Revisiting Word Embeddings in the LLM Era

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have recently shown remarkable advancement in various NLP tasks. As such, a popular trend has emerged lately where NLP researchers extract word/sentence/document embeddings from these large decoder-only models and use them for various inference tasks with promising results. However, it is still unclear whether the performance improvement of LLM-induced embeddings is merely because of scale or whether underlying embeddings they produce significantly differ from classical encoding models like Word2Vec, GloVe, Sentence-BERT (SBERT) or Universal Sentence Encoder (USE). This is the central question we investigate in the paper by systematically comparing classical decontextualized and contextualized word embeddings with the same for LLM-induced embeddings. Our results show that LLMs cluster semantically related words more tightly and perform better on analogy tasks in decontextualized settings. However, in contextualized settings, classical models like SimCSE often outperform LLMs in sentence-level similarity assessment tasks, highlighting their continued relevance for fine-grained semantics.

1 Introduction

Word2Vec (Mikolov et al., 2013a) and GLoVe (Pennington et al., 2014), which revolutionized the field of NLP and word embedding techniques by representing words as dense vectors. The complexity and scale of embedding models have since increased dramatically. Transformer-based architecture like BERT-based models (Devlin et al., 2018), RoBERTa (Liu et al., 2019) expanded language representation capabilities by providing context-aware embeddings for words and longer sequences. The most recent paradigm shift came with Large Language Models (LLMs) like GPT (Brown et al., 2020), PaLM (Chowdhery et al., 2022), LLaMA (Touvron et al., 2023), etc. A popular trend has emerged where NLP researchers extract word/sentence/document embeddings from these large decoder-only models for various inference tasks, yielding promising results. However, it remains unclear whether the performance improvement of LLM-induced embeddings is merely due to scale or whether the underlying embeddings they produce significantly differ from classical models.

To explore this, we conducted an in-depth investigation of word embedding similarity in two settings: 1) decontextualized and 2) contextualized for both classical models and LLMs. In the decontextualized setting, we generated embeddings for $\approx 80,000$ words, with curtailed datasets for pretrained Word2Vec ($\approx 50K$) and GloVe ($\approx 60K$) due to vocabulary limitations. We analyzed them using word-pair similarity and word analogy tasks. For the contextualized setting, we selected *anchor words* (verbs, nouns, or adjectives) and created multiple sentences using them to provide context. We then extracted the embeddings of these anchor words for evaluation. More specifically, we examined embedding similarity across nine diverse variational tasks, including *synonym*, *antonym*, *negation*, *jumbling*, *paraphrase*, *questionnaire*, *exclamation*, and *polysemy*. To compare the models in contextualized settings, we performed three distinct similarity analyses: 1. *Anchor Inter-Contextual Variance*: measuring the variance of an anchor word embedding across different contexts; 2) *Anchor Contextual Deviation*: Assessing how context influences anchor word embeddings compared to their decontextualized counterparts; 3) *Sentence Similarity*: Measuring a model's ability to capture linguistic variations at a sentence level.

Our results show that LLMs cluster semantically related words more tightly and perform better on analogy tasks in decontextualized settings. However, in contextualized settings, classical models like SimCSE outperform LLMs in sentence-level tasks, highlighting their continued relevance.

2 Related Work

Text representation is a fundamental pursuit in NLP research, and we have witnessed a remarkable evolution in text representation methodologies over the past decade. This transformation can be grouped into four generations: 1) Classic Decontexualized Word Embeddings like Word2Vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014); 2) Transformer-based contextualized Embeddings like BERT (Devlin et al., 2018), BART (Lewis et al., 2019), and RoBERTa (Liu et al., 2019); 3) Sentence Encoders such as LASER (Artetxe and Schwenk, 2019), Universal Sentence Encoder (USE) (Cer et al., 2018), and Sentence-BERT (SBERT) (Reimers and Gurevych, 2019); and 4) Large Language Model (LLM) induced embeddings like GPT (Brown et al., 2020), PaLM (Chowdhery et al., 2022), LLaMA (Touvron et al., 2023), OpenELM (Mehta et al., 2024), OLMo (Groeneveld et al., 2024) etc.

Previous work by Haber and Poesio (2021); Fournier et al. (2020); Haber and Poesio (2024); Ethayarajh (2019); Mahajan et al. (2023); Sarkar et al. (2022) have investigated how transformer-based models capture word context to varying degrees. In contrast, previous work by Peters et al. (2018); Li and Armstrong (2024); Miaschi and Dell'Orletta (2020) has focused on extracting context-independent word representations for tasks such as word analogy.

Recent LLMs, with their unprecedented scale and capabilities, have demonstrated remarkable success across various NLP tasks (Bubeck et al., 2023; Dai et al., 2022; Du et al., 2022; Smith et al., 2022; Sarkar et al., 2023; Akter et al., 2023). This has motivated multiple NLP researchers to extract word/sentence embeddings from these decoder-only models and use them for other downstream tasks different from text generation (Jiang et al., 2023b; An et al., 2024). Despite these advancements, the fundamental medium of written language has remained constant. While the similarity and relatedness of words have not inherently changed, the models' approach to treating words and their similarities has evolved significantly. This raises important questions about the nature of embeddings generated by LLMs compared to those created by traditional encoding models like Word2Vec or Sentence-BERT. Indeed, little is known about the fundamental nature of these LLM-induced embeddings and how they differ from classical embeddings. It is also unclear how these word embeddings differ from each other in both contextualized and decontextualized settings.

3 Comparing Decontextualized Embeddings: LLM vs. Classical

We conduct a comparative study of two groups of models: 1) Large Language Models (LLMs) (decoder models with over 1B parameters) and 2) "Classical" (models with under 1B parameters) in terms of their decontextualized word embeddings. To be more specific, we selected thirteen models for our analysis, including seven LLMs and six classical models. The LLMs include: LLaMA2-7B and LLaMA3-8B (both dim = 4096) from Meta AI (Touvron et al., 2023), OpenAI's embedding model ADA-002 (dim = 1536), and Google's PaLM2 embedding model Gecko-001 (dim = 768) (Anil et al., 2023), OLMo-8B (dim = 4096) (Groeneveld et al., 2024), OpenELM-3B (dim = 3072) (Mehta et al., 2024) and, Mistral-8B (dim = 4096) (Jiang et al., 2023a). To more clearly see the differences between these models and older ("classical") ones, Meta AI's LASER (dim = 1024) (Artetxe and Schwenk, 2019), Universal Sentence Encoder (USE) (dim = 512) (Cer et al., 2018), SimCSE (dim = 1024) (Gao et al., 2021), SBERT (dim=384) (Reimers and Gurevych, 2019), Word2vec (dim=300) (Mikolov et al., 2013a) and GloVe (dim=300) (Pennington et al., 2014).

Decontextualized embeddings are obtained by inputting single words into each model's tokenizer. For models using single-token inputs, we utilize the final hidden state. For subword tokenization, we average the final hidden states of the tokens. Using these decontextualized embeddings, we conduct the following three comparative analyses.

- *Word-Pair Similarity Comparison*
- *Analogy Task Based Comparison*
- *Similarity Correlation Analysis*

3.1 Word-Pair Similarity Comparison

RQ-1: *How do LLM-induced decontextualized embeddings differ from classical ones in terms of the expected cosine similarity for a randomly chosen pair of words?*

To analyze decontextualized embeddings, we computed the cosine similarity for all ≈ 6.4B word pairs among $80,000$ distinct WordNet (Fellbaum, 1998) words. The raw similarity distributions revealed that many LLMs exhibit higher baseline similarities than classical models (refer appendix for the figure 5).

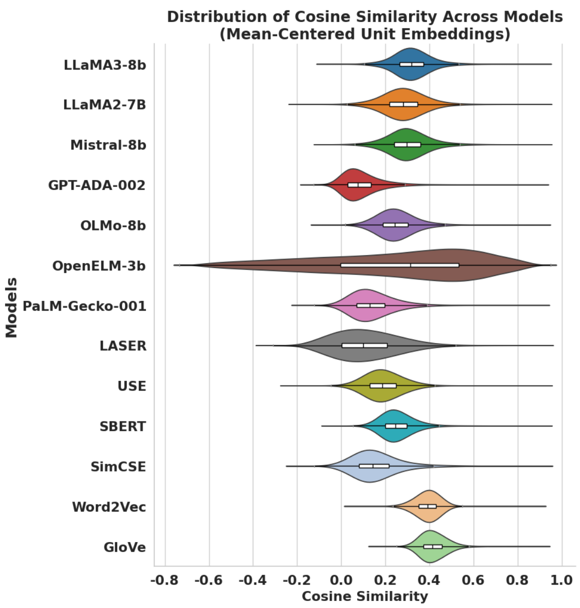


Figure 1: The Mean-Centered Embedding distribution of cosine similarities between all pairs of words.

To determine if this was a global shift or an intrinsic property, we performed mean-centering on unit-normalized embeddings (figure 1), revealing fundamental structural differences. For most LLMs (e.g., LLaMA, Mistral), this adjustment reduced but did not eliminate similarity inflation, suggesting it is an inherent characteristic. In contrast, mean-centering brought the mean similarity for GPT-ADA and PaLM near zero, aligning their distributions with classical models, while unexpectedly increasing inflation for SBERT, Word2Vec, and GloVe. The LLMs whose distributions centered near zero (GPT-ADA and PaLM) also demonstrated stronger performance and alignment with human expectations, indicating that embedding space structure is a key differentiator tied to model performance and interpretability.

***Finding-1**: LLMs show higher baseline similarities than classical embeddings, but only some (like GPT-ADA and PaLM) align with human expectations after mean-centering.*

RQ-2: *Do LLM-based decontextualized embeddings capture similarity better than classical ones?*

We evaluated word-pair similarity on the BATS dataset (Gladkova et al., 2016), categorizing pairs as *Morphologically Related*, *Semantically Related*, or *Uncategorized (random pairs)*. The uncategorized pairs are created using WordNet. Figure 2 shows the distribution of cosine similarities for these categories across 11 embedding models.

Figure 2 shows that Word2vec, GloVe, SBERT, and PaLM exhibit the greatest separation between related pairs (both morphologically and semantically related) and unrelated pairs, which is the desired outcome. Other models, especially LLMs like OpenELM and GPT-ADA struggle to differentiate between categories, finding all more similar. In contrast, classical models performed better at distinguishing morphological categories but did not perform well on semantic categories, as their distributions resembled those of random word pairs.

***Finding-2**: LLMs are not always better than classical models in capturing semantic similarity. PaLM (LLM) and SBERT (Classical) can effectively distinguish semantically related and unrelated pairs, whereas most other models (both LLM-based and Classical) struggle with the same.*

3.2 Analogy Task Based Comparison

RQ-3: *Do LLMs improve the performance of decontextualized word embeddings on analogy tasks?*

To answer this question, we followed the original word analogy task format set out by Mikolov et al. (2013b) and comprehensively evaluated the eleven embedding models on the word pairs from the BATS dataset. For words a, b, c, d, analogy $a : b :: c : d$ and embedding function $f(x)$, it is expected that $f(b) - f(a) + f(c) \approx f(d)$, which we will refer to as the **3CosAdd** method. Other approaches have been introduced for this task, including **Pair Distance** and **3CosMul** (introduced by Levy and Goldberg (2014)). Later, Drozd et al. (2016) introduced new methods called **3CosAvg** and **LRCos**, which achieved excellent performance in their experiments on classical models. For a detailed explanation, refer to appendix (Sec. A.2).

Method	3CosAdd	3CosAvg	3CosMul	LRCos	PairD
GPT-ada	**0.4123**	**0.4465**	**0.4238**	0.3750	**0.2319**
LLaMA2	0.1449	0.2000	0.1454	0.1310	0.0526
LLaMA3	0.0496	0.0590	0.0480	0.0530	0.0018
Mistral	0.0494	0.0620	0.0476	0.0635	0.0025
OLMo	0.0525	0.0645	0.0499	0.0665	0.0018
OpenELM	0.0165	0.0350	0.0141	0.0135	0.0020
PaLM	**0.3981**	**0.4575**	**0.4171**	**0.5340**	**0.1929**
SBERT	0.2431	0.2605	0.2667	**0.4870**	0.0856
SimCSE	0.0248	0.0385	0.0217	0.0315	0.0012
USE	0.1739	0.2120	0.1873	0.4500	0.0251
LASER	0.2271	0.2600	0.2369	0.2840	0.1214
GloVe	0.3481	0.4290	0.3452	0.4875	0.1523
Word2Vec	0.3229	0.3855	0.3096	0.4605	0.1203

Table 1: Performance on BATS Analogy. **Blue** denotes the best accuracy; **black** denotes the second best.

For all methods, the 3 words used as the input for the analogy were excluded from the answers, and top-1 accuracy was measured. For fairness, the same Wordnet corpus from section 3.1 was used for each model, and the arithmetic results for each

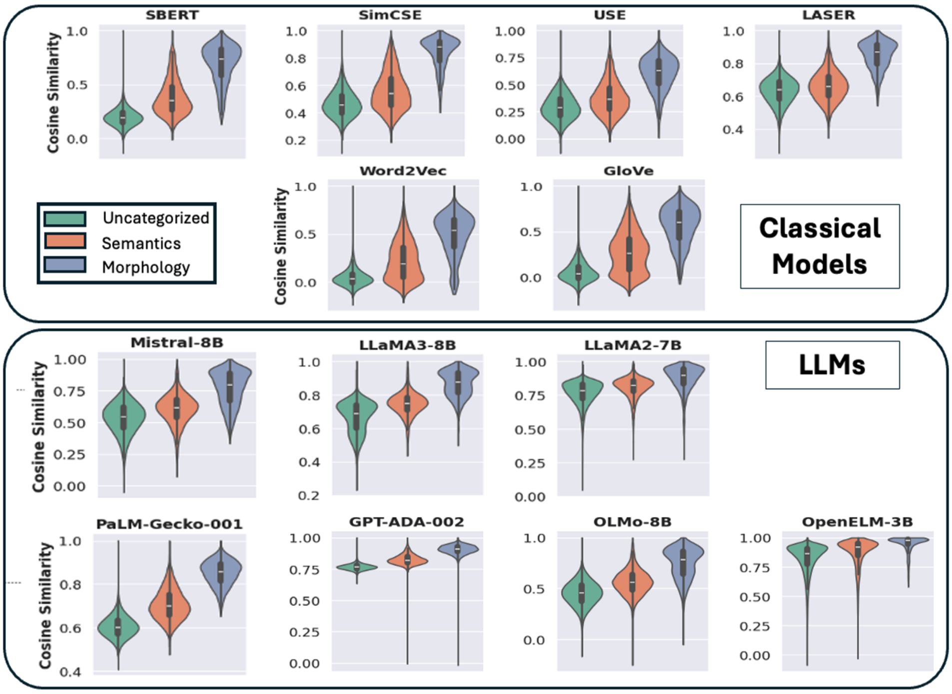


Figure 2: Violin box plot showing the distribution of cosine similarities for random, morphologically related, and semantically related pairs of words for each model.

method were used to find the nearest neighbor in the corpus. These results are shown in Table 1, with the best-performing embedding for each method shown in blue. Both ADA and PaLM performed very well, while OpenELM performed the worst in the LLM category. Among classical embeddings, SBERT and LASER performed quite well, often ranked higher than all open-source LLMs. Full information about each model's accuracy in each category can be found in the appendix (Table 4).

***Finding-3**: ADA and PALM outperform classical models on word analogy tasks. However, SBERT, GloVe, and Word2Vec often rank higher than open-source LLMs, indicating that smaller models can be alternatives resource efficient*

3.3 Similarity Correlation Analysis

RQ-4: *Do LLMs produce very different decontextualized word embeddings than the classical models?*

To further investigate whether LLMs offer something new/very different in terms of decontextualized embeddings, we computed statistical measures of correlation between each pair of models (both LLMs and Classical) in terms of their actual word embeddings. First, the cosine similarities of all pairs of words from the Wordnet corpus (see section 3.1) were computed for each embedding model. The correlation between two different embedding models was computed based on word pair similarities. Figure 3 shows the Spearman's ρ between each pair of embedding models (Kendall's τ correlation is reported in the appendix Figure 7 due to lack of space). Interestingly, these results show that the LLaMA family and Mistral are the most semantically similar, while SimCSE and LLaMA3 are the most different. Also, SimCSE and SBERT showed decent correlations with both ADA/PaLM. To ensure a fair comparison, Word2Vec and GloVe models were excluded due to their significantly different vocabulary sizes.

