Chinese Political Science Review
https://doi.org/10.1007/s41111-025-00327-w

ORIGINAL PAPER

Rolling Memory: A New Approach to Annotation with Generative LLMs in Social and Political Research

Joan C. Timoneda[1] · **Sebastián Vallejo Vera**[2]

Received: 29 September 2025 / Revised: 5 December 2025 / Accepted: 7 December 2025

Abstract

Generative Large Language Models (LLMs) have shown promising results in text annotation tasks, which is of interest to social scientists. The most commonly used approaches, zero-shot and few-shot learning, do not sufficiently exploit the in-context learning capabilities of these models. Extant work demonstrates that allowing these models to retain memory throughout the annotation task increases performance considerably. In this article, we propose a refinement to the memory approach from Timoneda et al. (Memory Is All You Need: Testing How Model Memory Affects LLM Performance in Annotation Tasks. arXiv preprint: 2503.04874, 2025) that leads to significant performance gains over the original version. We implement a two-run method where we keep 100 highly informative examples from the first run in the memory at the start of a second run. During the first run, the model improves its performance as it progresses through the annotation task, a phenomenon we attribute to in-context learning. Then, during the second run, performance is high and consistent throughout the task, improving by a further 3.56% over the .standard memory approach and a remarkable 13.26% over few-shot learning with chain-of-thought reasoning, the current gold standard. These results have important implications for applied researchers looking to improve measurement accuracy in annotation tasks

Keywords LLMs · ChatGPT · NLP · Text Classification · Text Annotation · Deep Learning

✉ Joan C. Timoneda
timoneda@purdue.edu

[1] Department of Political Science, Purdue University, 100 N. University St, 2230 Beering Hall, West Lafayette, IN 47907, USA

[2] Department of Political Science, University of Western Ontario, London, Canada

Published online: 07 January 2026

1 Introduction

Recent research introduces memory-based approaches to text classification tasks using generative Large Language Models (LLMs) (Timoneda and Vallejo Vera 2025b). Using OpenAI's GPT-4o and Meta's Llama−3.3-70B models, the authors show that allowing a generative LLM to retain a memory of its own previous classifications, regardless of accuracy, yields performance improvements of up to 25% when compared to few-shot learning with chain-of-thought (CoT) reasoning (Wei et al. 2022; Diao et al. 2023). This finding has important implications for theoretical and applied researchers alike, as memory-based approaches improve classifier performance by redefining how we provide examples to the model.[1] In this article, we ask *why* memory-based approaches significantly improve performance, and provide a refinement on the memory-based approach that boosts performance by up to 18.15% when compared to few-shot with CoT.

Our main argument builds on the literature on in-context learning (ICL): LLMs can update their behavior through examples in the prompt, without parameter updates (Mueller et al. 2023; Braccaioli et al. 2025). Previous research has demonstrated that LLMs can update their understanding of a given task from few examples, and without weights updates (see Mueller et al. 2023). Different studies have shown how this process, known as ICL, helps improve performance, especially when focusing on a single task (Braccaioli et al. 2025; Zhang et al. 2025). In annotation tasks, a sufficient number of in-context examples allow for ICL to emerge. In particular, when the model is provided with enough information about the task (e.g., annotation of a particular label), the transformer is able to map a 'task vector' that represent the 'rule' for that task (Hendel et al. 2023). From a cognitive-memory perspective, each past annotation becomes part of the model's short-term memory after being appended to the prompt (Shan et al. 2025).[2] While the model's weights are not changing, the information retained in the prompt allows the model to identify the task that is being carried out through the examples provided. This aligns with research on in-weight learning (i.e., learning from training) suggesting that in regions of the input space where data is scarce and in-weight learning is not possible, ICL emerges and increases performance by providing more diverse information (Chan et al. 2024).

One aspect of the memory-based approach that Timoneda and Vallejo Vera (2025b) overlooked in their original article is that classification accuracy ($F1$-score) should increase *as the classification task progresses.* This means that $F1$-scores are lower at the beginning of the task, when memory is smaller and the model has had fewer examples to improve its own classifications. Performance should gradually improve, peaking toward the end of the context window, as the model better understands the nuance of the task. We also predict a relatively constant progression before reaching peak performance, which underscores the role of memory: as the model becomes more experienced in the task at hand, it becomes more consistent and bet-

[1] 'Memory', in this case, refers to the information retained in the prompt.

[2] It is important to note that 'in-context learning' does not modify the weights of the original model. Whenmentioning 'in-context learning', we are referring to LLMs inferring patterns dynamically (rather thanchanging weights, which would imply 'learning' in the neural-network sense).

ter at predicting the correct responses. This also implies, however, that there is still performance to be gained if we can improve the *F*1-scores at the beginning of the run, where the model has less memory and therefore performs worse. In this article, we propose a *two-run* approach to solve this issue and increase performance substantially. The two-run approach can be implemented in two ways. First, we take the last 100 classified observations in memory from run 1 and use them as the initial memory for run 2. The classification for run two is then final. We call this first method *last 100*. Second, we calculate the set of 100 observations from run 1 that reached peak performance and use that set as the initial memory for run 2. We call this second variation *peak 100*. In both cases, model performance at the start of the run should be improved, leading to higher overall performance.

