# Varying Sentence Representations via Condition-Specified Routers

**Ziyong Lin**[1,2*], **Quansen Wang**[1,3], **Zixia Jia**[1✉], **Zilong Zheng**[1✉]
[1]National Key Laboratory of General Artificial Intelligence, BIGAI
[2]Tsinghua University [3]Peking University
linziyon22@mails.tsinghua.edu.cn
{wangquansen,jiazixia,zlzheng}@bigai.ai
https://github.com/bigai-nlco/CSR

## Abstract

Semantic similarity between two sentences is inherently subjective and can vary significantly based on the specific aspects emphasized. Consequently, traditional sentence encoders must be capable of generating conditioned sentence representations that account for diverse conditions or aspects. In this paper, we propose a novel yet efficient framework based on transformer-style language models that facilitates advanced *conditioned* sentence representation while maintaining model parameters and computational efficiency. Empirical evaluations on the Conditional Semantic Textual Similarity and Knowledge Graph Completion tasks demonstrate the superiority of our proposed framework.

## 1 Introduction

Sentence Textual Similarity (STS) (Agirre et al., 2012), which predicts the semantic similarity between two sentences, has long been a prevalent and crucial task for evaluating the quality of sentence representations and the semantic understanding capabilities of Natural Language Processing (NLP) models. However, traditional STS evaluation methods focus on the overall meaning of entire sentences, thereby neglecting the fine-grained semantics related to different aspects. To address this limitation, Deshpande et al. (2023) introduced Conditional STS (C-STS), which measures the similarity between two sentences within the context of given condition sentences. For instance, the two sentences "*A hotel room decorated in silver and white has a large mirror over the headboard of the bed.*" and "*A small bedroom suite in a hotel setting with a bed, small table, and two chairs.*" are considered similar given the condition "*The name of the place*" since both sentences describe rooms located in a hotel. Conversely, these sentences are considered dissimilar given the condition "*The number of chairs*" as the first sentence does not mention any chairs, whereas the second sentence explicitly mentions two chairs.

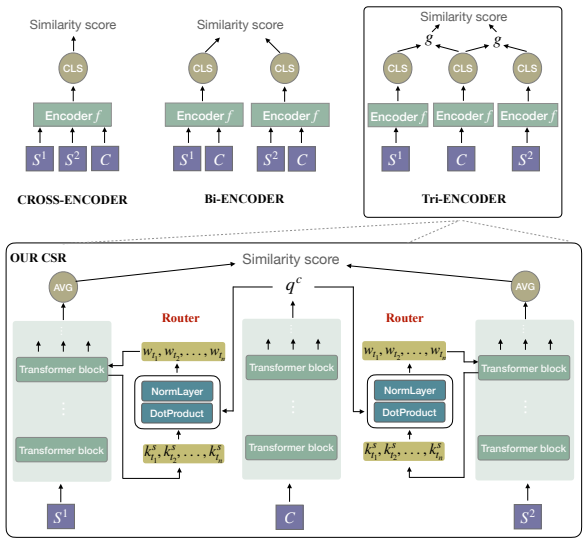



Figure 1: Illustrations of CROSS-ENCODER, BI-ENCODER, TRI-ENCODER architectures, with our conditioned sentence representation framework. $g$ generally is the compositional function.

Solving the C-STS task necessitates that the sentence embedding model be capable of generating distinct representations for the same sentence under different conditions. Previous studies (Deshpande et al., 2023) have introduced three architectures designed to incorporate the influence of conditions into sentence embeddings: CROSS-ENCODER, BI-ENCODER, and TRI-ENCODER, as depicted in Fig. 1. As mentioned in Reimers and Gurevych (2019), the CROSS-ENCODER architecture concatenates each sentence pair (additional with each condition in C-STS) to calculate similarity scores between every two sentences, leading to computational inefficiency. Similarly, the BI-ENCODER in the C-STS task concatenates each sentence with

*Work was done during LZ's internship at BIGAI.
✉ Correspondence to Zixia Jia and Zilong Zheng.



*Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17390–17401
November 12-16, 2024 

each condition, taking the limitation that the representation must be generated for every unique combination of sentences and conditions (Yoo et al., 2024). In contrast, the TRI-ENCODER employs a more computationally efficient method: it precomputes embeddings of sentences and conditions independently and then uses a separate composition function to merge the semantics of the sentence and the condition.

However, Deshpande et al. (2023) utilize the Hadamard product or concatenation operations on pre-computed sentence and condition embeddings to derive conditioned sentence representations for similarity calculation, observing a significant performance gap compared to the BI-ENCODER architecture. Yoo et al. (2024) enhanced the TRI-ENCODER architecture by integrating Hypernetworks (Ha et al., 2016) to project the original sentence embeddings into a specific condition subspace, thereby reducing the performance disparity with the BI-ENCODER architecture. Nevertheless, the use of Hypernetworks introduces an external parameter set three times larger than the backbone model (SimCSE (Gao et al., 2021)), which is necessary to achieve their improved performance. More related work can be found in Appendix D.

In this paper, we propose a **Conditioned Sentence Representation (CSR)** method based on the TRI-ENCODER architecture, with the goal of enhancing its performance without introducing external parameters while maintaining computational efficiency. Unlike TRI-ENCODER models in Deshpande et al. (2023), which focus on merging the semantics of the sentence and condition, our approach posits that the condition semantics ought to play the role of influencing which tokens in the sentence should contribute to the final condition-specific sentence embedding. Drawing inspiration from the router and heavy-light attention mechanism in Ainslie et al. (2023), we design a condition-relevant router that calculates scores for each token in the sentence, selecting the most relevant tokens that reflect the semantics emphasized by a given condition. Then, the relevant tokens additionally perform "heavy attention", while the remaining tokens perform "light attention" by multiplying the normalized router scores. This approach obtains different score distributions for a sentence based on different conditions, thereby generating varied conditioned sentence representations.