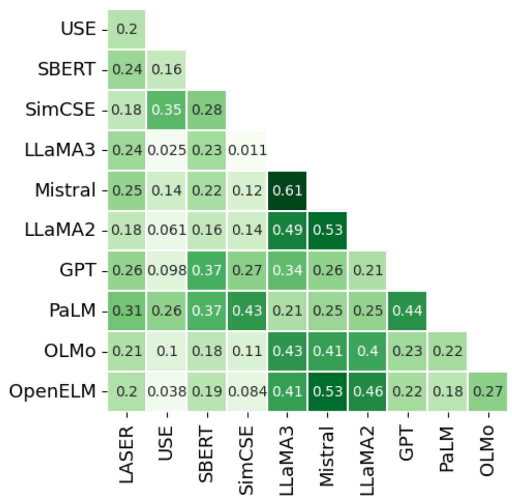


Figure 3: Spearman's ρ for each model pair, calculated from $2.1B$ randomly selected word pairs out of a total of $6.4B$ word pairs from the Wordnet (RQ1) corpus.

In another effort, we investigated how both types of models (LLMs and Classical) agreed/disagreed with each other regarding the similarity ranks of

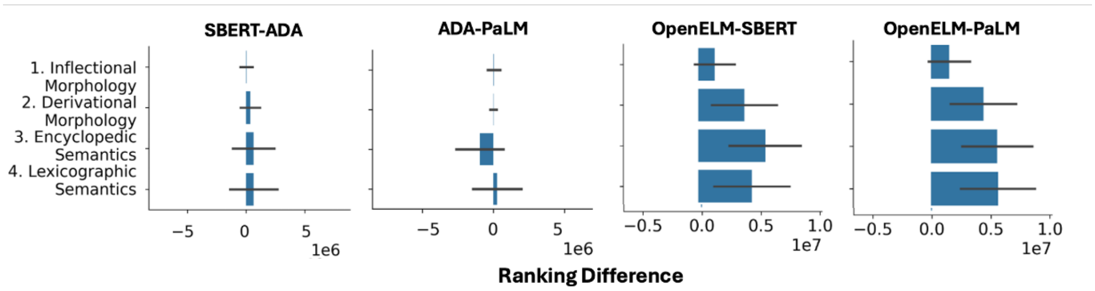


Figure 4: Mean-Variance plot of the difference in Word Pair Similarity Ranks for the BATS corpus. For all other model comparisons refer to appendix figure 6.

specifically related word pairs. More specifically, we computed the average difference of similarity ranks between pairs of words with three types of relations, morphological/semantic/random, for each pair of embedding models, where the rank is determined from the collection of all words in the BATS corpus (section 3.1). For example, if the 5th closest word to "bad" according to ADA-002's embedding was "worst", while "worst" was the 10th nearest word to "bad" according to LLaMA, we would compute a difference of -5 for that word pair while comparing ADA vs. LLaMA. If two models mostly tend to agree on the similarity ranks of word pairs, we would expect an average value of 0 with a small variance.

Figure 4 presents these results for SBERT/ADA and ADA/PALM pairs (Figure 6 shows all pairs in the appendix due to lack of space), revealing that all models—except OpenELM—agree reasonably well on the similarity of words related by morphology. Notably, some model pairs such as PaLM-ADA, LLaMA3-LASER, and SBERT-ADA/PaLM exhibit greater agreement. It is surprising that ADA, PaLM, and SBERT demonstrate the highest levels of agreement despite substantial differences in model size and semantic space, suggesting that SBERT has a semantic space very similar to those of LLMs like ADA and PaLM. In contrast, there were significantly more disagreements among the models for semantic relations.

***Finding-4**: Two LLMs, PaLM and ADA, tended to agree with each other in the decontextualized setting, additionally yielding a high correlation with SBERT, suggesting that SBERT is still an efficient choice when resources are constrained.*

4 Comparing Contextual Embeddings: LLM vs. Classical

In the contextualized setting, we compare LLM vs. Classical word/sentence embeddings across nine different variational tasks. This way allows us to examine how context influences different embedding models across various linguistic scenarios[1]. The variational tasks include:

4.1 Variational Tasks

- **Lexical Variations:**
  - **Synonym Task**: Generate sentence S_1 containing an anchor word. Create S_2 by replacing a word before the anchor word in S_1 with its *synonym*. Compare the anchor word embeddings from both sentences.
  - **Antonym Task**: Similar to the Synonym Task, but replace the word with its *antonym*.
  - **Negation Task**: Generate S_2 by adding a *negation* before the anchor word in S_1. Compare the anchor word embeddings.
- **Tone Variations:** First, Generate S_1 with an anchor word, and then -
  - **Exclamation Task**: Create four *exclamatory* variations of S_1 with the anchor word. Compare the anchor word embeddings.
  - **Question Formation Task**: Create four *interrogative* sentences based on S_1 containing the anchor word. Compare the anchor word embeddings.
  - **Active-Passive Task**: Generate S_1 in *active voice*. Create four *passive voice* versions of S_1, keeping the anchor word. Compare the anchor word embeddings.
- **Semantic Variations:**
  - **Jumbling Task**: Generate S_1 with an anchor word. Create the following sentences by:
    - S_2: Shuffling words before the anchor word.
    - S_3: Shuffling the entire sentence.
    - S_4 and S_5: Exchanging one or two words around the anchor word.

[1] Contextualized embeddings are generated by processing sentences from the nine contextual tasks and extracting the embeddings corresponding to the anchor words. Due to API limitations, closed-source models like GPT-ADA-002 and PaLM2-Gecko were excluded from the contextualized analysis. Similarly, classical models such as USE and LASER, which do not readily provide contextualized word embeddings, were omitted from this part of the study.

Finally, compare the anchor word embeddings.

- **Paraphrasing Task**: Generate S_1 with an anchor word. Create four *paraphrases* of S_1, all containing the anchor word. Compare the anchor word embeddings across these sentences.
- **Polysemy Task**: Generate five sentences using the anchor word in different *senses* (polysemy). Compare the embeddings to assess how models capture multiple meanings.

Due to LLMs' causal attention mechanism, we applied all variations before the anchor word, except for jumbling. Since causal attention computes embeddings based on preceding words, this ensures the perturbations influence the anchor word's embedding. Next, for each variational task, we compute 3 different similarity scores, as follows.

1) Anchor Inter-Contextual Variance: Here, we measure the variance of anchor word embeddings across different contexts. First, we extracted the embedding of each anchor word from all generated sentences. We then designated the embedding from the first sentence as the reference embedding. Subsequently, we computed the cosine angle between this reference embedding and the anchor word embeddings from the remaining sentences. The average of these cosine angles quantifies how differently the model represents the anchor word across various contexts.

2) Anchor Contextual Deviation: Here, we compute the cosine angle between the standalone (decontextualized) anchor word embedding and the anchor word contextual embeddings extracted from each generated sentence. We then averaged these cosine angles to obtain a measure of how much the contextualized representations deviate from the decontextualized ones.

3) Sentence Meaning Variance: Here, we measure how the sentences overall are semantically similar/different by computing the cosine angle between them. The average cosine angle between two sentence embeddings is reported.

4.2 Dataset Generation

To facilitate our contextual analyses, we created a synthetic dataset by randomly sampling $1,200$ anchor words (nouns, verbs, or adjectives) from WordNet. We then used the Claude-sonnet 3.5 model (Anthropic, 2024) to generate sentences for each variational task based on these words, ensuring a diverse and comprehensive set of contextual scenarios. The prompts to generate the dataset are shown in the appendix section B.2.

For lexical variational tasks, we generated only two sentences (one reference and one variational) for each anchor word, as have a very high word overlap between sentences. For the remaining six categories, we created five sentences for each anchor word (refer to Section 4.1). Each set of sentences shared the same anchor word, but in different contexts (see examples in Appendix 6).

To compute cosine angles, we extracted three types of embeddings: 1) Decontextualized anchor word embeddings from each model. 2) Contextualized anchor word embeddings (token-level anchor word embedding from the last hidden layer of each model), and 3) Sentence embeddings (overall embedding for each generated sentence). This multi-faceted approach allows us to compare word representations in both contextualized and decontextualized settings across different models and variational tasks, which not only provides a nuanced understanding of each model's strengths and limitations but can serve as predictive indicators for downstream model performance, offering actionable guidance for efficient and cost-effective model selection and evaluation.

4.3 Research Questions and Findings

RQ-5: *How do LLMs differ from classical embeddings for single lexicon variations?*

To examine how models handle single lexicon variations, we analyze the *Synonym, Antonym, and Negation* variational tasks and compare cosine angle (see Table 2). These tasks modify sentences by replacing a word with its synonym or antonym or by introducing a negation before the anchor word, which affects contextual understanding.

For all variations (*Synonym, Antonym, and Negation*), we expect a high value for *Anchor Contextual Deviation* (i.e., contextual word embeddings should be somewhat different from the corresponding decontextualized ones), and found LLaMA2 excelling in this aspect.

For synonym variations, we expect a low value for *Anchor Inter-contextual Variance* and *Sentence Meaning Variance*, as the overall meanings are typically unaltered. Our experiments aligns with these expectation, with the classical model SimCSE showcasing the lowest cosine angle (low variance) in the inter-contextual setting. For antonyms and negations, we anticipated greater variance due to their opposite meanings. However, as shown in Table 2, none of the models exhibited the expected high variance, likely because high word overlap

Lexical Variations	Synonym Inter. ↓	Synonym Deviation ↑	Synonym Sim. ↓	Antonym Inter. ↑	Antonym Deviation ↑	Antonym Sim. ↑	Negation Inter. ↑	Negation Deviation ↑	Negation Sim. ↑
SBert	10.74	45.69	18.13	18.45	46.64	**27.21**	24.41	47.53	**38.48**
SimCSE	**9.87**	47.77	**9.39**	22.33	49.07	21.00	29.12	50.92	**26.75**
LLaMA3	15.26	**69.41**	12.40	22.89	67.79	17.04	**30.42**	**69.74**	21.84
LLaMA2	15.21	**81.99**	12.94	22.23	**78.15**	16.76	28.93	**80.58**	22.80
Mistral	15.14	60.13	11.15	**22.95**	59.40	14.87	28.77	59.55	19.70
OLMo	16.51	58.62	13.78	**25.10**	57.37	18.66	**31.51**	57.26	24.50
OpenELM	**10.33**	68.23	**8.58**	16.89	**67.90**	9.88	20.18	68.41	13.01

Tone Variations	Exclamatory Inter. ↓	Exclamatory Deviation ↑	Exclamatory Sim. ↓	Questionnaire Inter. ↓	Questionnaire Deviation ↑	Questionnaire Sim. ↓	Active-Passive Inter. ↓	Active-Passive Deviation ↑	Active-Passive Sim. ↓
SBert	**23.81**	44.89	38.61	**21.28**	45.05	33.01	**20.24**	44.76	25.12
SimCSE	**24.52**	47.64	**27.00**	**21.75**	47.19	**21.74**	**18.13**	47.53	**15.51**
LLaMA3	38.66	64.76	30.34	39.53	63.22	30.45	43.65	65.80	27.09
LLaMA2	39.60	**71.21**	30.78	38.87	**69.50**	29.46	45.82	**73.90**	27.94
Mistral	35.80	55.68	27.71	36.65	56.21	26.74	41.00	57.29	24.01
OLMo	42.85	54.74	34.54	44.01	54.96	33.06	46.85	56.60	31.10
OpenELM	27.54	**67.04**	**19.50**	27.99	**67.10**	**17.39**	29.65	**66.74**	**15.53**

Semantic Variations	Polysemy Inter. ↑	Polysemy Deviation ↑	Polysemy Sim. ↑	Paraphrase Inter. ↓	Paraphrase Deviation ↑	Paraphrase Sim. ↓	Jumbling Inter. ↑	Jumbling Deviation ↑	Jumbling Sim. ↑
SBert	46.33	56.38	**75.49**	**26.05**	45.11	42.59	17.41	51.40	19.45
SimCSE	54.62	58.59	**57.81**	**24.99**	47.98	26.16	17.56	51.63	15.03
LLaMA3	55.64	**78.97**	52.49	39.22	65.20	27.17	**52.86**	**73.19**	**38.32**
LLaMA2	**59.51**	**88.18**	48.16	40.61	**73.40**	26.44	**56.75**	**73.38**	**51.89**
Mistral	58.60	71.09	41.95	37.48	57.18	**25.88**	42.68	63.39	27.91
OLMo	**63.58**	67.90	55.07	43.18	55.83	31.16	47.95	60.04	34.68
OpenELM	51.13	71.12	28.77	28.41	**67.13**	**20.05**	29.65	66.74	15.53

Table 2: Comparison of different models across various tasks in the Contextualized Evaluation setting. The values represented are the **Average Cosine Angle**. Arrows (↑↓) indicate expected behavior: ↑ suggests a lower cosine angle is desirable, and ↓ is the opposite. The lower the angle, the higher the cosine similarity. Here, **'Inter.'** represents **Anchor Inter-Contextual Variance**, **'Deviation'** represents **Anchor Contextual Deviation**, **'Sim'** stands for **Sentence Meaning Variance**, The **best** and **2nd** best scores in each category are highlighted in respective colors.

between sentence pairs led models to overlook the single-word differences phenomenon also reported in (Mahajan et al., 2024; Zhang et al., 2023). Also, when comparing antonym to synonym tasks, all models showed increased angles, indicating some sensitivity to opposite meanings. SimCSE, in particular, had the highest percentage change in angle ($\sim$ 50%), reflecting strong antonym differentiation, while OpenELM showed a smaller change ($\sim$ 15%), suggesting it may struggle more with antonym variations. For negation tasks, the addition of negation words led to higher angles, indicating a degree of sensitivity to negation, though the extent varied by model.

Finding-5: *In single lexicon variations, LLaMA2 led in Anchor Contextual Deviation. For Antonym and Negation tasks, OLMo had superior Inter-Contextual Variance, and SBERT excelled in Sentence Meaning Variance.*

RQ-6: *How do LLMs differ from classical embeddings for linguistic tone variations?*

We examine the *Exclamatory, Questionnaire, and Active-Passive* variational tasks, each involving five sentences per anchor word. The first sentence is the reference generated using the anchor word, while the remaining four are tailored to each category, sharing the anchor word in common. For these tasks, wider angles are desired for *Anchor Contextual Deviation*, but, lower angles for *Anchor Inter-Contextual Variance* and *Sentence Meaning Variance* (similar to the synonym task) as these are just tonal variations of the reference.

Finding 6: *For tone variations, classical models (SimCSE, SBERT) achieve desired low inter-contextual variance for anchor words. Among LLMs, OpenELM shows low sentence meaning variance, while LLaMA models (especially LLaMA2) excel at anchor contextual deviation.*

RQ-7: *How do LLMs differ from classical embeddings for overall semantic variations?*

We computed the cosine angles across all three fronts (*Inter-Contextual Variance*, *Anchor Contextual Deviation*, and *Sentence Meaning Variance*) for the 3 variational tasks: *Jumbling*, *Paraphrasing*, and *Polysemy*. In all these tasks, wider angles are desired for all 3 measures across all 3 tasks, with the only expectation that lower angles are desired for *Anchor Inter-Contextual Variance* and *Sentence Meaning Variance* in the case of *Paraphrasing* task.

Consistent with previous findings, LLaMA2 achieved the highest Anchor Contextual Deviation

Model	Sbert	SimCSE	LlaMA2-7b	LlaMA3-8b	Mistral-7b	OLMo-7b	OpenELM-3B
ARC-e	-	-	84.0	**92.4**	**90.8**	65.4	59.89
BoolQ	-	-	86.1	**87.5**	**89.3**	74.4	67.4
MMLU	-	-	46.2	**66.6**	**64.0**	40.5	26.76
PIQA	-	-	57.8	77.2	**80.6**	**78.4**	78.24
Clustering	42.35	29.04	45.24	**46.45**	**54.93**	32.0	18.71
Reranking	**58.04**	46.47	57.38	**59.68**	50.15	33.91	37.0
STS	78.9	74.33	**83.73**	83.58	**84.77**	27.04	38.31
Summarization	30.81	**31.15**	28.49	30.94	**36.32**	20.83	18.71

Table 3: Model Evaluation Across Various downstream tasks. The extended table can found in appendix table 5

for all tasks, as seen in Table 2. In fact, LLaMA2 performed the best across all three variance measures for the Jumbling task, suggesting its superior capability in capturing word order. All models demonstrate somewhat high Inter-Contextual Variance for polysemous word context (a desired behavior), with OLMo performing particularly well, suggesting it is adept at detecting polysemy. Finally, results were mixed for the paraphrasing task.

Finding-7*: LLaMA2 is best for word order (Jumbling task). For Polysemy, classical models lead in sentence-level similarity, while LLMs like OLMo are better at token-level disambiguation, revealing a trade-off. Paraphrasing results were mixed.*

5 Discussions and Final Words

In this paper, we compared word/sentence embeddings from 7 LLMs and 6 classical models (total 13) in both contextualized and decontextualized settings. In the decontextualized setting, we used WordNet and the BATS dataset to create a corpus of 80,000 unique words and 6.4 billion word pairs. Our results show that LLM-based models PaLM and ADA performed the best on word analogy tasks, surprisingly aligning with SBERT, suggesting SBERT as a resource-efficient alternative. Mean-centering allowed models like GPT-ADA and PaLM to produce similarity distributions closer to human expectations, yet other LLMs still showed higher baseline similarities than classical models.

