Answer is All You Need: Instruction-following Text Embedding via Answering the Question

Anonymous ACL submission

Abstract

This work aims to build a text embedder that can capture characteristics of texts specified by user instructions. Despite its tremendous potential to deploy user-oriented embeddings, none of previous approaches provides a concrete solution for it. This paper offers a new viewpoint, which treats the instruction as a *question* about the input text and encodes the *expected answers* to obtain the representation accordingly. Intuitively, texts with the same (implicit) semantics would share similar answers following the instruction, thus leading to more similar embeddings. Specifically, we propose InBedder that instantiates this embed-via-answering idea by only fine-tuning language models on abstractive question answering tasks. InBedder demonstrates significantly improved instruction-following capabilities according to our proposed instruction awareness tests and instruction robustness tests, when applied to both large language models (LLMs) (e.g., `llama-2-7b`) and smaller encoder-based LMs (e.g., `roberta-large`). Additionally, our qualitative analysis of clustering outcomes, achieved by applying different instructions to the same corpus, demonstrates a high degree of interpretability.

1 Introduction

Text embedders play a crucial role in large-scale textual data analysis and management. While existing models (Reimers and Gurevych, 2019a; Gao et al., 2021; Ni et al., 2022a,b; Wang et al., 2022; Xiao et al., 2023) demonstrate strong effectiveness in representing texts in general, they lack the ability to address user-specific objectives. This limitation hinders their application in more intricate scenarios where the embedding task requires the model to represent particular characteristics of the texts (Wang et al., 2023; Zhang et al., 2023b). Consider Figure 1, where a single set of reviews is required to be clustered in three distinct manners to derive meaningful insights. In response, we attempt to equip the text embedders with instruction-following capability in this paper.

One straightforward solution is to embed the concatenated instruction and input. Nonetheless, generic textual embeddings represent the texts in a form that can be used in textual similarity tasks, search and clustering, etc, rather than following instructions. Even for those that are trained with a multi-task contrastive objective (Su et al., 2023), there are no guarantees to generalize to new instructions due to the inevitably restricted diversity of training instructions written by humans.

We offer a novel viewpoint, which treats the instruction as a **question** about the input text and encodes the **expected answers**. Specifically, using the instructed input as the **prompt** to generative language models, we argue that the generated answers can be natively utilized to model semantic similarity under different instructions. For instance, given the sentences "*I love cats*" and "*I love dogs*", the instruction "*Do they love animals?*" will lead to a uniform response of "*Yes/Certainly/...*"; Conversely, distinct answers would be generated in response to "*What animals do they love?*". Therefore, we believe that the expectation of answer representations given the prompt can serve as an instruction-following embedding. We support this hypothesis by our empirical observations in Section 4.2 on existing instruction-tuned LLMs (Ouyang et al., 2022; Chung et al., 2022; Zhang et al., 2023a; Touvron et al., 2023a,b) [1] which have demonstrated that hidden states corresponding to the generated answers show considerably better instruction-awareness compared to those derived from the prompt.

Our observations indicate that function words and phrases in the answers do not contribute to better embedding quality. For instance, the introductory phrase "*Sure! Based on the input provided...*" is irrelevant to the answers and is commonly found

[1] For simplicity, we use LLMs to refer to instruction-tuned LLMs for the rest of the paper.

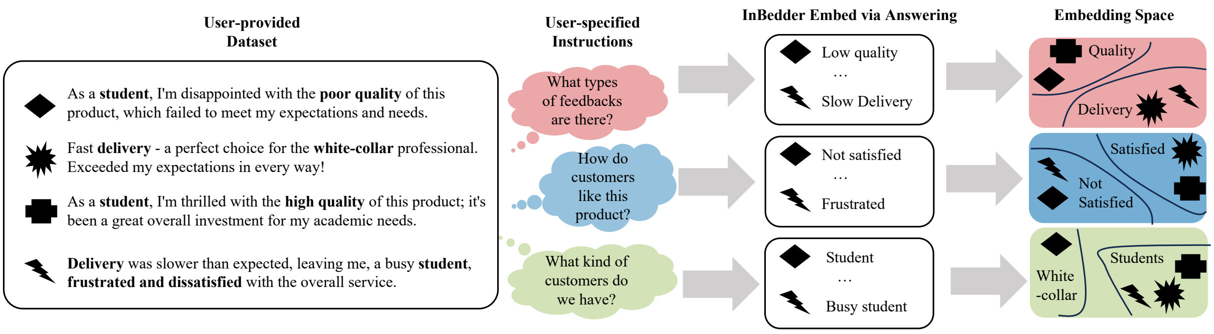


Figure 1: An example workflow of INBEDDER. INBEDDER takes in both user-provided dataset and user-specified instructions to produce personalized clusterings from which the user can extract insights about the dataset.

across various inputs. This redundancy can lead to inefficiency due to an increased decoding length, emphasizing the importance of answer brevity.

To effectively instantiate the embed-via-answering idea, we propose an **In**struction-following **Embedder** framework (INBEDDER), which is compatible with both large language models (LLMs) and smaller encoder-based LMs such as RoBERTa. Specifically, INBEDDER fine-tunes the LM on a union of 11 abstractive question answering (QA) datasets with $\sim 200,000$ paragraph-question-answer triplets where the answers are usually short and informative. To facilitate the model to learn (implicit) semantics, we choose abstractive QA in particular, as the answers cannot be directly extracted. We further simplify the answers by removing the stopwords, resulting in an average answer length of 2.89.

Due to the scarcity of evaluations focusing on instruction-following capabilities in the literature, we introduce a suite of tasks aimed at testing the ability of embedders to be instruction-aware, including (1) a triplet task that selects the closer sentence to the anchor sentence based on two different instructions, (2) an instruction-following sentence similarity task, and (3) a task for clustering the same corpus under various instructions. Furthermore, we evaluate INBEDDER s' robustness to the instructions by testing it on clustering datasets with either correct, implicit, or incorrect instructions. Our model is compared with both traditional text embedders as well as LLM-based embedders. The results demonstrate that our model can effectively process user instructions while generating high-quality embeddings. Moreover, we empirically observe that the hidden states corresponding to the first generated token can already effectively follow instructions, which makes it as efficient as traditional embedder methods by only requiring one forward pass of the LM. Finally, we propose to interpret the embedding clusters via post-processing on the generations of INBEDDER, and we observe that the clusters can reflect instruction-following capability when applying multiple instructions to the same corpus.

Our contributions are the following:

- We address a novel and challenging problem: instruction following of text embeddings and propose a framework, INBEDDER, to handle it by learning to answer user questions given inputs.
- We provide a comprehensive assessment for instruction-following text embedders, including instruction awareness tests and instruction robustness tests, which intuitively reflect the models' instruction-following capability.
- We propose an approach for extracting explanations from embedding clusters. We show that these explanations further reflect instruction-following capability.