To test these claims, we use one of OpenAI's state-of-the-art generative models at the time of writing, GPT-4o. Following Timoneda and Vallejo Vera (2025b), we apply it to the classification of political nostalgia in political party manifestos (Müller and Proksch 2024). Our main result and contribution is that the two-run memory approach maximizes performance in annotation tasks using generative LLMs such as GPT-4o. We also confirm and refine the testing from Timoneda and Vallejo Vera (2025b), providing a better link to ICL and showing results with reshuffled data. These methods do not require previously hand-labeled data (other than for validation) and yield high performance metrics in annotation tasks, two characteristics that are valued by social science scholars.

2 Why Memory-Based Approaches Significantly Improve Performance

Research using LLMs for supervised text classification has undergone deep shifts in recent years due to the fast-moving nature of the field. Old models using bag-of-words (see Zhang et al. 2010) gave way to word embeddings (Mikolov et al. 2013), which began the trend to treat texts as numeric representations and which still see innovations today (Rodriguez and Spirling 2022; Wu 2024). Deep neural network models, especially Bi-LSTMs, began to explore sentences as a whole, leading to significant increases in classifier performance. Everything changed, however, with the advent of the Transformers architecture, a type of neural network that developed the idea of attention heads to allow the model to recognize and understand words in context (Vaswani et al. 2017). Google's BERT revolutionized the field of Natural Language Processing, and RoBERTa and DeBERTa improved on BERT's performance (Timoneda and Vera 2025; Timoneda and Vallejo Vera 2025a), especially in specialized tasks with political text (Davila Gordillo et al. 2024; Escribà-Folch and Timoneda 2025). These models showed that Transformers-based approaches were capable of producing highly accurate classifications that mirrored the capacity of human coders (Gilardi et al. 2023).

Even more revolutionary, however, was the emergence of generative LLMs such as OpenAI's GPT and Meta's Llama family of models.[3] Generative LLMs are Trans-

[3] Other noteworthy examples are Anthropic's Claude, Mistral AI, DeepSeek, among others.

formers models trained on next-token prediction that excel at text-generation tasks such as summarization, paraphrasing, or translation, and others (Brown et al. 2020; Touvron et al. 2023). Recent research in the social sciences has examined the potential of generative LLMs as tools for text annotation, highlighting both their strengths and limitations (see Wang et al. 2025; Qin 2024), as well as the ethical implications of their use (Woods 2025; Hung 2025). Evidence suggests that generative LLMs can outperform human coders in certain tasks (Gilardi et al. 2023; Xu 2023; Liao et al. 2024; Xiao and Liu 2025) and achieve comparable accuracy when labeling political text (Ornstein et al. 2023; Vallejo Vera and Driggers 2025), including in multilingual contexts (Heseltine and Clemm von Hohenberg 2024). These studies typically design prompts with care, employing zero-shot or few-shot learning strategies combined with CoT reasoning to guide model outputs (Wei et al. 2022; Chen et al. 2023). While few-shot learning with CoT is the current state of the art, Timoneda and Vallejo Vera (2025b) have shown that the performance gains derived from these examples are limited when compared to the memory approach. Moreover, adding more examples to the few-shot learning CoT prompt leads to over-prompting, which reduces performance (Tang et al. 2025).

The alternative suggested by Timoneda and Vallejo Vera (2025b) builds on the literature on in-context learning (ICL) (Mueller et al. 2023; Braccaioli et al. 2025). Previous research has demonstrated that LLMs can update their understanding of a given task from few examples, and without weights updates (see Mueller et al. 2023).[4] Different studies have shown how this process, known as ICL, helps improve performance, especially when focusing on a single task (Braccaioli et al. 2025; Zhang et al. 2025). In annotation tasks, sufficient number of in-context examples allow for ICL to emerge. In particular, when the model is provided with enough information about the task (e.g., annotation of a particular label), the transformer is able to map a 'task vector' that represents the 'rule' for that task (Hendel et al. 2023).[5] From a cognitive-memory perspective, each past annotation becomes part of the model's short-term memory–i.e., the information retained in the prompt–after being appended to the prompt (Shan et al. 2025). Through this process, new information about the task and token representation is integrated, facilitating pattern recognition. This aligns with research on in-weight learning (i.e., learning from training) suggesting that in regions of the input space where data is scarce and in-weight learning is not possible, ICL emerges and increases performance by providing more diverse information on the task at hand (Chan et al. 2024).

Timoneda and Vallejo Vera (2025b) demonstrate that, for annotation tasks, maintaining the annotated text and the labels produced by the LLM in the prompt increases performance over traditional approaches (e.g., zero-shot and few-shot with CoT).[6] Allowing the LLM to retain a memory of its own past classifications, whether correct

[4] A typical example of in-context learning is a model prompted with the input "apple → red, limes → green, bananas →" that produces the output "yellow".

[5] For a review on other empirical and theoretical work on how ICL works, see Dong et al. (2022).

[6] Timoneda and Vallejo Vera (2025b) also show the performance gains of letting the LLM if it correctly or incorrectly labeled the text, in addition to maintaining the annotated text in the prompt. They call this 'reinforcement memory'.

or not, yields a 25% improvement when compared to few-shot learning with CoT reasoning. This memory-based approach allows LLMs to maximize their contextual understanding of a task. Annotation is a complex task: there is variation in the language (e.g., specialized or colloquial), the definitions of the labels, and the context in which the text is produced (e.g., social media, academic texts, or political speech). While LLMs are trained on troves of data, the training distribution during in-weight learning (IWL) is unknown (Chan et al. 2024). In annotation tasks, a lack of context can lead models to 'overfit' correct or incorrect responses from misidentifying the task.[7] Maintaining the memory of previous annotations in the prompt points LLMs to the correct task, allowing for ICL to maximize performance. Arguably, memory provides similar contextual knowledge to the generative LLM than the encoder layers do in encoder-only models like BERT or RoBERTa (see Timoneda and Vera 2025).