In the contextualized setting, we assessed 5 LLMs and 2 classical models across three variance measures: *Anchor Inter-Contextual Variance, Anchor Contextual Deviation, and Sentence Meaning Variance* across 9 variational tasks. We found that LLMs (especially LLaMA2) excel in *Anchor Contextual Deviation* across all contexts, demonstrating superior contextualized token-level analysis. Conversely, classical models (SimCSE and SBERT) outperformed many LLMs in terms of *Sentence Meaning Variance* for lexicon variation and *Polysemy tasks*, underscoring their continued relevance. Interestingly, OLMo achieved superior *Anchor Inter-Contextual Variance* in Antonym, Negation, and Polysemy tasks, demonstrating its superiority in properly contextualizing word embeddings in flipped-meaning scenarios.

5.1 Implications and Future Use

- **Accuracy-Interpretability Dilemma**: Our analysis quantifies model interpretability by measuring alignment with human expectations. For instance, in Antonym and Negation tasks, models like OLMo and SBERT exhibit high variance, correctly capturing the semantic shift and thus appearing more "interpretable." However, this desirable behavior doesn't always correlate with top performance on all benchmarks. We hypothesize this dilemma stems from model training: LLMs, optimized for generation on vast datasets, can over-generalize, leading to the inflated similarity scores we observed in our decontextualized analysis. This tendency harms fine-grained interpretability, creating a trade-off where a model might be accurate on broad similarity tasks but fail to make intuitive distinctions in practice. This suggests that relying solely on leaderboard scores can be misleading, and future work should aim to develop evaluation suites that reward both accuracy and interpretable, human-aligned reasoning.
- **Guidance for Model Selection:** Our findings offer actionable guidance for practitioners. The superiority of classical models like SimCSE in certain contextual tasks can likely be attributed to their task-specific contrastive training, which contrasts with the broader generative objectives of LLMs. This distinction is crucial for model selection. More broadly, our criteria can predict success on complex downstream tasks (see Table 3). The balanced performance of Mistral and LLaMA3 suggests that evaluating fundamental properties like inter-contextual variation—a direct result of a model's training paradigm—is an efficient way to predict its suitability for advanced applications. While these models are promising, further large-scale studies are needed.

6 Limitations

Despite providing a comprehensive comparison between classical embedding models and Large Language Models (LLMs) in both decontextualized and contextualized settings, our study has several limitations. First, due to computational constraints and API restrictions, we were unable to include some closed-source models and larger LLMs in the contextualized embedding analysis, which may limit the generalizability of our findings across all state-of-the-art models. Second, our evaluation focuses solely on the English language and uses synthetic sentences generated by the Claude-Sonnet model, which may not capture the full diversity and complexity of natural language in real-world contexts. Third, while we explored a range of linguistic tasks, this represents only a subset of the wide spectrum of linguistic evaluations that could be incorporated into future extensions of this framework.

Moreover, numerous works (Linzen, 2016; Fournier et al., 2020) have highlighted issues with using the standard analogy task to determine if semantic information is encoded in word embeddings. Therefore, we have refrained from making claims that one embedding is inherently "better" than another. Additionally, our reliance on cosine similarity as the primary metric assumes it adequately reflects semantic similarity between embedding vectors. While it is a popular choice in NLP literature, cosine similarity has inherent limitations, and our findings are constrained by this methodological assumption.

References

Mousumi Akter, Souvika Sarkar, and Shubhra Kanti Karmaker Santu. 2023. On evaluation of bangla word analogies. *CoRR*, abs/2304.04613.

Na Min An, Sania Waheed, and James Thorne. 2024. Capturing the relationship between sentence triplets for llm and human-generated texts to enhance sentence embeddings. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 624–638.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Anthropic. 2024. Claude3.5-sonnet.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Tom Brown, 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.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2022. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3519–3530, Osaka, Japan. The COLING 2016 Organizing Committee.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Louis Fournier, Emmanuel Dupoux, and Ewan Dunbar. 2020. Analogies minus analogy test: measuring regularities in word embeddings.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15, San Diego, California. Association for Computational Linguistics.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. 2024. Olmes: A standard for language model evaluations. *arXiv preprint arXiv:2406.08446*.

Janosch Haber and Massimo Poesio. 2021. Patterns of lexical ambiguity in contextualised language models. *arXiv preprint arXiv:2109.13032*.

Janosch Haber and Massimo Poesio. 2024. Polysemy—evidence from linguistics, behavioral science, and contextualized language models. *Computational Linguistics*, 50(1):351–417.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2023b. Scaling sentence embeddings with large language models. *arXiv preprint arXiv:2307.16645*.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Jiangtian Li and Blair C Armstrong. 2024. Probing the representational structure of regular polysemy via sense analogy questions: Insights from contextual word vectors. *Cognitive Science*, 48(3):e13416.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yash Mahajan, Naman Bansal, Eduardo Blanco, and Santu Karmaker. 2024. ALIGN-SIM: A task-free test bed for evaluating and interpreting sentence embeddings through semantic similarity alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7393–7428, Miami, Florida, USA. Association for Computational Linguistics.

Yash Mahajan, Naman Bansal, and Shubhra Kanti Karmaker. 2023. The daunting dilemma with sentence encoders: Success on standard benchmarks, failure in capturing basic semantic properties. *CoRR*, abs/2309.03747.

Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, et al. 2024. Openelm: An efficient language model family with open-source training and inference framework. *arXiv preprint arXiv:2404.14619*.

Alessio Miaschi and Felice Dell'Orletta. 2020. Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 110–119.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

MTEB. Mteb-leaderboard.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint arXiv:1808.08949*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Souvika Sarkar, Dongji Feng, and Shubhra Kanti Karmaker Santu. 2022. Exploring universal sentence encoders for zero-shot text classification. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing, AACL/IJCNLP 2022 - Volume 2: Short Papers, Online only, November 20-23, 2022*, pages 135–147. Association for Computational Linguistics.

Souvika Sarkar, Dongji Feng, and Shubhra Kanti Karmaker Santu. 2023. Zero-shot multi-label topic inference with sentence encoders and llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 16218–16233. Association for Computational Linguistics.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990.*

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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Hongyu Zhang, Harish Madabushi, and Massimo Poesio. 2023. Testing paraphrase models on recognising sentence pairs at different degrees of semantic overlap. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*.

A Decontextualized Evaluation Setting

A.1 Word-Pair Similarity Comparison

To analyze decontextualized embeddings, we computed the cosine similarity for all $\approx$ 6.4B word pairs among $80,000$ distinct WordNet (Fellbaum, 1998) words. The raw similarity distributions (refer figure 5) revealed clear differences in latent semantic spacing between model types. Classical static embeddings such as Word2Vec and GloVe, as well as transformer-based models like SBERT and USE, exhibited left-skewed similarity distributions, indicating lower similarity scores for random word pairs (see figure 5. In contrast, LLMs such as OpenELM and the LLaMA family showed higher overall similarity scores, resulting in right-skewed distributions. Due to vocabulary size constraints, Word2Vec and GloVe covered only about 50,000 and 60,000 words, respectively, so comparisons for these models were performed on smaller subsets

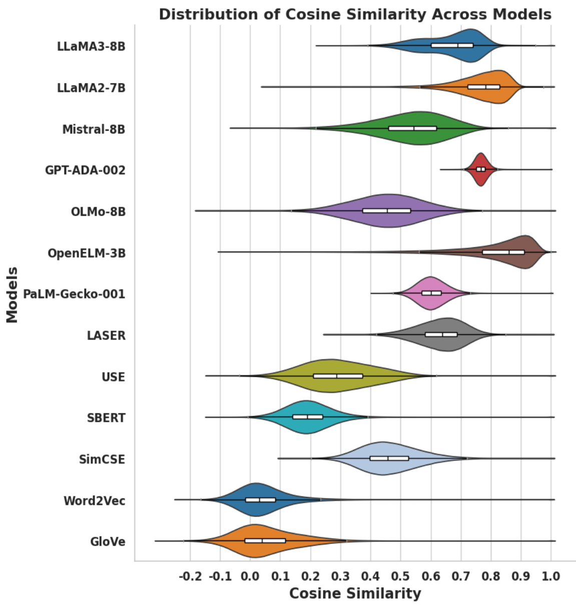


Figure 5: The distribution of cosine similarities between all pairs of words for each model.

A.2 Analogy-Task Based Comparison

Here we presented the exhaustive list of model accuracy on various evaluation methods of the Analogy task. See Table 4 for a more granular description of the performance of each model on specific categories of BATS. Here is the description of the Metric we used to evaluate the analogy task:

1. **3CosAdd**:
   The analogy $a : b :: c : d$ is solved by computing $f(b) - f(a) + f(c) \approx f(d)$. For example, in the analogy "king:man::queen:woman", the equation becomes $f(\text{man}) - f(\text{king}) + f(\text{queen}) \approx f(\text{woman})$.

2. **3CosAvg**:
   This extends 3CosAdd by averaging the trans-

formations over multiple analogy pairs. For "king:man::queen:woman", we take the average of multiple such pairs to improve accuracy:

$$f(d) \approx \text{avg}(f(b) - f(a) + f(c)).$$

3. **3CosMul**:
Similar to 3CosAdd but instead of adding, it multiplies cosine similarities:

$$\text{argmax}_d \frac{\cos(f(b), f(d)) \cdot \cos(f(c), f(d))}{\cos(f(a), f(d)) + \epsilon}.$$

4. **LRCos**:
A method using logistic regression to classify whether the analogy holds, using distances between embeddings.

5. **PairDistance**:
Measures the cosine distance between two pairs of words (a, b) and (c, d) to check how similar their relationship is. For "king:queen", the cosine distance is compared with "man:woman".

6. **SimilarToAny**:
Checks if d is similar to any of the words in the analogy (a, b, c). For "king:man::queen:?", it checks if $f(d)$ is similar to any of "king", "man", or "queen".

7. **SimilarToB**:
Checks if d is most similar to b in the analogy. For "king:man::queen:?", the method finds the word most similar to "man".

Below Table 6 showcase the extensive comparison of all the models on analogy task using various evaluation metrics.

The following sections in the appendix are organized as follows: Section A.2.1 presents the ranking comparison of models on the Word Analogy Task. Section A.3 provides Kendall's τ and Spearman's ρ correlations for model pairs on the word similarity task. Section B.1 gives examples of generated sentences for anchor words in contextualized evaluation. Section B.2 describes the prompting design for generating samples, and Section C presents the cosine similarity distribution across all evaluation metrics.

Model	Analogy Method	1. Inflectional Morphology	2. Derivational Morphology	3. Encyclopedic Semantics	4. Lexicographic Semantics
GPT3-Ada	3CosAdd	0.761	0.677	0.115	0.097
	3CosAvg	0.802	0.734	0.148	0.102
	3CosMul	0.776	0.697	0.122	0.100
	LRCos	0.606	0.482	0.280	0.132
	PairDistance	0.546	0.323	0.052	0.006
	SimilarToAny	0.155	0.044	0.005	0.029
	SimilarToB	0.276	0.134	0.038	0.090
LLaMA2	3CosAdd	0.230	0.271	0.055	0.023
	3CosAvg	0.326	0.362	0.086	0.026
	3CosMul	0.230	0.276	0.053	0.022
	LRCos	0.150	0.148	0.176	0.050
	PairDistance	0.066	0.130	0.013	0.001
	SimilarToAny	0.065	0.043	0.037	0.011
	SimilarToB	0.130	0.118	0.054	0.026
LLaMA3	3CosAdd	0.079	0.099	0.011	0.009
	3CosAvg	0.096	0.114	0.016	0.010
	3CosMul	0.076	0.097	0.010	0.009
	LRCos	0.044	0.058	0.104	0.006
	PairDistance	0.001	0.004	0.000	0.002
	SimilarToAny	0.053	0.059	0.010	0.008
	SimilarToB	0.100	0.112	0.018	0.016
Mistral	3CosAdd	0.084	0.093	0.010	0.010
	3CosAvg	0.102	0.116	0.018	0.012
	3CosMul	0.082	0.090	0.010	0.009
	LRCos	0.066	0.068	0.110	0.010
	PairDistance	0.001	0.006	0.000	0.003
	SimilarToAny	0.062	0.063	0.008	0.009
	SimilarToB	0.108	0.112	0.014	0.012
OLMo	3CosAdd	0.094	0.093	0.014	0.009
	3CosAvg	0.116	0.106	0.022	0.014
	3CosMul	0.090	0.089	0.012	0.009
	LRCos	0.074	0.078	0.100	0.014
	PairDistance	0.001	0.004	0.000	0.002
	SimilarToAny	0.065	0.057	0.012	0.008
	SimilarToB	0.116	0.108	0.022	0.016
OpenELM	3CosAdd	0.030	0.031	0.003	0.004
	3CosAvg	0.070	0.052	0.010	0.008
	3CosMul	0.025	0.027	0.002	0.003
	LRCos	0.002	0.002	0.046	0.004
	PairDistance	0.003	0.003	0.000	0.002
	SimilarToAny	0.040	0.035	0.007	0.005
	SimilarToB	0.066	0.054	0.012	0.008
PaLM	3CosAdd	0.743	0.609	0.118	0.122
	3CosAvg	0.794	0.668	0.232	0.136
	3CosMul	0.768	0.648	0.128	0.124
	LRCos	0.780	0.714	0.404	0.238
	PairDistance	0.466	0.249	0.048	0.008
	SimilarToAny	0.165	0.027	0.011	0.035
	SimilarToB	0.270	0.082	0.030	0.108

Model	Analogy Method	1. Inflectional Morphology	2. Derivational Morphology	3. Encyclopedic Semantics	4. Lexicographic Semantics
SBERT	3CosAdd	0.461	0.393	0.046	0.073
	3CosAvg	0.474	0.418	0.058	0.092
	3CosMul	0.506	0.424	0.062	0.074
	LRCos	0.808	0.642	0.270	0.228
	PairDistance	0.135	0.184	0.021	0.003
	SimilarToAny	0.178	0.065	0.003	0.019
	SimilarToB	0.302	0.154	0.020	0.088
SimCSE	3CosAdd	0.040	0.045	0.008	0.007
	3CosAvg	0.058	0.068	0.016	0.012
	3CosMul	0.035	0.039	0.007	0.006
	LRCos	0.024	0.026	0.070	0.006
	PairDistance	0.001	0.002	0.001	0.002
	SimilarToAny	0.036	0.037	0.010	0.007
	SimilarToB	0.056	0.068	0.014	0.012
USE	3CosAdd	0.397	0.156	0.039	0.103
	3CosAvg	0.442	0.190	0.084	0.132
	3CosMul	0.436	0.165	0.049	0.100
	LRCos	0.722	0.412	0.396	0.270
	PairDistance	0.076	0.012	0.008	0.005
	SimilarToAny	0.101	0.032	0.006	0.035
	SimilarToB	0.204	0.098	0.026	0.098
LASER	3CosAdd	0.431	0.434	0.022	0.022
	3CosAvg	0.484	0.506	0.030	0.020
	3CosMul	0.448	0.454	0.023	0.023
	LRCos	0.510	0.482	0.116	0.028
	PairDistance	0.230	0.245	0.009	0.003
	SimilarToAny	0.087	0.027	0.004	0.007
	SimilarToB	0.198	0.072	0.012	0.020
GloVe	3CosAdd	0.720	0.351	0.262	0.060
	3CosAvg	0.764	0.446	0.430	0.076
	3CosMul	0.770	0.366	0.228	0.017
	LRCos	0.880	0.544	0.440	0.086
	PairDistance	0.395	0.089	0.122	0.003
	SimilarToAny	0.233	0.059	0.089	0.051
	SimilarToB	0.324	0.124	0.132	0.062
Word2Vec	3CosAdd	0.775	0.319	0.137	0.062
	3CosAvg	0.828	0.376	0.266	0.072
	3CosMul	0.804	0.329	0.092	0.014
	LRCos	0.932	0.600	0.224	0.086
	PairDistance	0.355	0.054	0.070	0.003
	SimilarToAny	0.254	0.094	0.074	0.052
	SimilarToB	0.394	0.196	0.068	0.066

Table 4: BATS performance across categories with methods.