2 Related Works

Text Embedder Text embedders empower modern natural language processing systems with a wide variety of abilities like clustering (Aggarwal and Zhai, 2012) and information retrieval (Karpukhin et al., 2020). In the representation space of text embedders, similar texts are embedded close to each other. Thus, Siamese networks (Reimers and Gurevych, 2019b) and contrastive learning (Gao et al., 2021) are proposed to learn the relative position of texts in the latent embedding space. Text embedders are further strengthened by incorporating more weakly supervised text similarity annotations (Wang et al., 2022; Xiao et al., 2023), model structure variants (Ni et al., 2022b) and multi-task learning (Su et al., 2023). However, these mainstream text embedders only process general textual similarity, ignoring the changing view on textual similarity based on user demands. Our INBEDDER shows a strong instruction-following text embedder in instruction following by using expected answer distributions.

Instruction Tuning Instruction-following (Zhang et al., 2023a) of LLMs is one of the core abilities for them to capture the user intents, which makes LLMs popular among users. InstructGPT (Ouyang et al., 2022) is one of the first trials on instruction-following LLMs, which unearths the potential of LLMs to complete tasks under instructions from users. With an outstanding instruction-following ability from reinforcement learning with human feedback (RLHF), ChatGPT (OpenAI, 2023) achieves great success. The open-source instruction-following LLMs, like LLaMA chat (Touvron et al., 2023a,b), also provide valuable resources for researcher. Our work instead studies instruction-following abilities of text embeddings. Previously, it has been discovered that LLM can explore and manipulate various attributes of texts (Peng et al., 2023). Moreover, LLM hidden states can effectively represent space and time (Gurnee and Tegmark, 2023), an aspect of texts (Zou et al., 2023) or a task defined by input-output pairs (Todd et al., 2023). Despite the potential, it is still unknown how to produce instruction-following embeddings from LLMs.

Goal-Driven Clustering With the recent advancements of instruction-following LLMs, goal-driven clustering has been proposed to group text corpora according to a personalized goal (Wang et al., 2023). In order to address such a challenging yet novel problem, Goal-EX (Wang et al., 2023) applies a two-step pipeline that first proposes cluster explanations according to a user-oriented goal with GPT-4 and then selects clustering assignments with another LLM. Zhang et al. (2023b) proposes another method that can incorporate user instructions to first determine sentence relationships via a triplet selection task and then produce clusters via fine-tuning. These produced clusters can then benefit personalized multi-document summarization (Coavoux et al., 2019; Fabbri et al., 2019; Lu et al., 2020). Our paper instead directly produces embeddings that are shaped by different instructions which does not require calling APIs of LLMs and potentially saves costs.

3 Problem Formulation

3.1 Instruction-following Embedder

We introduce the definition of instruction-following embedder in this section. A vanilla text embedder (denoted $\mathbf{Emb}(\cdot) : \mathcal{X} \to \mathcal{Z}$) (Reimers and Gurevych, 2019a; Gao et al., 2021; Ni et al., 2022a,b; Wang et al., 2022; Xiao et al., 2023) embeds texts from token sequence space $\mathcal{X}$ into a D-dimensional vector space $\mathcal{Z} \subseteq \mathbb{R}^D$, where similarities between two pieces of texts can be measured by a certain metric $Sim(\cdot, \cdot) : \mathcal{Z} \times \mathcal{Z} \mapsto \mathbb{R}$. These embeddings are usually designed to generically represent texts, i.e., they aim to capture the overall meaning. Such an approach, while versatile, often fails to align with a specific downstream application, e.g., grouping a corpus according to a particular interest or customizing a search engine with a targeted aspect. In this paper, we assume these goals can be specified by a user instruction I and then used to shape the embedding space without any fine-tuning to the text embedder. Under this circumstance, the similarity scores are conditional, i.e., $Sim(\cdot, \cdot|I)$.

The most straightforward approach is to just embed the concatenated instruction and input, which we will hereafter refer to as **prompt**,

$$Sim(X, X'|I) = Sim(\mathbf{Emb}(I \oplus X), \mathbf{Emb}(I \oplus X'))$$

where $X, X' \in \mathcal{X}$ and $\oplus$ is concatenation. In order to assess the instruction-following ability, we will present a series of tasks in Section 5.3 that require the model to understand the instructions.

Instructor (Su et al., 2023), a previous work, utilized a contrastive objective alongside multi-task learning to develop a more general text embedder. Our experiments in Section 5.3 demonstrate that their model does not adequately comprehend instructions. This is not surprising given the limited instruction diversity and the lack of encouragement to follow instructions during training.

Our hypothesis. We hypothesize the responses of LLMs (Touvron et al., 2023a,b; OpenAI, 2023; Chung et al., 2022) can be embedded to produce instruction-following embedding. Specifically, the LLMs are prompted to generate a response Y,

$$Y = \mathbf{LLM}(I \oplus X)$$

where both $X, Y \in \mathcal{X}$ are from token sequence space. $\mathbf{LLM}(\cdot) : \mathcal{X} \to \mathcal{X}$ is a function that maps prompts to responses. Usually, there could be multiple valid Y for a given prompt. In order to accommodate instruction-following embedding, we offer a novel viewpoint, which treats the instruction I as a **question** about the input text X and encodes the

expected answers (i.e. the responses to the question). In this paper, we study how to effectively embed expected answers.

3.2 Instruction Awareness Tests

Traditional generic embedding evaluation benchmarks, such as MTEB (Muennighoff et al., 2023) and SentEval (Conneau and Kiela, 2018), lack the ability to assess instruction awareness. In this work, we propose a set of new tasks specifically designed to comprehensively evaluate the capabilities of embedding models in this regard. We discuss the task formulations below and leave the detailed dataset creation procedures in Appendix A.

IntentEmotion. Inspired by previous works (Zhang et al., 2023b), we employ triplet tasks with two contrasting criteria, *i.e.* the intent and emotion of an utterance. A triplet task is composed of three different utterances $\{u_1, u_2, u_3\}$ where u_1, u_2 have the same intent but different emotions while u_1, u_3 have the same emotion but different intents, or vice versa. A success is defined under criterion I^{int} if $d(z_1^{int}, z_2^{int}) < d(z_1^{int}, z_3^{int})$. On the other hand, it is said to be a success for criterion I^{emo} if $d(z_1^{emo}, z_2^{emo}) > d(z_1^{emo}, z_3^{emo})$. Notice that the ranking is reverted under the two criteria. We use the harmonic mean of two success rates as our metric.

InstructSTSB. Traditional Semantic Textual Similarity (STS) Benchmark (Cer et al., 2017) lacks a definitive criterion for annotators to rely on, resulting in the subjectivity of the ratings. Hence, we create another instruction-based STS task where the two sentences are similar or dissimilar based on different instructions. We measure the Spearman correlation from cosine similarities. Notice that a similar dataset was first proposed in Deshpande et al. (2023). The main differences are that (1) our dataset is created directly from the original test set of STSB [2] via brainstorming instructions; (2) our dataset only involves two ratings 0 and 1 indicating same or different, unlike the $1 \sim 5$ rating scale in their case, reducing subjectivity in the evaluation.