We evaluate our method on the C-STS task and Knowledge Graph Completion (KGC) task, demonstrating significant improvement over previous TRI-ENCODER frameworks (Deshpande et al., 2023; Yoo et al., 2024) while maintaining memory and computational efficiency.

## 2 Framework

Traditionally, the input for sentence embedding models (considering transformer-style language models in this paper) consists solely of the text itself (*e.g.*, sentences, paragraphs) and is expected to output a vectorized representation of the input sentence. **Conditioned sentence representation**, however, dynamically modifies the sentence embedding based on the influence of another text serving as a condition, thereby reflecting the conditioned sentence information.

**Motivation** Given a sentence set $\mathcal{S} = \mathbf{s}^i$ and a condition set $\mathcal{C} = \mathbf{c}^i$, obtaining every possible conditioned sentence representation requires $|\mathcal{S}| \times |\mathcal{C}|$ forward encoding processes in the BI-ENCODER architecture. In contrast, the TRI-ENCODER architecture requires only $|\mathcal{S}| + |\mathcal{C}|$ forward encoding processes, with an additional $|\mathcal{S}| \times |\mathcal{C}|$ **lightweight** composition computation (*e.g.*, Hadamard product). While the TRI-ENCODER is more computationally efficient, it suffers a performance drop due to the lack of direct interaction between sentences and conditions within the encoders as discussed in Section 1. Our motivation is to improve the trade-off between performance and computational efficiency, effectively integrating the advantages of both architectures to enhance the TRI-ENCODER's performance while maintaining efficient computation.

**Method** In this study, we employ a transformer-style encoder $f$ as the backbone of our CSR framework, depicted in Figure 1. We provide a detailed description of the process to derive the conditioned representation for a sentence $\mathbf{s}$.

Our CSR initially obtains the embedding of a given condition $\mathbf{c}$ using an encoder $f$. Unlike prior approaches, we do not utilize the final [CLS] embedding as the representation of the condition. Instead, we extract the query vector $\mathbf{q}^c \in \mathbb{R}^{1\times d_c}$ of the [CLS] token in the last layer for the subsequent router operation.

The sentence $\mathbf{s}$ is input into the same encoder $f$. However, starting from the $t$-th layer of the encoder, a router operation is introduced. For instance, considering the $t$-th layer, the query vector $\mathbf{q}^c$ of condition $\mathbf{c}$ queries each token of sentence $\mathbf{s}$ to

obtain the router score for each token:

$$\mathbf{score}_t = \frac{\mathbf{q}^c \cdot \mathbf{K}_t^{s\mathrm{T}}}{\sqrt{d_k}}, \quad \mathbf{w}_t = softmax(\mathbf{score}_t)$$

where $\mathbf{K}_t^s \in \mathbb{R}^{n \times d_k}$ means the key vector of $n$ tokens in sentence $\mathbf{s}$ in the $t$-th encoder layer. $d_k = d_c$ is the hidden size of $\mathbf{K}_t^s$ and $\mathbf{q}^c$. Then, the multi-head attention hidden state $\mathbf{h}_t^s$ of sentence $\mathbf{s}$ in the $t$-th layer is given by:

$$\mathbf{h}_t^s = (\mathbf{1} + \mathbf{w}_t) * \mathbf{h}_t^{'s}, \tag{1}$$

where $\mathbf{h}_t^{'s}$ is the original multi-head attention hidden state of the $t$-th transform block without router mechanism, $\mathbf{1} \in \mathbb{R}^{n \times 1}$ has all values being 1. Subsequently, $\mathbf{h}_t^s$ is passed to the following modules (*i.e.*, residual, norm, and feed-forward layers) of the $t$-th transformer block and then progresses to the next $(t+1)$-th transformer block.

From the $t$-th layer to the final $T$-th layer of encoder $f$, each layer's hidden state is affected by the condition-specified router. Finally, following the approach of Deshpande et al. (2023), we use an average pooling layer to obtain the conditioned sentence representation $\mathbf{r}_c^s$ from the hidden states output by the last layer of the encoder.

To ensure computational efficiency, we only incorporate routers into the final few layers, *i.e.*, $t \to T$. Empirical evidence indicates that integrating a router exclusively in the $t = T - 1$ layer has demonstrated the significant superiority of our framework. Refer to Section 3.3 for more details. We emphasize the advantages of our framework:

- ▷ Our framework does not introduce any additional parameters. All vectors $\mathbf{q}^c$ and $\mathbf{K}^s$ utilized in the router mechanism are derived directly from the original backbone encoder $f$.
- ▷ The router score calculation is lightweight. And we directly enhance the representation of those selected condition-relevant tokens (with higher router scores) by multiplying the normalized router scores, promising **simple, efficient, yet effective** framework.

**Training** Given a training sample $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{c}, y^*)$, where $y^*$ represents the ground truth similarity score between sentence $\mathbf{s}_1$ and sentence $\mathbf{s}_2$ under the condition $\mathbf{c}$, we adopt the Mean Squared Error (MSE) objective (Wang and Bovik, 2009) to train our framework: $\mathcal{L} = ||\phi(\mathbf{r}_c^{s_1}, \mathbf{r}_c^{s_2}) - y^*||_2^2$, where $\phi$ denotes the cosine similarity function. The Cache mechanism can be found in Appendix B.

