

A Taxonomy for training time markers

A.1 Categories for Training Markers

Length: `<length_tokens>`, `<length_sentences>`, and `<length_paragraphs>` models granular control in generation length. We tokenize using language-specific Spacy models[15] to obtain token and sentence counts. For paragraph counts, we use the "`\n\n`" delimiter. `<length_bucket>` categorizes generations into broader categories such as *concise* (under 300 tokens), *medium* (between 300 and 1,000 tokens) or *long* (over 1,000 tokens), providing a more general level of control when needed.

Format: `<format>` is used to describe generations with specific output structures such as: JSON, Markdown, or tabular formats. This is particularly useful to condition stricter format requirements needed for real world use cases.

Style: `<style>` captures tone and manner of communication, distinguishing between different modes of expression such as "*Formal*" and "*Informal*". We also add a "*Custom*" value to model for training examples where the user specifies a particular format. For instance, at inference if a user asks, "Respond like Yoda you will" this marker will allow the model to adapt its response to match the requested style. We annotate this marker by using dataset-related information.

Language: `<language>` describes the natural language of the generation, enabling us to model responses in specific languages during inference. We provide detailed markers across the 23 languages covered by our model. Our goal with this tag is to improve language-specific generations and reduce language switching where a prompt specified by a user in one language is not responded to in the same language in the completion. `<code_type>` is specifically used to model programming languages for coding-related tasks. We annotate this marker by using dataset-related information.

Quality: `<quality>` provides a measurable score indicating the quality of a sample, often derived from human annotations or a Reward Model (RM). We utilize a proprietary reward model⁵ to assign rewards to a subset of our training data. We also use these rewards to create a categorical marker `<quality_bucket>` by using quartiles within language-specific subsets into `{1, 2, 3, 4}`, offering a broader description of quality.

Source: `<source>` describes the origin of the data, distinguishing between human-generated content and other methods of data creation like synthetic and translation. We annotate this marker by using dataset-related information.

Domain: `<domain>` ensures that domain-specific knowledge is captured from training subsets, which can then be leveraged at inference to generate content that is relevant and accurate within a particular field. This is particularly crucial for inputs that could belong to multiple fields. For instance, when a user asks, "How do I calculate a factorial?", specifying the `<domain>` as either Code or Math provides valuable context for modeling the interaction. In cases where this marker cannot be obtained from the dataset information, we employ an LLM to annotate and provide our detailed prompt in E

Task: `<task>` defines the overall objective of the generation and helps capture task-specific behaviors, especially when outputs involve complex combinations of formats or actions. This marker is useful to model dataset-wise characteristics. We hypothesize this is particularly helpful for indicating complex workflows during inference. For example, distinguishing between Translation and CodeTranslation, or Rewrite and CodeFix, enhances the descriptiveness of datapoints within the same training pool. In cases where this marker cannot be obtained from the dataset information, we employ an LLM to annotate and provide our detailed prompt in E

B Additional details about experimental sections and key ablations.

Languages covered by Markers We cover 23 languages: *Arabic, Chinese (simplified & traditional), Czech, Dutch, English, French, German, Greek, Hebrew, Hindi, Indonesian, Italian, Japanese, Korean, Persian, Polish, Portuguese, Romanian, Russian, Spanish, Turkish, Ukrainian and Vietnamese.*

Training protocol Training for each variant spanned 8,000 steps, employed a cosine learning rate schedule with a warm-up phase, using a batch size of 32 and an evaluation batch size of 64. We train

