimation, less attention has been dedicated to systematic analyses of uncertainty estimation in this task. One approach based on a deep neural network proposes a supervised method that considers the matching score of the best candidate and its distance from the second-best to quantify confidence [20]. The authors show that computed confidence scores help prioritize links to revise with a progressive budget. To the best of our knowledge, recent approaches based on LLMs [7, 8, 9, 3, 5] have not investigated how to exploit uncertainty measures in generative approaches. Uncertainty measures may support the decision whether to link or not a top candidate (e.g., based on a threshold), including the detection of NIL entities (i.e., values associated with entities not in the KB) [4]. Additionally, little attention has been dedicated to the impact of the variability of the links predicted by LLMs under different generations within this task.

Uncertainty Estimation for LLMs. *Confidence Score* methods leverage single-shot proxies from output probabilities (e.g., entropy, log-probabilities, perplexity) but can be overconfident when wrong [21, 13, 22, 23, 24]; *Semantic Consistency* approaches, involving the generation of multiple outputs for the same prompt and measuring their consistency or semantic similarity, are effective but computationally expensive [22, 14]; *Supervised* methods learn calibrated uncertainty from features of generated text or hidden states and often outperform unsupervised heuristics, but require an annotated dataset [25, 26]; *Ensemble- and Bayesian-Inspired* approaches estimate approximate Bayesian uncertainty (e.g., deep ensembles, Monte-Carlo dropout) but are generally impractical at LLM scale [27, 28, 29]; *Verbalized and Self-Reported Uncertainty* methods improve interpretability by prompting models to report confidence, with mixed reliability across tasks [30, 31, 32]. Finally, [33, 34] employ *Uncertainty-Aware In-Context Learning* to filter/refine or to guide iterative prompting, improving reliability on open-ended tasks.

This work. Our approach intersects multiple-generations and supervised paradigms by leveraging a self-supervised uncertainty regressor that learns from multiple-generations outputs how to estimate uncertainty from a single generation. Although this work shares some conceptual similarities with [26], our method does not require supervised labels and focuses on tabular EL.

3. Methods

3.1. Problem Formulation

Let a tabular EL instance be defined by an input tuple (T, m, E_{KG}), where T is a table, m is a mention at coordinates (r, c), and $E_{\text{KG}} = \{e_1, \ldots, e_K\}$ is a retriever-provided set of candidate entities, dependent on the particular Knowledge Graph KG. An LLM M conditions on a prompt $\mathbf{x}$ built from (T, m, E_{KG}) and generates a textual answer y from which a deterministic post-processor extracts a selected candidate $\hat{e}(y) \in E_{\text{KG}}$. For non-deterministic decoding, repeated generations $\{y^{(i)}\}_{i=1}^{N}$ induce an empirical distribution $\hat{p}_{\text{ans}}(\cdot|\mathbf{x})$ over answers[3], from which an uncertainty $u(\cdot)$ can be estimated. Here, $u(\cdot)$ is the uncertainty score derived from the empirical distribution of generations[4]. Our goal is to output, for each $\mathbf{x}$, both a selected candidate $\hat{e}(y)$ and an uncertainty score $\hat{s}(y)$ that correlates with $u(\cdot)$ computed a posteriori from multiple generations. We learn this score via an uncertainty regressor h_ϕ from single-shot token-level features (cf. Section 3.4). At deployment, we use $\hat{s}(y)$ to flag instances for manual review. Algorithm 1 outlines the complete training (warm-up) and inference workflow.

3.2. Measures of Uncertainty

To quantify the uncertainty $u(\cdot)$ of LLM outputs, we adopt two widely recognized measures: Predictive Entropy (PE), which captures the uncertainty inherent to a model's answer distribution, and Semantic Entropy (SE), which considers semantic equivalence classes rather than raw answers for measuring uncertainty [14]. PE is defined as the entropy of the observed output sequences y, realizations of a random variable Y, conditional on an input sequence $\mathbf{x}$, such that:

$$PE(\mathbf{x}) = H(\hat{p}_{\text{ans}}(\cdot|\mathbf{x})) = -\sum_{y} \hat{p}_{\text{ans}}(y|\mathbf{x}) \log \hat{p}_{\text{ans}}(y|\mathbf{x}) \tag{1}$$

Lower PE indicates that the output distribution is concentrated around a single answer, while higher PE reflects uncertainty over multiple possible answers. SE is instead calculated over the distribution of answer meanings $\hat{p}_{\text{sem}}(\cdot|\mathbf{x})$[5], that is, over the semantic equivalence classes c observed on the output, where each class refers to a shared meaning:

$$SE(\mathbf{x}) = H(\hat{p}_{\text{sem}}(\cdot|\mathbf{x})) = -\sum_{c} \hat{p}_{\text{sem}}(c|\mathbf{x}) \log \hat{p}_{\text{sem}}(c|\mathbf{x}) \tag{2}$$

where $SE(\mathbf{x})$ is the same as in [14]. To enable a comparison between entropy measures computed over discrete distributions with varying support sizes, both measures are normalized by the logarithm of the number of unique outputs. We also report the sequence perplexity (PP) as a confidence baseline. For an answer $y = (y_1, \ldots, y_T)$ of T tokens,

$$PP(y) = \exp\left(-\frac{1}{T} \sum_{t=1}^{T} \log p\big(y_t | y_{<t}, \mathbf{x}\big)\right) \tag{3}$$

where $p(y_t|y_{<t}, \mathbf{x})$ is the model's next-token probability.

[3]In particular, $\hat{p}_{\text{ans}}(y|\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[y^{(i)} = y]$, where $\mathbb{I}[x] = 1$ if the predicate x holds, and 0 otherwise.
[4]In Section 3.2 we instantiate it using the entropy of $\hat{p}_{\text{ans}}$ and, via semantic grouping [14], of $\hat{p}_{\text{sem}}$.
[5]Let $g : \mathcal{Y} \to \mathcal{C}$ map an answer y to its semantic class $c = g(y)$ [14]. The empirical distribution over semantic classes is $\hat{p}_{\text{sem}}(c|\mathbf{x}) = \sum_{y \in \mathcal{S}(\mathbf{x})} \hat{p}_{\text{ans}}(y|\mathbf{x})\mathbb{I}[g(y) = c] = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[g(y^{(i)}) = c]$, where $\mathcal{S}(\mathbf{x})$ is the set of observed distinct answers.

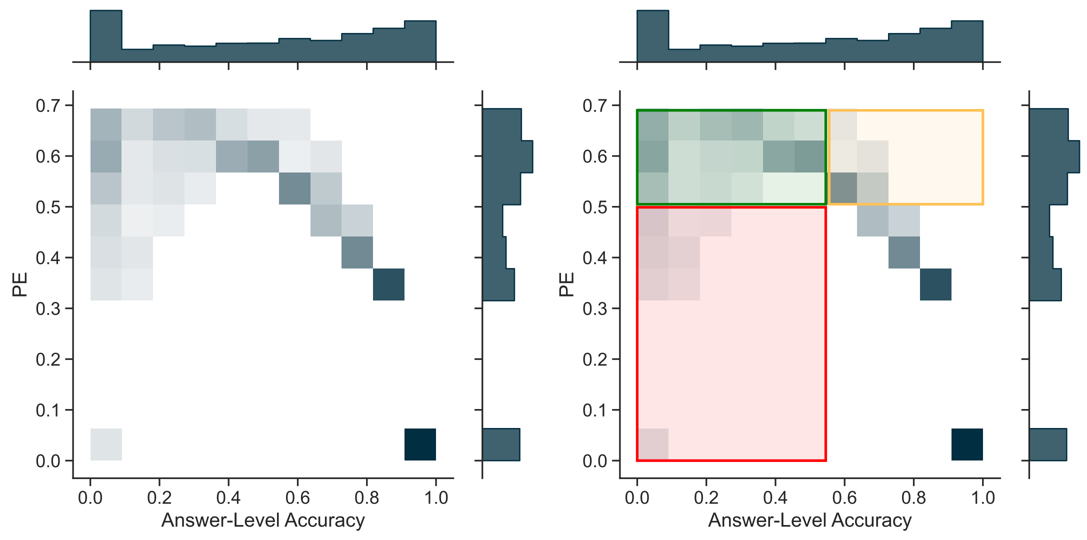

Figure 1: (left) Joint distribution of answer-level accuracy and Predictive Entropy (PE) per input for *Llama-3.1-8B-Instruct*, with marginal distributions on the axes. **(right)** We define the positive class as “flag for manual review”. Thresholding uncertainty yields: green = correctly flagged low-accuracy cases (true positives), yellow = incorrectly flagged high-accuracy cases (false positives), red = consistently wrong but low-uncertainty cases (false negatives; not recoverable by uncertainty thresholding).

3.3. Uncertainty Estimation

Our practical objective is to estimate uncertainty and use it to flag uncertain answers for manual review, without relying on an explicit ground truth. The underlying intuition is that higher uncertainty is correlated with lower accuracy, while low uncertainty alone does not imply higher accuracy. Accuracy is calculated as the average answer correctness across N generated outputs for each input prompt.