NYTClustering. We present the clustering results for the New York Times (NYT) dataset (Sandhaus, 2008), which is categorized according to two annotations: topic and location of the news articles. The results are reported using the harmonic mean of the V-measure for both clustering types.

[2] https://huggingface.co/datasets/mteb/stsbenchmark-sts/viewer/default/test

3.3 Instruction Robustness Tests

We further introduce an evaluation task specifically designed to assess the robustness of embedding models to various instructions. We employ clustering to evaluate model performance in response to correct, implicit, and incorrect instructions. For each clustering task, a set of 10 correct instructions is generated by instructing GPT-4 to paraphrase the original task instructions. Similarly, a set of 10 implicit instructions is produced by GPT-4 through the rephrasing of the instructions to convey them implicitly. Moreover, 10 incorrect instructions are created by prompting GPT-4 to formulate instructions that diverge from the original task objective. Examples of these instructions are illustrated in Figure 9. The difference in average performance between correct and incorrect instructions is denoted as Δ_{ci}, and the difference in average performance between implicit and incorrect instructions is denoted as Δ_{ii}. See Appendix B for details.

4 Methodology

In this section, we introduce INBEDDER that is derived from observations on LLMs. We first define ways to acquire sentence embeddings from LLMs in Section 4.1. Subsequently, we illustrate early observations in Section 4.2. Finally, we introduce INBEDDER that fine-tunes an LLM to an instruction-following embedder, in Section 4.4.

4.1 Encoding Methods

Contemporary LLMs are usually composed of one (encoder-/decoder-only) or two (encoder-decoder) transformer architectures with L layers. The input of the transformer is a sequence of embeddings $[h_0^1, \cdots, h_0^N]$ where N is the length of prompt $(I \oplus X)$. Each layer will then produce an intermediate hidden state $\mathbf{h}_l$ until the last layer which is used to predict the $(N+1)$th output token. We first introduce two strategies to acquire a single aggregated embedding from an off-the-shelf LLM.

Direct Encoding directly utilizes LLM hidden states. Since it is not obvious which hidden states contain the most relevant information to the prompt, we explore 5 aggregation methods for each layer:

1) The average of generation Y's hidden states with generation length N_g,

$$\mathbf{Emb}_l^{\text{avg-gen}} = \frac{1}{N_g + 1} \sum_{j=0}^{N_g} h_l^{(N+j)},$$

2) The average of prompt hidden states. This will serve as a direct comparison to "avg-gen".

$$\mathbf{Emb}_l^{\text{avg-ppt}} = \frac{1}{N-1} \sum_{i=1}^{N-1} h_l^i,$$

3) The hidden states used to predict the first token in generations,

$$\mathbf{Emb}_l^{\text{1st-gen}} = h_l^N,$$

4) The last generation hidden states,

$$\mathbf{Emb}_l^{\text{last-gen}} = h_l^{N+N_g},$$

5) The average of all hidden states,

$$\mathbf{Emb}_l^{\text{avg-all}} = \frac{1}{N+N_g} (\sum_{i=1}^{N} h_l^i + \sum_{j=1}^{N_g} h_l^{N+j}),$$

In practice, we adjust the aggregation methods with regard to the uniqueness of each architecture. See details in Appendix C.

While direct encoding is commonly applied for conventional encoders (Wang et al., 2022; Su et al., 2023), using only the input information might not reveal the implicit features that can be inducted by answering the prompt. Thus, we propose **Re-encoding**, which is a two-step approach that first produces the responses Y based on the prompts and then re-encode them using another embedder $\mathbf{Emb}_R$. Mathematically,

$$\mathbf{Emb}^{\text{re-enc}} = \mathbb{E}_{P(Y|I \oplus X)}[\mathbf{Emb}_R(Y)]$$

We then re-write the above with an empirical estimation,

$$\mathbf{Emb}^{\text{re-enc}} = \frac{1}{|\mathcal{S}_Y|} \sum_{Y \sim \mathcal{S}_Y} \mathbf{Emb}_R(Y)$$

where $\mathcal{S}_Y$ is sampled from response distribution $P(Y|I \oplus X)$. We choose $\mathbf{Emb}_R$ to be a (relatively) light-weight sentence transformer, thus the efficiency of re-encoding is similar to that of avg-gen. And when $N_g = 1$, all the aggregation methods possess the same efficiency.

4.2 Answer Speaks Louder

In this section, we show some early observations that guide us towards the design of InBedder. With the definitions in the previous section, we show the performance comparison among various aggregation methods on an existing LLM in Figure 2 left. To assess performance, we devised three

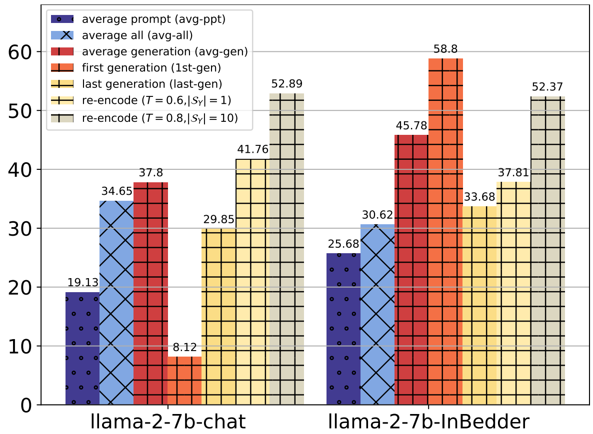


Figure 2: Instruction awareness tests performance (averaged over 3 datasets) for different encoding methods from last layer. T is the decoding temperature while $\mathcal{S}_Y$ is the sample size. **Observations:** (1) The generation/answer side (i.e., the checkerboard pattern) is more informative than the prompt side (i.e., the dark blue with dotted pattern); and (2) In `llama-2-7b-InBedder`, `1st-gen` seems to significantly outperform others.

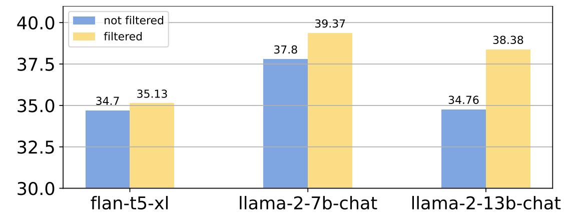


Figure 3: Filtered vs. not filtered (i.e., avg-gen on the last layer of each LLM). **Observations:** filtering hidden states associated with uninformative contents can marginally improve performance.

distinct tasks that require the embedders to comprehend not only the raw texts but also the instructions, which will be further elaborated in Section 5.3. It is evident that hidden states derived from generations (avg-gen) consistently surpass those from prompts (avg-ppt). Additionally, averaging all hidden states, denoted as avg-all, does not enhance performance. Finally, an examination of three distinct models in Figure 2 reveals that re-encoding consistently outperforms all direct encoding methods, while increasing the sample size $|\mathcal{S}_Y|$ will further boost performance. These observations manifest our hypothesis that answers are more important for instruction-following embedder, in other words "answers speak louder". See analysis of model depth in Figure 6.