Note that this does not require the user to provide information on the *correct* labels, only to maintain the task in the LLM's memory (prompt). Thus, it is not improving performance by 'training' the model, but rather by better contextualizing the task. Few-shot and CoT approaches provide (limited) contextual information about the task, and (limited) information about the correct label. We argue that ICL emerges primarily from the former, and exploit this condition by allowing the model to retain memory of the task for as long as possible.

In this article, we show the evolution of LLM performance as it accesses more memory. We demonstrate that performance exhibits decreasing marginal returns, with rapid increases at first followed by a plateau. Results hold after reshuffling the data to rule out confounding variation in the relevance of the examples (from random sampling) that inform the task. Furthermore, in light of these more robust findings, we propose a refinement to Timoneda and Vallejo Vera (2025b)'s memory approach that increases performance by a further 3.56% on the same dataset used by the authors, which translates to gains of 13.26% over few-shot learning with CoT. First, we do a full initial run with the full dataset of 600 observations, and select a more informative set of 100 observations from this run where the running F1 score is higher. This subset of observations is more internally consistent and more accurate on the whole, leading to performance gains from the start of the second run. The improvements derived from the memory approach, and this new two-run method in particular, represent a significant leap for annotation tasks using generative LLMs.

3 Data and Analysis

The analysis proceeds in two parts. First, we show that the memory-based approach improves LLM performance as the model classifies new texts within the same session—that is, average $F1$-scores will be higher toward the end of the session. We extend Timoneda and Vallejo Vera (2025b) in showing average results with data reshuffled 20 times. We compare this with two commonly used approaches: zero-

[7] In this context, by 'overfitting' we refer to the idea of 'pattern lock-in' or 'over-conditioning', where too many examples can cause the model to focus too much on matching patterns in the examples with those in new texts, decreasing performance (Tang et al. 2025).

shot annotation, where there is no memory retention from the model; and few-shot chain-of-thought (few-shot CoT) annotation, the current gold standard in annotation tasks with generative LLMs.[8] Second, we show that *two-run* approaches significantly improve classifier performance. We provide results using both the last 100 and the peak 100 variations, and contrast both approaches with a standard two-run approach where memory is reset at the beginning of the second run.

For our tests, we use data on political nostalgia from Müller and Proksch (2024), the same data used by Timoneda and Vallejo Vera (2025b) to test their memory-based methods. The data contain human-annotated labels of nostalgic and not nostalgic sentences from party manifestos in 24 European countries.[9] Nostalgia for the past is an important element in far-right party rhetoric, especially in Europe. Overall, the data contain 1,200 human-annotated sentences, with 151 coded as nostalgic and 1,049 as not nostalgic. Following Timoneda and Vallejo Vera (2025b) we created a subsample of 600 observations, keeping all 151 nostalgic observations and drawing a random sample of 449 non-nostalgic ones. Sentences from the sample were 20.83 words long on average, with a minimum of 3 and a maximum of 174 words, similar to the original sample. The dataset is unbalanced, which is not a problem *per se* but does make it more difficult for LLMs to classify the underrepresented category correctly. Also, the reliability results align with our expectation that political nostalgia is moderately challenging to identify. Coders showed moderate agreement on categorical judgments ($K = 0.56$), while the aggregated ratings demonstrated good overall reliability ($ICC(2,k) = 0.84$, 95% CI [0.82, 0.85]). These values place the task between average and above average in terms of coding difficulty.

For all the approaches tested in this article, the system prompt we employ is as follows: "Please classify the following sentence as containing 'nostalgic' or 'not nostalgic' ideas or subtexts about politics, society, or the past. [New line] Return only the name of the category." Each sentence is then submitted to the LLM as a new user message. For each of these tests, we use the OpenAI library in Python to query the LLM and obtain an annotation, using GPT-4o as the model of choice.[10] We use a temperature of 0.7 across all models to maintain consistency. We keep the rest of the hyperparameters at their default values.

We start with the two baseline classification strategies: zero-shot and few-shot CoT. For the zero-shot approach we submit each new classification request as an independent user message, with the model storing no memory of previous interactions. For the few-shot CoT approach, we add ten examples to the prompt, which

[8] See (Alizadeh et al. 2023) for further details on applying few-shot learning with CoT.

[9] The authors define nostalgia as a 'predominately positive emotion that is associated with recalling memories of important or momentous events, usually experienced with close others' (Müller and Proksch 2024). A sentence is classified as nostalgic when at least three of the four coders agreed. The number of manifestos per country ranges from 16 (Poland) to 183 (Denmark), with an average of 68 manifestos and 11 elections per country. Additionally, the authors translated each non-English sentence into English before classification. There are no additional pre-processing steps reported by the authors, and we do not modify the original text. The following text is labeled as nostalgic (and is used as presented here): "It is the only party with stability and consistency has demonstrated for over nine decades that is by the people and would not betray him."