Uncertainty–Accuracy Relationship. Figure 1 highlights the effect of setting a threshold on uncertainty to flag items for manual review. Interpreting the flag decision as a binary classifier, the **green** area contains correctly flagged low-accuracy cases (true positives). Reviewing these comes at the cost of also reviewing cases in the **yellow** area (false positives), where accuracy would be satisfactory but uncertainty is high. Cases in the **red** area are consistently wrong but with low variability; these are false negatives and are not recoverable by uncertainty-based thresholding. Intuitively, lowering the uncertainty threshold expands the region of flagged cases. On the other hand, raising the threshold shrinks both green and yellow zones, reducing workload at the risk of missing additional bad cases. Notably, the subset of red points with zero uncertainty, which are consistently wrong across generations, remains invariant to the threshold and unrecoverable by uncertainty thresholding alone. Sweeping the threshold traces the ROC analysis discussed later.

Efficient Uncertainty Estimation via Self-Supervised Regression. A large class of uncertainty estimation methods, including widely used metrics like PE and SE, depend on repeated sampling for their computation, which comes with significant resource overhead. We therefore propose an efficient self-supervised method to approximate such uncertainty measures using information from a single generation, making them more suitable for downstream use. The target variable – either PE or SE, measured on N independent runs of an LLM – is regressed from observable features using a Random Forest model implemented via `XGBRFRegressor`, with 100 estimators [35, 36]. We optimize the Mean Squared Error (MSE) between the predicted score $\hat{s}(y)$ and the normalized multi-shot target $u(\hat{p}_{\text{ans}})$ (Section 3.2). We utilize 10-fold cross-validation grouped by prompt to avoid data leakage and to obtain reliable performance estimates on unseen cases. The regressor is trained in a warm-up phase, during which N generations per prompt are collected and the derived target variable is learned. At runtime, the regressor estimates the uncertainty from a single generation in response to the same prompt. We denote the learned uncertainty regressor by h_ϕ, which maps features $F(M, \mathbf{x})$ extracted from a target model M's single generation $y = M(\mathbf{x})$ to an estimate $\hat{s}(y) = h_\phi(F(M, \mathbf{x}))$. The method is self-supervised as it does not require externally labeled data. Moreover, the regression incurs negligible computational overhead. Further considerations regarding learning convergence are provided in Section 4.5.

We collect the features, as defined in Section 3.4, from N independent runs of an LLM. We benchmark the proposed method considering: (1) the first 10 generated tokens, and (2) all the tokens from the fixed-width tail of the prompt[6]. This allows us to investigate the contribution of different prompt segments to the quality of the uncertainty estimates. We also evaluate different feature sets as input, as detailed in Section 4.2, to assess the relative importance of individual feature groups.

3.4. Features for Uncertainty Estimation

We rely on features $F(M, \mathbf{x})$ that are observable or easily computable during the inference process of an LLM M. We compute token-wise features for specific portions of the prompt and generated tokens. Instead of aggregating the features at the sequence level [13], we maintain the features at the token level. Let V be a vocabulary of tokens and let $\mathbf{x}$ be a sequence of tokens from V. Given the input prompt $\mathbf{x}$, an LLM computes a logit vector $\mathbf{z} = M(\mathbf{x}) \in \mathbb{R}^{|V|}$ for each token. It then applies a softmax function to derive the stochastic vector $\mathbf{p}$, i.e., the probability of t being the next token for every $t \in V$:

$$\mathbf{p} \in \mathbb{R}^{|V|} \text{ s.t. } p_t = \sigma(\mathbf{z})_t = \frac{\exp(z_t/\tau)}{\sum_{v \in [|V|]} \exp(z_v/\tau)}$$

where τ is the temperature parameter, which controls the randomness of the output distribution. We calculate for each token[7]:

- *Max probability.* The maximum probability observed across the output vocabulary, measured at the output layer L of the model.

$$\mathrm{M}(\mathbf{p}) = \max_{t \in [|V|]} p_t$$

- *Entropy.* The entropy of the token probability distribution

$$\mathrm{H}(\mathbf{p}) = -\sum_{t \in [|V|]} p_t \log p_t$$

  over all vocabulary tokens at the output layer L. Lower entropy indicates higher confidence, while higher entropy reflects more uncertainty.
- *LogitLens* [37]. The Kullback-Leibler (KL) divergence between the probability distribution at each intermediate layer l and the probability distribution at the output layer L [37].

$$\mathrm{KL}(\mathbf{p}^l \parallel \mathbf{p}^L) = \sum_{t \in [|V|]} p_t^l \log \frac{p_t^l}{p_t^L}$$

  where $\mathbf{p}^l$ is measured for each layer $l \in [L-1]$ and $\mathbf{p}^L$ is measured on L, resulting in $L-1$ divergence measures quantifying the deviation between the distribution at the intermediate layers and the output layer. At each layer l, $\mathbf{p}^l = \sigma(\mathbf{z}^l) = \sigma\left(\mathrm{LayerNorm}[\mathbf{h}^l] W_U\right)$ is obtained by applying the last LayerNorm to the hidden representation $\mathbf{h}^l$ and multiplying it by the unembedding matrix W_U.

[6]The token indices collected in (2) are fixed across models, whereas those in (1) are model-dependent.
[7]We use the notation $[K]$ to indicate $\{1, ..., K\}$.

Workflow overview for self-supervised uncertainty estimation

Warm-up (offline)

1. For each prompt $\mathbf{x}$, collect N independent generations from the LLM M to obtain $\{y^{(i)}\}_{i=1}^{N}$.
2. Compute the a posteriori uncertainty target $u(\hat{p}_{\text{ans/sem}})$ either as normalized PE or SE (Equations 1 and 2, respectively).
3. For each generation $i \in [N]$, extract per-generation features $F^{(i)}(M, \mathbf{x})$ using token-level observables (Section 3.4) over selected segments (e.g., *Postilla*, first G *Generated* tokens).
4. Form training pairs $\left(F^{(i)}(M, \mathbf{x}),\ u(\hat{p}_{\text{ans/sem}})\right)$ for $i = 1, \ldots, N$ and for all prompts $\mathbf{x}$; train the regressor h_ϕ on the training pairs.

Inference (online)

1. For each new prompt $\mathbf{x}$, run a single generation to produce an answer $y = M(\mathbf{x})$ and extract $F(M, \mathbf{x})$.
2. Return y and predict $\hat{s}(y) = h_\phi(F(M, \mathbf{x}))$ and use it to decide if the item should be reviewed.

3.5. Runtime Considerations

At inference time, the proposed approach reduces the number of LLM generations used to estimate uncertainty from N to 1 and adds only a lightweight regression pass on features extracted during generation. In typical settings, the regression overhead is negligible compared to a single forward pass of the LLM; thus, single-shot estimated PE/SE achieves most of the benefit of multi-shot uncertainty at substantially lower cost. For a theoretical analysis of Transformer time complexity with and without KV-cache during generation, see Appendix F.

4. Experimental Evaluation

In this work, we address the following research questions:

Q1 Do our single-shot uncertainty estimates identify low-accuracy answers?
Q2 How much does uncertainty-guided manual correction improve accuracy under a budget B?
Q3 How much warm-up data is needed to learn the uncertainty regressor h_ϕ effectively?
Q4 How does temperature τ affect the uncertainty/accuracy trade-off?

Reproducible code and plotting scripts are available at: https://github.com/carloalbertobono/llm-uncertainty.

4.1. Dataset

We address the Entity Linking (EL) task on tabular data, formalized in Section 3.1, where each table mention must be linked to its corresponding entity in a Knowledge Base. The dataset used, *TableInstruct-EL-2K*, is adapted from the TableInstruct EL test set [5], which is formatted for LLMs and contains 2,000 mentions annotated with Wikidata entities, with exactly one correct entity per mention. The original dataset included $\sim$600 mentions with only one candidate (i.e., the correct one), which limits our ability to evaluate answer variability. To address this issue, the candidates were enriched with those retrieved using LamAPI [38], a full-fledged retriever that returns richer candidate sets. In the resulting dataset, 1,650 mentions ($\sim$91%) include at least 45 candidate entities each. Prompts follow the "Entity Linking" template from [5] and include the following segments: an *Instruction* that provides context and task guidelines; an *Input* table, Markdown-serialized; a *Question* asking which of the provided referent candidates corresponds to a specific table mention; a fixed-width *Postilla* that clarifies the expected answer format. The *Generated* segment contains the model's predicted answer. To support understanding, a minimal, self-contained illustrative example of the prompt, candidate list, and expected answer format is provided in Appendix A.