⁵The RM is competitive with leading reward models on the RewardBench Leaderboard [24](<https://huggingface.co/spaces/allenai/reward-bench>)

category	definition	values
<quality>	Score indicating the quality assigned to a sample as annotated by a human or a RewardModel(RM).	<i>float</i>
<quality_bucket>	<quality> bucketed into quartiles (calculated by language). 1 indicating <i>highest</i> quality and 4 indicating <i>lowest</i> quality	{1,2,3,4}
<length_tokens>	# of tokens	<i>int</i>
<length_sentences>	# of sentences	<i>int</i>
<length_paragraphs>	# of paragraphs	<i>int</i>
<length_bucket>	<length_tokens> bucketed by defined response length ranges	concise, medium, long
<task>	Task-related information	{OpenEnded, Explanation, Translation, Classification, CreativeWriting, QuestionAnswering, InformationExtraction, Summarization, Rewrite, Reasoning, CodeGeneration, CodeFix, CodeTranslation, CodeExplanation}
<domain>	Domain-related information	{Sciences, Technology, SocialSciences, Culture, Medical, Legal, Unspecified, Conversation, Code, Math}
<code_type>	(<i>coding tasks</i>) Programming languages	{python, javascript, cpp, cobol, java, go, rust, swift, csharp, php, typescript, shell, c, kotlin, ruby, haskell, sql}
<format>	To model desired generation format	{MCQAnswer, ChainOfThought, XML, JSON, Enumeration, Tabular, Markdown, Latex}
<source>	To model the data-generation/annotation source	{Human, Translation, Synthetic, Others}
<style>	To model tone and style of the generation	{Formal, Informal, Custom}
<lang>	To model the 23 languages present in the training data	{English, French, Spanish, Italian, German, Portuguese, Japanese, Korean, Arabic, Chinese, Russian, Polish, Turkish, Vietnamese, Dutch, Czech, Indonesian, Ukrainian, Romanian, Greek, Hindi, Hebrew, Persian}

Table 8: **Comprehensive taxonomy for training time markers:** Our taxonomy contains 13 categories shown with their descriptions and values.

for 2 epochs with a peak learning rate of at 2.5×10^{-4} , achieved through 10 warm-up steps starting from a learning rate of 0.0, and then decay back to 1.25×10^{-4} . One fine-tuning run using 8,000 steps on 128 Nvidia H100 GPUs takes around 6 hours.

Length Evaluations Given the original instruction in the AlpacaEval-LI dataset [61] contains the exact constraint, our **TreasureMarked** and **TreasureMarked (fixed)** both contain explicit reference to the constraint. For **TreasureMarked**, we present the original length-instructed prompt, allowing the model to deduce the associated tags. This approach evaluates the model’s ability to extrapolate tags from instructions. In contrast, for **TreasureMarked (fixed)**, since the original instruction contains the exact constraint, we investigate an additional control strategy where we provide the constraint in the marker template if the taxonomy directly supports it. We remove the length instruction and append the corresponding `<length_tokens>` tag with the appropriate value. Table 3 provides an example of an edited prompt. This strategy assesses the model’s adherence to known templates and its ability to follow explicit length requirements that are only provided via the marker template.

Language Control We insert training markers present in the data into the prompt, but leave out the `<lang>` marker, since it is already present in the prompt.

C Extended Related Work

From one- to multi-dimensional training data tagging The idea of tagging inputs in neural sequence prediction goes back to early applications in machine translation and language modeling. The motivation there was to leverage discrete features to overcome data sparsity or imbalance and introduce levers of control. In early neural LMs, often tokens were added to target a very specific attribute such as topic [37] or auxiliary features [3] including genre and length. A specific token for example was added at inference time to control a feature in translation like the target language [17], or desired output quality [6, 42, 36, 25, 13] and text complexity [1, 34] but also language-specific nuanced attributes like politeness [44, 11], the voice [58], gender [23], domains [19, 4], or diversity [47] of translations. Other works enriched the input representation during training with discrete linguistic features [43] or document information [16] for a better contextualization at inference time. Where and how tags should be placed best differed across applications [16, 56]. Some tags were combined into multidimensional tagging [50, 40] All of these were individual efforts that target one or two dimensions at a time, highly specialized for one trained target model and with training data for one particular task. In contrast, our focus is on a much more general framework with a vast training corpus that targets general performance. Our approach is similarly general, where instead of a single feature, we want to enable a flexible approach that can be used for any text generation task. Furthermore, our goal is to explicitly target improving performance on the long-tail of underrepresented features.