4.3 Answer Brevity Matters

One notable issue for using LLMs as embedders is their propensity to produce content that, while enhancing readability for humans, may not be directly relevant to the task at hand. For instance, `llama-2-7b-chat` frequently initiates responses with introductory phrases such as "Based on the input provided..." or "The topic of the news article is..." which are common across various requests. It is thus plausible to conjecture that the hidden states responsible for generating these superfluous contents contribute no useful information to the embedding task. Following this intuition, we conducted a simple experiment to validate the impact of filtering out hidden states associated with such content. Specifically, we compiled a list of candidate tokens for exclusion, which includes tokens present in the instruction, stopwords, and common phrases like "Based on". While calculating "`avg-gen`", we disregard hidden states linked to the generation of tokens from this list. The outcomes, depicted as green bars in Figure 3, indicate a marginal improvement in the performance of the three evaluated models upon the removal of non-informative content, thereby validating the assumption that these hidden states are indeed redundant.

4.4 Our INBEDDER

In order to effectively instantiate the embed-via-answering, we propose a novel fine-tuning framework leveraging existing curated question-answering (QA) datasets. Specifically, we collect a set of 11 abstractive QA datasets [3], which sum up to $\sim 200,000$ paragraph-question-answer triplets. As discussed in Section 3.3, we treat the paragraph as the input, the question as the instruction, and generate the answers. Note that, we pre-process the answers so that all the stopwords are removed, which results in an average response length of 2.89. As will be demonstrated in the experiments, such a pre-processing step significantly contributes to our method. We then fine-tune the LM with an autoregressive objective.

We emphasize three inherent advantages of INBEDDER: (1) QA datasets usually have concise outputs that will promote the LLMs to respond eagerly without considering too much about readability. Refer to Figure 8 for an example. (2) Compared to multi-task datasets introduced in Su et al. (2023), our dataset offers significantly greater diversity in instructions, attributed to the variety of questions associated with each input paragraph, unconstrained by question format. And most importantly, the questions are publicly available without any extra costs. (3) The auto-regressive objective induces better interpretability of generated embeddings via mining explanations from its generations.

5 Experiments

We explain our experimental setup in Sections 5.1 and 5.2. Next, we share outcomes from our tests on instruction awareness and robustness in Sections 5.3 and 5.4. Lastly, we compare results for general embedding tasks in Section 5.5.

5.1 Implementations

We fine-tune INBEDDER from various language models such as (1) `roberta-large`, (2) `opt-1.3b`, (3) `opt-2.7b`, and (4) `llama-2-7b` [4]. For masked language modeling-based roberta-large, we adapt our framework by appending mask tokens behind prompts with the same length as target tokens and then training with mask token prediction loss. During testing, we append 3 mask tokens to represent the answer. We consistently train for 1 epoch with a learning rate of 2×10^{-5}. For INBEDDER, we always employ the same pattern to feed the inputs to the models, i.e. "### Input:\n{input}\n\n### Instruction:\n{instruction}\n\n### Response:". For `llama-2` chat models, we provide an extra prefix to induce shorter answers: "Your task is to give an answer according to the instruction and input. Please keep your answer short.". At test time, we allow the maximum generation length to be 40 for `llama-2` chat models and 3 for our INBEDDER. We exclude hidden states corresponding to special tokens. We consistently use `e5-large-v2` (Wang et al., 2022) as our re-encoder. Lastly, we set the maximum prompt length to be 512 (including instruction, input, and the words in the pattern). We train and evaluate these models with at most 4×A100 (PCIe). Training can be finished in about 10 hours for `llama-2-7b-InBedder`.

5.2 Compared Methods

We compare with generic sentence embedding models: E5 (Wang et al., 2022) and Instructor (Su et al., 2023) [5]. We also compare with instruction-tuned models: `llama-2` chat models (Touvron

[3]We also include several multiple-choice QA datasets but remove all the wrong choices.

[4]huggingface ids: "roberta-large", "facebook/opt-1.3b", "facebook/opt-2.7b", "meta-llama/Llama-2-7b-hf".

[5]huggingface ids: "intfloat/e5-large-v2", "hkunlp/instructor-large"

Model	I.STSB	IntEmo	NYT	Avg
e5-large-v2(w/o instruction)	0.00	30.24	50.07	26.77
instructor-large	-15.02	47.96	49.96	27.63
roberta-large-alpaca(avg-gen)	8.43	90.34	21.60	40.12
roberta-large-InBedder (avg-gen)	14.81	**91.07**	51.18	52.35
opt-1.3b-alpaca(avg-gen)	-1.81	71.51	12.88	27.53
opt-1.3b-InBedder (1st-gen)	7.47	89.96	53.13	50.19
opt-2.7b-alpaca(avg-gen)	3.95	75.09	13.32	30.79
opt-2.7b-InBedder (1st-gen)	10.45	84.54	59.43	51.47
llama-2-7b-chat(re-enc)	16.56	79.32	29.41	41.76
llama-2-13b-chat(re-enc)	19.76	73.60	32.74	42.03
llama-2-7b-w/o-process(1st-gen)	21.10	83.64	52.72	52.49
llama-2-7b-InBedder (1st-gen)	**22.07**	89.68	**64.65**	**58.80**

Table 1: Instruction awareness tests results. The best encoding methods are shown in parentheses for each non-sentence-transformer model. We only consider the last layer in this table. I.STSB is short for InstructSTSB.

Model	AskU.	SciD.	StackO.	20news	Avg
e5-large-v2(w/o instruction)	59.01	83.84	50.60	47.94	60.35
instructor-large	63.48	81.83	50.50	53.51	62.33
roberta-large-alpaca(avg-gen)	56.29	73.02	41.66	40.61	52.90
roberta-large-InBedder (avg-gen)	55.50	73.80	41.00	41.93	53.06
opt-1.3b-alpaca(avg-gen)	55.89	69.68	42.43	38.49	51.62
opt-1.3b-InBedder (1st-gen)	59.09	71.33	43.08	46.45	54.99
opt-2.7b-alpaca(avg-gen)	55.65	76.26	42.45	32.11	51.62
opt-2.7b-InBedder (1st-gen)	59.94	75.33	41.93	49.07	56.57
llama-2-7b-chat(re-enc)	55.26	75.81	41.43	25.34	49.46
llama-2-13b-chat(re-enc)	53.69	77.64	38.84	30.77	50.24
llama-2-7b-w/o-process(1st-gen)	61.25	83.13	44.39	50.68	59.86
llama-2-7b-InBedder (1st-gen)	60.32	80.61	44.77	52.33	59.51

Table 2: Generic sentence embedding task performance. The best encoding methods are shown in parentheses for each non-sentence-transformer models. We only consider the last layer in this table.

et al., 2023b) that are fine-tuned with RLHF [6]. For `roberta-large` and `opt` models we compare with those checkpoints tuned on Alpaca (Taori et al., 2023). For Alpaca fine-tuning, we follow the original dataset and hyperparameters. [7]

5.3 Instruction Awareness Tests Results

In Figure 2 right, quite unexpectedly, we observe that using `1st-gen` in InBedder achieves the best performance and it outperforms the other encoding methods by a significant amount. We hypothesize that although `1st-gen` is utilized solely for decoding the first token in the generations, it may contain the most relevant information due to the model being trained on concise outputs. Further qualitative analysis in Table 4 shows that the first generated tokens usually correspond to the answer. We then present comparisons across various models in Table 1. Fine-tuning InBedder appears to be effective across a range of model sizes, from the 355M model `roberta-large` to the 1.3/2.7b `OPT` and the 7b `llama-2`. We can also observe that without pre-processing, the performance will be significantly degraded on instruction-awareness according to `llama-2-w/o-process`, which further validates that conciseness of outputs is important.