| Model | Spearman | Pearson | # Params |
|---|---|---|---|
| *Tri-encoder Architecture* | | | |
| **Baseline**✦ | $28.9_{\pm 0.7}$ | $27.8_{\pm 1.2}$ | 125M |
| **Hyper-CL**✦ | $33.8_{\pm 0.1}$ | $33.1_{\pm 0.3}$ | +453M |
| **Our CSR**✦ | $34.1_{\pm 0.2}$ | $34.0_{\pm 0.6}$ | +0M |
| **Baseline**◗ | $35.3_{\pm 1.0}$ | $35.6_{\pm 0.9}$ | 355M |
| **Hyper-CL**◗ | $39.6_{\pm 0.2}$ | $40.0_{\pm 0.3}$ | +1076M |
| **Our CSR**◗ | $\mathbf{42.0}_{\pm 0.8}$ | $\mathbf{42.7}_{\pm 0.8}$ | +0M |
| ***Hidden Router***◗ | $39.9_{\pm 0.2}$ | $39.4_{\pm 0.1}$ | +0M |
| ***Heavy Router***◗ | $40.5_{\pm 0.2}$ | $39.9_{\pm 0.2}$ | +2M |
| *Bi-encoder Architecture* | | | |
| **Baseline**✦ | $43.4_{\pm 0.2}$ | $43.5_{\pm 0.2}$ | 124M |
| **Our CSR**✦ | $43.7_{\pm 0.2}$ | $43.6_{\pm 0.1}$ | +0M |
| **Baseline**◗ | $47.5_{\pm 0.1}$ | $47.6_{\pm 0.1}$ | 354M |
| **Our CSR**◗ | $\mathbf{47.9}_{\pm 0.2}$ | $\mathbf{47.8}_{\pm 0.2}$ | +0M |

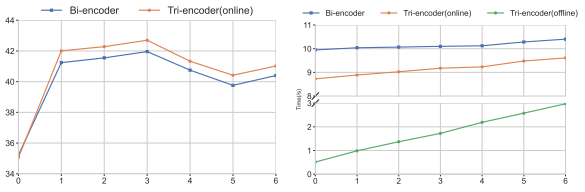
Table 1: Results of our framework in C-STS task. The best results are **BOLD**. Rows with ✦ indicates results with DiffCSE$_{base}$ as backbone encoder; rows with ◗ are results with SimCSE$_{large}$ as backbone encoder. Results of **Baseline** and **Hyper-CL** are from Deshpande et al. (2023) and Yoo et al. (2024) respectively. ***Heavy Router*** and ***Hidden Router*** are two variants of **Our CSR**.

## 3 Experimental Results

### 3.1 Conditional Sentence Textual Similarity

**Settings** We evaluate our CSR framework on the Conditional Sentence Textual Similarity (C-STS) task (Deshpande et al., 2023) with different backbones (SimCSE (Gao et al., 2022) and DiffCSE (Chuang et al., 2022)). The base architectures, dataset splits, hyper-parameters, and evaluation metrics (Spearman and Pearson correlations) are consistent with Deshpande et al. (2023) (**Baseline**). We select the specific hyper-parameter $t$ and the best model checkpoint based on validation set performance. Although the CSR framework is designed for TRI-ENCODER architecture, it can be extended to BI-ENCODER architecture by using the attention weight of the condition's [CLS] upon sentence tokens as the router score $\mathbf{w}$. And the sentence hidden states are computed according to Eq. (1). We also demonstrate the effectiveness of our router mechanism in BI-ENCODER. All results are averaged on three runs. We do not consider CROSS-ENCODER due to its computational inefficiency and subpar performance (43.2% in Deshpande et al. (2023) and 43.8% in Anonymous (2024)) compared to BI-ENCODER.

**Results** The experimental results of our CSR framework on the C-STS task are shown in Table 1. Compared to C-STS baseline (Deshpande et al., 2023), our CSR achieves significant improvements, namely a **6.7%** and **7.1%** improvements on




(a) Model performance (b) Total inference time (2834 examples)

Figure 4: Performances and Inference times of varying number of router layers. The x-coordinate equals $T - t$, representing the number of transformer layers with the router, where $t = 0$ corresponds to the base model.

| | MRR | Hit@1 | Hit@3 |
|---|---|---|---|
| **SimKGC (Bi-encoder)** | 66.6 | 58.7 | 71.7 |
| **SimKGC (Tri-encoder)** | 33.5 | 22.6 | 38.2 |
| **Hyper-CL (Tri-encoder)** | 61.6 | 50.6 | 69.0 |
| **Our CSR (Tri-encoder)** | 66.1 | 58.1 | 71.1 |

Table 2: Results of our framework in KGC task.

the Spearman and Pearson correlation coefficients respectively. Even though the Hyper-CL (Yoo et al., 2024) tripled the number of parameters, our CSR without any additional parameters still surpassed it. It is worth mentioning that integrating our router mechanism with BI-ENCODER architecture also demonstrates the advantage compared to Deshpande et al. (2023). We conducted significance tests ($p < 0.05$) based on the T-test by re-running our CSR and the baseline ten times. The results yielded a p-value of 0.0007 and 0.0139 corresponding to Spearman and Pearson respectively, indicating our CSR is significantly superior to the baseline. Some case studies can be found in Appendix A.

### 3.2 Knowledge Graph Completion

**Settings** To prove the generalization of our CSR, we evaluate it on another task, Knowledge Graph Completion (KGC). Based on the Simple Contrastive Knowledge Graph Completion (SimKGC) mechanism (Wang et al., 2022a), which transforms the original KGC task into the formulation of comparing the similarity of head-relation embedding with tail embedding. Since our CSR could provide text embeddings, we consider the head node as a sentence and the relation as the condition, aiming for the model to output a relation-specific sentence embedding to compare with the tail embedding. We follow the experiment settings and evaluation metrics (including the Mean Reciprocal Rank (MRR), Hit@1, and Hit@3) of the original SimKGC (Wang et al., 2022a) paper and Hyper-CL (Yoo et al., 2024).

| Weight Strategy | Spearman | Pearson |
|---|---|---|
| **Baseline** | 35.3 | 35.6 |
| **Our CSR** | 42.0 | 42.7 |
| **Random Weight** | 34.7 | 34.3 |
| **Only Weight** | 39.28 | 38.3 |

Table 3: Model performance under different weighting strategies

**Results** As Table 2 indicates, our framework achieves consistent improvement in both computational and parameter efficiency for the SimKGC task, as the MRR improves by 4.5 and the Hit@1 increases by 7.5 compared to Hyper-CL. It is exciting that our CSR based on TRI-ENCODER architecture achieves nearly comparable performance with BI-ENCODER architecture.