Table 1
Distribution of answer-level accuracy over N=10 non-deterministic runs, grouped by recoverability. *Unrecoverable* cases ("Always correct" and "Never correct (w/o Unc.)") have zero observed output variability; *Recoverable* cases ("Never correct (w/ Unc.)" and "Sometimes correct") exhibit variability that uncertainty thresholding can surface. The rightmost column reports the review-eligible fraction (Recoverable total $=$ *Never correct (w/ Unc.)* $+$ *Sometimes correct*). Rows sum to 1.

Model	**Unrecoverable**		**Recoverable**		**Recoverable total**
	Always correct	*Never correct (w/o Unc.)*	*Never correct (w/ Unc.)*	*Sometimes correct*	
Gemma-2-2B-Instruct	0.18	0.06	0.36	0.40	0.76
Gemma-2-9B-Instruct	0.67	0.10	0.09	0.14	0.23
Llama-3.1-8B-Instruct	0.15	0.01	0.17	0.67	0.84
Qwen2.5-7B-Instruct	0.40	0.02	0.15	0.43	0.58
TableLlama	0.79	0.03	0.01	0.17	0.18

4.2. Experimental Settings

Experiments are performed using the following models: *Gemma-2-2B-Instruct*, *Gemma-2-9B-Instruct*, *Llama-3.1-8B-Instruct*, *TableLlama*, *Qwen2.5-7B-Instruct*. These models were chosen as a representative set of instruction-tuned, open-source language models. Additionally, *TableLlama* is considered state-of-the-art in the Entity Linking task on tabular data [3]. The number of generations for each prompt is set to $N = 10$, while the temperature τ is set to 1.0. Features (observables) are extracted over different portions of tokens, specifically from the *Postilla* and from the first 10 tokens of the *Generated* segment[8], considering features from the output layer alone ($\mathrm{M}(\mathbf{p})$ and $\mathrm{H}(\mathbf{p})$), from the intermediate layers alone (LogitLens), and combined. These combinations are motivated by the hypothesis that uncertainty may manifest differently across prompt segments and feature groups, necessitating an empirical assessment of their impact. We compare the entropy values derived with our method against the following baselines: PE, SE, PP, and an oracle with access to the true answers.

In the remainder of the paper, we utilize two different concepts of accuracy. **Answer-level** accuracy refers to the accuracy of the answers to a given prompt, computed over multiple generations. Answer-level accuracy can then be aggregated at the level of a set of prompts; we refer to **dataset-level accuracy** to indicate the average answer-level accuracy computed over the whole dataset.

4.3. Task Accuracy and Recoverable Errors

Regarding the performance of the different LLMs on the task, Table 1 summarizes the proportion of answers with and without uncertainty – *Recoverable* and *Unrecoverable*, respectively – considering $N = 10$ runs per item. Answers with no associated uncertainty are grouped into *Always correct* and *Never correct*; answers with associated uncertainty are grouped into *Never correct* and *Sometimes correct*. The rightmost column (Recoverable total) adds *Never correct (w/ Unc.)* and *Sometimes correct* to give the recoverable fraction, i.e., the share of items that uncertainty thresholding can, in principle, surface for manual correction. It should be noted that *Never correct* items are partitioned into zero-variance items (*w/o Unc.*) and items with observed uncertainty (*w/ Unc.*).

Gemma-2-2B-Instruct, the smallest model in terms of parameters, and *Llama-3.1-8B-Instruct* show the lowest *Always correct* proportion, suggesting limited reliability. In contrast, *TableLlama* and *Gemma-2-9B-Instruct* show the highest *Always correct* proportion, with *Qwen2.5-7B-Instruct* falling in between. The Recoverable total highlights the best-case pay-off of uncertainty-guided review. In general, the fraction of unrecoverable-never-correct cases remains modest.

[8]The performance differs depending on whether features are extracted from the *Postilla* or *Generated* tokens. A more detailed analysis can be found in Appendix C.

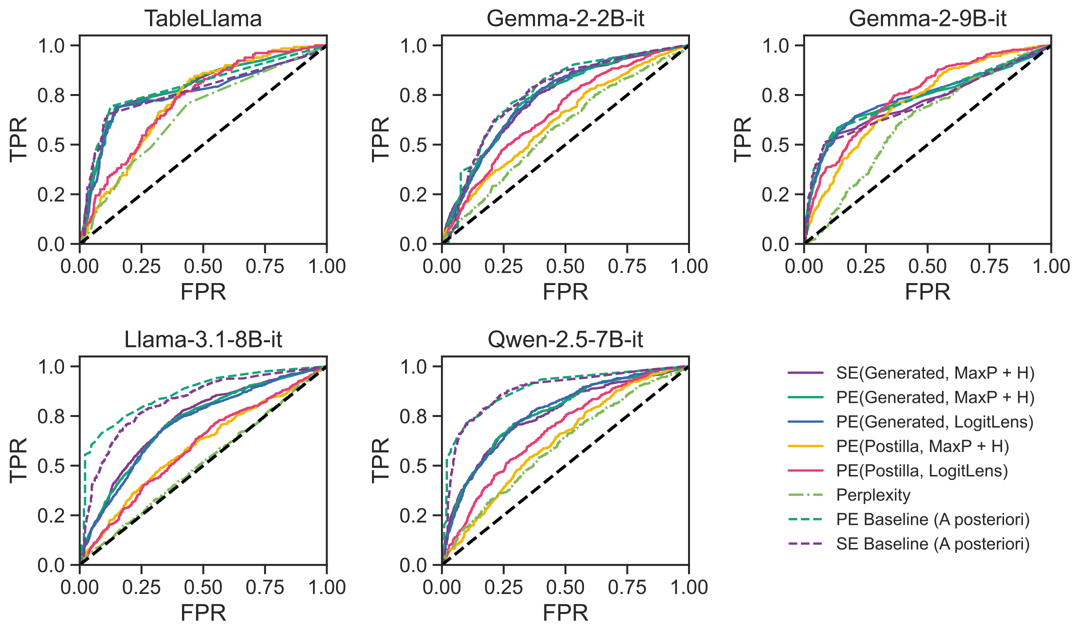

Figure 2: ROC analysis targeting low-accuracy (< 0.5) cases for selected models. Positive = "flag for manual review". In the legend, the notation "`Target(Segment, Observable)`" indicates that the target variable *Target* was predicted using a regressor trained on *Observable* features extracted from the *Segment* portion of the prompt. Dashed lines referring to *PE/SE Baseline (a posteriori)* represent the multiple-generations PE and SE computed over $N = 10$ generations.

4.4. Q1 and Q2: Uncertainty Estimates Assessment

We assess the reliability of our estimates through a series of targeted experiments. First, we test whether the uncertainty measures can identify mentions with low answer-level accuracy, where "low" means that the answer-level accuracy is below 0.5 over $N = 10$ generations. To this end, we perform a ROC analysis [39] on both the estimated and baseline entropy measures. We report the results in terms of true positive rate (TPR) and false positive rate (FPR), where TPR corresponds to the actual low-accuracy cases that the uncertainty-based method correctly flags. At the same time, FPR is the fraction of high-accuracy cases (accuracy ≥ 0.5) that the method wrongly flags as low-accuracy. Here, the positive class corresponds to the decision "flag for manual review." In the ROC plot, each curve traces the trade-off between TPR and FPR as a threshold on the uncertainty score—used to decide "flag or not"—is varied. The diagonal line represents random chance; curves that bow upward (high TPR at low FPR) indicate effective uncertainty signals for identifying low-accuracy items.

Moreover, we assess the impact of the uncertainty measures on budget-dependent manual correction. To this end, we investigate the average accuracy obtained by selecting the most uncertain items up to a budget B, according to a given uncertainty measure. We assess the aggregate accuracy at the dataset level after correcting a fraction B of the items.