5.4 Instruction Robustness Tests Results

Figure 4 presents the results obtained across three models. Compared to `instructor-large` and `llama-2-7b-chat`, our model demonstrates larger values of Δ_{ci} and Δ_{ii}, as well as superior average performance when applying correct instructions. This indicates that InBedder exhibits a better understanding of correct or implicit instructions and possesses greater robustness against incorrect ones.

5.5 Generic Sentence Embedding Tasks

Finally, we also compare performances on generic sentence embedding tasks. We choose a subset of tasks from the MTEB (Muennighoff et al., 2023) benchmark, including: "TwentyNewsgroupsClustering", "AskUbuntuDupQuestions", "SciDocsReranking" and "StackOverflowDupQuestions". The first task is a clustering task with V-measure as its metric. The others are reranking tasks that require the model to correctly identify the sentences that are close to the query semantically. We follow MTEB to use the "mean average precision (MAP)" as our metric. For each task, we design a task-level prompt that describes the requirements. We observe in Table 2 that InBedder has a closer performance to state-of-the-art embedders E5 (Wang et al., 2022) and Instructor (Su et al., 2023) than other LLM-based embedders, even though it was not trained with a contrastive objective as most sentence transformers do.

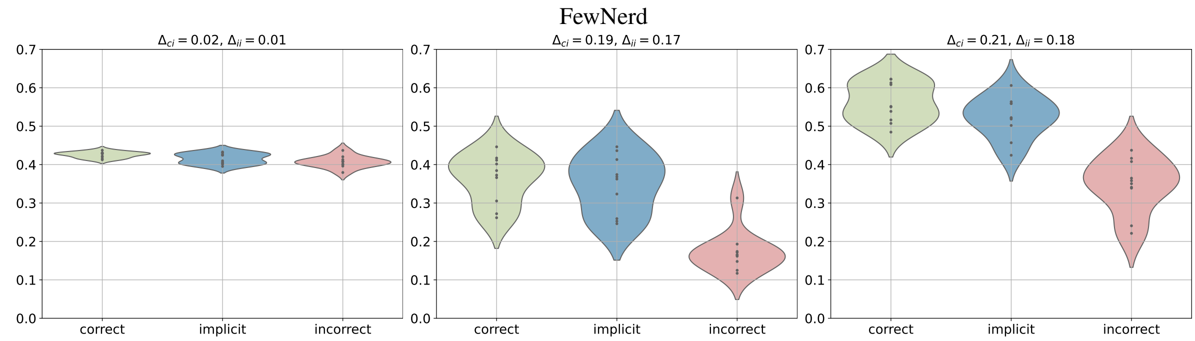
6 Embedder Clustering Interpretation

Interpreting neural embeddings has long been an aspiration in numerous research endeavors (Panigrahi et al., 2019; Trifonov et al., 2018). We show in this section that InBedder naturally possesses interpretability due to its instruction-following training objective. In this section, we propose a method to "extract answers" from produced semantic clusters of InBedder.

Specifically, we directly post-process the generated sequences of InBedder to collect identifi-

[6] huggingface id: "meta-llama/Llama-2-7b-chat-hf" and "meta-llama/Llama-2-13b-chat-hf"

[7] `https://github.com/tatsu-lab/stanford_alpaca/tree/main`



(a) instructor-large (b) llama-2-7b-chat (re-enc) (c) llama-2-7b-InBedder (1st-gen)

Figure 4: Instruction robustness tests results. Three set of instructions are tested: correct, implicit and incorrect. Δ_{ci} denotes the separation between mean of correct and incorrect. Δ_{ii} denotes the separation between mean of implicit and incorrect. InBedder shows better robustness and performance overall. See more datasets in Figure 7

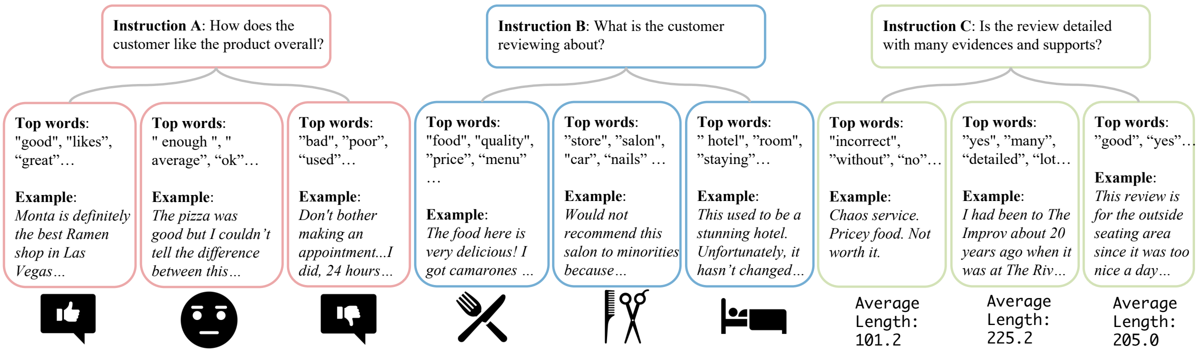


Figure 5: Instruction-following clustering with `llama-2-7b-InBedder` on Yelp reviews. The results are produced by simply instructing the model. 3 clusters along with top words and examples are shown for each instruction where we can observe clear accountability to the instructions.

able information about a cluster. To differentiate clusters, we initially collect outputs from each cluster following K-means clustering and concatenate these outputs into a single document per cluster. Subsequently, we employ Tf-idf to vectorize these K documents, resulting in K feature vectors. The dimensions of each vector denote the relative frequency of a word's occurrence in one document compared to its occurrence in others. Hence, we rank feature words according to the corresponding value in the feature vector, which will then be designated as cluster keywords.