### 3.3 Analysis

**Variants of Our Framework** We show the performance of two variants of CSR in Table 1. **Heavy Router** represents we use additional parameters to calculate the router scores. Specifically, we randomly initialized $\mathbf{W}_q \in \mathbb{R}^{d_c \times d_c}$ and $\mathbf{W}_k \in \mathbb{R}^{d_c \times d_c}$ for calculating the condition queries and the token keys of sentences. Suppose that the hidden states of a condition's [CLS] token and a sentence token $s_i$ are $\mathbf{h_c}$ and $\mathbf{h}_i$ respectively, then the condition query and token key are $\mathbf{W}_q\mathbf{h}_c^{\mathrm{T}}$ and $\mathbf{W}_k\mathbf{h}_i^{\mathrm{T}}$ respectively. $\mathbf{W}_q$ and $\mathbf{W}_k$ were updated by back-propagation according to the current loss function. **Hidden Router** represents we use the hidden states of the condition and sentence to calculate router scores rather than the condition's query and sentence tokens' keys. The Hidden Router variant performed poorly, suggesting that hidden states may not effectively serve as queries for selecting relevant tokens. The Heavy Router, which uses additional parameters, also underperformed compared to the standard router, likely due to the dataset's insufficient size or inadequate training.

**Impact of Weighting Strategy** To explore the efficacy of our weighting method, we also investigate two distinct settings. The **Random Weight** bypasses the weights suggested by the router and instead opts for a random initialization process. The **Only Weight** means that we replace $\mathbf{1} + \mathbf{w}_t$ to $\mathbf{w}_t$ in Eq. (1). As shown in Table 3, the Random Weight harms the baseline performance, indicat-

| Model | Pearsonr | Spearmanr |
|---|---|---|
| **Baseline** | 3.1 | 3.4 |
| **Our CSR** | 5.5 | 5.5 |
| **Llama2-7B** | 0.4 | 1.9 |
| **Llama3-8B** | 0.6 | 2.9 |
| **Text-embedding-3 (large)** | 6.0 | 5.6 |

Table 4: Results without fine-tuning on various methods.

ing that it can not catch condition-relevant token information. Under the Only Weight setting, the performances still have some improvement compared to the baseline but are under our best results. The reason is likely that the condition query tends to pick the most relevant tokens (which are sparse) related to the condition topic, but the semantics of some tokens as context are still important for the whole sentence representation. Therefore, while prioritizing the most salient terms, we must not overlook the remaining contextual tokens.

**Impact of Various Router Layers** We conducted experiments by progressively incorporating the router mechanism starting from the final layer (*i.e.,* $t = T$), analyzing the influence of hyperparameter $t$ (shown in Fig. 4 (a)) with the backbone SimCSE$_{large}$. We can find that only integrating our router in the $t = T - 1$ layer has demonstrated the superiority of our framework. As the number of layers with the router mechanism increased, the performance gains plateaued and even slightly declined. This suggests that just incorporating a few layers of condition-guided information in the sentence self-attention process is effective for controlling sentence representation. The subsequent decline might be attributed to the overabundance of control signals, which could obscure certain less prominent yet crucial words in the sentence.

It is noteworthy that the addition of routers only slightly increases the model's inference time. It can be seen that adding routers in the last 3 layers achieves the best performance, where the inference time is still far less than BI-ENCODER baseline. Our model has the potential efficiency when more conditioned sentence representations should be computed. The reason for the relatively small difference in inference time between BI-ENCODER and TRI-ENCODER settings is that there are few repeated conditions and sentences in the C-STS dataset. As a result, the time advantage of the tri-encoder in an online setting (caching while computing) is not as pronounced. If there are $|\mathcal{C}|$ conditions and $|\mathcal{S}|$ sentences, and conditioned sentence representation have to be computed for every pair, the time advantage would become more noticeable due to multiple times of reusing cache. For additional clarification, we also calculate the inference time in an offline setting, where both condition and sentence representations are precomputed and cached. The results are shown in Fig. 4 (b). It is obvious that the inference time of the Tri-encoder in the offline setting is largely less than the Bi-encoder. Our framework also promises computational efficiency compared to the Tri-encoder baseline ($T - t = 0$).

**Analysis without Finetuning** As shown in Table 4, the inclusion of a single router layer in our CSR led to a modest improvement in performance without further training. This suggests that the router mechanism possesses the inherent ability to select the relevant tokens in the sentences with respect to the condition semantics. We also conducted experiments leveraging Llama2-7B, Llama3-8B and Text-embedding-3-large[1]. For Llama2-7B, we first concatenate sentences and conditions. Then, we pass the concatenated text through the model and use the last layer representation of the last token as the representation of the entire sentence due to the next-token-prediction nature of the decoder-only model. Finally, we compute the cosine similarity of the two sentences for the final prediction. The result is shown in Table 4. The performance of the Llama2-7B and Llama3-8B models falls short of expectations, indicating that these LLMs lack the ability to directly generate satisfactory conditioned sentence representations. Our proposed method achieves similar performance as Text-embedding-3-large on the zero-shot setting with a much smaller model size. More discussion about LLMs for continued sentence representation can be found in Appendix C.

## 4 Conclusion

This paper introduces a novel Conditioned Sentence Representation method with a router mechanism to achieve substantial performance gains of TRI-ENCODER architecture in C-STS task and KGC task without introducing any additional parameters. Furthermore, it maintains computational efficiency, highlighting simplicity, efficiency, and effectiveness.

[1] https://platform.openai.com/docs/guides/embeddings/embedding-models

## Limitations

While our CSR method has yielded promising results on the C-STS task, there are several limitations to consider. First, more experiments on other tasks: many other tasks face the same requirement of conditioned sentence representations, such as the Aspect-Based Sentiment Analysis (ABSA) (Pavlopoulos, 2014) task which identifies and extracts the sentence sentiment of specific aspects. We slightly modify our framework to adapt to these tasks in future work. Second, as discussed in Appendix C, LLMs face challenges in the conditional sentence representation task, suggesting that our CSR method could also enhance the LLMs' ability to generate conditioned sentence representation. We take this exploration as future work.