[10] Like the authors of the original paper, we use this model because it offers a good balance between performance and cost, and we consider it to be a sufficiently novel model to make findings generalizable.

include their text, the correct classification, and a detailed reasoning behind the correct classification. We then ask the model to classify each new text into one of the two categories.[11] For memory prompting, a replication of Timoneda and Vallejo Vera (2025b), we append every annotation to the conversation history after classification and run the model again asking it to classify a new text, keeping the set of previous annotations in the conversation history.[12]

For the main contribution of this article, the two-run extension to the memory prompting approach, we implement the following procedure. First, we use the same system-level prompt as the few-shot CoT baseline, including the simple text classification instructions. Second, we do one full run using the original memory approach. This yields a set of fully LLM-coded annotations. Third, we then do a second run on the same data, but keeping a set of 100 high-information annotations in the conversation history at the start of the second run.[13] To select these high-information annotations, we suggest both the *last 100* and the *peak 100* approaches. For *last 100*, we keep the last 100 annotations from run 1 in the memory to start run 2. In the second run, we use a rolling memory: each new LLM annotation is appended to the memory set of 100 observations, and the oldest entry is dropped. As a result, by the time the model reaches the high-information subset in the second run, the entire initial set of 100 items has been replaced. This ensures that the model never annotates an item that is still present in its memory. For the peak 100 observations, we replicate this same approach but using the set of 100 observations from run 1 that had the highest average F-1 score after the first 100 observations.[14]

Lastly, we evaluate model performance by comparing it to the human-annotated ground truth. Since we are interested in changes in performance as the model advances in the annotation task, we report the average $F1$ scores across a rolling set of 200 observations.[15] To ensure that the reported results are not a product of random sampling from the dataset, we do 20 runs for each of the approaches, reshuffling the dataset every time to correct for potential biases stemming from the order of the observations on a given run.[16] For every instance, we plot all runs and report the average $F1$-score from the 20 runs, showing also the trends for all the runs in

[11] Each example sentence is indicated with 'text:'; each correct classification with 'Answer:', and each reasoning text with 'Reasoning:'. All these elements are always separated by a new line.

[12] That is, first we append the previous classification to the conversation history. Then, we append a new text to classify to the conversation history as a new user prompt. Lastly, we make a new request to the model passing the conversation history, which includes both previous interactions plus the latest text to classify.

[13] We do not tell the model that the annotations in the memory buffer are its own previous classifications. Also, we limit the set to 100 annotations to avoid over-fitting, comply with the token limited imposed by LLMs, and maintain costs and/or computational resources low.

[14] This, again, ensures that the model will not classify texts that are already in the memory.

[15] $F1$ is the harmonic mean between precision and recall and is a better metric than accuracy in unbalanced samples. We estimate the $F1$ score for the first 200 observations, and from there we estimate the $F1$ score dropping the first observations and adding the last one. Thus, the second $F1$ score would be estimated using observations 2 to 201, and so on.

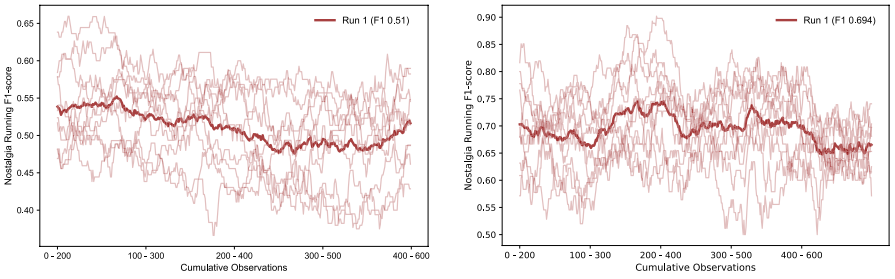
[16] For instance, in one run, the dataset may contain observations that are harder to classify toward the top of the dataset, and easier ones toward the bottom. In this case, we would observe improving performance through the task, but that would be due to text complexity and not to memory. By reshuffling the dataset

the background. Naturally, this approach introduces visible variability across runs, as LLM outputs are probabilistic and the dataset is reshuffled each time. Examining results over multiple runs allows us to capture this natural variation and evaluate performance within that context. The observed differences are thus a consequence of both the model's stochastic output and the reshuffling of the data. By aggregating over many runs, we can determine whether differences between approaches are meaningful. When computational resources allow, we encourage researchers to adopt similar multi-run evaluation strategies.

4 Results—In-Session Memory Improvement

Fig. 1 shows the running $F1$-score averages (y-axis) for our two benchmarks: zero-shot learning without examples or memory—plot (a)—and few-shot learning with CoT—plot (b).[17] Both plots show results for the nostalgic category (nostalgic = 1), but the overall $F1$-scores follow an identical patterns.[18] In the x-axis are all possible sequential sets of 200-observations (from 0-200 to 400-600), and increase by 1 each time.[19] Performance from zero-shot learning is as expected, with no discernible pattern as the model advances in the labeling task (see Fig. 1a). The average $F1$-score for the zero-shot approach is low, at 0.51. Conversely, few-shot CoT improves the average $F1$-score for the full run to 0.694, a significant gain derived from the ten examples and the reasoning provided to the model. However, there is no clear positive or negative pattern as the model classifies more observations. In fact, despite some minor fluctuations, the overall pattern is flat— few-shot CoT starts at 0.685 and ends at 0.682, with the average only slightly above these figures.

20 times and calculating the running average $F1$-scores for all the runs, we ensure it is not text complexity that is driving the results.



(a) Sequential Session Restart (b) Few-Shot CoT

Fig. 1 Running $F1$-score averages in blocks of 200 observations for the nostalgic $= 1$ category for the two benchmark tests. All runs displayed in light red and average performance highlighted in dark red

[17] To ensure that there is no memory retention, we reset the session every time a new text is classified.

[18] The only difference is that the overall $F1$-score is higher than the nostalgic $F1$-score.