Table 3 in Appendix presents explanations derived from `llama-2-7b-InBedder`. When compared to the label components of each cluster, the top words collected effectively capture the characteristics of each cluster. To showcase the instruction-following capability of InBedder, cluster explanations are further illustrated with three distinct instructions in Figure 5 using the Yelp review dataset (Zhang et al., 2015) (originally designed for sentiment analysis). The top words distinctly delineate the differences between clusters, in accordance with the provided instructions. For example, under "Instruction B" various products that are being reviewing are revealed from clusters. On the other hand, under "Instruction C", variations in average sentence length are observed, indicating the degree of detail present in the review.

7 Conclusions and Future Work

Our work addresses a novel problem, text embedding with instruction-following. We propose InBedder to produce desirable embeddings from LLMs via generating expected answers. The method is inspired by observations on existing LLMs. Our text embedder model llama-2-7b-InBedder outperforms both traditional sentence transformers and aggregated embeddings from LLMs on instruction-awareness tests, and instruction robustness tests and achieves close performance on traditional generic tasks. We also show that InBedder is inherently applicable for embedding cluster explanation which will facilitate user-oriented dataset analysis. We encourage future works to investigate more efficient solutions which is important in large-scale retrieval systems.

Limitations

Efficiency. Our model is not sufficiently efficient for large-scale retrieval tasks. In retrieval, corpus is usually encoded as vector embeddings beforehand, the only operation conducted is to encode the query and to compute the cosine similarities between the query and corpus. However, INBEDDER requires encoding the entire corpus *w.r.t.* each user query which results in significant latency. However, one possible solution is to first select the most similar candidates and then use INBEDDER as a query-dependent reranker.

Effectiveness on generic tasks. The results in Table 2 show that INBEDDER does not surpass traditional sentence transformers on especially generic reranking tasks. (1) Our ambition is to provide an instruction following embedder that could potentially facilitate user-oriented tasks rather than optimizing for high-performing sentence embedding and we leave the exploration on that dimension in future works. (2) INBEDDER might benefit from better prompt design or task description which we have discussed in Section 5.4.

Ethical Considerations

Our work focuses on the instruction-following abilities of text embedders. Note that the proposed framework does not guarantee the model to generate safe contents for embedding interpretability. Nonetheless the produced embeddings do not contain (understandable) harmful contents by themselves. Furthermore, we will release our code, datasets and checkpoints upon acceptance in order to facilitate reproducibility.

References

Charu C. Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 77–128. Springer.

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on NLP for ConvAI - ACL 2020*. Data available at https://github.com/PolyAI-LDN/task-specific-datasets.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Maximin Coavoux, Hady Elsahar, and Matthias Gallé. 2019. Unsupervised aspect-based multi-document abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 42–47, Hong Kong, China. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Ameet Deshpande, Carlos Jimenez, Howard Chen, Vishvak Murahari, Victoria Graf, Tanmay Rajpurohit, Ashwin Kalyan, Danqi Chen, and Karthik Narasimhan. 2023. C-STS: Conditional semantic textual similarity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5669–5690, Singapore. Association for Computational Linguistics.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.

Wes Gurnee and Max Tegmark. 2023. Language models represent space and time. *arXiv preprint arXiv:2310.02207*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Yao Lu, Yue Dong, and Laurent Charlin. 2020. Multi-XScience: A large-scale dataset for extreme multi-document summarization of scientific articles. In

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 8068–8074, Online. Association for Computational Linguistics.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022a. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2022b. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 9844–9855. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Abhishek Panigrahi, Harsha Vardhan Simhadri, and Chiranjib Bhattacharyya. 2019. Word2Sense: Sparse interpretable word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5692–5705, Florence, Italy. Association for Computational Linguistics.

Letian Peng, Yuwei Zhang, and Jingbo Shang. 2023. Generating efficient training data via llm-based attribute manipulation. *CoRR*, abs/2307.07099.

Nils Reimers and Iryna Gurevych. 2019a. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019b. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1102–1121. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. 2023. Function vectors in large language models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Valentin Trifonov, Octavian-Eugen Ganea, Anna Potapenko, and Thomas Hofmann. 2018. Learning and evaluating sparse interpretable sentence embeddings. In *Proceedings of the 2018 EMNLP Workshop*

BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 200–210, Brussels, Belgium. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *CoRR*, abs/2212.03533.

Zihan Wang, Jingbo Shang, and Ruiqi Zhong. 2023. Goal-driven explainable clustering via language descriptions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10626–10649, Singapore. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighof. 2023. C-pack: Packaged resources to advance general chinese embedding. *CoRR*, abs/2309.07597.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2023a. Instruction tuning for large language models: A survey. *CoRR*, abs/2308.10792.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023b. ClusterLLM: Large language models as a guide for text clustering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13903–13920, Singapore. Association for Computational Linguistics.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

A Instruction Awareness Tests Creation

The datasets employed in this section have proper license for further modification. Note that our created datasets are intended for research purpose only.

IntentEmotion We use BANKING77 (Casanueva et al., 2020) test set as our base dataset to create triplets. We prompt `gpt-4-0613` to create utterances that have the same intent but two different emotions "optimistic" and "frustrating", denoted as u^1_{opt} and u^1_{fru}, with the following prompt.

> *Could you modify the emotion (one optimistic and one frustrating) of following utterance without changing the intent ("[INTENT]")?*
> *"[TEXT]"*
> *Please output a JSON object containing keys "optimistic" and "frustrating", and no other things.*

For each generated utterance, we then prompt the same LLM again to modify the intent of the utterance, denoted as u^2_{opt} and u^2_{fru} with the following prompt.

> *Modify the intent of the above utterances (i.e. from "[INTENT]" to another one that you brainstormed. Usually by modifying the objects or actions) without changing the emotions. Same as before, output a JSON object containing keys "optimistic" and "frustrating", and no other things.*

This will result in 4 generated utterances (disregarding the original utterance), then we group these utterances into 4 triplets according to two criteria:

$$\{u^1_{opt}, u^2_{opt}, u^1_{fru}\}, \{u^1_{fru}, u^2_{fru}, u^1_{opt}\},$$
$$\{u^1_{opt}, u^1_{fru}, u^2_{opt}\}, \{u^1_{fru}, u^1_{opt}, u^2_{fru}\}$$

In each triplet, the first one is the anchor, the second is the positive and the last is the negative. Thus the first two triplets follow emotion criterion while the last two follow intent criterion. As a result, there are 12,320 triplets in total, half for emotion and half for intent. We calculate the triplet success rates for both criteria separately, and then calculate the harmonic mean.

InstructSTSB We use STSb (Cer et al., 2017) test set as out base dataset to generate sentence pairs. We generate two instructions, one that can discriminate the sentence pair and the other that can not. To achieve that, we prompt `gpt-4-1106-preview` sequentially with the following two instructions.

> The following two sentences have similar surface forms:
>
> 1. [SENTENCE1]
> 2. [SENTENCE2]
>
> In order to discriminate the two sentences, what question would you ask? (e.g. what is the subject of the sentence?) Please output a JSON object that contains the key "question".
>
> Similar to the above, in order to make the answers to the two sentences immune to discrimination, what question would you ask? (e.g. what is the subject of the sentence?) Please output a JSON object that contains the key "question".