## Acknowledgements

The authors thank the reviewers for their insightful suggestions on improving the manuscript. This work presented herein is supported by the National Natural Science Foundation of China (62376031).

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity.* sem 2012: The first joint conference on lexical and computational semantics—. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012), Montréal, QC, Canada*, pages 7–8.

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontanon, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, et al. 2023. Colt5: Faster long-range transformers with conditional computation. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Anonymous. 2024. SEAVER: Attention reallocation for mitigating distractions in language models for conditional semantic textual similarity measurement. In *Submitted to ACL Rolling Review - April 2024*. Under review.

Sihao Chen, Hongming Zhang, Tong Chen, Ben Zhou, Wenhao Yu, Dian Yu, Baolin Peng, Hongwei Wang, Dan Roth, and Dong Yu. 2023. Sub-sentence encoder: Contrastive learning of propositional semantic representations. *Preprint*, arXiv:2311.04335.

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljacic, Shang-Wen Li, Scott Yih, Yoon Kim, and James Glass. 2022. DiffCSE: Difference-based contrastive learning for sentence embeddings. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4207–4218, Seattle, United States. 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. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 6894–6910. Association for Computational Linguistics (ACL).

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings. *Preprint*, arXiv:2104.08821.

David Ha, Andrew M Dai, and Quoc V Le. 2016. Hypernetworks. In *International Conference on Learning Representations*.

OpenAI. 2022. Introducing chatgpt.

Ioannis Pavlopoulos. 2014. Aspect based sentiment analysis. *Athens University of Economics and Business*.

Nils Reimers and Iryna Gurevych. 2019. 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)*, pages 3982–3992.

Jingxuan Tu, Keer Xu, Liulu Yue, Bingyang Ye, Kyeongmin Rim, and James Pustejovsky. 2024. Linguistically conditioned semantic textual similarity. *Preprint*, arXiv:2406.03673.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022a. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*.

Zhou Wang and Alan C Bovik. 2009. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117.

Young Hyun Yoo, Jii Cha, Changhyeon Kim, and Taeuk Kim. 2024. Hyper-cl: Conditioning sentence representations with hypernetworks. *Preprint*, arXiv:2403.09490.

## A Case Study for Effectiveness of Router in Token Selection

We investigated the effectiveness of the router by visualizing the scores it assigns to each token. As shown in Fig. 5 , without the router, the sentence encoder can only guess which tokens to focus on, often leading to selections that may not align with the condition. In contrast, the [CLS] of the condition carries guiding information, allowing it to select tokens that are more relevant to the specific condition (Fig. 6 ).

We also examined whether different conditions could select their own useful tokens within the same sentence. As shown in Fig. 7 , different conditions do indeed focus on different tokens, further demonstrating the effectiveness of the router.

## B Cache

Traditional TRI-ENCODER architectures cache condition embeddings and sentence embeddings once computed, subsequently reusing them whenever conditioned representations related to them need to be computed. Yoo et al. (2024) cache the entire hypernetwork corresponding to a condition, maintaining computational efficiency but resulting in substantial memory usage. To balance memory usage and computation trade-offs, we cache the condition query and sentence token key vectors in the $t$-th layer. Compared to traditional TRI-ENCODER architectures, our approach maintains a similar cache size and incurs only additional forward computations for the $T - t$ layers. For efficiency, $t$ can be set to $T - 1$, requiring only lightweight additional calculations and ensuring almost the same performance. Empirically, we find that a very small value for $T - t$ (i.e., less than 3, see Section 3.3 for more details) is sufficient.

## C Discussion of transformer-style Encoder and Large Language Models (LLMs)

Deshpande et al. (2023) has conducted few-shot in-context learning utilizing various LLMs like Flan-T5 (Chung et al., 2024), Tk-Instruct (Wang et al., 2022b), ChatGPT-3.5 (OpenAI, 2022), and GPT-4 (Achiam et al., 2023), finding that it is challenging for LLMs by prompting them to verify the conditional sentence similarity. Except for GPT-4, all the LLMs perform less than 30% Spearman correlation score no matter the detailed instruction or more demonstrations. The best performance of GPT-4 is 43.6% Spearman in 4-shot. See more details from Deshpande et al. (2023). Besides, we try to fine-tune the Llama2-7B[2] through the prompting strategy, only achieving 24.5% Spearman correlation score. Furthermore, these methods can not output a proper conditioned sentence representation that we mainly focus on. *How well do the sentence representations that LLMs provide?* In view of this problem, we utilized Text-embedding-3-large on the C-STS task. The results are shown in Table 4. Our proposed method achieves similar results as ext-embedding-3-large on zero-shot setting with a much smaller model size. We also tried to fine-tune a Llama2-7B model with the proposed framework to acquire **Conditioned Sentence Representation**, but the performance is unsatisfactory. Upon experimenting we believe that the attention mechanism of the decoder-only model like Llama2-7B failed to incorporate half of the total tokens, so the sentence representation acquired does not best reflect the meaning of the entire sentence conditioned on another piece of text. We take exploring more appropriate fine-tuning methods for LLMs to obtain better conditioned sentence representation as future work.

## D Related Work

Recent studies have focused on various sentence representation methods to address semantically complex tasks, such as the C-STS task, which requires considering the effect of conditions on sentence representation. Tu et al. (2024) employs a QA-based approach by converting conditions into questions and using LLMs to generate answers

[2]https://huggingface.co/meta-llama/Llama-2-7b

that extract sentence semantics related to the condition. However, this approach does not explicitly model vectorized sentence representations. Chen et al. (2023) addresses the sub-sentence representation problem by constructing datasets containing sub-sentence pieces and subsequently performing pretraining to develop a sub-sentence encoder. The most relevant work to ours is Hyper-CL (Yoo et al., 2024), which focuses on conditioned sentence representation by constructing a condition-specific hypernetwork with substantial parameters. Additionally, for the C-STS task, SEAVER (Anonymous, 2024) proposes a cross-encoder approach, leveraging a self-reweighting technique to augment sentence representations with the cross-attention of condition text.