[19] That is, the first observation is 0-200, the second 1-201, the third 2-202, and so on.

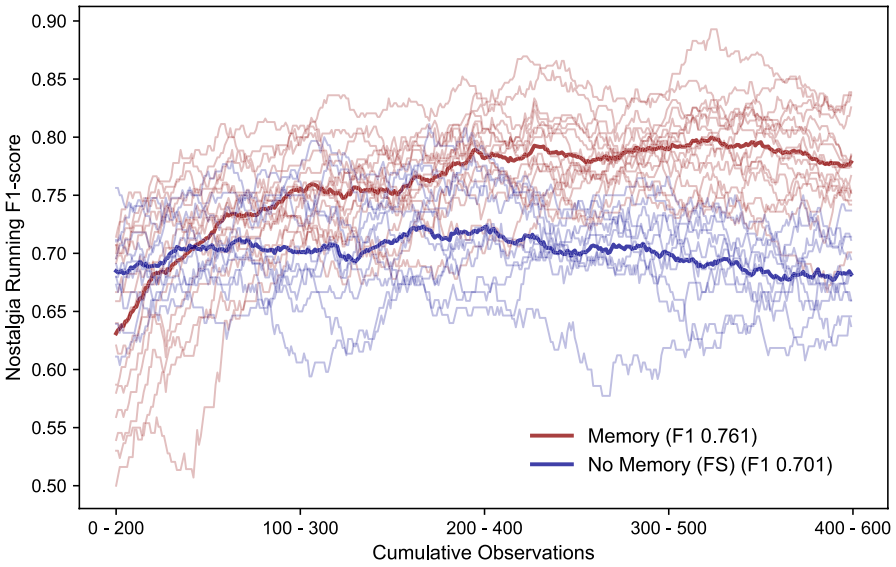


Fig. 2 Running $F1$-score averages in blocks of 200 observations for the nostalgic $= 1$ category. Every run plotted, with the average performance highlighted in red (memory approach) and blue (few-shot CoT)

Figure 2 plots the running $F1$-score averages for the nostalgic $= 1$ category from the memory approach (red) and few-shot CoT (blue). As Timoneda and Vallejo Vera (2025b) show, the memory approach improves overall performance by 9%, on average after reshuffling the data 20 times. Moreover, the curve for the memory approach follows a clear diminishing returns pattern, with steep initial gains followed by a plateau after around 400 observations. In fact, the $F1$-score of the first 250 observations for memory is below few-shot CoT, but then surpasses it for the remaining observations. Memory ultimately outperforms few-shot CoT by 14.15% ($p \leq 0.05$) at the end of the run. At their respective peaks, memory outperforms few-shot CoT by 10.65% ($p \leq 0.05$) –memory peaks at exactly 0.8 while few-shot CoT peaks at 0.723. More generally, Fig. 2 suggests that allowing the model to retain a memory of its own previous annotations increases performance over the correct gold standard in few-shot with CoT.

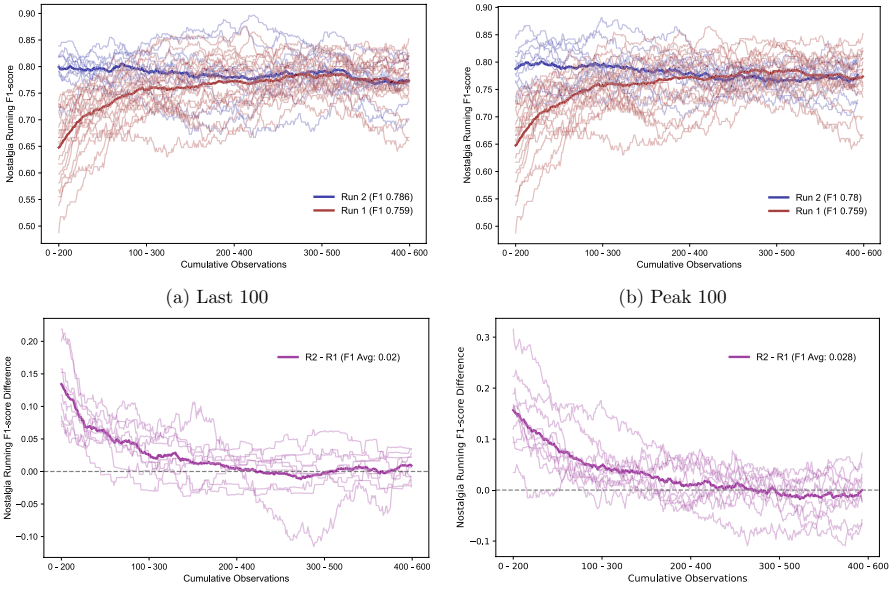
5 Results—Two-run Approach

The running $F1$-score average of the memory approach in Fig. 2 suggests that memory plays a key role in improved LLM annotation performance. As with a human coder, earlier annotations are less accurate as the model has a lower understanding of the task at hand. It then improves significantly as it annotates more texts. While highly valuable, this approach suffers from one weakness, namely, that performance for the first 250 observations or so is still below few-shot CoT performance. We argue this is due to the examples and reasoning furnished to the model when using few-shot

CoT, which help it become better at the task early on. In light of this, we propose a refinement to the memory approach that would improve its performance for the first 250 observations. Doing so would then lead to higher overall performance (average $F1$-score). As described earlier, the refinement uses informative memory from run 1 to start a second run (run 2). We use both the *last 100* and *peak 100* observations from run 1 to improve initial performance in run 2. As a benchmark, we run the few-shot CoT approach continuously twice[20] to determine whether doing a second run increases performance on its own, which would call the memory results into question. We find that this is not the case, as we report below.