A.2.1 Word Analogy Task Ranking

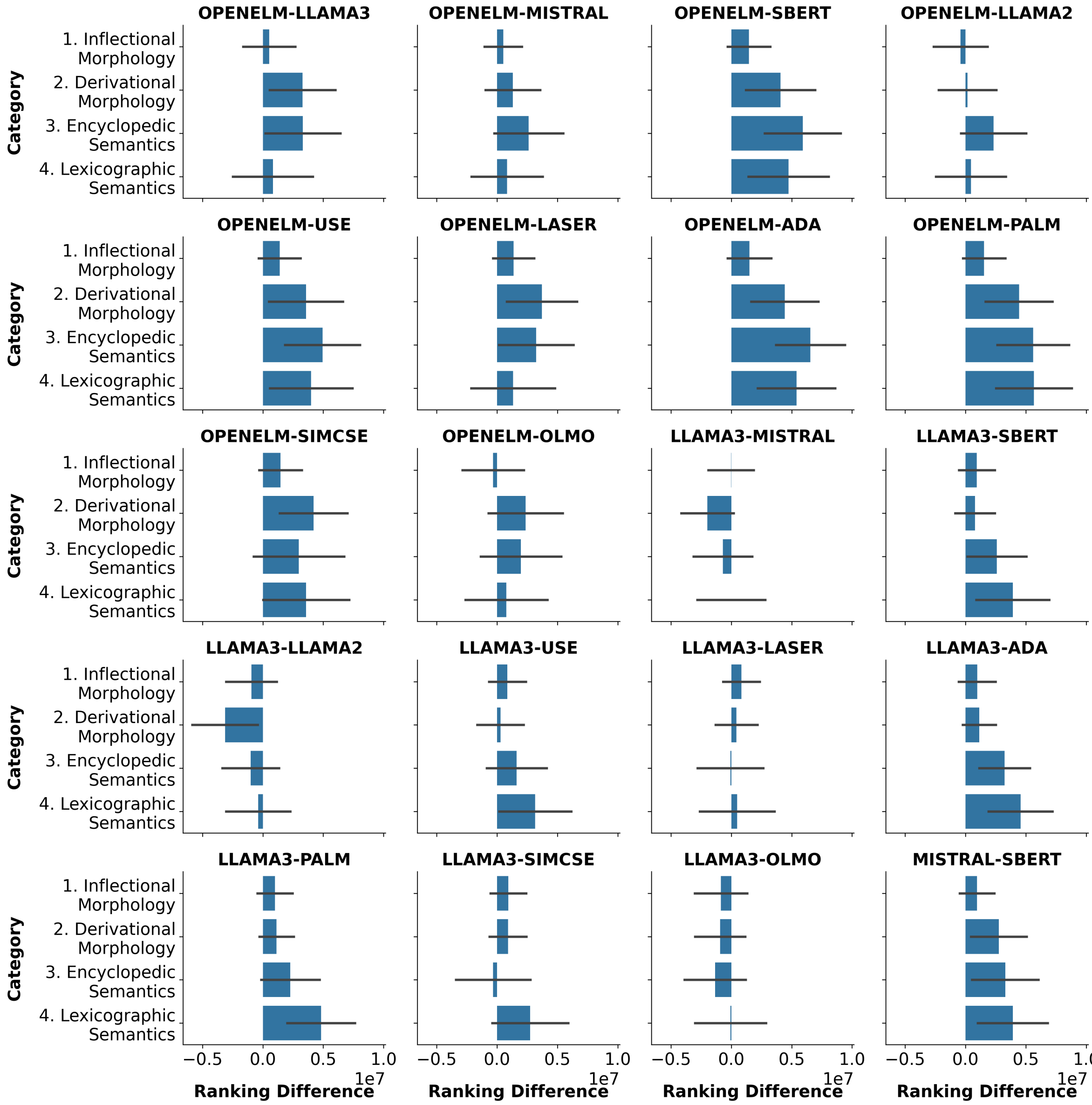


Figure 6: For each model, the cosine similarity of related words was found and ranked according to all pairs of words. Here, the difference in ranking between model pairs for certain BATS categories is shown.(*Continued*)

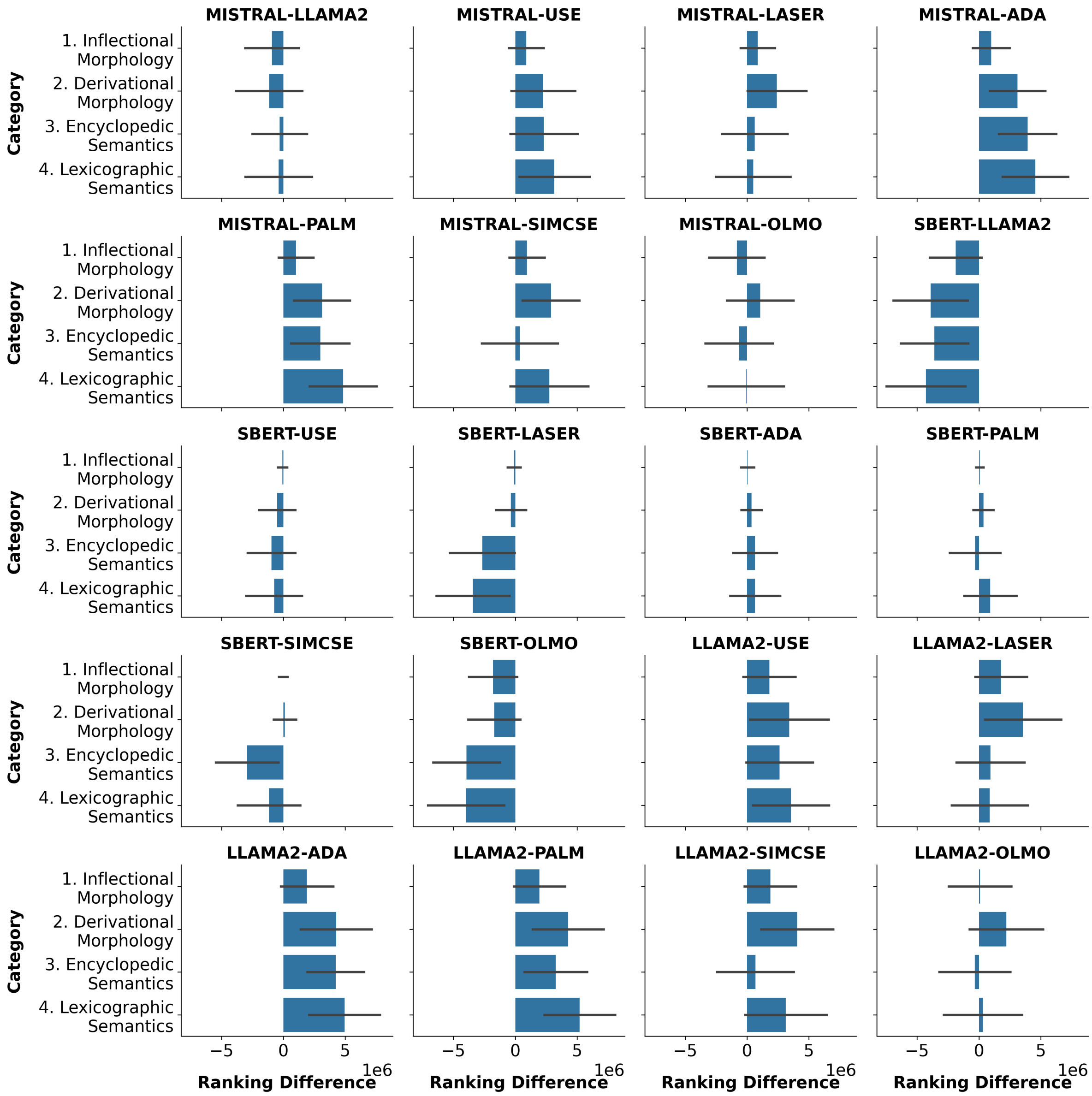


Figure 6: For each model, the cosine similarity of related words was found and ranked according to all pairs of words. Here, the difference in ranking between model pairs for certain BATS categories is shown. (*Continued*)

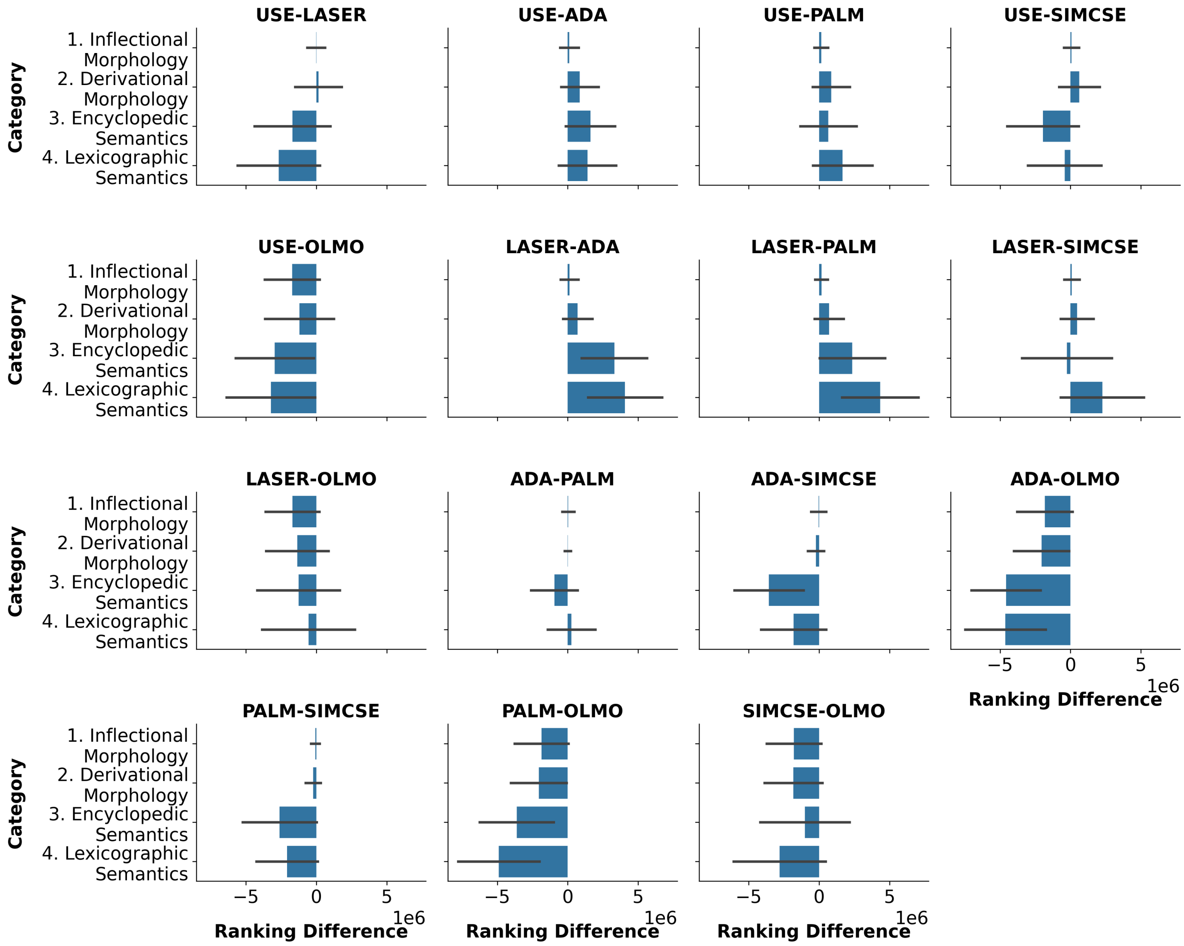
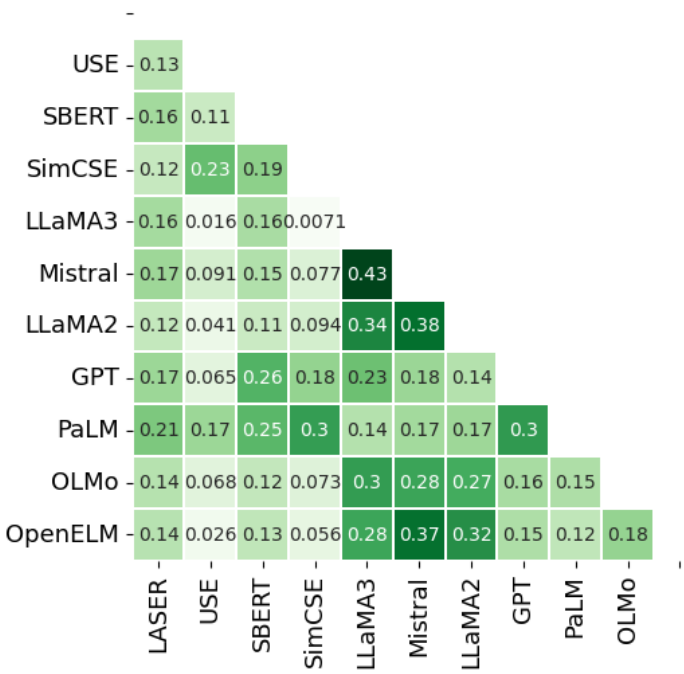
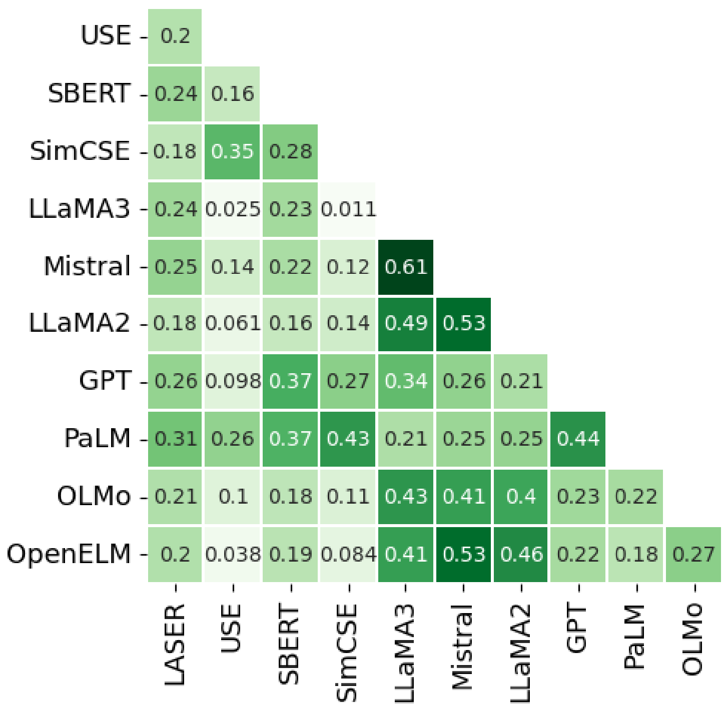


Figure 6: For each model, the cosine similarity of related words was found and ranked according to all pairs of words. Here, the difference in ranking between model pairs for certain BATS categories is shown.

A.3 Similarity Correlation Analysis



(a) Kendal τ



(b) Spearman ρ

Figure 7: Correlation coefficients for each pair of models, found using a large dataset of pairs of words.

B Contextualized Evaluation Setting

To contextualize our findings, Table 5 presents a holistic evaluation of the models on a wide variety of benchmarks that assess both embedding quality and general reasoning capabilities. The embedding quality were reported on MTEB Leaderboard (Muennighoff et al., 2022; MTEB) and reasoning benchmark were reported by (Gu et al., 2024). We also ran our own evaluations for OLMo and OpenELM on the MTEB dataset where performances were not readily available. This broad benchmark assessment is crucial, as it validates that our proposed criteria can serve as predictive indicators for a model's success on complex downstream tasks. For instance, the balanced performance of models like Mistral and LLaMA3 on our criteria reflects their strong, adaptable results on the advanced reasoning and summarization benchmarks shown in the table. This connection underscores that fundamental characteristics, such as the balance between inter-contextual variation and deviation, are not just theoretical but are indicative of a model's practical suitability for sophisticated applications.

Model	Sbert	SimCSE	LlaMA2-7b	LlaMA3-8b	Mistral-7b	OLMo-7b	OpenELM-3B
ARC-c	-	-	54.2	**79.3**	**78.6**	48.5	35.58
ARC-e	-	-	84.0	**92.4**	**90.8**	65.4	59.89
BoolQ	-	-	86.1	**87.5**	**89.3**	74.4	67.4
HellaSwag	-	-	78.9	**81.8**	**83.0**	76.4	72.44
MMLU	-	-	46.2	**66.6**	**64.0**	40.5	26.76
PIQA	-	-	57.8	77.2	**80.6**	**78.4**	78.24
SIQA	-	-	77.5	81.6	**82.8**	78.5	**92.7**
WinoGrande	-	-	71.7	**76.2**	**77.9**	67.9	65.51
Clustering	42.35	29.04	45.24	**46.45**	**54.93**	32.0	18.71
Pair classification	82.37	70.33	**88.03**	87.8	**88.59**	49.32	56.71
Reranking	**58.04**	46.47	57.38	**59.68**	50.15	33.91	37.0
STS	78.9	74.33	**83.73**	83.58	**84.77**	27.04	38.31
Summarization	30.81	**31.15**	28.49	30.94	**36.32**	20.83	18.71

Table 5: Model Evaluation Results Across Various Tasks. **Blue** is top scorer and **black** is second best.