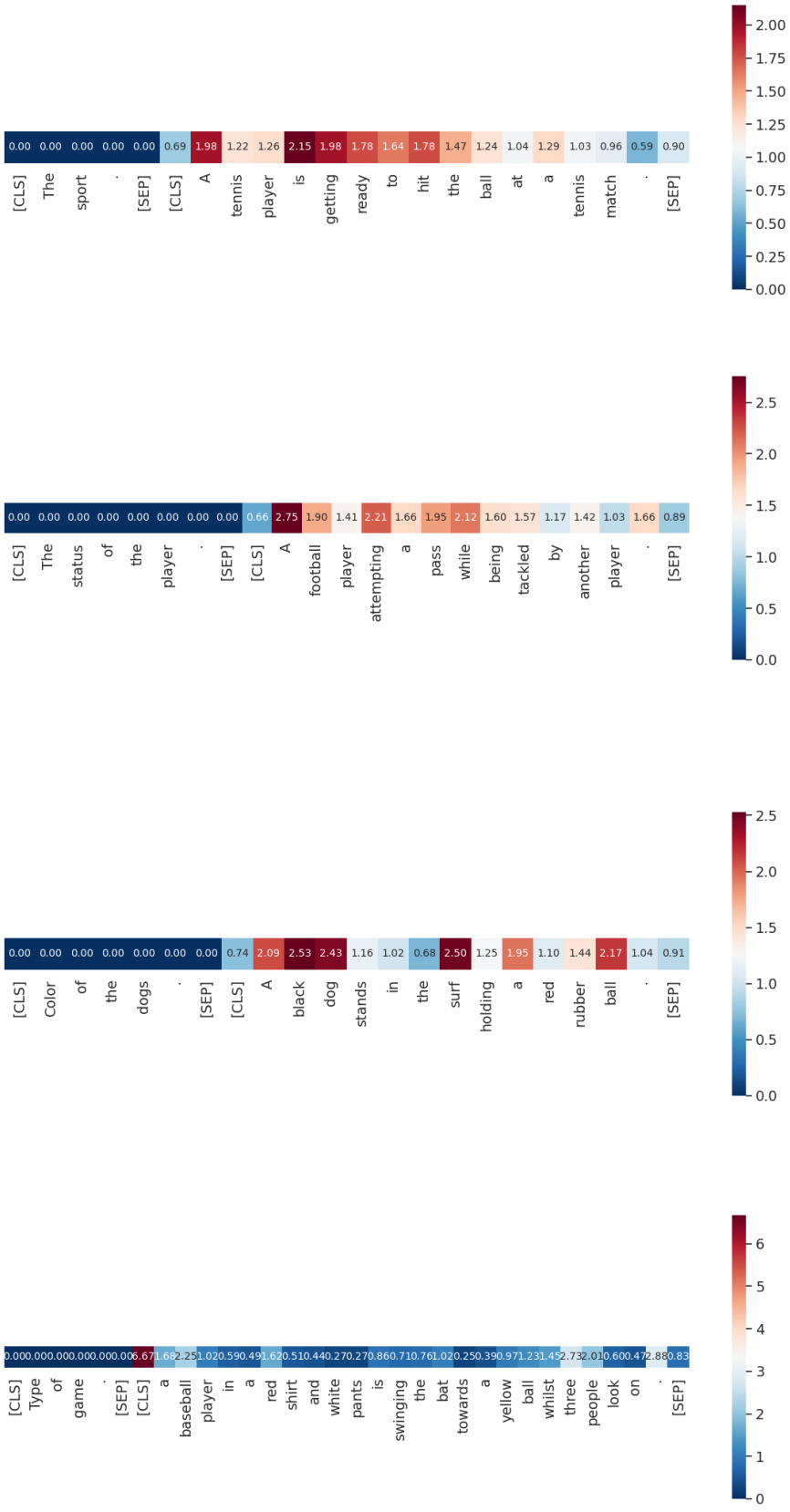



Figure 5: Visualization of sentence CLS token attention to different tokens without using the router (since the sentence cannot see the condition in the tri-encoder architecture, the condition part has a weight of 0).