As a result, there are 2758 sentence pairs in total. We then set the ratings for discriminative pairs to 0 and 1 for non-discriminative pairs. Following previous implementation (Muennighoff et al., 2023), we use spearman correlation as our metric and cosine similarity as similarity measurement.

NYTClustering There are no further modifications to this dataset since it already contains two sets of annotations, one for location and one for topic.

B Instruction Robustness Tests Creation

The datasets employed in this section have proper license for further modification. Note that our created datasets are intended for research purpose only. We adopt clustering datasets FewNerd, FewRel and FewEvent from Zhang et al. (2023b). We adapt clustering datasets RateMyProf and Feedbacks from Wang et al. (2023). All these datasets are clustered under a complex task instruction such as entity type or the aspect of the review or the reason to (dis)like. Since the original paper (Wang et al., 2023) does not provide the annotations, we use `gpt-4-1106-preview` to select annotations for them and then we post-process the dataset so that the clusters are equal in size. As a result, Feedbacks contains 3 clusters and 756 human feedbacks to machine generated data. RateMyProf contains 4 clusters and 2,296 reviews from RateMyProfessor. Lastly, we provide various instructions that are correct, implicit or incorrect by prompting GPT-4 (webpage) to generate similar, implicit, or dissimilar instructions.

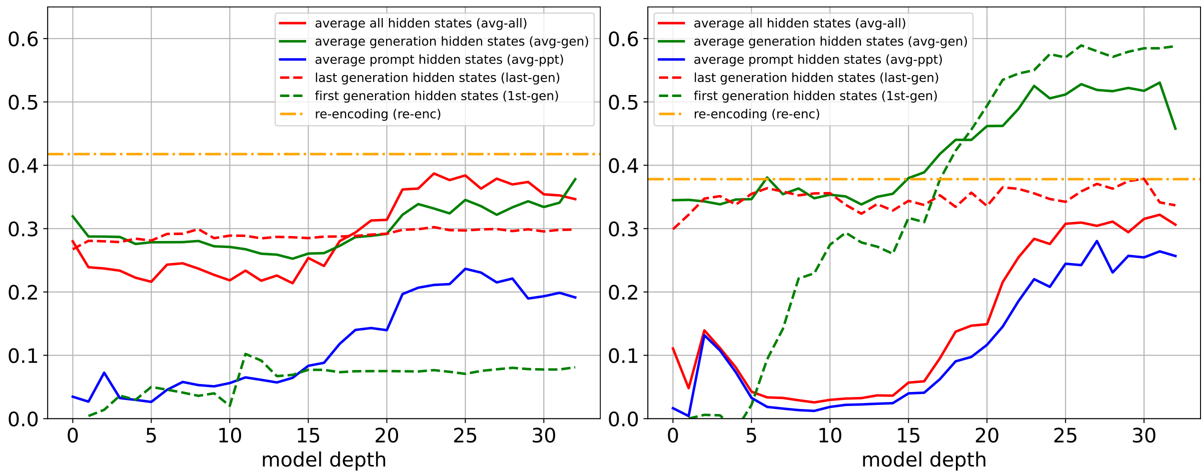
C Details on Direct Encoding

The direct encoding proposed in Section 4.1 are all compatible with decoder-only transformers. For encoder-decoder models such as `flan-t5`, because of the two separated models, we remove `avg-all` since the hidden states are not in the same space. Besides, we extract `avg-ppt` from encoder and `avg-gen&1st-gen&last-gen` from decoder respectively. Notice that for `1st-gen`, we use the hidden states for the BOS token in the decoder side. For encoder-only models, we remove `1st-gen` and `last-gen`. We implement the sentence embedding function by generating tokens first [8] and then cache the intermediate hidden states for further compute. Considering the efficiency, `avg-ppt&1st-gen` only require single forward pass while the others require iterative generations and thus depending on the generation length.

> Input: Did you know that vegetables can grow in the climates they are not used to? … What these engineers have been using is very simply cold sea water. How did they use it? …
>
> Instruction: What is the report mainly about?
>
> Output: use sea water

Figure 8: An example from our training data.

[8] Notice that, in huggingface (Wolf et al., 2019), both decoder-only and encoder-decoder model can use "generation" function: `https://huggingface.co/docs/transformers/main_classes/text_generation`. For encoder-only, we simply concatenate the "[MASK]" tokens after the prompts for generation.



(a) `llama-2-7b-chat` (b) `llama-2-7b-InBedder`

Figure 6: Instruction awareness tests results vs. model depth.

RateMyProf				
cluster	1 (lowest entropy)	2	3	4 (highest entropy)
label components	personal qualities:327 ease or difficulty:31 assessment-related:15 amount of work:10	amount of work:331 assessment-related:215 ease or difficulty:128 personal qualities:12	assessment-related:173 personal qualities:99 ease or difficulty:77 amount of work:62	ease or difficulty:338 amount of work:171 assessment-related:171 personal qualities:136
top words	teaching,personality, classroom,quality, good,teacher, skills,student...	assignments,homework, workload,expectations, tests,difficulty, professor,exams...	teaching,quality, grade, ability, style,grading, lectures,enough...	difficulty,professor level,coursework lectures,tests class,course...