B.1 Synthetic Data Generation Samples

Task		Examples
Synonym	**Anchor Word**:	**adored**
	Word Replaced:	*deeply*
	Word Replaced with:	*profoundly*
	Sentence-1:	The actress was *deeply* adored by her fans for her talent and humility.
	Sentence-2:	The actress was *profoundly* adored by her fans for her talent and humility.
Antonym	**Anchor Word**:	**adored**
	Word Replaced:	*cherished*
	Word Replaced with:	*despised*
	Sentence-1:	The brilliant sunset over the ocean was a sight everyone on the beach deeply *cherished* and **adored**.
	Sentence-2:	The brilliant sunset over the ocean was a sight everyone on the beach deeply *despised* and **adored**.
Negation	**Anchor Word**:	**adored**
	Negation Added:	***not*** **adored**
	Sentence-1:	The famous musician was **adored** by millions of fans worldwide.
	Sentence-2:	The famous musician was ***not*** **adored** by millions of fans worldwide.
Jumbling	**Anchor Word**:	**adored**
	Sentence-1:	The famous actor was **adored** by millions of fans worldwide for his charismatic performances on the silver screen.
	Sentence-2:	*was the famous actor* **adored** by millions of fans worldwide for his charismatic performances on the silver screen.
	Sentence-3:	*on millions performances for the was silver screen. his* ***adored*** *charismatic actor of the by fans famous worldwide*
	Sentence-4:	the famous worldwide was **adored** by millions of fans actor for his charismatic performances on the silver screen.
	Sentence-5:	the the charismatic was **adored** by millions of fans worldwide for his actor performances on famous silver screen.
Active-Passive	**Anchor Word**:	**adored**
	Sentence-1:	The talented musician was **adored** by fans for her soulful performances.
	Sentence-2:	Fans **adored** the talented musician for her soulful performances.
	Sentence-3:	Soulful performances were what fans **adored** about the talented musician.
	Sentence-4:	The musician's soulful performances made her **adored** by countless fans.
	Sentence-5:	The talented musician was enthusiastically **adored** by fans for delivering soulful performances.

Table 6: Task Examples (Part 1) (*Continued*)

Task		Examples
Paraphrase	**Anchor Word**:	**adored**
	Sentence-1:	The famous actor was **adored** by millions of fans worldwide for his charismatic performances on the silver screen.
	Sentence-2:	Legions of admirers cherished the renowned celebrity, who was **adored** for his magnetic screen presence and captivating portrayals.
	Sentence-3:	The iconic star was **adored** by countless devotees for his spellbinding acting prowess and mesmerizing big screen appearances.
	Sentence-4:	Multitudes of enthusiasts revered the legendary performer, whose **adored** on-screen personas and enthralling acting talents left an indelible mark.
	Sentence-5:	The revered thespian was **adored** by a global fanbase for his captivating performances and charismatic screen presence that enthralled audiences worldwide.
Questionnaire	**Anchor Word**:	**adored**
	Sentence-1:	The famous celebrity was **adored** by millions of fans worldwide.
	Sentence-2:	Was the famous celebrity **adored** by millions of fans worldwide?
	Sentence-3:	Did the famous celebrity was **adored** by millions of fans across the globe?
	Sentence-4:	Were there millions of fans worldwide who **adored** the famous celebrity?
	Sentence-5:	Has the famous celebrity been **adored** by a vast number of fans globally?
Exclamation	**Anchor Word**:	**adored**
	Sentence-1:	The **adored** celebrity was swarmed by fans seeking autographs and selfies.
	Sentence-2:	How **adored** the celebrity was by the fans who swarmed them for autographs and selfies!
	Sentence-3:	What an **adored** celebrity, to be swarmed by so many fans seeking autographs and selfies!
	Sentence-4:	How the fans **adored** the celebrity, swarming them for autographs and selfies!
	Sentence-5:	**adored** beyond measure, the celebrity found themselves swarmed by fans - what a scene of autographs and selfies!
Polysemic	**Anchor Word**:	**address**
	Sentence-1:	The CEO delivered an inspiring **address** to the company employees during the annual meeting.
	Sentence-2:	Could you please provide me with your current residential **address** for our records?
	Sentence-3:	The computer program accessed the memory **address** to retrieve the data.
	Sentence-4:	The speaker began her **address** by thanking the audience for attending.
	Sentence-5:	Please **address** the envelope carefully to ensure it reaches the correct destination.

Table 6: Task Examples (Part 2)

B.2 Synthetic Data Generation Prompts

B.2.1 Questionnaire

Questionnaire Task Generation Prompt:
'System Prompt':
Using the anchor word, create a sentence S1 that includes the anchor word. After generating S1, generate four more questionnaire sentences of S1. It's crucial that all sentences retain the anchor word in its original form in all sentences.

Here is an example. For a given anchor word 'forum', the generated S1 and S2 sentences are:
{
'sentence1': "The online forum provides a platform for experts to discuss emerging technologies.",
'sentence2': "Does the online forum provide a platform for experts to discuss emerging technologies?",
'anchor_word': 'forum'
}

The output should be in the following json format:
{'sentence1: S1,
'sentence2': S2,
'sentence3: S3,
'sentence4': S4,
'sentence5': S5,
'anchor_word': anchor_word
}

User: Here is the anchor word: word. Note that, The anchor word must appear unchanged in all sentences.

B.2.2 Active-Passive

Active-Passive Task Generation Prompt:
'System Prompt':
Using the anchor word, create an active voice sentence S1 that includes the anchor word. After generating S1, generate four passive voice sentences of S1. It's crucial that all sentences retain the anchor word in its original form in all the sentences.

Here is an example, for a given anchor word 'forum', the generated S1 and S2 sentences are:
{ 'sentence1': "Experts frequently share their knowledge in the online forum about emerging technologies.",
'sentence2': "Knowledge about emerging technologies is frequently shared by experts in the online forum.",
'anchor_word': 'forum' }

The output should be in the following json format:
{'sentence1: S1,
'sentence2': S2,
'sentence3: S3,
'sentence4': S4,
'sentence5': S5,
'anchor_word': anchor_word
}

User: Here is the anchor word: word. Note that, The anchor word must appear unchanged in all the sentences.

B.3 Polysemy

Polysemous Pair Generation Prompting:
'System Prompt':
Using the anchor word, generate five sentences that are polysemous. Note that, the anchor word should appear in all the sentences but with different meanings. Ensure that the polysemous anchor word is positioned either in the middle or near the end of each sentence.

Here is the example:
{ 'sentence1': "The ancient Roman forum was a bustling center of public life and political debate.",
'sentence2': "The online forum became a heated battleground for discussing the latest tech trends.",
'anchor_word': 'forum' }

The output should be in the following json format:
{'sentence1: S1,
'sentence2': S2,
'sentence3: S3,
'sentence4': S4,
'sentence5': S5,
'anchor_word': anchor_word }

User: Here is the anchor word: word.

B.3.1 Paraphrase

Paraphrase Task Generation Prompt:
'System Prompt':
Using the anchor word, create a sentence S1 that includes the anchor word. After generating S1, create four paraphrased sentences of sentence S1. All four sentences should convey the same overall meaning as S1. It's crucial that all the sentences retain the anchor word in its original form.

For a given anchor word 'forum', the generated S1 and S2 sentences are:
{'sentence1': "The online forum provided a platform for experts to share their knowledge and engage in lively discussions about emerging technologies.",
'sentence2': "A digital meeting place, the forum enabled specialists to disseminate their expertise and participate in animated conversations regarding cutting-edge innovations.",
'anchor_word': 'forum'}

The output should be in the following json format:
{'sentence1: S1,
'sentence2': S2,
'sentence3: S3,
'sentence4': S4,
'sentence5': S5,
'anchor_word': anchor_word
}

User: Here is the anchor word: word.

B.3.2 Jumbling

Jumbling Task Data Generation:
To create the Jumbling Task dataset, we used sentence 1 from the polysemous task dataset as the reference sentence for the Jumbling task. Next, using the reference sentence S_1, we generated four unique sentences by shuffling the reference sentence in four different ways:

1. S_2: We first identified the location of the anchor word and then shuffled all the words present before the anchor word.
2. S_3: We completely shuffled the entire sentence.
3. S_4 and S_5: We identified the anchor word and then exchanged one or two words around the anchor word, respectively.

B.3.3 Synonym

Synonym Pair Generation Prompting:
'System Prompt':
Using the anchor word, generate a sentence S1 of at least 15 words with the anchor word placed near the end. Next, keeping the anchor word unchanged in S2, generate a sentence S2 with the same meaning as S1 by replacing one word (other than the anchor word) with its synonym, ensuring that all word replacements occur before the anchor word in S2.

"Note: Keep the anchor word unchanged in both sentences S1 and S2." Here is an example:
For a given anchor word 'forum', the generated S1 and S2 sentences are:
{ 'sentence1': "Several of the questions asked by the audience in the fast-paced forum were new to the candidates.",
'sentence2': "Numerous of the questions asked by the audience in the fast-paced forum were new to the candidates.",
'word_replaced': 'Several',
'word_replaced_with': 'Numerous',
'anchor_word': 'forum' }

The output should be in the following json format:
{'sentence1: S1,
'sentence2': S2,
'word_replaced': word,
'word_replaced_with': new_word,
'anchor_word': anchor_word }

User: Follow the instructions and replace a word other than the anchor word. Here is the anchor word:{***word***}. Make sure both sentences S1 and S2 have the anchor word in it."

B.3.4 Negation

Negation Pair Generation Prompting:
'System Prompt':
Using the anchor word, generate a sentence S1 with the anchor word in it. Next, generate a sentence S2 with an opposite meaning to S1 by adding a negation word before the anchor word in S2. Make sure the negation word is appropriate to the context of the sentence. Also, ensure that S1 and S2 should have the same words except for the negation word in S2.
Note: Do not modify or change the anchor word in both sentences.

Here is an example: For a given anchor word 'forum', the generated S1 and S2 sentences are:
{'sentence1': "The talented artist was adored by fans for her captivating performances.",
'sentence2': "The talented artist was not adored by fans due to her underwhelming performances.",
'anchor_word': 'adored',
'negation_added': 'not adored' }

The output should be in the following json format:
{'sentence1: S1,
'sentence2': S2,
'anchor_word': anchor_word
'negation_added': negation_word }

User: Here is the anchor word: word.

B.3.5 Antonym

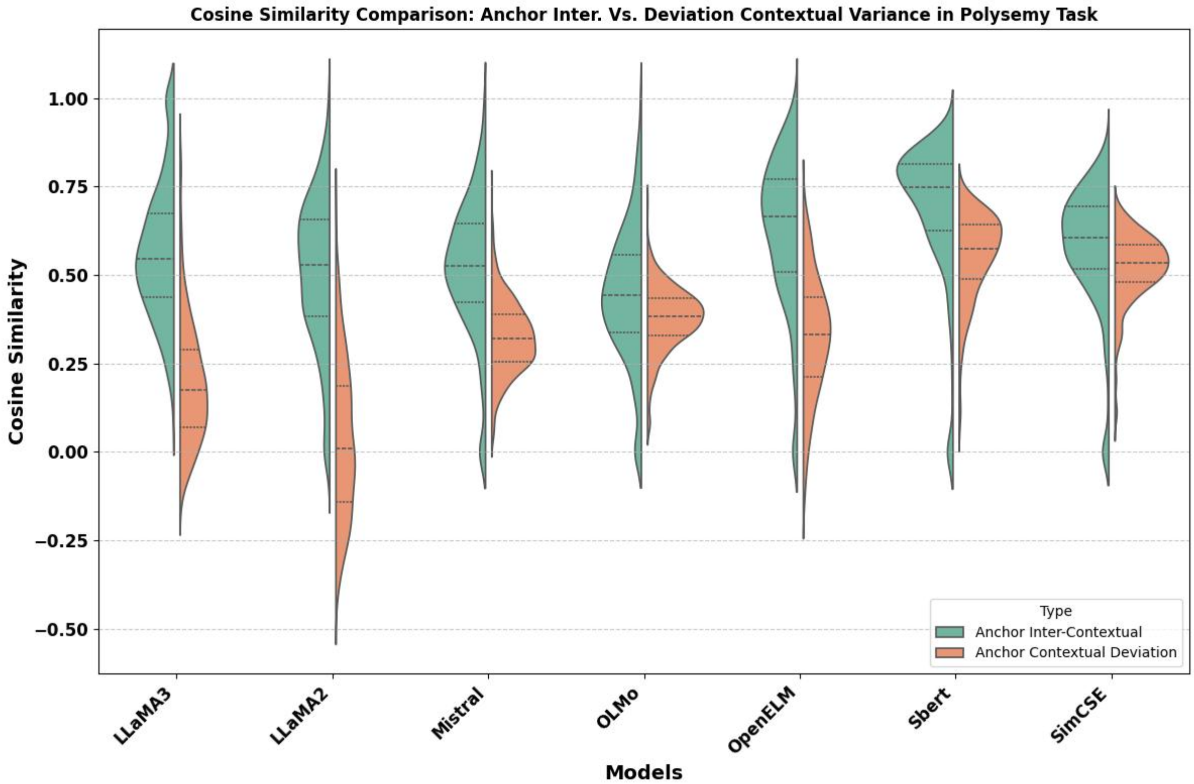
Antonym Pair Generation Prompting:
'System Prompt':
Using the anchor word, generate a sentence S1 of at least 15 words with the anchor word placed near the end. Next, keeping the anchor word unchanged in S2, generate a sentence S2 with an opposite meaning to S1 by replacing one word (other than the anchor word) with its antonym, ensuring that all word replacements occur before the anchor word in S2.

Note: Do not modify or change the anchor word in both sentences.
Here is an example: For a given anchor word 'forum', the generated S1 and S2 sentences are:
{ 'sentence1': "Several of the questions asked by the audience in the fast-paced forum were new to the candidates.",
'sentence2': "Few of the questions asked by the audience in the fast-paced forum were new to the candidates.",
'word_replaced': 'Several',
'word_replaced_with': 'Few' }

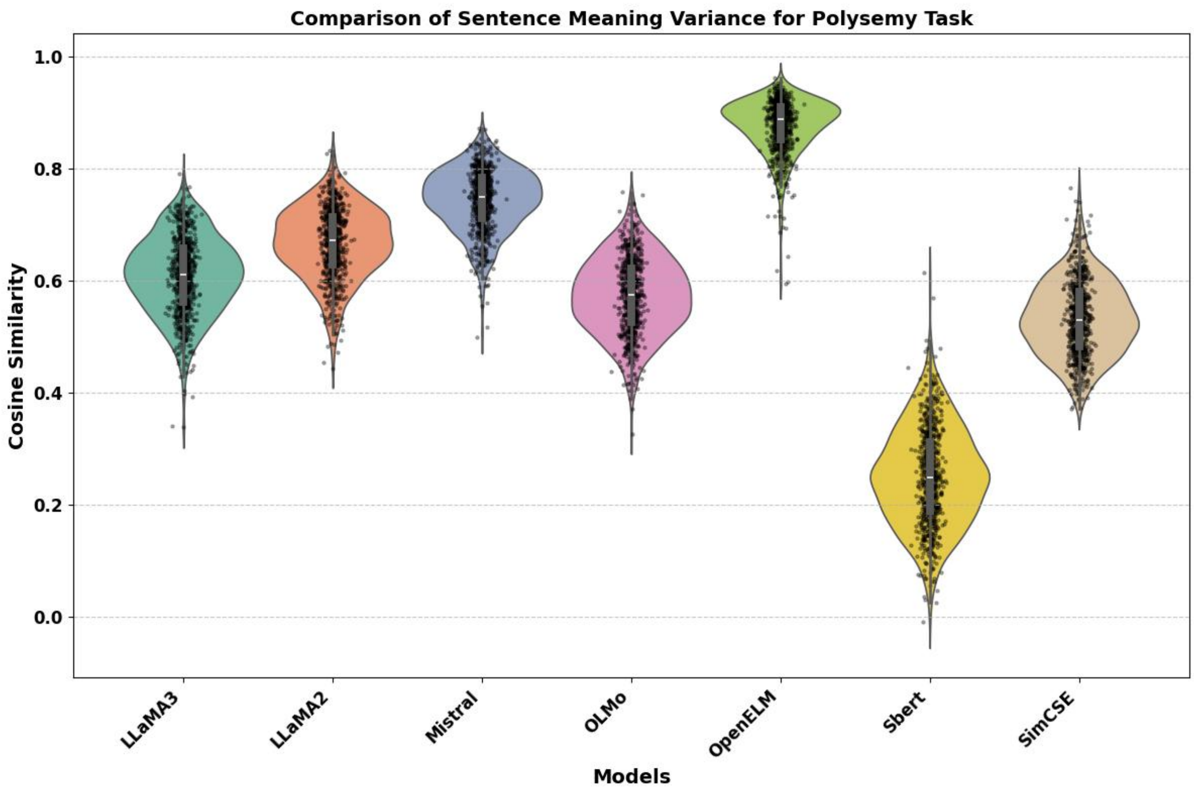
The output should be in the following json format:
{'sentence1: S1,
'sentence2': S2,
'anchor_word': anchor_word
'word_replaced': word, 'word_replaced_with': new_word }

User: Here is the anchor word: word.