From control in pretraining to control in instruction finetuning In LLM research, there are several related works that experiment with adding prefixes for *control in pretraining*: [18] add control codes for desired text features in pretraining of a LLM derived from the structure of their source, i.e., subdomains or links of online texts and specific task labels for translation and QA. At inference time, values for these control codes are specific to steer the generation. [14] further propose a cooldown schedule in pretraining going from tagged data to untagged data in order to not require prefixes at inference. [61] focus on length control by adding natural language length specification templates to *fine-tuning* data for DPO. In our work we focus on the instruction finetuning stage and incorporate nuanced multi-dimensional tags (i.e. the user can specify length *and* domain *and* format). We circumvent a cooldown schedule by simply introducing tag dropout, hence requiring a much smaller volume of tagged data at training time, and not a complete population of tags at inference time.

From encoded to inferred meta-information Related prefix and prompt tuning methods [28, 26] use continuous embeddings learned for special tokens representing tags in training to condition predictions for specific tasks at inference time. [45] further break those into separate tags for domain and function tags. In our case, we directly embed prefixes with the same vocabulary as the LLM, smoothly integrating them into the sequence. In our experiments, we find that this helps format following even when specified in natural language and not tags.

1068 Attribute-based control in LLM generations has also been pursued with other methods, such as
1069 attribute classifiers [9] or learned attribute vectors [59] — see [62] for a comprehensive survey.

1070 D Limitations

1071 Firstly, although we design a comprehensive taxonomy for our training markers, there can be different
1072 categorizations of domains and tasks, and further data properties. However, our work primarily
1073 showcases the benefits of various data markers and how to leverage them during training (to model)
1074 and inference (to guide model outputs). Therefore, we believe that an extended taxonomy would
1075 further improve the results we demonstrate.

1076 Secondly, we conduct experiments using a 7-billion parameters language model. We opt for 7B
1077 parameter scale to be able to run more experiments with different test cases as presented in the
1078 paper, instead of using a much larger model size with very limited experiments due to the underlying
1079 compute costs. However, considering a larger model could learn to model training-time markers with
1080 higher performance, we believe that our findings hold for larger model sizes.

1081 Finally, we showcase the effectiveness of learning training-time markers in the supervised fine-tuning
1082 phase. We did not experiment with learning these markers in the pre-training phase as pre-training
1083 requires multiple degrees of higher costs. Therefore, we leave this exploration to the future work.

1084 E LLM Annotation

1085 For the following training markers : <domain>, <task>, <format> we annotate using the multi-
1086 lingual open-weights model Command R+.

1087 We provide definitions and multilingual few-shot examples (except for <format>) to obtain high-
1088 quality annotations from the LLM. The prompt used for tagging is as follows:

1089 E.0.1 <domain>

You are a helpful assistant whose goal is to classify the given prompt into
↪ a single class given the following definitions

`Sciences` : Topics related to the broad area of knowledge encompassing all
↪ scientific disciplines, including biology, chemistry, physics, earth
↪ sciences, and astronomy, which study the natural world through
↪ observation, experimentation, and analysis, aiming to understand
↪ fundamental principles and phenomena across various scales and aspects
↪ of the universe

`Technology` : Topics related to the broad area of knowledge encompassing
↪ all engineering and technical disciplines, including Computer Science,
↪ Software Engineering, Internet of Things(IoT), Cybersecurity, Data
↪ Science, Artificial Intelligence, Machine Learning and various
↪ engineering disciplines like Mechanical Engineering, Civil Engineering
↪ and Biotechnology

`SocialSciences` : Topics related to the broad area of knowledge
↪ encompassing all academic disciplines dedicated to the systematic study
↪ of human society, social relationships, and the structures that shape
↪ them, including fields like anthropology, economics, political science,
↪ psychology, and sociology, all focused on understanding how individuals
↪ and groups interact within a society and the factors influencing their
↪ behavior, cultural norms, and societal institutions