Detecting Low-Accuracy Answers We evaluate how well the uncertainty estimate can identify *low accuracy* items (answer-level accuracy ≤ 0.5). Figure 2 shows the ROC curves for the considered LLMs, comparing the different uncertainty estimates and baselines. ROC curves are a well-established metric for assessing classification performance, and illustrate the trade-off between true positive rate and false positive rate across different thresholds, in our case, on uncertainty estimates. PE and SE based on N generations (dashed lines) show the highest performance. The gap between these baselines and the estimated PE/SE (solid lines) captures the performance loss due to estimating the multiple-generations entropy with the information from a single generation. Notably, when using the best-performing models, *TableLlama* and *Gemma-2-9B-Instruct*, PE and SE computed on the N multiple generations closely approximate the estimated PE and SE (cf. green and purple solid and dashed lines). However, across

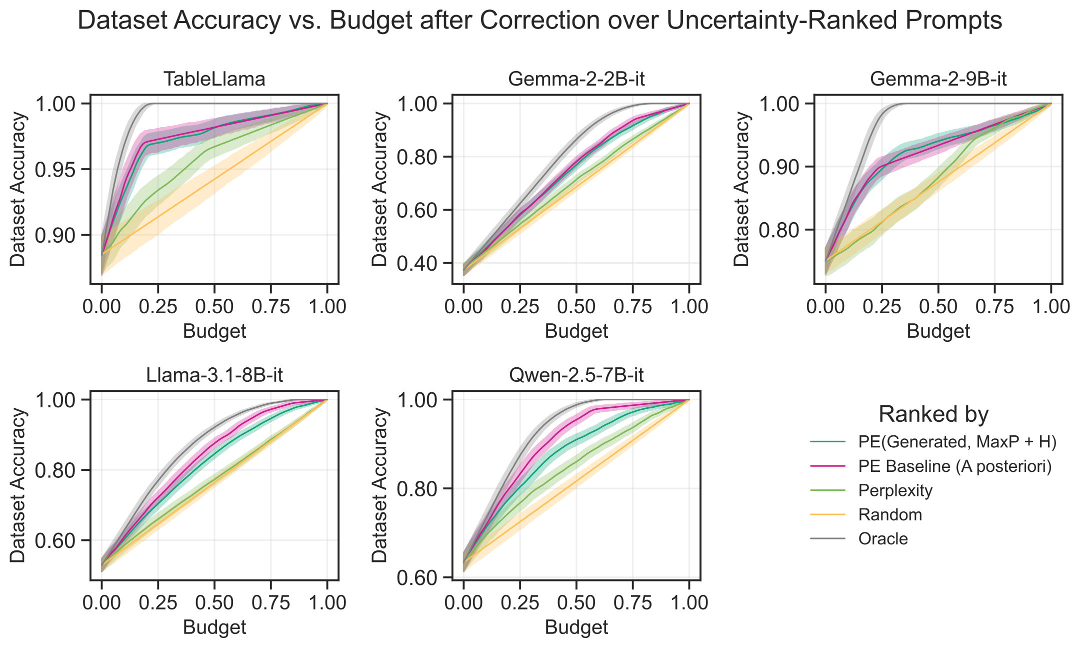

Figure 3: Dataset accuracy as a function of the budget B of items corrected, where ranking is based on the measures shown in the legend. The shaded areas correspond to the 95% C.I. estimated via $1{,}000$ bootstrap resampling iterations [40]. Each curve illustrates how accuracy improves as more high-uncertainty prompts are corrected. In the legend, the notation "`Target(Segment, Observable)`" indicates that the target variable *Target* was predicted using a regressor trained on *Observable* features extracted from the *Segment* portion of the prompt. *PE/SE Baseline (a posteriori)* represent the multiple-generations PE and SE computed over $N = 10$ generations.

other models, the PE and SE baselines show a better ability to distinguish high- and low-accuracy cases. Moreover, PP is consistently outperformed by other methods. In general, the performance of our method falls between PE/SE and PP. We recall that the estimates of our method are obtained at a fraction of the computational cost, by computing $N = 1$ generations instead of $N = 10$.

Regarding the use of features derived from different stages of the generation process, estimates based on the features observed in the *Postilla* segment appear to be generally less informative compared to the *Generated* segment. However, focusing on the utilization of the LogitLens features, we highlight that their contribution is substantial when using the *Postilla* features, while it becomes negligible when using the *Generated* features.

Uncertainty-Guided Correction under Budget B We evaluate how much accuracy improves when a human annotator—assumed to be always correct—uses a limited budget B to manually revise the most uncertain cases. Items are ranked by the various uncertainty signals, and the top B fraction is corrected; the resulting dataset-level accuracy after these ideal corrections is shown in Figure 3. Each curve corresponds to a different ranking method: the proposed PE estimate (e.g., PE(Generated, MaxP+H)), the a posteriori PE/SE baseline, perplexity, random selection, and an oracle that ranks strictly by true low-accuracy severity. The gray oracle curve defines the upper bound, while the yellow random curve gives a reference for uninformed correction. Across the evaluated models, uncertainty-guided correction substantially outperforms random selection, with the largest marginal gains at small budgets. The single-shot regressors closely track the multi-shot PE/SE baselines, recovering most of their improvement while reducing LLM calls from N to 1, whereas the Perplexity baseline produces consistently lower curves. TableLlama and Gemma-2-9B exhibit steeper initial slopes, indicating a high concentration of correctable errors among the most-uncertain items, while other models obtain comparable gains only at larger budget fractions B.

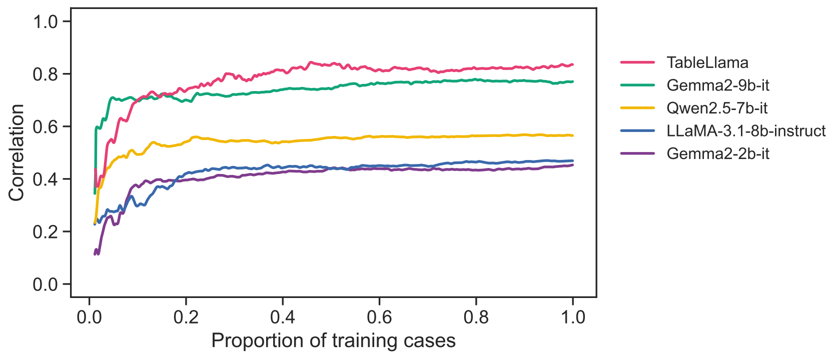

Figure 4: Spearman correlation (ρ) with multiple-generations PE when training the proposed method over an increasing number of training cases, average over 10-fold cross-validation.

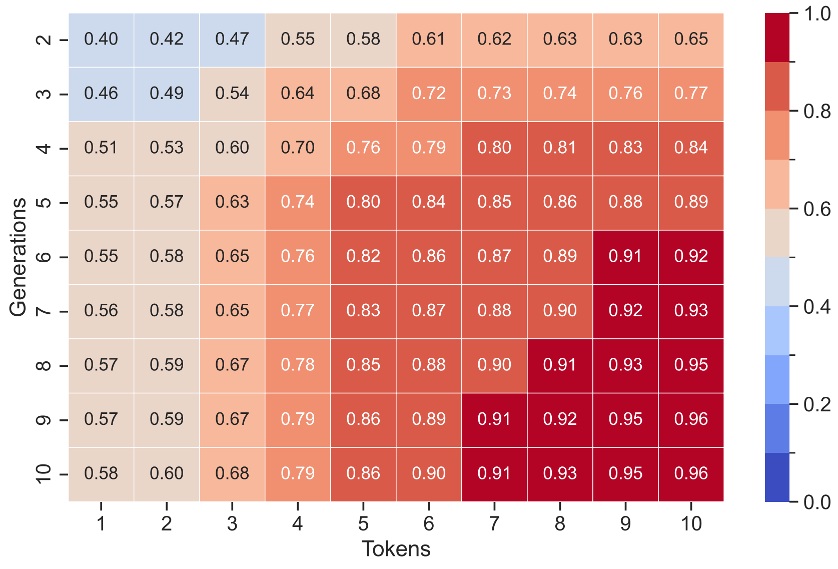

Figure 5: Spearman correlation (ρ) with PE, obtained with $N = 10$ generations from TableLlama using all the tokens, as a function of the number of tokens (x-axis) and generations (y-axis) used to compute a truncated PE.

4.5. Q3: Learnability of the uncertainty regressor h_ϕ

Since our self-supervised regressor h_ϕ requires a warm-up phase for learning, we conducted two supplementary experiments to assess its practical applicability. First, we estimated the number of examples that are necessary to reach the stability of the regressors' performance. As a complementary analysis, we assess if a proxy target measure can be derived, trading some accuracy for a reduced computational cost.

Regressor Convergence as a Function of Training Size We train the regressor with an increasing number of cases and assess its Spearman correlation with the target. We perform a 10-fold cross-validation, where in each fold we keep the validation set fixed and train the regressor by expanding the training set one-by-one. Figure 4 reports the average performance over all the folds for each model. We regress the PE target only, using the $\mathrm{M}(\mathbf{p})$ and $\mathrm{H}(\mathbf{p})$ features on the first 10 *Generated* tokens. Stable performance is reached with a limited number of items, e.g., $10\% - 20\%$ of the dataset, depending on the model. This highlights how, even with a limited number of examples, the approach can successfully learn to estimate uncertainty. This, in turn, impacts the overall efficiency and sustainability of the method, since the number of multiple-generations cases necessary to train the model appears to be limited. We recall that, since the method is self-supervised, it does not rely on external annotations.