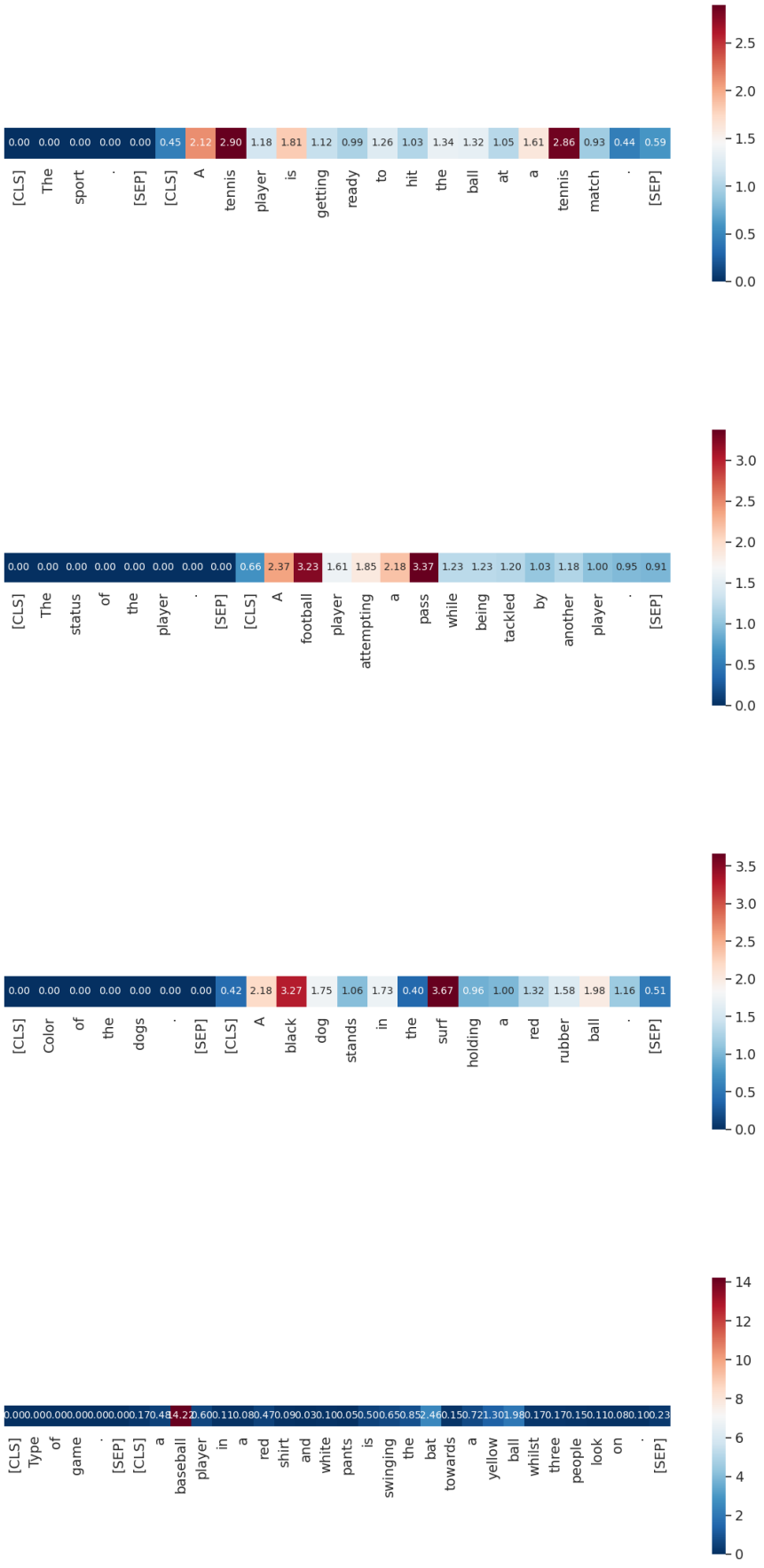
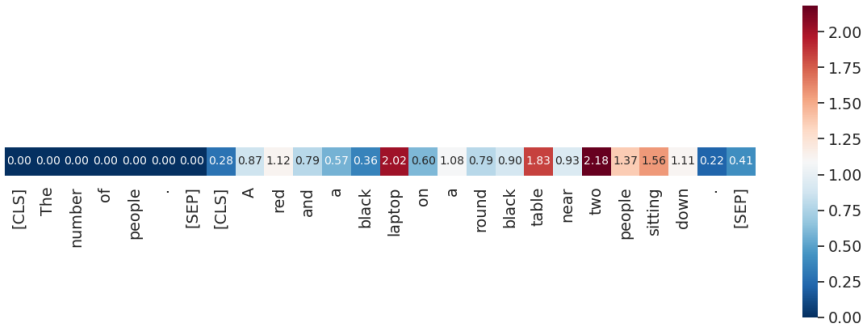
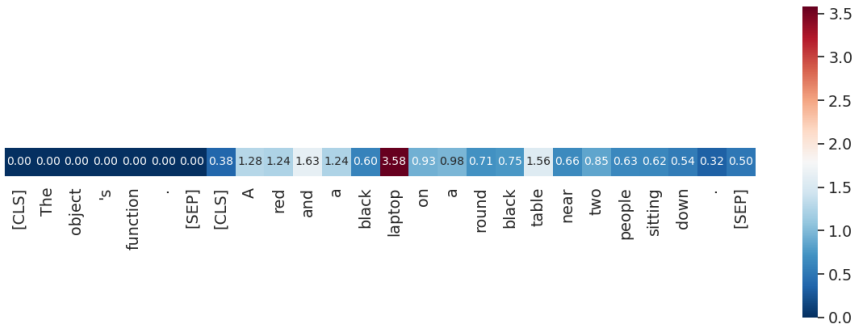
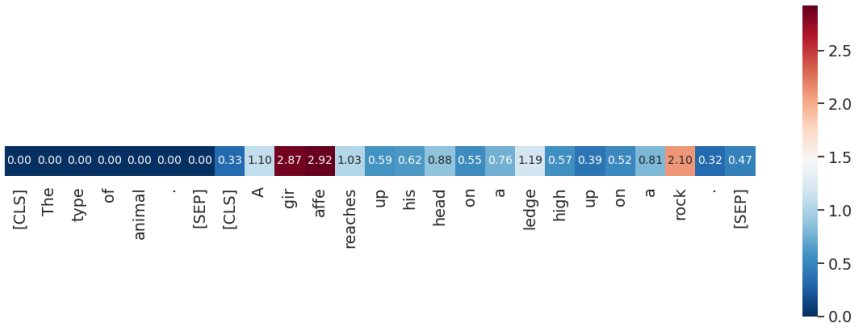
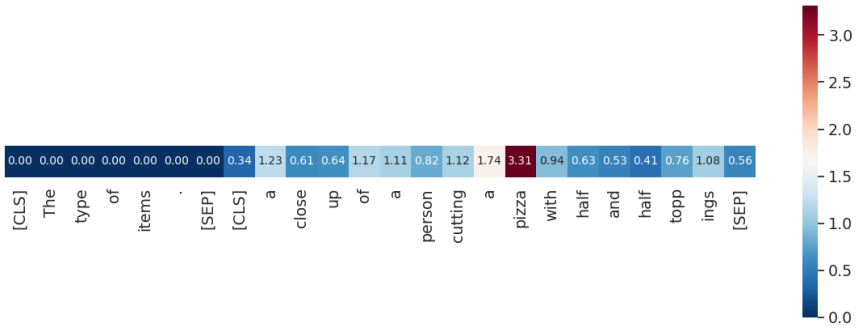



Figure 6: Visualization of how the CLS token from the condition attends to different sentence tokens when using the router mechanism.