Figure 3a, b show the results for both two-run approaches. We observe a similar common pattern: for the second run, early performance gains are strong and statistically significant. After that, as expected, there is practically no improvement throughout the task, as the model's average running $F1$-score starts much higher than in run 1 and performance then remains high throughout run 2. Plots (c) and (d) in Fig. 3 confirm this, showing that early gains between 0.13 and 0.16 are significantly different from zero across the repeated reshuffled runs. Importantly, because the second run improves annotation performance in the first observations in the dataset, overall performance for the model thus increases significantly for both approaches using two runs with memory. In fact, performance increases a further 3.56% using last 100 and

[20] We do not restart the API session between the two runs.



(a) Last 100 (b) Peak 100

(c) Last 100 (d) Peak 100

Fig. 3 Running $F1$-score averages in blocks of 200 observations for the nostalgic = 1 category, two runs using alternative memory approaches

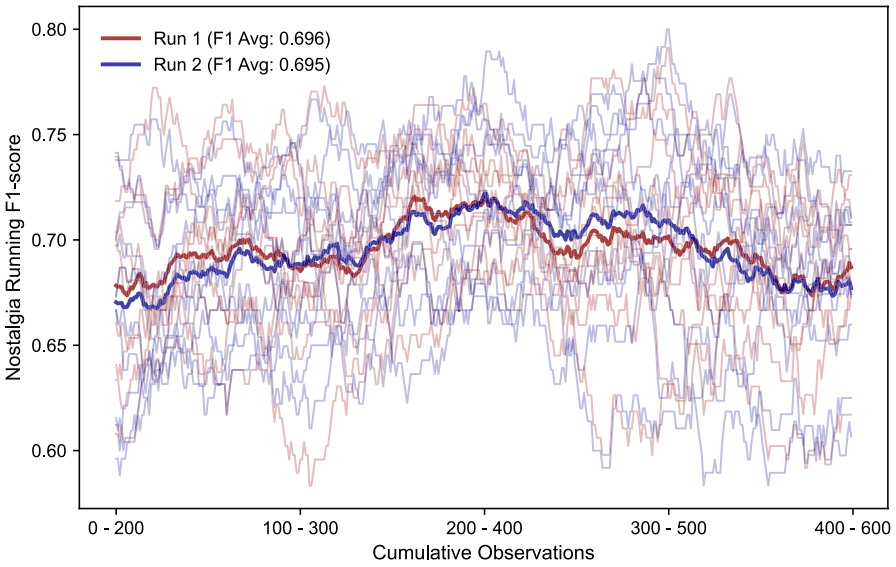


Fig. 4 Two runs with few-shot learning (no memory): Running $F1$-score averages in blocks of 200 observations for the nostalgic = 1 category

2.77% using peak 100. Both approaches now improve on few-shot learning with CoT by a remarkable 13.26% and 12.39%, respectively. These gains are especially striking considering that performance is close to 0.7 with few-shot learning with CoT (0.694). The results indicate that the last-100 approach yields stronger performance compared to the peak-100 alternative.[21] We therefore recommend using the last-100 approach in practice.[22] For both approaches, however, performance converges towards the end of the run, pointing to the structural ceilings in terms of performance when using generative LLMs in annotation tasks. We believe that these early gains in the second run can only be explained through the informative memory the model kept from run 1. Moreover, while we observe variability across runs due to the probabilistic nature of the model and dataset reshuffling, the results consistently show that the average performance under the rolling memory (two-run) approach is significantly higher than the original memory approach. In Fig. 3, for instance, the mean trajectories of the peak-100 and last-100 *second* runs lie clearly above all corresponding *first* runs, and the mean of the differences across runs is distinctly different from zero. These

[21] The average $F1$-scores for the last-100 and peak-100 approaches are 0.786 and 0.780, respectively. A t-test indicates the difference is statistically significant, but we consider the substantive effect of 0.006 to be small. A Cohen's d of 0.56 shows only a moderate effect. We think these results reinforce the recommendation to use last 100, but also confirm that the differences are substantively small and that both approaches improve on the original one-run memory approach significantly.

[22] The last-100 approach is simpler and yields more consistent results than the peak-100 approach. In contrast, peak-100 scores may be inflated by easier stretches of data rather than true performance.

consistent directional differences show that our findings are not driven by stochastic variation alone.[23]

Figure 4 confirms this by showing that a second run does not improve the results for few-shot learning with CoT (no memory). The figure shows the running average $F1$-scores for two runs using few-shot learning with CoT. We do not restart the session for the second run. The two lines follow practically identical trajectories, which means that the first run does not in itself influence the results of a second run. There is, therefore, no memory leakage from one run to the next. Unless explicitly specified, as in Fig. 3, the model does not maintain meaningful memory to improve on its own performance on the same data. This serves as a robustness check to ensure that running the same task twice does not drive the results observed in Fig. 3a, b. The only way to explain the results in Fig. 3a, b, therefore, is that the informative memory we keep from run 1 (either the last 100 or peak 100 observations) yields significant improvements for performance early on in the second run.