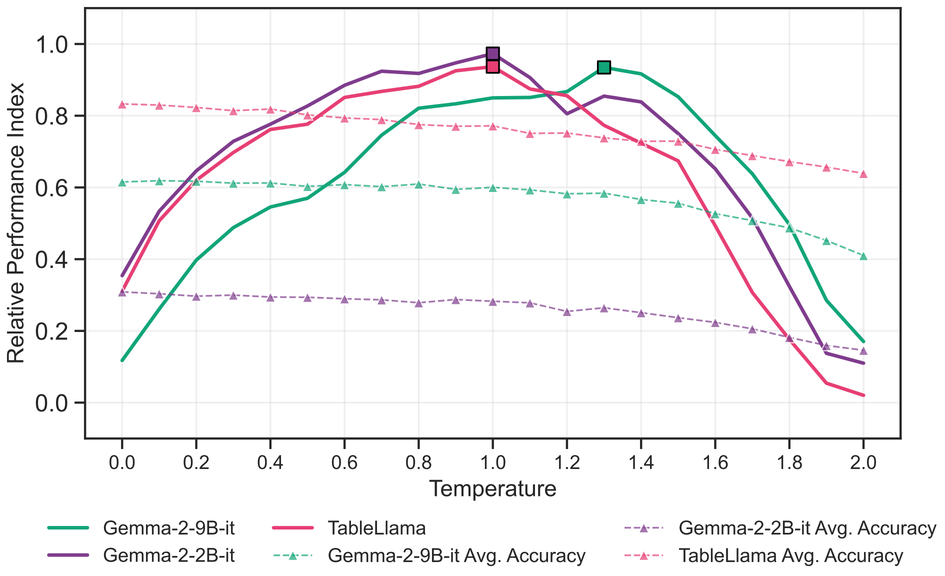

Figure 6: Sensitivity of model performance to temperature. Model performance is summarized as the area under the curve of accuracy as a function of budget (cf. Figure 3), with selected items ranked by PE, and rescaled to [0, 1] for each model. The square markers highlight the maximum Relative Performance Index.

Approximated Ground Truth We finally consider whether reducing the resources dedicated to the multiple generations can retain enough information to approximate the full uncertainty – that is, the uncertainty observed with the original N generations when considering all the generated tokens. For this experiment, performed for illustration purposes, we focus on the best-performing model, TableLlama, and examine the PE target only. We cap the number of generations and the number of generated tokens, both of which impact the computational burden linearly. Figure 5 shows the Spearman correlation between the approximated PE and the full PE. In this setup, even under computational constraints, the derived approximate PE is reasonably correlated with the full PE, i.e., computed on the complete set of generations and tokens. This highlights how an approximate target signal could be derived by trading accuracy for efficiency.

4.6. Q4: Temperature Sensitivity

Our approach proposes a measure of uncertainty related to output variability. However, the temperature setting affects the trade-off between output variability – needed to estimate uncertainty – and task accuracy. In this section, we evaluate the impact of the temperature on the task performance. We summarize the performance as the area under the curve (AUC) that describes dataset-level accuracy as a function of correction budget B (cf. Figure 3 for comparison). We chose this metric as it summarizes the practical downstream usability of the proposed method.

We systematically sweep the temperature value in the range $0.0 \leq \tau \leq 2.0$ with steps of 0.1 to assess its effect on the AUC. Due to computational budget constraints, we utilize a subset of 200 elements from the original dataset and focus on PE. Figure 6 reports the AUC over the temperature and the average overall accuracy obtained with selected models. It can be observed that the most convenient performance is obtained for balanced τ values, for which dataset-level accuracy does not change drastically, implying that temperature variations up to a certain level do not compromise the task performance. Setting smaller τ values degrades the performance, since lower output variability directly reduces observable uncertainty. Similarly, setting a higher τ value leads to higher observed uncertainties, which are related to noisy outputs and do not reflect the actual answer uncertainty. Based on these observations, we set $\tau = 1.0$ for all the experiments previously discussed in this paper. Nonetheless, this experiment illustrates how temperature can be adjusted depending on the specific scenario.

5. Conclusions and Future Work

In this work, we propose a lightweight, self-supervised approach for obtaining uncertainty estimates. An approximate entropy measure is regressed by leveraging features that are observable at inference time, based on final- and intermediate-layer token probability distributions. This approximation is calculated, after an initial warm-up phase, without relying on multiple, costly generation runs, making it suitable for practical use in real-world EL workflows.

We validate the proposed method on the task of Entity Linking for tabular data, showing a strong correlation with uncertainty observed over multiple generations estimates, with significantly reduced computational overhead. Empirical evaluations, performed across several instruction-tuned LLMs, demonstrate that the method is highly effective at identifying low-accuracy outputs, in particular when using features derived from generated tokens. We also quantitatively assess the number of cases needed for learning to predict uncertainty, showing that a limited warm-up phase is sufficient. Finally, we measure how the LLM temperature influences the trade-off between uncertainty and the overall accuracy of the method.

The proposed method is general, and it may benefit other closed-form tasks beyond the considered context. Future experiments, aimed at testing its generalizability across tasks and domains – particularly in open-ended EL settings, where different uncertainty measures could have different behaviors and efficiency trade-offs – could extend the results provided in this study. A warm-up phase, involving multiple LLM generations, is required for training, which introduces a computational cost. However, this cost is mitigated by the rapid convergence of the learning phase, maintaining the efficiency of the overall process. Additionally, evaluating the transferability of learned regressors across datasets related to the same task may provide further insights into the robustness of the method.

While our method is model-agnostic and applies unchanged to any model exposing token log-probabilities (and optionally hidden states), we leave the profiling and evaluation of larger-capacity models to future work. When only output-layer probabilities are available (e.g., closed models), our output-layer-only variant remains applicable.

We also show that some of the features related to the intermediate layers during the generation process are partially able to fill the performance gap observed between prompt-related and generation-related tokens. The use of internal LLM state features to anticipate properties of the generated output represents a promising research direction. Overall, the results reinforce the practical applicability of uncertainty-aware methods in LLM-based EL workflows, particularly in contexts where quality, efficiency, and scalability are critical. Beyond mention-level scores, a natural extension is to target the specific decision tokens (e.g., the span that selects a candidate, or yes/no tokens in classification prompts) to obtain position-aware confidence; for multi-answer outputs, per-span uncertainties could further support selective review and partial acceptance.

Acknowledgments

The work has received funding from the European Union's Horizon Europe research and innovation programme un- der grant agreements No. 101189771 (DataPACT) and No. 101070284 (enRichMyData), and the Italian PRIN project Discount Quality for Respon- sible Data Science (202248FWFS).

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (GPT-4-turbo) in order to perform grammar and spelling checks. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] C. S. Bhagavatula, T. Noraset, D. Downey, TabEL: Entity linking in web tables, in: International Semantic Web Conference, Springer, 2015, pp. 425–441.

[2] N. Gupta, S. Singh, D. Roth, Entity linking via joint encoding of types, descriptions, and context, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 2681–2690.

[3] F. Belotti, F. Dadda, M. Cremaschi, R. Avogadro, M. Palmonari, Evaluating LLMs on entity disambiguation in tables, arXiv preprint arXiv:2408.06423 (2024).

[4] J. Liu, Y. Chabot, R. Troncy, V.-P. Huynh, T. Labbé, P. Monnin, From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods, Journal of Web Semantics 76 (2023) 100761.

[5] T. Zhang, X. Yue, Y. Li, H. Sun, TableLlama: Towards open large generalist models for tables, in: K. Duh, H. Gomez, S. Bethard (Eds.), Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), Association for Computational Linguistics, Mexico City, Mexico, 2024, pp. 6024–6044. URL: https://aclanthology.org/2024.naacl-long.335/. doi:`10.18653/v1/2024.naacl-long.335`.

[6] P. Li, Y. He, D. Yashar, W. Cui, S. Ge, H. Zhang, D. R. Fainman, D. Zhang, S. Chaudhuri, Table-GPT: Table-tuned GPT for diverse table tasks, arXiv preprint arXiv:2310.09263 (2023).

[7] M. Cremaschi, F. D'Adda, A. Maurino, stEELlm: An LLM for generating semantic annotations of tabular data, ACM Transactions on Intelligent Systems and Technology (2025). URL: https://doi.org/10.1145/3719206.

[8] I. Jayawardene, R. Avogadro, A. Soylu, D. Roman, Tablinkllm: An llm-based approach for entity linking in tabular data, in: 2024 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE, 2024, pp. 206–214.

[9] R. Nararatwong, N. Kertkeidkachorn, R. Ichise, Evaluating tabular and textual entity linking in financial documents, in: 2024 IEEE 18th International Conference on Semantic Computing (ICSC), IEEE, 2024, pp. 130–133.

[10] X. Deng, H. Sun, A. Lees, Y. Wu, C. Yu, TURL: Table understanding through representation learning, ACM SIGMOD Record 51 (2022) 33–40.

[11] J.-C. Klie, R. Eckart de Castilho, I. Gurevych, From Zero to Hero: Human-In-The-Loop Entity Linking in Low Resource Domains, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 6982–6993. URL: https://aclanthology.org/2020.acl-main.624/. doi:`10.18653/v1/2020.acl-main.624`.

[12] S. Arnold, R. Dziuba, A. Löser, Tasty: Interactive entity linking as-you-type, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, 2016, pp. 111–115.

[13] P. Manakul, A. Liusie, M. J. Gales, SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023, pp. 9004–9017.