C Comparison of Models in Contextualized Settings

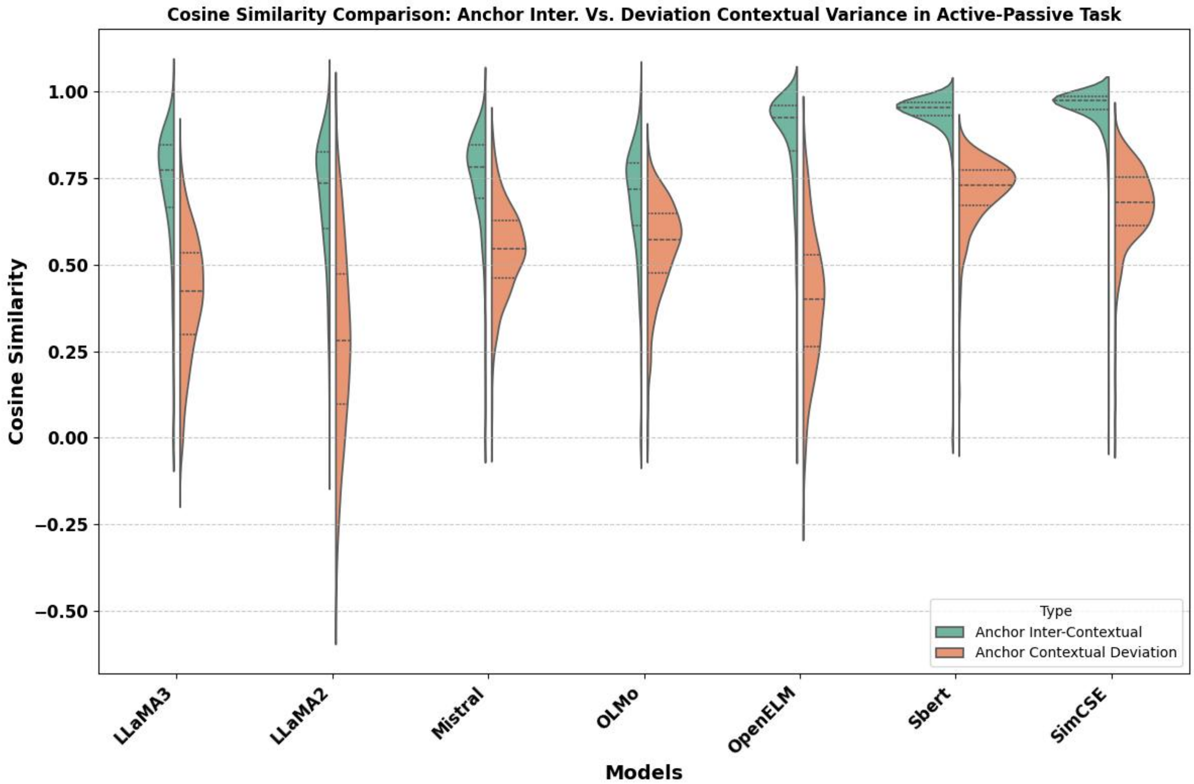


(a) The distribution of cosine similarities between Anchor Inter-Contextual Variance and Anchor Contextual Deviation words.

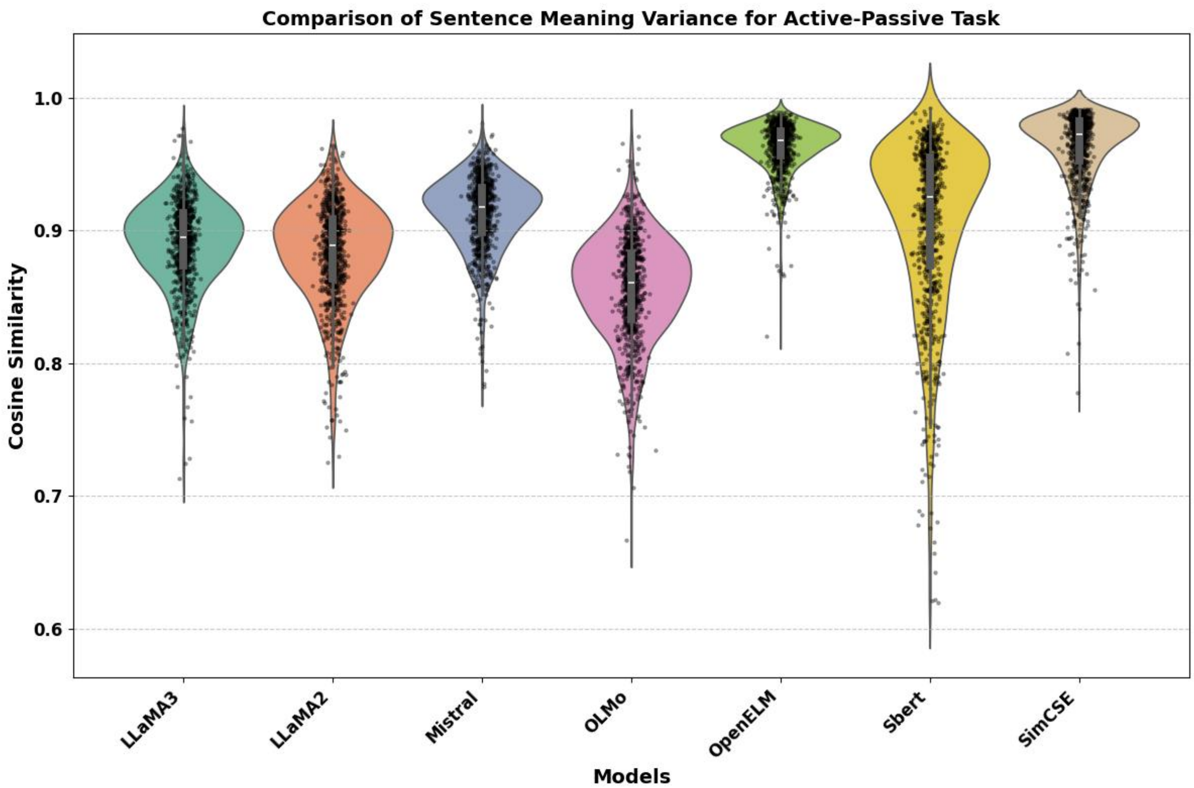


(b) The distribution of cosine similarities between sentences in Sentence Meaning Variance.

Figure 8: Polysemy Task comparison

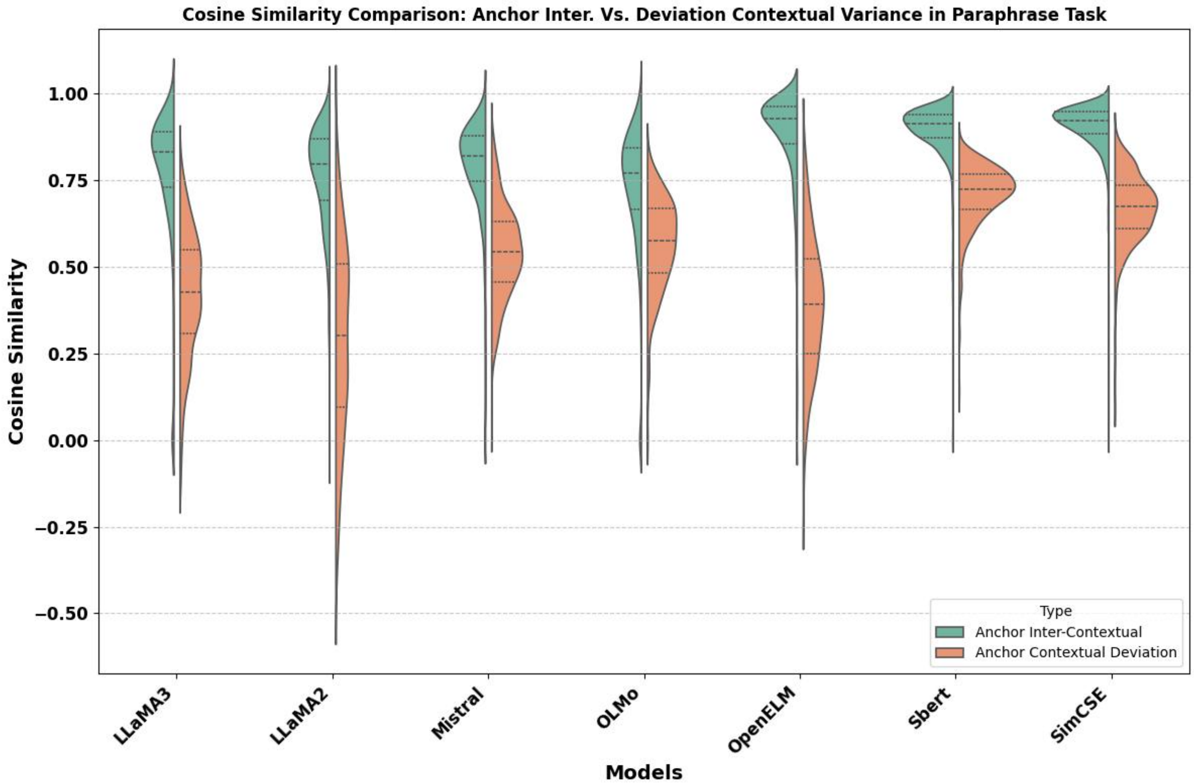


(a) The distribution of cosine similarities between Anchor Inter-Contextual Variance and Anchor Contextual Deviation words.

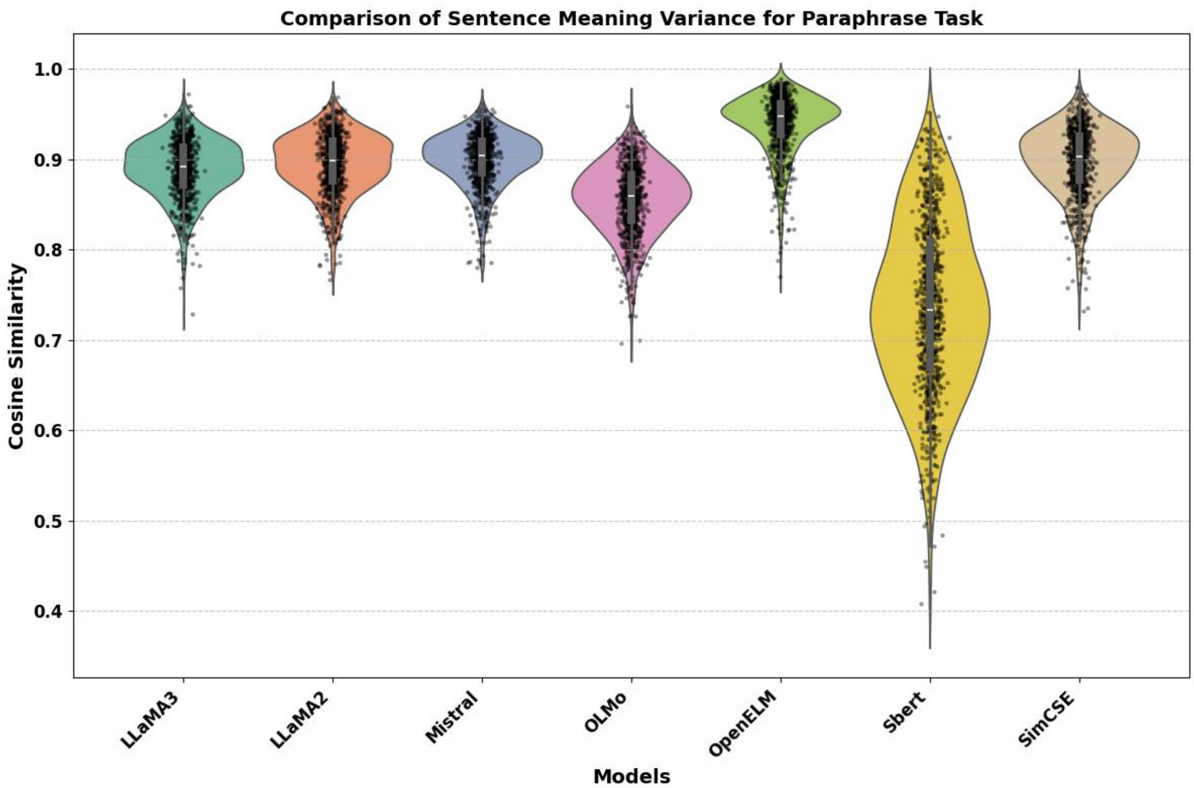


(b) The distribution of cosine similarities between sentences in Sentence Meaning Variance.

Figure 9: Active-Passive Task comparison

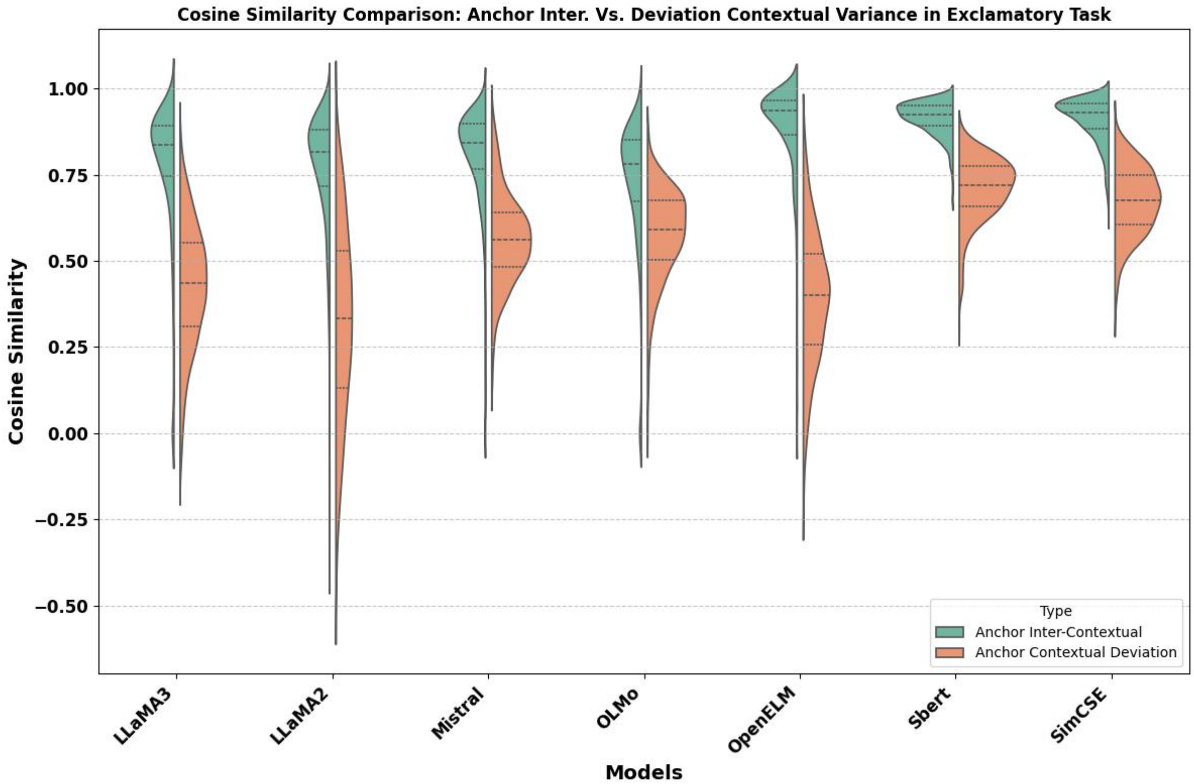


(a) The distribution of cosine similarities between Anchor Inter-Contextual Variance and Anchor Contextual Deviation words.