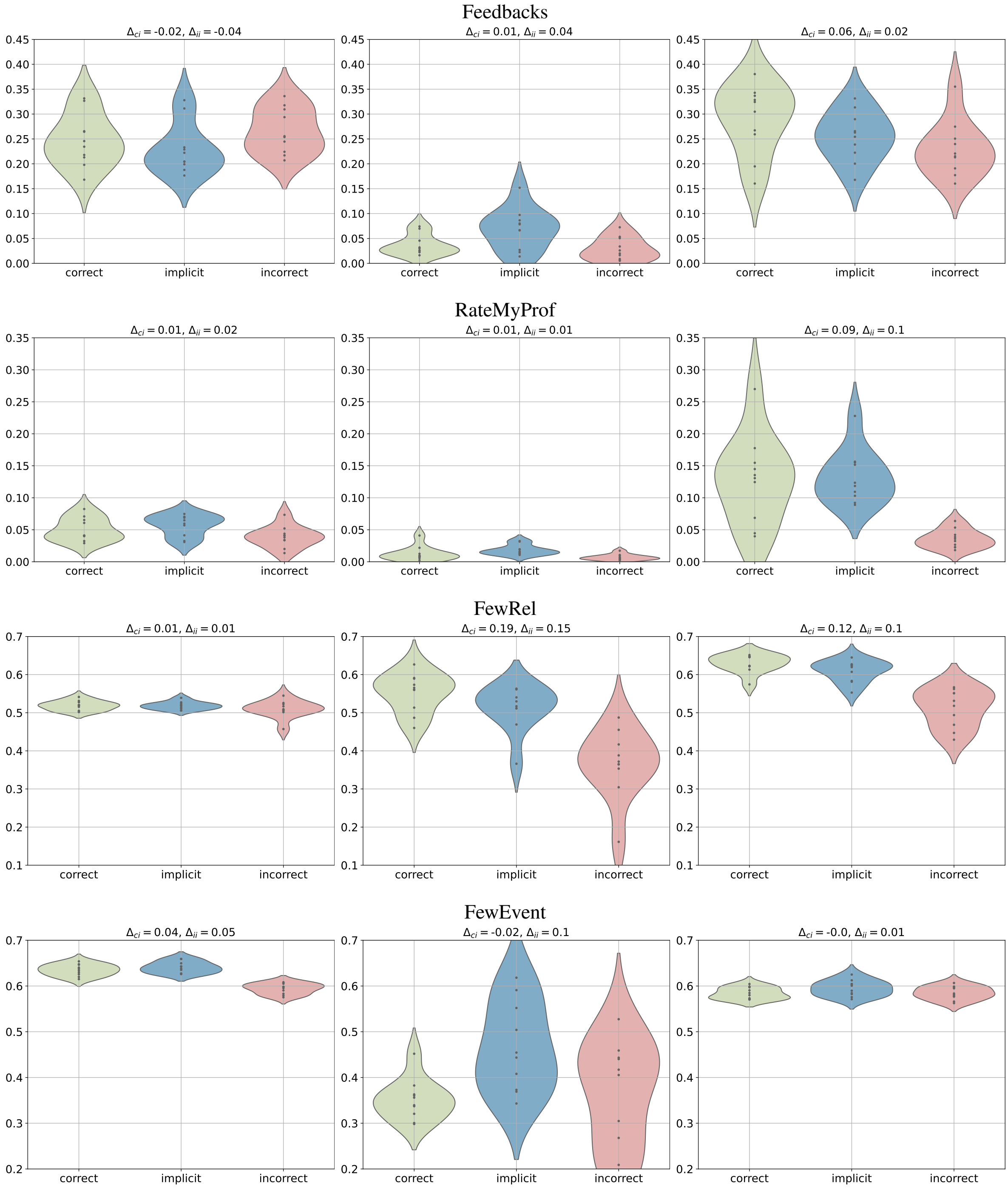
Feedbacks			
cluster	1 (lowest entropy)	2	3 (highest entropy)
label components	Structure, Coherence:229 Inclusion of Main Points:9 Content Accuracy:7	Content Accuracy:148 Inclusion of Main Points:134 Structure, Coherence:10	Inclusion of Main Points:109 Content Accuracy:97 Structure, Coherence:13
top words	sentence,better, structure,written, improved,flow, unclear,read...	dislike,accurate, author,generated, mention,machine, accuracy,inaccurate...	like,feedback, post,likes, good,human, advice,relationship...

FewRel					
cluster	1 (lowest entropy)	2	3	63	64 (highest entropy)
label components	taxon rank:24	heritage designation:70 located in the administrative...:1 location:1	taxon rank:46 instance of:1 said to be the same as:1 country of citizenship:1	movement:9 religion:6 work location:6 said to be the same as:5	language of work or name:23 said to be the same as:16 followed by: =10 applies to jurisdiction:7
top words	family,species, gastropod,marine, sea,urothoidae, psolidae,feed...	historic,registe, places,listed, national,historical, house,significance...	family,subfamily, order,genus, families,tribe, sent,orders...	friends,beowulf*, personalities,jewish, slave,personality, independence,owner...	language,minor, wikipedia,candela, french,translation, flag,major...

FewNerd						
cluster	1 (lowest entropy)	2	3	56	57	58 (highest entropy)
label components	Geo-Political:139 film:1 car:1 education: 1	Geo-Political:92 company:1 government:1 sports team:1	award:35 living thing:1 broadcast program:1	language:33 Geo-Political:8 written art:4 software:4	artist, author:47 scholar:25 actor:9 Geo-Political:9 ...	film:25 broadcast program:12 Geo-Political:9 written art:6 ...
top words	city,america, europe,continent usa,state, north,county...	country,europe, jordan,ireland, india,america, uk,germany...	award,prize, awards,best, given,show, film,category...	language,spoken dialect,sentence languages,skerry, dialects,french...	person,artist dan,name paris,well, professor, etc...	film,movie comedy,actor series,tv, directed,play...

FewEvent					
cluster	1 (lowest entropy)	2	3	33	34 (highest entropy)
label components	Military Service:150	Marry:162 Leadership:2 Place Lived:1	Olympic Medal Honor:189 Education:3 Olympic Athlete Affiliation:3	Leadership:19 Employment Tenure:9 Education:8 Place Lived:6	Sentence:14 Transfer Money:13 Charge Indict:13 Transport person:9
top words	soldier,medal, war,honor, received,sailor, vietnam,killed...	wife,birth, died,maria, child,king, marriage,queen...	olympics,medal, gold,summer, winner,relay, competition,race...	event,speech, triggered,words, talk,rallies, raise,appended...	sentence,trigger, charged,penalty, crime,event, prison,guilty...

Table 3: Cluster explanation results using generations from `llama-2-7b-InBedder`. Notice that we simplify some label names for presentation. Clusters are ordered by increasing entropy which is determined by the distribution of labels within each cluster.



(a) instructor-large (b) llama-2-7b-chat (re-enc) (c) llama-2-7b-INBEDDER (fst-gen)

Figure 7: Instruction robustness more results.

Prompt	top-10 first decoding	GPT-4 answer
### Input: The Justice Department filed suit Thursday against the state of Mississippi for failing to end what federal officials call "disturbing" abuse of juveniles and "unconscionable" conditions at two state-run facilities. ### Instruction: What specific language or descriptors does the first sentence use to describe the abuse and conditions at the juvenile facilities? ### Response:	['Dist', 'dist', 'des', 'D', 'Des', 'un', 'Un', 'use', 'uses', 'specific']	"disturbing," "unconscionable"
### Input: "Further testing is still under way, but at this stage, given the early detection, the outlook in such instances would be positive," the specialist said yesterday. ### Instruction: What additional information is provided in the first sentence that is not present in the second sentence? ### Response:	['fur', 'ear', 'testing', 'out', 'stage', 'first', 'F', 'd', 'special', 'information']	Further testing, early detection
### Input: Frank Quattrone, the former Credit Suisse First Boston technology investment-banking guru, reportedly pleaded not guilty Tuesday to charges of obstruction of justice and witness tampering. ### Instruction: Who pleaded not guilty to charges of obstruction of justice and witness tampering? ### Response:	['Fran', 'former', 'Form', 'F', 'Cred', 'f', 'Qu', 'Mr', 'cred', 'ex']	Frank Quattrone

Table 4: Qualitative analysis on the top-10 tokens decoded at the first position. We also present the answers from GPT-4 (webpage) by prompting it to “answer this question within 5 words”.

Original: What is the topic of news?

Correct: Could you summarize the key topic of the article?

Implicit: What's making the headlines in today's paper?

Incorrect: Are there any significant data or statistics mentioned in the article?

Figure 9: An example from our prompt robustness tests.