[14] S. Farquhar, J. Kossen, L. Kuhn, Y. Gal, Detecting hallucinations in large language models using semantic entropy, Nature 630 (2024) 625–630.

[15] H.-Y. Huang, Y. Yang, Z. Zhang, S. Lee, Y. Wu, A survey of uncertainty estimation in LLMs meets practice, arXiv preprint arXiv:2410.15326 (2024).

[16] S. Luohe, H. Zhang, Y. Yao, Z. Li, H. Zhao, Keep the cost down: A review on methods to optimize LLM's KV-cache consumption, in: First Conference on Language Modeling, 2024, p. 19.

[17] C. Batini, M. Scannapieco, et al., Data and information quality, Cham, Switzerland: Springer International Publishing 63 (2016).

[18] G. Li, Human-in-the-loop data integration, Proceedings of the VLDB Endowment 10 (2017) 2006–2017.

[19] J. Fan, J. Tu, G. Li, P. Wang, X. Du, X. Jia, S. Gao, N. Tang, Unicorn: a unified multi-tasking matching

model, ACM SIGMOD Record 53 (2024) 44–53.

[20] R. Avogadro, M. Ciavotta, F. De Paoli, M. Palmonari, D. Roman, Estimating link confidence for human-in-the-loop table annotation, in: 2023 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE, 2023, pp. 142–149.

[21] Y. Huang, J. Song, Z. Wang, S. Zhao, H. Chen, F. Juefei-Xu, L. Ma, Look before you leap: An exploratory study of uncertainty analysis for large language models, IEEE Transactions on Software Engineering 51 (2025) 413–429. doi:10.1109/TSE.2024.3519464.

[22] Z. Lin, S. Trivedi, J. Sun, Generating with confidence: Uncertainty quantification for black-box large language models, Transactions on Machine Learning Research (2023).

[23] H. Ma, J. Chen, G. Wang, C. Zhang, Estimating LLM uncertainty with logits, 2025. URL: https://arxiv.org/abs/2502.00290. arXiv:2502.00290.

[24] B. Plaut, N. X. Khanh, T. Trinh, Probabilities of chat LLMs are miscalibrated but still predict correctness on multiple-choice Q&A, 2025. URL: https://arxiv.org/abs/2402.13213. arXiv:2402.13213.

[25] A. Vazhentsev, L. Rvanova, I. Lazichny, A. Panchenko, M. Panov, T. Baldwin, A. Shelmanov, Token-level density-based uncertainty quantification methods for eliciting truthfulness of large language models, in: Proceedings of NAACL 2025, 2025, pp. 2246–2262.

[26] L. Liu, Y. Pan, X. Li, G. Chen, Uncertainty estimation and quantification for LLMs: A simple supervised approach, 2024. URL: https://openreview.net/forum?id=g3aGMMFHW0.

[27] E. Fadeeva, R. Vashurin, A. Tsvigun, A. Vazhentsev, S. Petrakov, K. Fedyanin, D. Vasilev, E. Goncharova, A. Panchenko, M. Panov, et al., LM-Polygraph: Uncertainty estimation for language models, arXiv preprint arXiv:2311.07383 (2023).

[28] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, Advances in neural information processing systems 30 (2017).

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (2014) 1929–1958.

[30] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. E. Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, D. Ganguli, D. Hernandez, J. Jacobson, J. Kernion, S. Kravec, L. Lovitt, K. Ndousse, C. Olsson, S. Ringer, D. Amodei, T. Brown, J. Clark, N. Joseph, B. Mann, S. McCandlish, C. Olah, J. Kaplan, Language models (mostly) know what they know, CoRR (2022).

[31] S. Lin, J. Hilton, O. Evans, Teaching models to express their uncertainty in words, arXiv preprint arXiv:2205.14334 (2022).

[32] M. Xiong, Z. Hu, X. Lu, Y. Li, J. Fu, J. He, B. Hooi, Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs, arXiv preprint arXiv:2306.13063 (2023).

[33] Y. Yang, H. Li, Y. Wang, Y. Wang, Improving the reliability of large language models by leveraging uncertainty-aware in-context learning, arXiv preprint arXiv:2310.04782 (2023). arXiv:2310.04782.

[34] Y. A. Yadkori, I. Kuzborskij, A. György, C. Szepesvári, To believe or not to believe your LLM, arXiv preprint arXiv:2406.02543 (2024).

[35] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32.

[36] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'16, Association for Computing Machinery, New York, NY, USA, 2016, p. 785–794. URL: https://doi.org/10.1145/2939672.2939785. doi:10.1145/2939672.2939785.

[37] nostalgebraist, Interpreting GPT: The logit lens, https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020. LessWrong, 2020.

[38] R. Avogadro, M. Cremaschi, F. D'Adda, F. De Paoli, M. Palmonari, et al., Lamapi: a comprehensive tool for string-based entity retrieval with type-base filters., in: OM@ ISWC, 2022, pp. 25–36.

[39] K. Hajian-Tilaki, Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation, Caspian journal of internal medicine 4 (2013) 627.

[40] E. S. Keeping, Introduction to statistical inference, Courier Corporation, 1995.

A. Illustrative Example

The example below illustrates the approach to resolve a mention in a table to its corresponding entity, using an LLM. A high-level **instruction**, describing the entity linking task, is provided. A **question** contains the **mention** to be linked, together with a list of entity **candidates** extracted from a retriever, e.g., Wikidata Lookup Service or LamAPI. The candidates are provided in the following format: `<label [DESC] description [TYPES] type1, type2, ..., typeK>`. A snapshot of the **input** table is provided, i.e., N rows above and below the row containing the mention, in Markdown format. The LLM selects a candidate from the predefined list and outputs it verbatim as an **answer**.

Entity Linking Prompt Example

Instruction: *This is an entity linking task. The goal for this task is to link the selected entity mention in the table cells to the entity in the knowledge base. You will be given a list of referent entities, with each one composed of an entity name, its description and its type. Please choose the correct one from the referent entity candidates.* [...]

Input:
```
[TLE] List of high schools in South Dakota.
col:  |school|type|city|county|mascot|
row 0:  |Aberdeen High School|Private|Aberdeen|Brown|Knights|
row 1:  |Agar High School|Public|Agar|Sully|
[...]
row 35:  |Crazy Horse High School|Public|Wanblee|Jackson|Chiefs|
row 36:  |Crow Creek High School|Public|Stephan|Hyde|Chieftains|
row 37:  |Custer High School|Public|Custer|Custer|Wildcats|
[...]
```

Question: *The selected entity mention is: 'Hyde'. The column name for 'Hyde' is 'county'. The referent candidates are:*
```
<Dr. Jekyll and Mr. Hyde [DESC] fictional characters [TYPE] group of fictional characters>
<Hyde v Hyde [DESC] landmark case of the English Court of Probate and Divorce [TYPE] legal case>
<Hyde County [DESC] county in South Dakota, United States [TYPE] county of South Dakota>
<Douglas Hyde [DESC] first President of Ireland (1860-1949) [TYPE] linguist>
<Hyde County [DESC] county in North Carolina, United States [TYPE] county of North Carolina>
<Hyde [DESC] civil parish in Bedfordshire, UK [TYPE] civil parish>
<Hyde Park [DESC] town in Dutchess County, New York, United States [TYPE] town of New York>
<Strange Case of Dr Jekyll and Mr Hyde [DESC] novel by R. L. Stevenson [TYPE] literary work>
<Hyde Park [DESC] neighborhood in Chicago, Illinois [TYPE] neighborhood>
[...]
```

What is the correct referent entity for 'Hyde'?

Postilla: *Answer with just a candidate, selected from the provided referent entity candidates list, and nothing else. The selected candidate must be reported verbatim from the list provided as input. Each candidate in the list is enclosed between < and > and reports [DESC] and [TYPE] information.*

Answer: `<Hyde County [DESC] county in South Dakota, United States [TYPE] county of South Dakota>`

B. Representative Qualitative Examples

We provide representative examples of the strengths and shortcomings of our method, reporting for each example: input table, mention, candidate list, generated answers with their frequencies, baseline, and estimated uncertainties. **Recoverable error 1** reports the standard case: a model's output is uncertain, with borderline accuracy (0.5), and our one-shot prediction successfully highlights such uncertainty. **Recoverable error 2** reports a more complex case, in which the same mention is referred to over multiple seasons. As a consequence, the model struggles to fix on one single entity. Again, our method is able to recover this case by predicting an uncertain outcome. We also report an unrecoverable error case, in which there is no answer variability. Our method correctly predicts low answer variability and, consequently, the case is not marked for correction. However, the answer is wrong, and the error cannot be recovered. Notably, the regressor accurately captures uncertainty across all the examples.

Recoverable error 1: high answer variability → low accuracy

party	city	province	registration
Democracia Galega	Oleiros	A Coruña	1997-01-13
Partido Democrático Español	Madrid	Madrid	1997-01-13
Partido Nacional Republicano	Valladolid	Valladolid	1997-01-13
Partido del Amor Universal	Barcelona	Barcelona	1997-02-12
Els Verds de Formentera	Formentera	Balearic Islands	1997-02-20
Els Verds d'Eivissa	Ibiza	Balearic Islands	1997-02-20
	...		