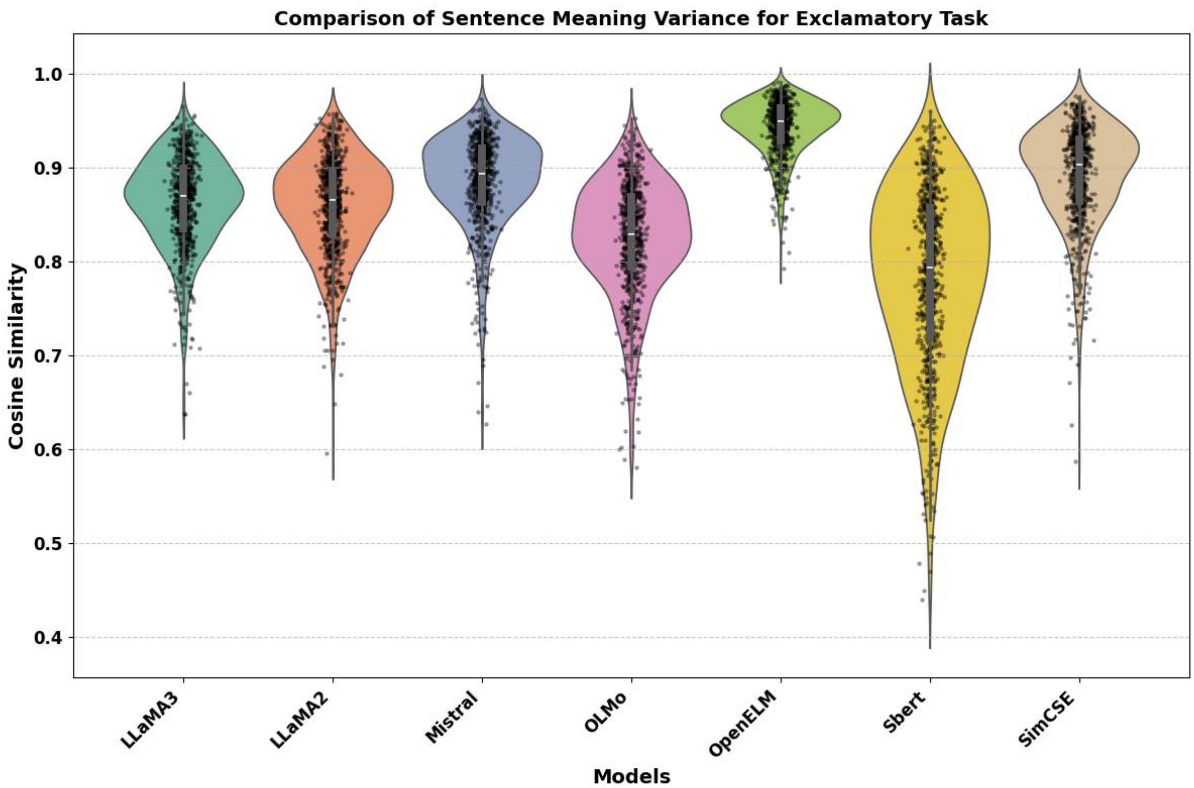


(b) The distribution of cosine similarities between sentences in Sentence Meaning Variance.

Figure 10: Paraphrase Task comparison

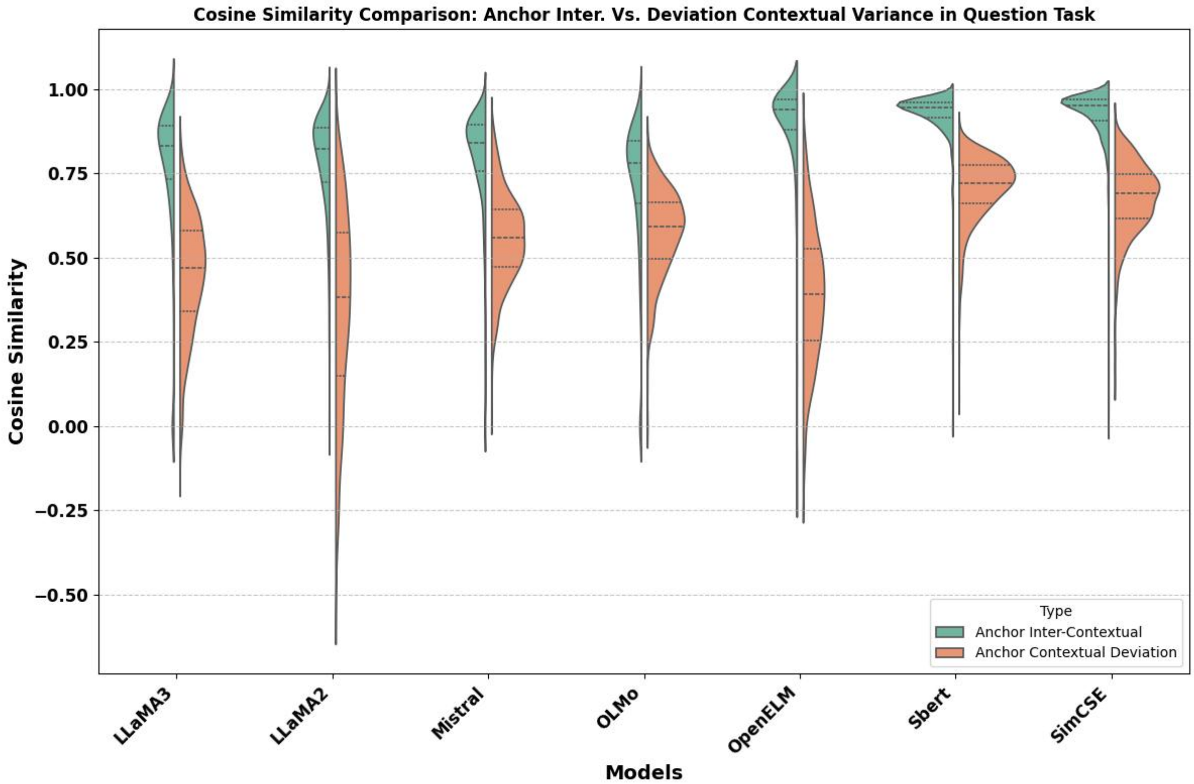


(a) The distribution of cosine similarities between Anchor Inter-Contextual Variance and Anchor Contextual Deviation words.

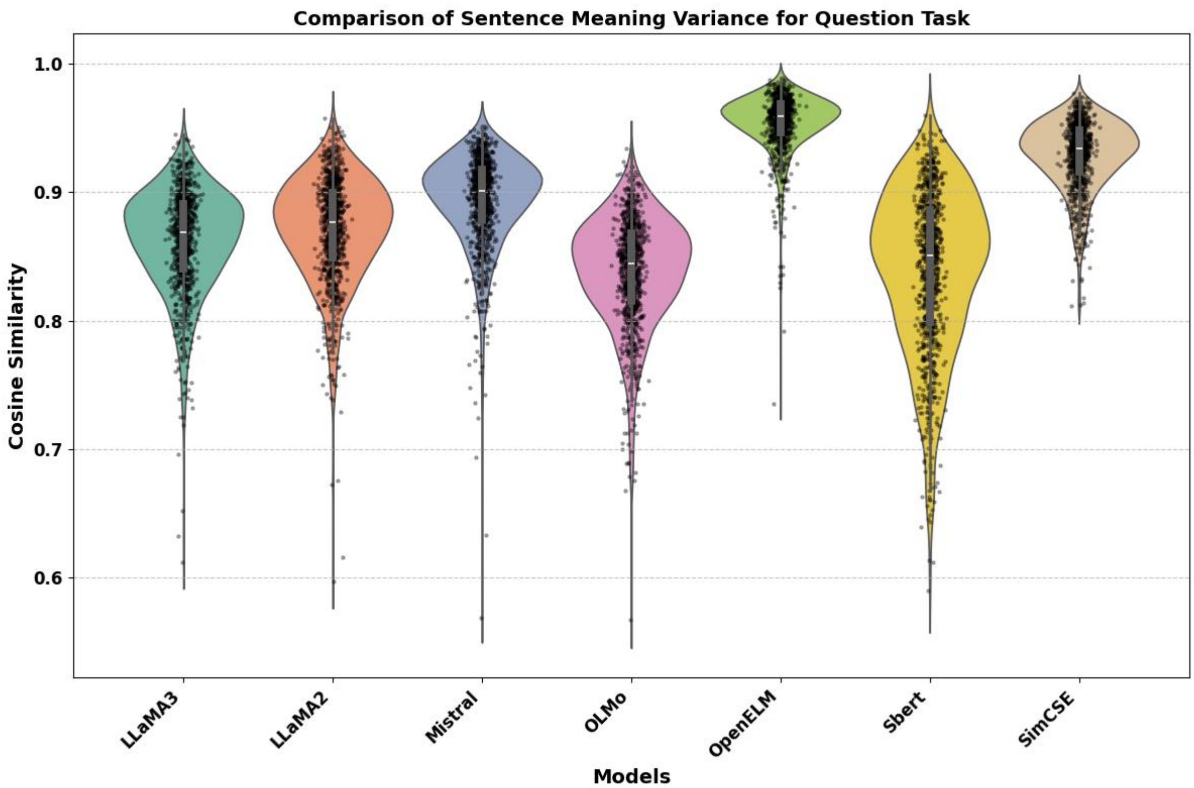


(b) The distribution of cosine similarities between sentences in Sentence Meaning Variance.

Figure 11: Exclamatory Task comparison

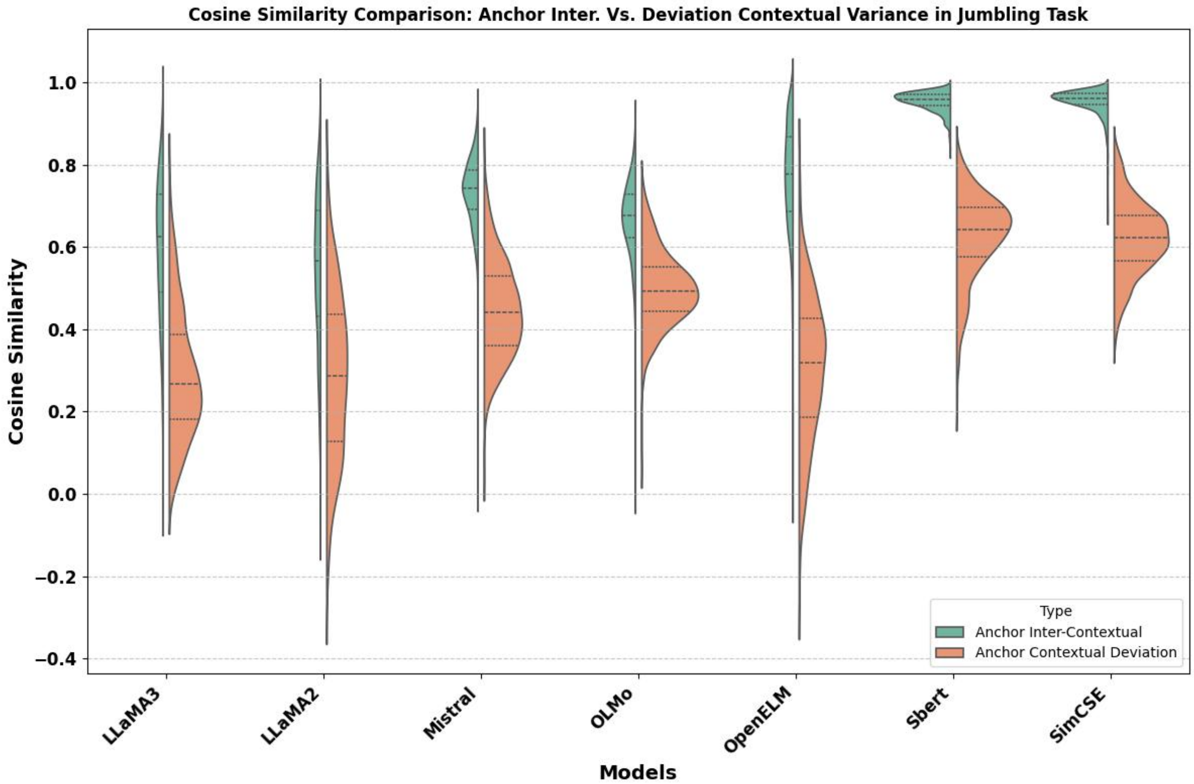


(a) The distribution of cosine similarities between Anchor Inter-Contextual Variance and Anchor Contextual Deviation words.

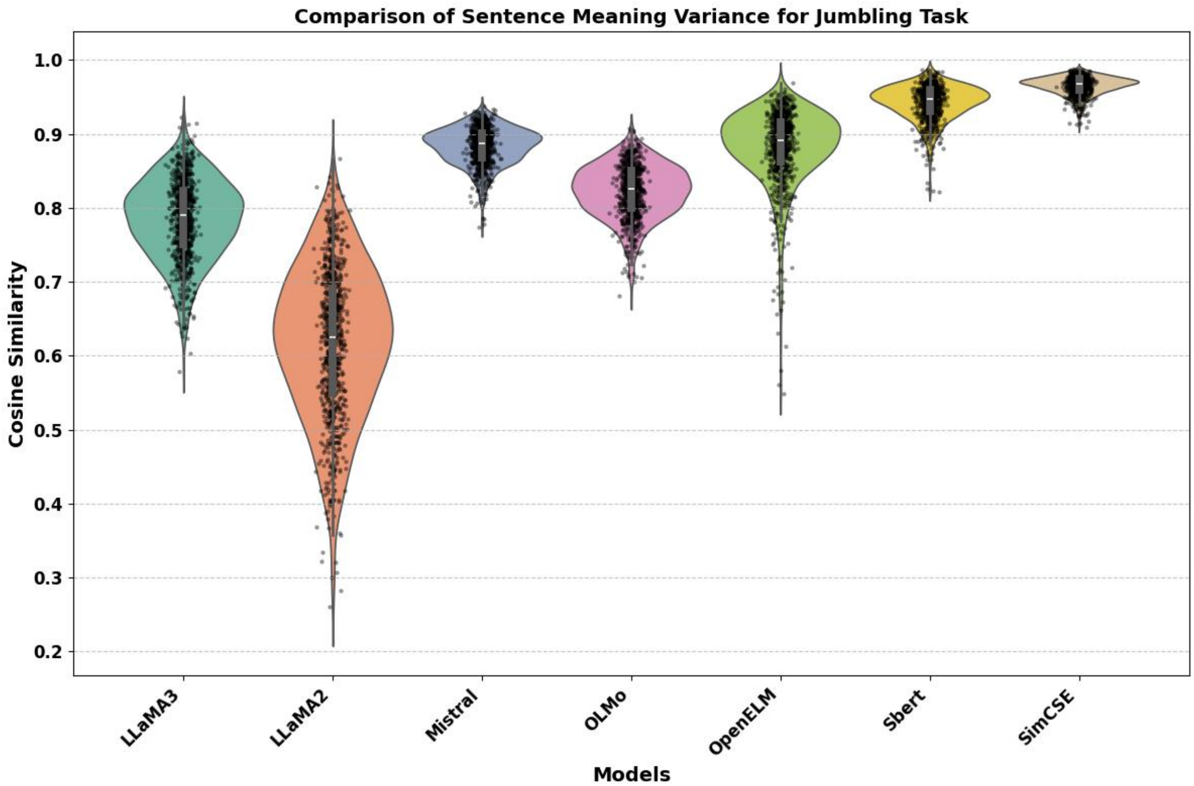


(b) The distribution of cosine similarities between sentences in Sentence Meaning Variance.

Figure 12: Questionnaire Task comparison

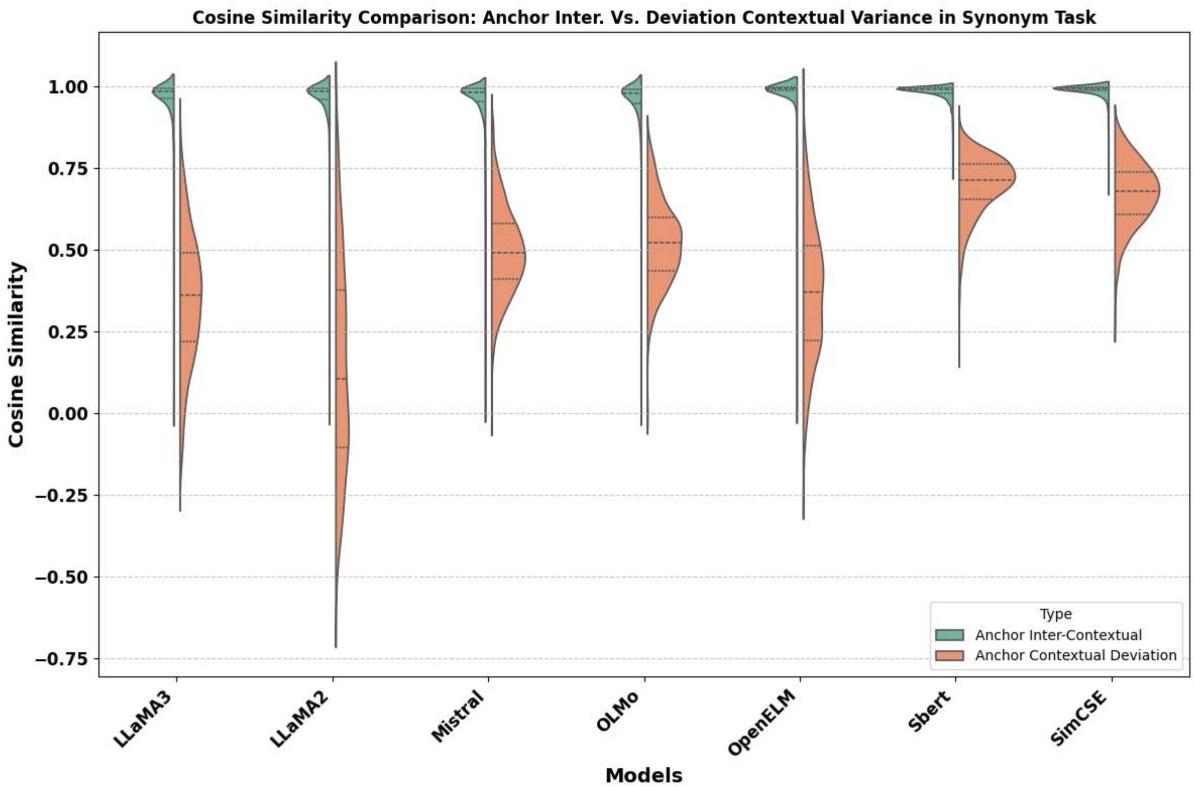


(a) The distribution of cosine similarities between Anchor Inter-Contextual Variance and Anchor Contextual Deviation words.