`Culture` : Topics related to the broad area of knowledge encompassing all
↪ cultural practices or beliefs within societies, including related
↪ concepts or behaviors that people within a culture group share and
↪ understand as belonging together, like food, art, language, family
↪ structure, societal norms or religious rituals

`Medical` : Topics related to the broad area of knowledge and practice
 ↳ encompassing all medicine and healthcare, including diagnosing and
 ↳ treating diseases, preventative measures, specialties like surgery,
 ↳ cardiology, oncology, pediatrics, and more, all built upon the
 ↳ foundation of basic medical sciences and patient care principles
 `Finance` : Topics related to the broad area of knowledge encompassing
 ↳ activities like managing money, business ethics, investing, borrowing,
 ↳ lending, trading, budgeting, saving, and forecasting, essentially
 ↳ focusing on the acquisition, allocation, and management of capital
 ↳ within businesses
 , individuals, and governments across various financial markets and
 ↳ instruments
 `Legal` : Topics related to the broad area of knowledge encompassing
 ↳ Private, Public and Criminal Law, Criminal Justice, Law Enforcement,
 ↳ Policing, Justice Systems or Crime
 `Conversation` : Topics related to Conversation, Chit-Chat or Roleplay
 `Code` : Topics related to a specific subject/field within computer
 ↳ programming where software is designed and developed to solve problems
 ↳ related to a particular industry, business function, or area of
 ↳ expertise, essentially defining the target audience and unique
 ↳ requirements for the code being written including tasks like Code
 ↳ Generation, Code Fix and Code Explanation
 `Math` : Topics related to the broad field of study that uses numbers,
 ↳ shapes, and formulas to describe and quantify the world, including
 ↳ areas like Logical Reasoning, Quantitative Calculation, Pattern
 ↳ Recognition, Formulating Conjectures, Arithmetic, Algebra, Geometry,
 ↳ Number Theory, Set Theory and Analysis

If you are unable to confidently assign one of the above classes, you will
 ↳ simply respond with `Unspecified` and nothing else.

Note:

- You are only to respond with the name of the class you believe best
 ↳ matches the domain of the example.
- You are only allowed to classify the example into one of the following
 ↳ tags :
 [`Sciences`, `Technology`, `SocialSciences`, `Culture`, `Medical`,
 ↳ `Finance`, `Legal`, `Conversation`, `Code`, `Math`, `Unspecified`]

Here are a few examples :

Prompt : What is photosynthesis?
 Answer : `Sciences`

Prompt : What is the TCP/IP protocol and how does it work ?
 Answer : `Technology`

Prompt : How has globalization affected social cohesion ?
 Answer : `Social Sciences`

Prompt : What is an example of a popular dish that is available in multiple
 ↳ communities but known under different names?
 Answer : `Culture`

Prompt : How long does one have to fast before a fasting sugar blood test?
 Answer : `Medical`

Prompt : Analyze the impact of microfinance initiatives on poverty
 ↳ alleviation in developing countries.
 Answer : `Finance`

Prompt : What is the difference between a first-degree crime and a
 ↳ second-degree crime?
 Answer : `Legal`

Prompt : Hey! How are you?
 Answer : `Conversation`

Prompt : Given a variable x=3.142 in Python, how would I use an f-string to
 ↳ show just 1 decimal value?
 Answer : `Code`

Prompt : Solve the quadratic equation: $x^2 + 5x - 6 = 0$
 Answer : `Math`

Prompt : Use ABC notation to write a melody in the style of a folk tune.
 Answer :

1090 E.0.2 <task>

You are a helpful assistant whose goal is to classify the given prompt into
 ↳ a single class given the following definitions