Table: Registered political parties in Spain, mention highlighted

Name	Description	Type
Madrid	Spanish Congress Electoral District	electoral district of the Spanish Congress
Madrid	city in Iowa, United States	city in the United States
Madrid Province	province of Spain (1833-)	province of Spain
Madrid	constituency of the Senate of Spain	constituency of the Senate of Spain
Madrid	poem by Alfred de Musset	version, edition, or translation
Madrid	capital city of Spain	municipality of Spain
Madrid	None	passenger ship
Madrid	None	electoral district
Madrid	film	film
Madrid	mountain in South Africa	mountain
Madrid	encyclopedia article	encyclopedia article
	...	

Candidate entities with description and type (right and wrong answers)

Observed answers (count):

`<Madrid [DESC] capital city of Spain [TYPE] municipality of Spain>`	**(5)**
`<Madrid [DESC] Spanish Congress Electoral District [TYPE] electoral district of the Spanish Congress>`	**(3)**
`<Madrid [DESC] None [TYPE] electoral district>`	**(2)**

```
Predictive Entropy:     0.649  Predicted using our method (avg): 0.588
Semantic Entropy:       0.478  Predicted using our method (avg): 0.421
```

Recoverable error 2: high answer variability → low accuracy

Season	Club	League	Apps	Goals
1986-87	Fiorentina	Serie A	5	1
1987-88	Fiorentina	Serie A	27	6
		...		
1994-95	Juventus	Serie A	17	8
1995-96	Milan	Serie A	28	7
		...		
1998-99	Inter	Serie A	23	6
1999-00	Inter	Serie A	18	6
		...		

Table: Roberto Baggio career stats, mention highlighted

Name	Description	Type
Serie A	top Italian football league	annual sporting event
Serie A (basketball) 2003-04	sports season	sports season
Serie A (basketball) 2007-08	None	sports season
Serie A	2nd tier of Italian women's rugby union	national championship
Serie A	top Italian pallapugno league	sports competition
1984-85 Serie A	sports season	sports season
1986-87 Serie A	sports season	sports season
1990-91 Serie A	sports season	sports season
1994-95 Serie A	sports season	sports season
1997-98 Serie A	sports season	sports season
1998-99 Serie A	sports season	sports season
1999-2000 Serie A	sports season	sports season
	...	

Candidate entities with description and type (right and wrong answers)

Observed answers (count):

<1986-87 Serie A [DESC] sports season [TYPE] sports season>	**(4)**
<1994-95 Serie A [DESC] sports season [TYPE] sports season>	**(3)**
<Serie A [DESC] top Italian football league [TYPE] annual sporting event>	**(2)**
<1998-99 Serie A [DESC] sports season [TYPE] sports season>	**(1)**

Predictive Entropy:	0.639	Predicted using our method (avg):	0.571
Semantic Entropy:	0.638	Predicted using our method (avg):	0.559

Unrecoverable error: no answer variability & zero accuracy

name	family	language	region	country	pop (M)
Agaw	Cushitic	Agaw	Horn of Africa	Ethiopia. Eritrea	1
Amhara	Semitic	Amharic	Horn of Africa	Ethiopia	20
Beja	Cushitic	Beja	Horn of Africa	Sudan. Eritrea	2
Bilen	Cushitic	Bilen	Horn of Africa	Eritrea	0.2
Gurage	Semitic	Gurage	Horn of Africa	Ethiopia	1.9
Oromo	Cushitic	Afan Oromo	Horn of Africa	Ethiopia. Somalia. Sudan. Kenya	30
Saho	Cushitic	Saho	Horn of Africa	Eritrea. Ethiopia	0.2
		...			

Table: Ethnic groups in Horn of Africa, mention highlighted

Name	Description	Type
Bilen i dag	Swedish periodical	periodical
Blågula Bilen	aid agency	aid agency
Bilen & miljön	Swedish periodical	periodical
Handsfreetelefoni i bilen	motion by Inger Jarl Beck et al. 2005	individual motion
Bilen people	ethnic group	ethnic group
Bilen	1992 film by John Goodwin	film
Ensam i bilen	article in Drömmen om bilen (1997)	article
Blin	language	modern language
När bilen drabbade landsbygden	article in Drömmen om bilen (1997)	article
Bilen	family name	family name
Mobilförbud i bilen	motion by Helena Bargholtz 2009	individual motion
	...	

Candidate entities with description and type (right and wrong answers)

Observed answers (count):

```
<Bilen people [DESC] ethnic group [TYPE] ethnic group>        (10)
```

```
Predictive Entropy:    0.0   Predicted using our method (avg): 0.044
Semantic Entropy:      0.0   Predicted using our method (avg): 0.041
```

C. Position-Dependent Feature Contributions

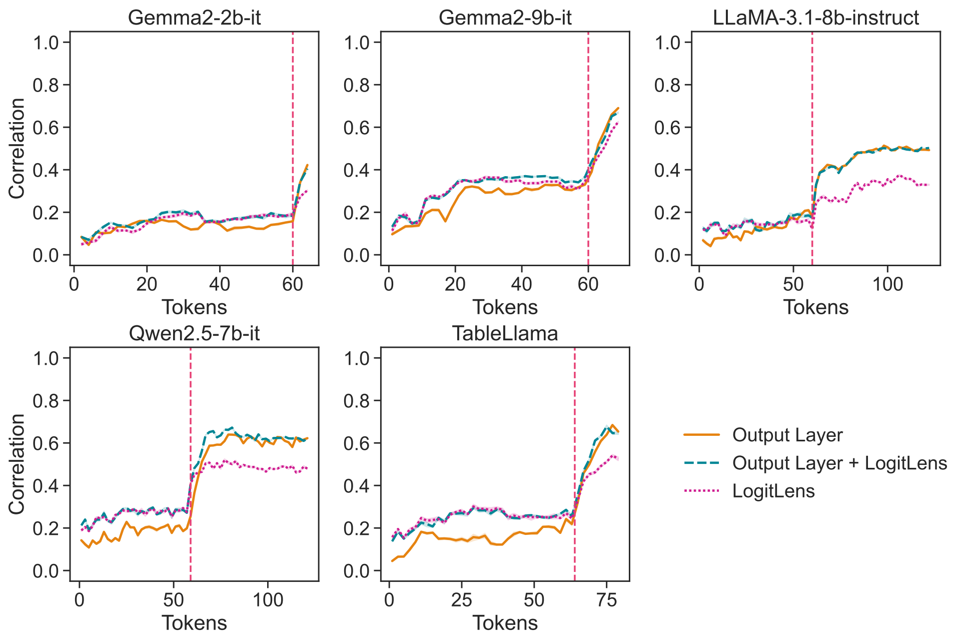

Figure 7: Spearman correlation with PE when training on a progressive window of tokens spanning the concatenated *Postilla* and *Generated* segments. The vertical dashed line marks the boundary between the two segments.

Figure 2 reveals that performance differs depending on whether features are extracted from the *Postilla* or *Generated* tokens. To investigate this further, we designed an experiment that progressively expands a sliding window over the concatenated *Postilla* and *Generated* segments, measuring how information accumulates as more tokens are included. We evaluate three feature configurations: (1) output-layer features, (2) LogitLens features from intermediate layers, and (3) their combination. Figure 7 shows the Spearman correlation between each configuration and the baseline PE as a function of the window size. Correlation increases gradually while the window traverses the *Postilla* tokens, with a pronounced jump once the *Generated* tokens are reached. The improvement over *Postilla* is non-uniform: certain positions—especially the final tokens—contribute disproportionately, indicating that the informative signal is unevenly distributed. Additionally, LogitLens features provide a benefit within the *Postilla* segment but not after entering the *Generated* portion, suggesting that during generation, the output-layer features already capture sufficient information, whereas intermediate-layer representations are more useful prior to generation. These observations imply that feature selection should be adapted based on token origin and position to maximize effectiveness. This also motivates extensions that localize uncertainty at the answer-bearing token(s), enabling per-token confidences and finer-grained handling of multi-answer outputs.