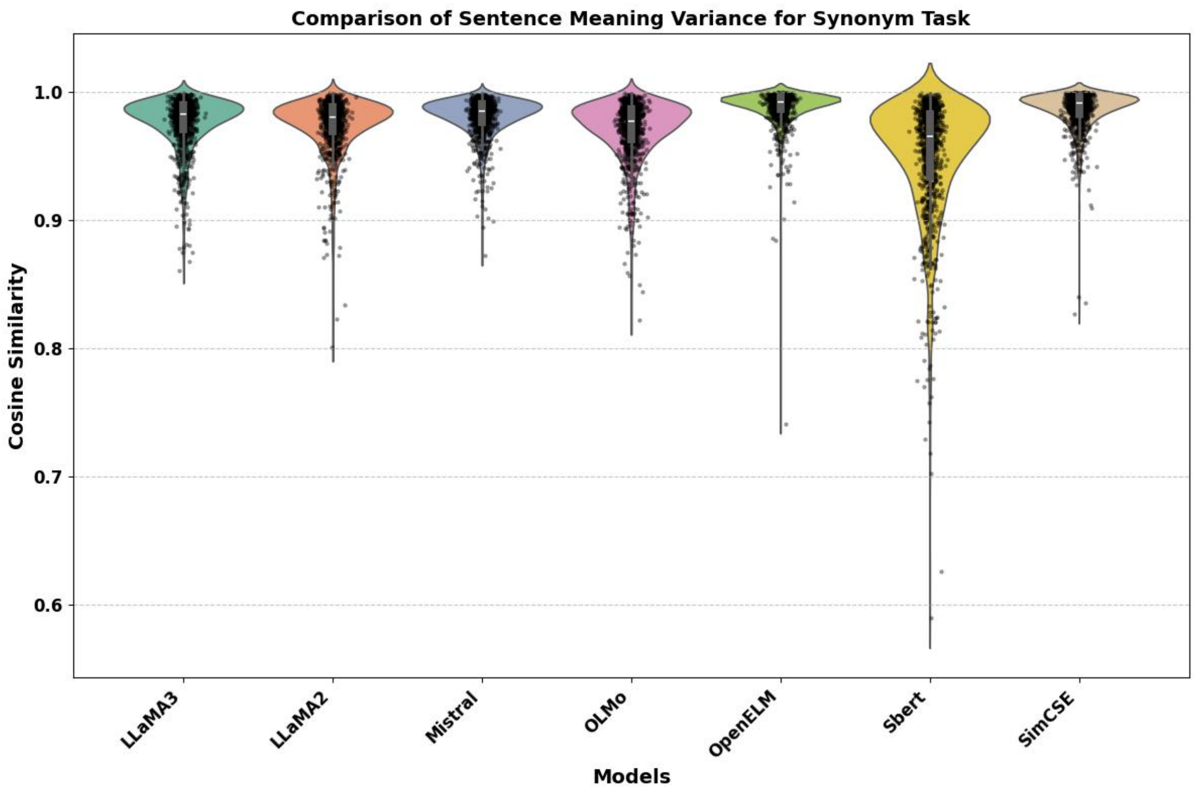


(b) The distribution of cosine similarities between sentences in Sentence Meaning Variance.

Figure 13: Jumbling Task comparison

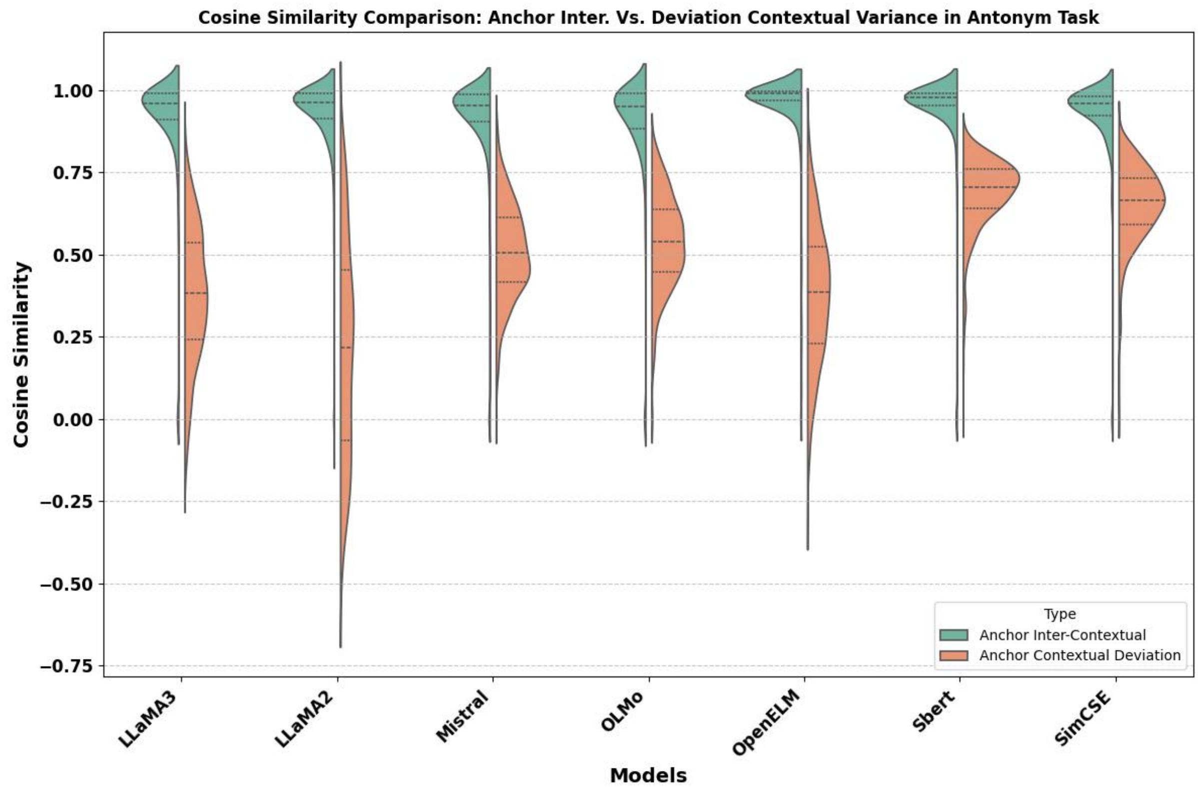


(a) The distribution of cosine similarities between Anchor Inter-Contextual Variance and Anchor Contextual Deviation words.

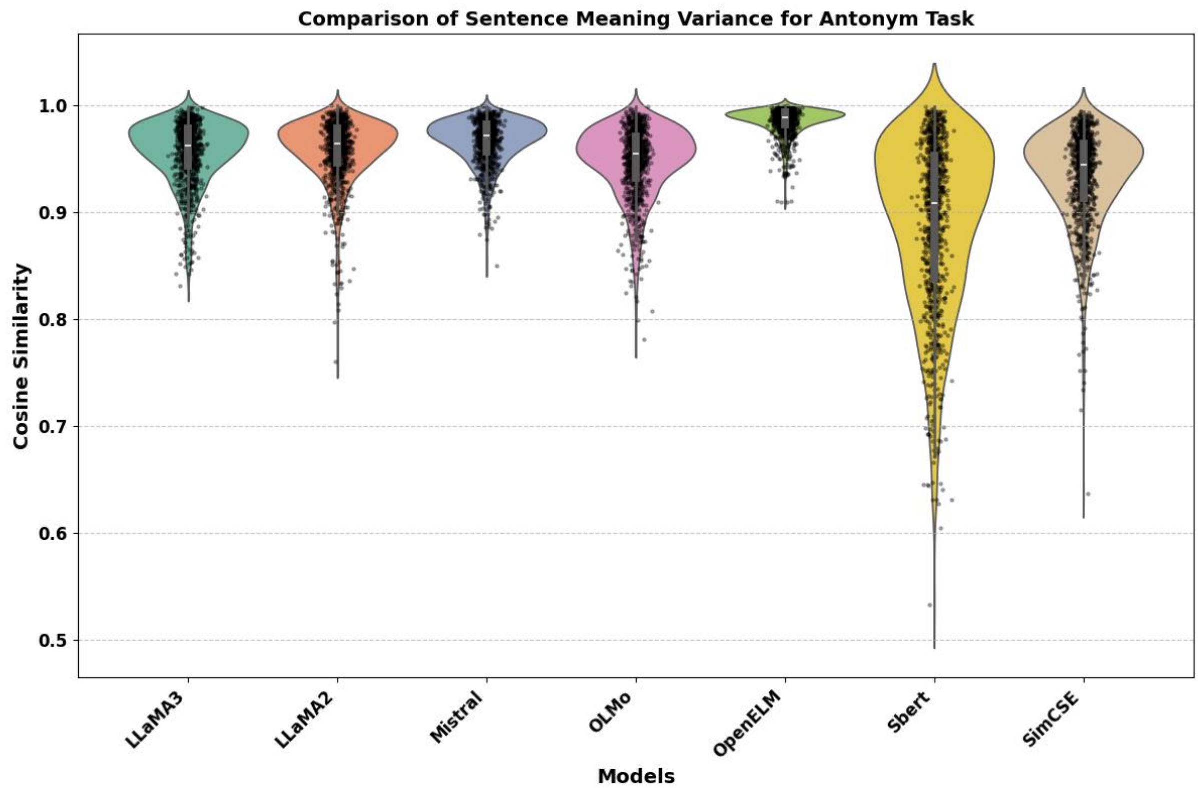


(b) The distribution of cosine similarities between sentences in Sentence Meaning Variance.

Figure 14: Synonym Task comparison

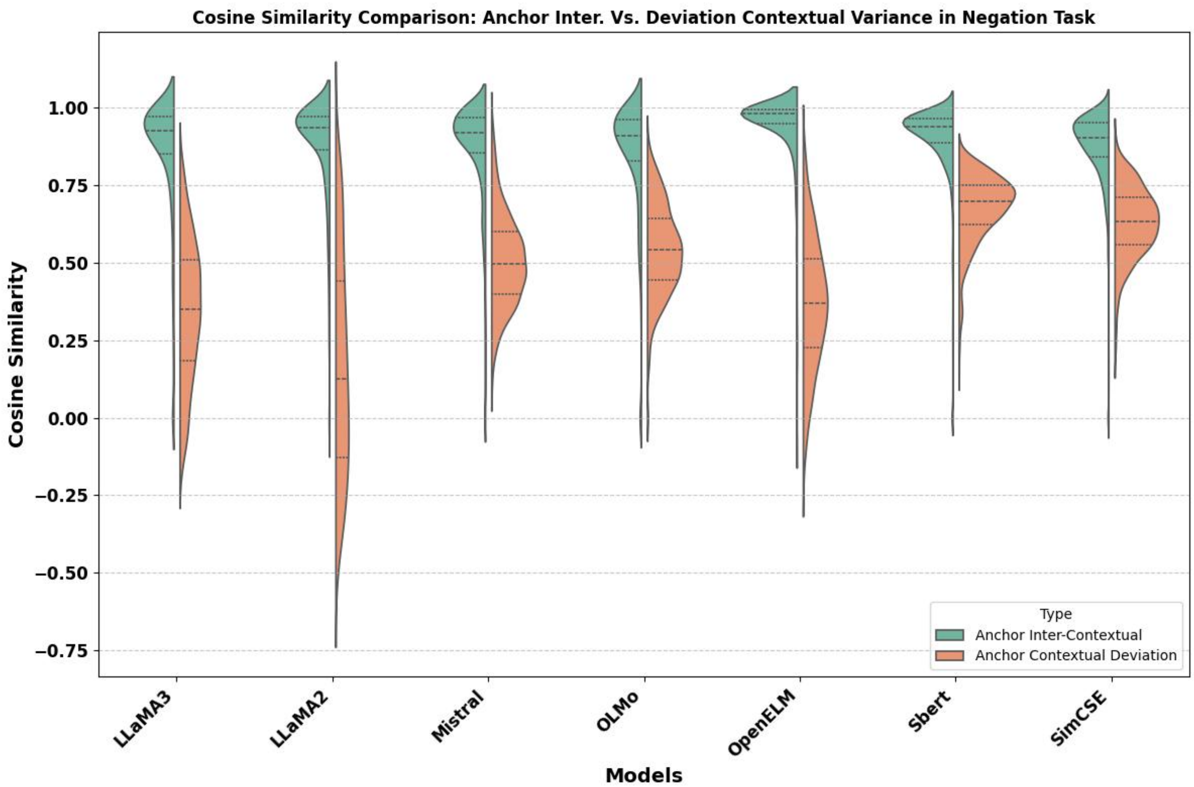


(a) The distribution of cosine similarities between Anchor Inter-Contextual Variance and Anchor Contextual Deviation words.

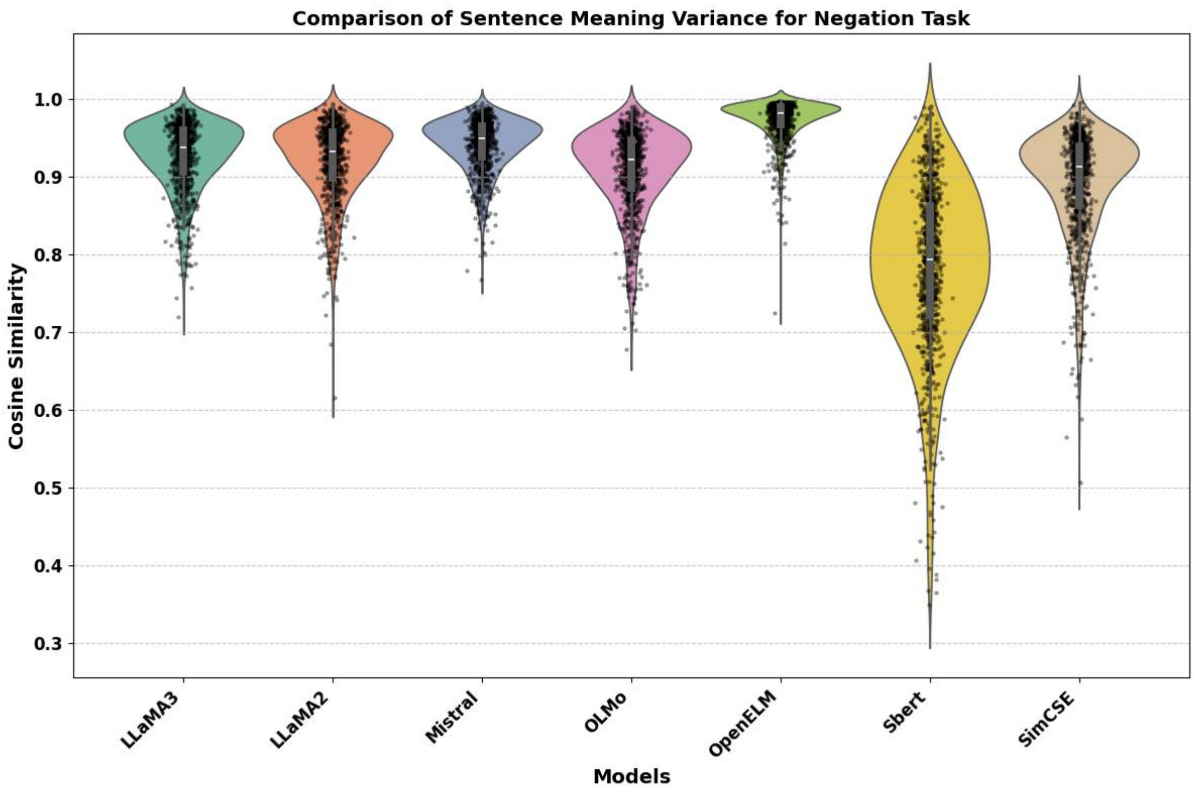


(b) The distribution of cosine similarities between sentences in Sentence Meaning Variance.

Figure 15: Antonym Task comparison



(a) The distribution of cosine similarities between Anchor Inter-Contextual Variance and Anchor Contextual Deviation words.



(b) The distribution of cosine similarities between sentences in Sentence Meaning Variance.

Figure 16: Negation Task comparison