6 Discussion and Conclusions

The promise of LLMs as annotation tools has been widely documented and is of particular interest to social scientists, as are its challenges (see Gilardi et al. 2023; Walker and Timoneda 2024). Taking advantage of in-context learning, Timoneda and Vallejo Vera (2025b) demonstrate that *memory* approaches (i.e., retaining previous iterations of a task) improve LLMs performance in annotation tasks. In this article, we expand and refine the memory methodology by maximizing the performance of LLMs across *all* observations, including those toward the beginning of the dataset where the traditional memory approach underperformed. In our proposed refinement, we apply the standard memory approach to the *first* run of an annotation task, but then perform a *second* run through the same data while keeping 100 observations from the first run in the memory. This two-run memory approach shows significant gains in performance when compared to the standard memory approach and few-shot learning with CoT.

We present evidence that the memory retained in the model triggers *in-context learning* (ICL)—an update on behavior through examples in the prompt without parameter updates (Mueller et al. 2023; Braccaioli et al. 2025). When provided with no examples (e.g., zero-shot) or a few examples plus reasoning (e.g., few-shot CoT), model performance remains flat across the full set of observations. As models retain more example in their memory, accuracy improves, suggesting a learning process like ICL.

We offer several takeaways from this extension. First, our two-run refinement performs similarly well to what Timoneda and Vallejo Vera (2025b) call 'reinforcement memory', where they not only keep observations in the model's memory, but also provide information to the model about the correct answers. The reinforcement

[23] Although formal modeling of the variance would provide additional nuance, such analysis is beyond the present scope, as the current tests already show highly significant differences across approaches. We do view systematic quantification of this variance as an important direction for future work.

memory approach, however, requires additional hand-labeled data to feed the model. Our *last 100* extension does not, allowing researcher to reduce the need for hand-labeled data (and use it, rather, for validation). Second, we use multiple iterations and data *reshuffling* to test the memory approach, and these multiple runs show differences in performance due to the model's stochastic nature. In real-world applications, researchers do not know whether the results they obtain from one run of their data are high or low in this distribution. When there are no resource constrains, researchers are advised to reshuffle the dataset and run the process at least 10 times, which is akin to performing k-fold cross-validation. They can then take the run that is closest to the overall mean of all runs, or adjudicate mismatches across observations as is commonly done with human annotators.

Given the promising results, there are still multiple avenues for future research. In this article, we use 100 observations to keep in memory, which is an arbitrary number derived from resource and token limit considerations. It is unclear what is the effect of increasing or decreasing that number, how it may be related to cost, overfitting, and the internal memory limitations of LLMs. Another important line of inquiry is the degree to which memory approaches can reduce the effect of the wording in the prompt—that is, memory may reduce the need for prompt engineering. Achieving higher consistency regardless of the prompt is not only important for downstream tasks, but also for replication. Finally, in this article we provide a proof of concept for our rolling memory approach on one task. Future research should test the degree to which the performance gains from our example generalize to more complex tasks in terms of substance, number of labels, low-resource languages, and rare categories. Since our method leverages in-context learning, we expect it to improve annotation performance across different task setups. Prior research on in-context learning has similarly shown performance gains across a range of tasks (see Mueller et al. 2023; Hendel et al. 2023).

Acknowledgments The authors thank Gael Le Mens, Bryce Dietrich, Eddie Yang, the participants at the Fudan AI workshop June 7-8, 2025, and two anonymous reviewers for their helpful comments and suggestions. The authors also thank Purdue University for access to computing resources.

Data Availability Data and code can be found at https://github.io/joantimoneda/rolling_memory_CPSR

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no Conflict of interest.

References

Alizadeh, Meysam, Maël Kubli, Zeynab Samei, Shirin Dehghani, Juan Diego Bermeo, Maria Korobeynikova and Fabrizio Gilardi. 2023. Open-source large language models outperform crowd workers and approach ChatGPT in text-annotation tasks. arXiv preprint arXiv:2307.02179 101.

Braccaioli, Lorenzo, Anna Vettoruzzo, Prabhant Singh, Joaquin Vanschoren, Mohamed-Rafik Bouguelia and Nicola Conci. 2025. Meta-Learning Transformers to Improve In-Context Generalization. arXiv preprint arXiv:2507.05019 .

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems* 33:1877–1901.

Chan, Bryan, Xinyi Chen, András György and Dale Schuurmans. 2024. Toward understanding in-context vs. in-weight learning. arXiv preprint arXiv:2410.23042 .

Chen, Jiuhai, Lichang Chen, Heng Huang and Tianyi Zhou. 2023. When do you need chain-of-thought prompting for chatgpt? arXiv preprint arXiv:2304.03262 .

Davila Gordillo, Diana, Joan Timoneda and Sebastian Vallejo Vera. 2024. Machines Do See Color: A Guideline to Classify Different Forms of Racist Discourse in Large Corpora. arXiv preprint: 2401.09333 .

Diao, Shizhe, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. arXiv preprint arXiv:2302.12246 .

Dong, Qingxiu, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu et al. 2022. A survey on in-context learning. arXiv preprint arXiv:2301.00234 .

Escribà-Folch, Abel and Joan C Timoneda. 2025. Can Social Media Help Incumbents Subvert Democracy? *Politics & Society* p. 00323292251362676.

Gilardi, Fabrizio, Meysam Alizadeh, and Maël. Kubli. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences* 120 (30) : e2305016120.

Hendel, Roee, Mor Geva and Amir Globerson. 2023. In-context learning creates task vectors. arXiv preprint arXiv:2310.15916 .

Heseltine, Michael, and Bernhard Clemm von Hohenberg. 2024. Large language models as a substitute for human experts in annotating political text. *Research & Politics* 11 (1): 20531680241236240.