D. Spearman Correlation Between Estimated and True PE/SE

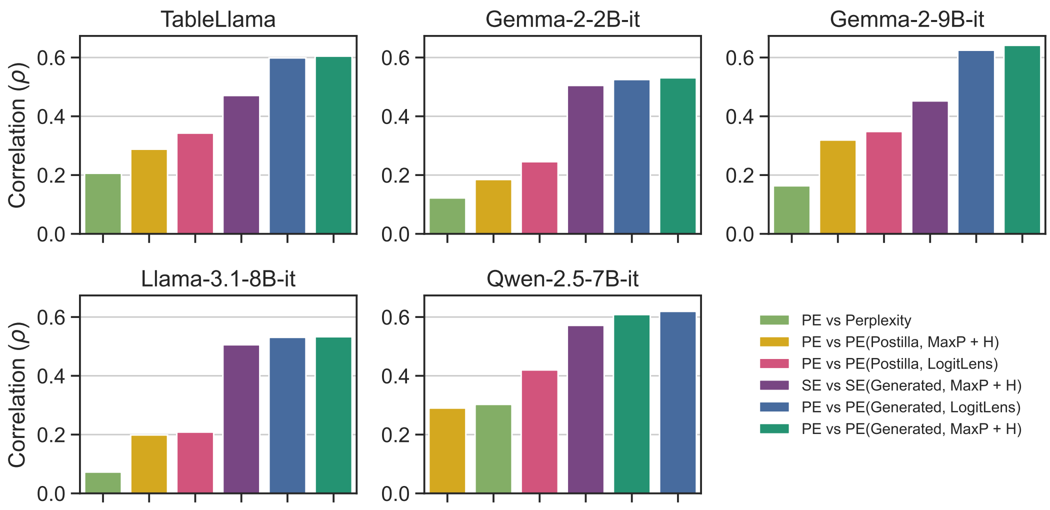

Figure 8: Spearman's rank correlation coefficient (ρ) between the estimated entropy (PE or SE) and the multiple-generations entropy (PE or SE) across models. In the legend, the notation "`Target(Segment, Observable)`" indicates that the target variable *Target* was predicted using a regressor based on *Observable* features extracted from the *Segment* segment of the prompt.

E. Answer Variability as a Function of Temperature

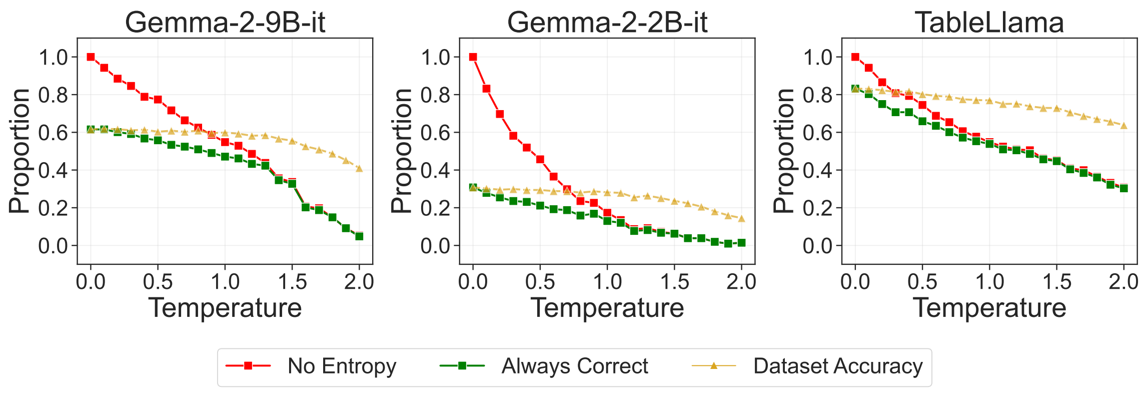

Figure 9: Sensitivity to temperature of prompts with no output variation (*red*), prompts with correct answers only – still no output variation (*green*), and prompts with correct answer on average (≥ 0.5, *yellow*), **(left)** Gemma-2-9b-it, **(center)** Gemma-2-2b-it, and **(right)** TableLlama.

To evaluate the variability of answers and recoverable cases, we systematically sweep temperature values in the range $0.0 \leq \tau \leq 2.0$ with steps of 0.1 to assess the temperature effect on (1) the number of items that show output variability and (2) the average accuracy. For capacity reasons, we utilize a subset of 200 elements from the original dataset. Figure 9 shows the results of the experiment for two selected LLMs. The *yellow* series shows average accuracy, highlighting how higher τ values deteriorate the performance on the task. At the same time, lower τ settings yield the highest proportion of always correct cases (*green* line). However, again at low τ, output variability is also lowest; the uncertainty of these cases cannot be estimated, making this setting unsuitable for recovery. Operationally, a reasonable trade-off for τ is to minimize the accuracy loss while maximizing the number of cases recoverable through uncertainty, that is, those above the *red* line. Another way to view this trade-off is by examining the difference between the *red* and *green* lines: this difference represents the proportion of unrecoverable cases, that is, cases that are always wrong and have no output variability. Making these cases recoverable requires "paying" by reducing always correct cases, while trying to preserve average accuracy (*yellow*) as much as possible. This experiment illustrates how temperature can be adjusted depending on specific combinations of model, task, and application constraints.

F. Transformer-Based Time Complexity

To further assess the practicality and usability of our approach, in this section we derive the time complexity of a Transformer-based architecture, such as the ones used by the models considered in this work. We consider a Transformer with L layers, and d hidden size. The context length is set to N, while the number of generated tokens is G. The time complexity of a single forward pass over a prompt $X \in \mathbb{R}^{N\times d}$can be decomposed into the following components:

- **Self-attention:** Given the $Q, K, V \in \mathbb{R}^{N\times d}$ matrices, the time complexity of the self-attention mechanism is $O(N^2 \cdot d)$, where N is the context length and d is the hidden size. The attention scores are computed as QK^T, and the output is computed as $\sigma(QK^T)V$. The time complexity of this operation is $O(N^2 \cdot d)$.
- **Feed-forward:** During the feed-forward step, the time complexity is $O(8N \cdot d^2) = O(N \cdot d^2)$ overall, where d is the hidden size. In this we can include also the projection of the input to the QKV space and the final projection to the output space.

In total one has a time complexity of $O(N^2 \cdot d + N \cdot d^2)$ for a single Transformer layer, which becomes $O(L[N^2 \cdot d + N \cdot d^2])$, where L is the number of layers.

If we now suppose to generate G tokens without the use of a KV-cache, the time complexity of the g-th generation step is $O(L[(N+g)^2 \cdot d + (N+g) \cdot d^2])$, for every $g \in [G]$. If we then sum over all the generations, we have:

$$O\left(\sum_{g=1}^{G} L\left[(N+g)^2 d + (N+g)d^2\right]\right) = \tag{4}$$

$$O\left(L\sum_{g=1}^{G} \left[(N+g)^2 d + (N+g)d^2\right]\right) = \tag{5}$$

$$O\left(L\sum_{g=1}^{G} \left[(N^2 + 2Ng + g^2)d + (N+g)d^2\right]\right) = \tag{6}$$

$$O\left(L\left[(GN^2 + 2NG^2 + G^3)d + (GN + G^2)d^2\right]\right) = \tag{7}$$

$$O\left(L\left[G(N+G)^2 d + G(N+G)d^2\right]\right) = \tag{8}$$

$$O\left(LG\left[(N+G)^2 d + (N+G)d^2\right]\right) \tag{9}$$

which shows that the time complexity of generating G tokens is quadratic in the number of overall tokens $N+G$, when $G \ll N$, otherwise it would become cubic in the number of generated ones.

When a KV-cache is used, while the time for processing the prompt is the same, a major computational saving is obtained during the generation phase. In this case, the time complexity of the g-th generation step can be decomposed into the following components:

- **Self-attention:** During the self-attention, Q reduces to a single vector $q_g \in \mathbb{R}^{1\times d}$, while K, V becomes $K_g, V_g \in \mathbb{R}^{(N+g-1)\times d}$. Overall, the time complexity of this operation is $O((N+g) \cdot d)$.

- **Feed-forward:** Since the output of the Self-Attention has a dimension of $1 \times d$, the feed-forward step has a time complexity of this operation is $O(d^2)$, which includes the projection of the input to the QKV space and the final projection to the output space.

If we then sum over all the generations $g \in [G]$, we have:

$$O\left(\sum_{g=1}^{G} L\left[(N+g)d + d^2\right]\right) = \tag{10}$$

$$O\left(L\sum_{g=1}^{G}\left[(N+g)d + d^2\right]\right) = \tag{11}$$

$$O\left(L\left[G(N+G)d + Gd^2\right]\right) = \tag{12}$$

$$O\left(LG\left[(N+G)d + d^2\right]\right) \tag{13}$$

which shows that the time complexity of generating G tokens is linear in the number of overall tokens $N+G$, when $G \ll N$, otherwise it would become quadratic in the number of generated ones.

Table 2
Time complexity of the Transformer-based architecture. L is the number of layers, N is the number of tokens in the prompt, G is the number of generated tokens, and d is the hidden size.

Phase	Without KV-cache	With KV-cache
Prompt processing	$O(L[N^2 \cdot d + N \cdot d^2])$	$O(L[N^2 \cdot d + N \cdot d^2])$
Generation	$O(LG[(N+G)^2 d + (N+G)d^2])$	$O(LG[(N+G)d + d^2])$

This simple derivation shows that our proposed approach to learn to estimate the multiple-generations Predictive Entropy (PE) is still computationally promising even when advanced KV-cache techniques are employed during inference.