[CLS] The number of people . [SEP] [CLS] A red and a black laptop on a round black table near two people sitting down . [SEP]
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.28 0.87 1.12 0.79 0.57 0.36 2.02 0.60 1.08 0.79 0.90 1.83 0.93 2.18 1.37 1.56 1.11 0.22 0.41
2.00 1.75 1.50 1.25 1.00 0.75 0.50 0.25 0.00


[CLS] The object 's function . [SEP] [CLS] A red and a black laptop on a round black table near two people sitting down . [SEP]
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.38 1.28 1.24 1.63 1.24 0.60 3.58 0.93 0.98 0.71 0.75 1.56 0.66 0.85 0.63 0.62 0.54 0.32 0.50
3.5 3.0 2.5 2.0 1.5 1.0 0.5 0.0


[CLS] The type of animal . [SEP] [CLS] A gir affe reaches up his head on a ledge high up on a rock . [SEP]
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 1.10 2.87 2.92 1.03 0.59 0.62 0.88 0.55 0.76 1.19 0.57 0.39 0.52 0.81 2.10 0.32 0.47
2.5 2.0 1.5 1.0 0.5 0.0


[CLS] The type of items . [SEP] [CLS] a close up of a person cutting a pizza with half and half topp ings [SEP]
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.34 1.23 0.61 0.64 1.17 1.11 0.82 1.12 1.74 3.31 0.94 0.63 0.53 0.41 0.76 1.08 0.56
3.0 2.5 2.0 1.5 1.0 0.5 0.0

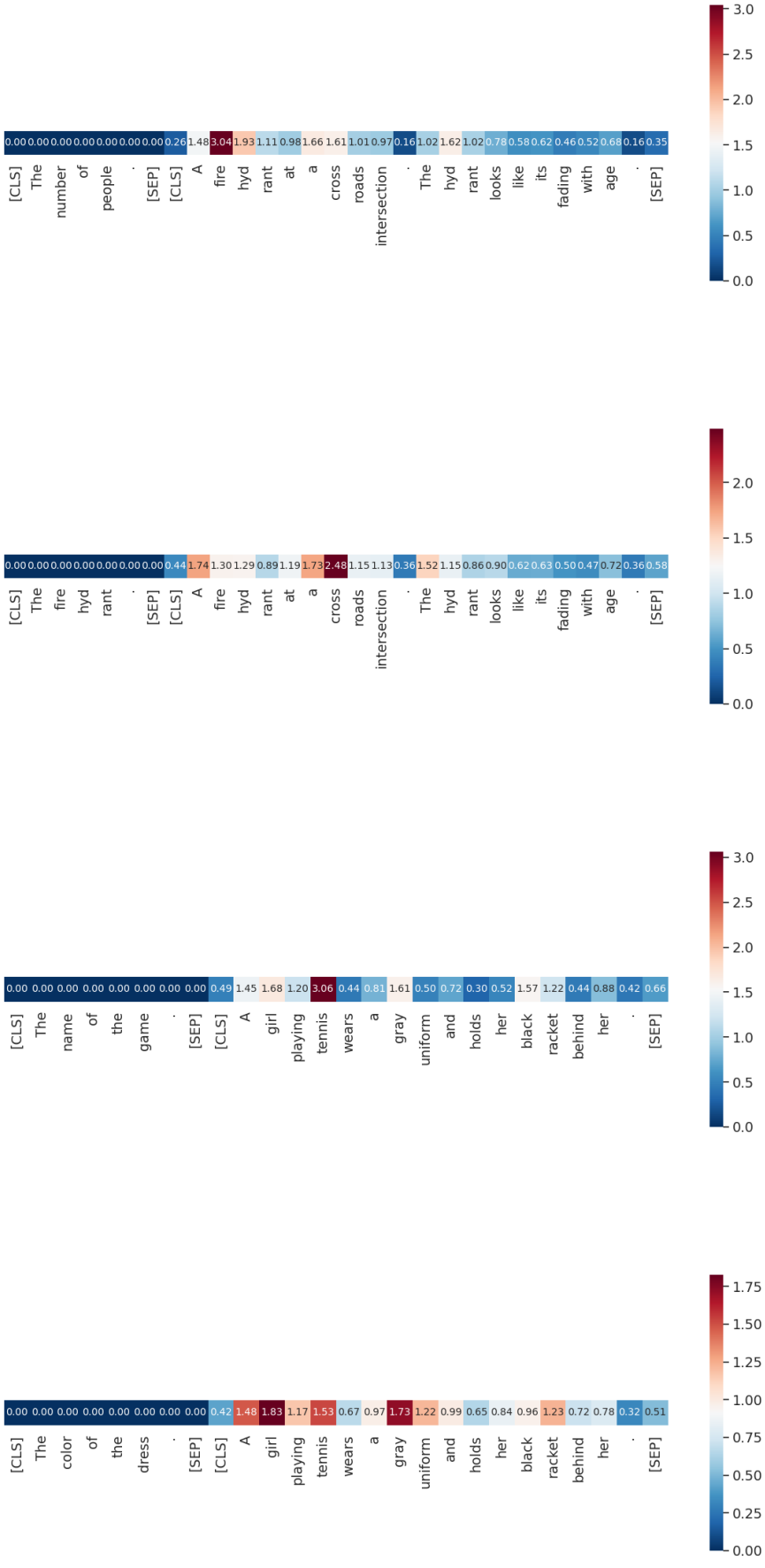



Figure 7: Visualization of CLS attention on different tokens of the same sentence under various conditions using the router mechanism.