Hung, A. T. W. 2025. Too Perfect to Be Like a Human? Ethical Challenges of Emotional AI in Health Care. *Fudan Journal of the Humanities and Social Sciences* .

Liao, X., H. Song and D. E. Bard. 2024. Predicting Successful Treatment Completion Using Baseline Case Characteristics through Machine Learning and Ensemble Modeling: A Two-Step Approach. *Chinese Political Science Review* .

Mikolov, Tomas, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 .

Mueller, Aaron, Albert Webson, Jackson Petty and Tal Linzen. 2023. In-context learning generalizes, but not always robustly: The case of syntax. arXiv preprint arXiv:2311.07811 .

Müller, Stefan, and Sven-Oliver. Proksch. 2024. Nostalgia in European party politics: a text-based measurement approach. *British Journal of Political Science* 54 (3): 993–1005.

Ornstein, Joseph T, Elise N Blasingame and Jake S Truscott. 2023. How to train your stochastic parrot: Large language models for political texts. *Political Science Research and Methods* pp. 1–18.

Qin, X. 2024. Intelligent Technologies and Methodological Transformations in the Social Sciences. *Chinese Political Science Review* 9:1–17.

Rodriguez, Pedro L., and Arthur Spirling. 2022. Word embeddings: What works, what doesn't, and how to tell the difference for applied research. *The Journal of Politics* 84 (1): 101–115.

Shan, Lianlei, Shixian Luo, Zezhou Zhu, Yu Yuan and Yong Wu. 2025. Cognitive memory in large language models. arXiv preprint arXiv:2504.02441 .

Tang, Yongjian, Doruk Tuncel, Christian Koerner and Thomas Runkler. 2025. The Few-shot Dilemma: Over-prompting Large Language Models. arXiv preprint arXiv:2509.13196 .

Timoneda, Joan C., and Sebastián Vallejo Vallejo. Vera. 2025. Behind the mask: Random and selective masking in transformer models applied to specialized social science texts. *PloS one* 20 (2) : e0318421.

Timoneda, Joan C., and Sebastián Vallejo. Vera. 2025. BERT, RoBERTa, or DeBERTa? Comparing Performance Across Transformers Models in Political Science Text. *The Journal of Politics* 87 (1): 347–364.

Timoneda, Joan C. and Sebastian Vallejo Vera. 2025*b*. Memory Is All You Need: Testing How Model Memory Affects LLM Performance in Annotation Tasks. arXiv preprint: 2503.04874 .

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar et al. 2023. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 .

Vera Vallejo, Sebastián, and Hunter Driggers. 2025. LLMs as annotators: the effect of party cues on labelling decisions by large language models. *Humanities and Social Sciences Communications* 12 (1): 1–11.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30.

Walker, Christina and Joan C Timoneda. 2024. Identifying the sources of ideological bias in GPT models through linguistic variation in output. arXiv preprint arXiv:2409.06043 .

Wang, Z., D. Wang, Y. Xu et al. 2025. Intelligent Computing Social Modeling and Methodological Innovations in Political Science in the Era of Large Language Models. *Journal of Chinese Political Science* .

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35:24824–24837.

Woods, D. 2025. Stag Hunt in the Digital Wilds: Legitimizing Global AI Governance Amidst Diverse Terrains. *Fudan Journal of the Humanities and Social Sciences* .

Wu, V. C. S. 2024. Leveraging Computational Methods for Nonprofit Social Media Research: A Systematic Review and Methodological Framework. *Journal of Chinese Governance* 9 (3): 303–327.

Xiao, Y., and H. Liu. 2025. A Mixture Modeling Approach to Detect Different Behavioral Patterns for Process Data. *Fudan Journal of the Humanities and Social Sciences* 18:79–113.

Xu, Y. 2023. What Can AI Learn from Psychology and When Can AI Neglect It? *Fudan Journal of the Humanities and Social Sciences* 16:495–513.

Zhang, Xingxuan, Haoran Wang, Jiansheng Li, Yuan Xue, Shikai Guan, Renzhe Xu, Hao Zou, Han Yu and Peng Cui. 2025. Understanding the Generalization of In-Context Learning in Transformers: An Empirical Study. arXiv preprint arXiv:2503.15579 .

Zhang, Yin, Rong Jin, and Zhi-Hua. Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics* 1 (1): 43–52.

Joan C. Timoneda is assistant professor of political science at Purdue University. He is a comparative and methodology scholar who works on authoritarian politics, democratic backsliding and computational social science. His recent substantive research analyzes the personalization of power in both dictatorships and democracies. His political methodology work focuses on computational and big data methods, especially Transformer-based LLMs. His recent publications cover the technical aspects of LLMs, their biases, and specific applications to specialized political science tasks. His work has appeared at the Journal of Politics, Political Analysis, British Journal of Political Science, Political Science Research and Methods, Plos One, Comparative Politics, and Democratization, among others.

Sebastián Vallejo Vera is an assistant professor of political science at the University of Western Ontario. His research focuses on gender and racial barriers of access and participation in political institutions. As a computational social scientist, his work covers the application of Transformer based Large Language Models (LLMs) in the social sciences, as well as human-LLM interactions and their effect on users' perceived political preferences. His work has appeared at the Journal of Politics, Comparative Politics, Digital Journalism, Scientific Reports, Plos One, Political Research Quarterly, and Electoral Studies, among others. His most recent project, "LLMs, Rabbit Holes, Echo Chambers, and Political Bias", is funded by a Social Sciences and Humanities Research Council (SSHRC) grant.