`CodeTranslation` : Tasks related to the process of converting source code
 ↳ from one programming language to another while preserving the code's
 ↳ functionality
 `CodeExplanation` : Tasks related to the specific process of explaining a
 ↳ snippet of code in a programming language
 `CodeGeneration` : Tasks related to the specific process of generating a
 ↳ snippet of code in a programming language
 `Explanation` : Tasks related to explaining a concept in any domain
 `CreativeWriting` : Tasks related to any form of writing that employs
 ↳ creative, literary or poetic techniques that displays imagination or
 ↳ invention including role-play
 `QuestionAnswering` : Tasks related to any form of question answering,
 ↳ including open-ended questions, closed-ended questions about a given
 ↳ context and requests for information about a particular entity. This
 ↳ will also generally include your `what`, `which`, `who`, `when` type
 ↳ questions
 `OpenEnded` : Tasks related to any form of open-ended text generation like
 ↳ chat, conversation or chit-chat
 `InformationExtraction` : Tasks related to any form of information
 ↳ extraction usually involving some context
 `Summarization` : Tasks related to any form of summarization including but
 ↳ not limited to abstractive summarization, extractive summarization or
 ↳ concise descriptions of content
 `CodeFix` : Tasks related to the specific process of correcting/fixing a
 ↳ piece of code to achieve the desired result.
 `Reasoning` : Tasks involving any form of Ideation, Reasoning, Problem
 ↳ Solving, Instruction Following or Chain-of-Thought(CoT) in order to
 ↳ achieve the desired result.
 `Rewrite` : Tasks involving any form of
 ↳ re-writing/re-phasing/re-wording/re-framing in order to achieve the
 ↳ desired result.

`Classification` : Tasks related to specific request of classification
↪ where you are required to assign a thing to one of several groups
`Translation` : Tasks related to specific request of translating a given
↪ piece of text from one language to another language
If you are unable to confidently assign one of the above classes, you will
↪ simply respond with `Unspecified` and nothing else.

Note:

- You are only to respond with the name of the class you believe best
↪ matches the domain of the example.
- You are only allowed to classify the example into one of the following
↪ tags :
[`CodeTranslation`, `CodeExplanation`, `CodeGeneration`, `Explanation`,
↪ `CreativeWriting`, `QuestionAnswering`, `OpenEnded`,
↪ `InformationExtraction`, `Summarization`, `CodeFix`, `Reasoning`,
↪ `Rewrite`, `Classification`, `Translation`, `Unspecified`]

Here are a few examples :

Prompt : Translate the following Python function to equivalent JavaScript
↪ code that checks if a string is a palindrome.

```
def is_palindrome(str):  
    return str == str[::-1]
```

Answer : `CodeTranslation`

Prompt : Explain the following python function :

```
def is_palindrome(str):  
    return str == str[::-1]
```

Answer : `CodeExplanation`

Prompt : Generate a Python function to check whether a string is a
↪ palindrome.

Answer : `CodeGeneration`

Prompt : Explain briefly how the water cycle works

Answer : `Explanation`

Prompt : Translate the following sentence from English to Spanish, using a
↪ formal tone: 'We are pleased to announce the new partnership with our
↪ company.'

Answer : `Translation`

Prompt : You're a talk show host. Pick two guests that are wildly different
↪ from each other. Briefly introduce them

Answer : `CreativeWriting`

Prompt : Classify the sentiment of the following review as positive,
↪ negative, or neutral: 'The product exceeded my expectations!'

Answer : `Classification`

Prompt : What is the capital city of France?

Answer : `QuestionAnswering`

Prompt : Describe your ideal work environment

Answer : `OpenEnded`

Prompt : From the following news article, extract the names of the
↪ companies involved in the recent merger, along with the date the merger
↪ was announced.

Context: In a significant development in the tech industry, two leading
 ↳ companies have announced their merger, marking a new era of innovation
 ↳ and collaboration. The merger, which was officially announced on March
 ↳ 15, 2025, brings together TechInnovate Inc. and DigitalSolutions Corp,
 ↳ two giants in their respective fields. TechInnovate Inc, known for its
 ↳ cutting-edge research and development in artificial intelligence and
 ↳ machine learning, has been at the forefront of technological
 ↳ advancements. With a team of over 5,000 engineers and scientists, the
 ↳ company has consistently delivered groundbreaking solutions that have
 ↳ transformed various industries. DigitalSolutions Corp, on the other
 ↳ hand, is renowned for its expertise in software development and digital
 ↳ transformation. The company has a proven track record of helping
 ↳ businesses across the globe to modernize their
 operations and enhance their digital capabilities. With a workforce of
 ↳ over 10,000 professionals, DigitalSolutions Corp. has been a key
 ↳ player in driving digital innovation. The merger is expected to create
 ↳ a powerhouse in the tech industry, combining the strengths of both
 ↳ companies to offer comprehensive solutions that address the evolving
 ↳ needs of businesses and consumers. The combined entity will leverage
 ↳ TechInnovate's AI and machine learning capabilities with
 ↳ DigitalSolutions' software development expertise to develop
 ↳ next-generation technologies. Industry analysts predict that this
 ↳ merger will lead to significant advancements in areas such as
 ↳ autonomous systems, smart cities, and personalized healthcare. The
 ↳ synergy between the two companies is anticipated to drive innovation,
 ↳ improve efficiency, and create new opportunities for growth.

Answer : `InformationExtraction`

Prompt : Given the following story, provide a title that summarizes the
 ↳ idea behind the story:

Context:

There once was a girl who was frustrated with life and asked her father for
 ↳ advice. He asked her to bring an egg, two tea leaves, and a potato. He
 ↳ then started boiling water in three separate vessels. He put the egg,
 ↳ potato, and tea leaves in one vessel each. After a few minutes, he
 ↳ asked her to peel the egg and potato and strain the leaves. He
 ↳ explained to his daughter that:
 The soft egg was now hard.
 The hard potato was now soft.
 The tea had changed the water itself.
 When adversity is at our door, we can respond to it in different ways.

Moral: We decide how to respond to difficult situations.

Answer : `Summarization`

Prompt : Fix the code below to correctly identify a palindrome:

```
def is_palindrome(str):
    return str == str[-1]
```

Answer : `CodeFix`

Prompt : John has one pizza, cut into eight equal slices. John eats three
 ↳ slices, and his friend eats two slices. How many slices are left?

↳ Explain your reasoning step by step.

Answer : `Reasoning`

Prompt : Exaggerate this product description : 'Our new sneakers are

↳ comfortable, lightweight, and stylish.' to a paragraph that can be used
 ↳ by the marketing team

Answer : `Rewrite`

Prompt : Use ABC notation to write a melody in the style of a folk tune.
Answer :

1091 **E.0.3 <format>**

You are a helpful assistant whose goal is to classify the given prompt into
↪ a single class given the following definitions

`MCQAnswer` : Tasks related to multiple-choice type question answering.
↪ These prompts will typically contain multiple choices provided either
↪ in bullet form or eumerated numerically/alphabetically. This also
↪ contains multiple-choice question answer generation tasks.
`ChainOfThought` : Tasks related to Chain-of-Thought(CoT) style question
↪ answering. This also contains CoT style question answer generation
↪ tasks
`Enumeration` : Tasks that involve enumeration, bullet points, lists or
↪ itemization of any form
`XML` : Tasks that involve XML generation, validation or processing in any
↪ form
`Tabular` : Tasks that involve table generation, validation or processing
↪ in any form
`JSON` : Tasks that involve JSON generation, validation or processing in
↪ any form
`Markdown` : Tasks that involve Markdown generation, validation or
↪ processing in any form

If you are unable to confidently assign one of the above classes, you will
↪ simply respond with `Unspecified` and nothing else.

Note:

- You are only to respond with the name of the class you believe best
↪ matches the domain of the example.
- You are only allowed to classify the example into one of the following
↪ tags :
[`MCQAnswer`, `ChainOfThought`, `Enumeration`, `XML`, `Tabular`, `JSON`,
↪ `Markdown`, `Unspecified`]

Prompt : Use ABC notation to write a melody in the style of a folk tune.
Answer :