

**Dissertation**  
**Improving Interpretability and Accuracy of Artificial Intelligence  
in Natural Language and Image Understanding**

by

Thang M. Pham

A dissertation submitted to the Graduate Faculty of  
Auburn University  
in partial fulfillment of the  
requirements for the Degree of  
Doctor of Philosophy

Auburn, Alabama

August 3, 2024

Keywords: natural language understanding, transformer, masked language model, word order sensitivity, feature attribution, contextual phrase embeddings, phrase similarity, phrase retrieval, phrase sense disambiguation, part-based image classifiers, language bottleneck, explainable and editable networks, zeroshot learning, text-image grounding, multimodality

Copyright 2024 by Thang M. Pham

Approved by

Anh Nguyen, Chair, Associate Professor of Computer Science and Software Engineering  
Shubhra Kanti Karmaker, Assistant Professor of Computer Science and Software Engineering  
Yang Zhou, Assistant Professor of Computer Science and Software Engineering  
Ashish Gupta, Professor of Business Analytics and Information Systems  
Jessica McDonald, Associate Professor of Department of Mathematics and Statistics

## Abstract

Transformer-based models, such as BERT [29], have revolutionized natural language processing (NLP) by setting new standards for accuracy and capability. BERT has achieved state-of-the-art (SOTA) results on many NLP tasks and benchmarks, such as GLUE [155], even surpassing human performance. Despite these successes, there remain questions about whether these models truly understand natural languages like humans do.

Our findings reveal that BERT-based classifiers often disregard the sequential order of words when evaluated on GLUE [115]. Using LIME [134], an attribution method to visualize how much a token contributes towards models’ prediction, we find that instead of understanding sentence meaning, these models rely on superficial cues. Additionally, incorporating BERT into attribution methods to yield more plausible counterfactuals when interpreting text classifiers has proven problematic. Our research shows that BERT is not particularly useful in this context unless the attribution method, such as LIME, produces out-of-distribution samples [116]. The limitations of current NLP benchmarks like GLUE are evident as they do not require models to understand the surrounding context before making predictions. To address this, we introduce the Phrase-in-Context (PiC) benchmark [117]. PiC forces models to comprehend the context first before interpreting the meaning of a phrase, as the meaning is context-dependent. This benchmark poses a significant challenge to models, with even GPT-4 (as of March 2023) only achieving a 64–75% accuracy.

Moreover, recognizing the value of text-based explanations, we propose using part-based object descriptors generated by GPT-4 [107] to explain image classification systems. By grounding these texts into specific regions of an object image, we aim to enhance both interpretability and performance. This approach is exemplified by PEEB [118], a part-based image classifier also based on transformer. PEEB translates class names into descriptive texts, matching visual parts to these descriptions for improved classification. It significantly outperforms existing explainable models,

particularly in zero-shot settings, and allows for classifier customization without retraining, thereby advancing both interpretability and accuracy in fine-grained image classification.

## Acknowledgments

Completing my Ph.D. journey has been one of the most challenging and rewarding experiences of my life. It would not have been possible without the support and guidance of many remarkable individuals, to whom I am deeply grateful.

First and foremost, I want to thank my advisor – Professor Anh Nguyen for his unwavering guidance, insightful feedback, and constant encouragement. Your dedication to excellence and profound knowledge have been instrumental in shaping my research and academic growth. I am forever grateful for your mentorship and the numerous hours you devoted to helping me navigate the complexities of this journey.

My sincere appreciation goes to Professors Shubhra Kanti Karmaker, Yang Zhou, Ashish Gupta, and the university reader, Jessica McDonald, for graciously agreeing to be my committee members. Your willingness to review my work and provide critical feedback has been crucial in refining my research. Your expertise and perspectives have enriched my dissertation, and I am deeply thankful for your contributions.

I am immensely grateful to Professor Xiao Qin for his generous support and dedication in handling all the procedures and paperwork associated with my Ph.D. program. Your meticulous attention to detail and willingness to help me understand intricate requirements have been invaluable. Your patience and commitment have significantly eased my path through this challenging endeavor.

I would like to extend my gratitude to my lab members: Michael Alcorn, Naman Bansal, Qi Li, Peijie Chen, Pooyan Rahmanzadehgervi, Giang Nguyen, Tin Nguyen, Hung Nguyen, and Mohammad Reza Taesiri. Your support and valuable feedback during lab meetings have been essential in shaping my research. The collaborative environment and intellectual discussions we shared have significantly contributed to my academic and personal growth.

Special thanks to Dr. Trung Bui, Dr. Long Mai, and Dr. David Seunghyun Yoon for being exceptional collaborators and mentors during my internships at Adobe. The invaluable experience and essential skills I acquired during these internships have prepared me for a successful career in the industry. Your guidance and mentorship have been pivotal in my professional development, and I am profoundly grateful for the opportunities you provided.

To my family, your unwavering support and encouragement have been my pillar of strength throughout this journey. You have stood by me through the stresses and challenges, providing me with the emotional and mental fortitude to persevere. Your faith in my abilities has been a constant source of motivation, and I am deeply thankful for your love and support.

A special and heartfelt thank you to my wife, Phat Nguyen. Your unwavering support, love, and patience have been my anchor during the most difficult times. You have not only taken care of our home but also assisted me in my research work. Your confidence in me has been a source of immense strength, and together we eagerly await the arrival of our daughter, Taylor Pham. There are no words to express how much I appreciate everything you have done for me. I am incredibly grateful for your love and sacrifices, and I look forward to starting the next phase of our lives together with excitement and joy.

This Ph.D. journey has been a collective effort, and I am deeply thankful to each one of you for your contributions. Your support, guidance, and encouragement have been the foundation upon which this achievement stands. Thank you!

## Table of Contents

Abstract . . . . .	ii
Acknowledgments . . . . .	iv
List of Tables . . . . .	xi
List of Figures . . . . .	xvii
1 Introduction . . . . .	1
1.1 Motivations . . . . .	1
1.2 Contributions . . . . .	2
1.3 Organization . . . . .	4
2 Out of Order: How Important is the Sequential Order of Words in a Sentence in Natural Language Understanding Tasks? . . . . .	6
2.1 Overview . . . . .	6
2.2 Related Work . . . . .	8
2.3 Methods . . . . .	10
2.3.1 Datasets . . . . .	10
2.3.2 Classifiers . . . . .	10
2.3.3 Constructing sets of real and shuffled examples for experiments . . . . .	11
2.4 Experiments and Results . . . . .	13
2.4.1 How much is word order information required for solving GLUE tasks? . . . . .	13
2.4.2 How sensitive are models trained to predict the similarity of two sentences? . . . . .	15
2.4.3 How important are words to classification after their context is shuffled? . . . . .	15
2.4.4 If not word order, then what do classifiers rely on to make correct predictions? . . . . .	18
2.4.5 Does increasing word-order sensitivity lead to higher model performance? . . . . .	21
2.5 Discussion and Conclusion . . . . .	23

3	Double Trouble: How to not Explain a Text Classifier’s Decisions using Counterfactuals Synthesized by Masked Language Models? . . . . .	25
3.1	Overview . . . . .	25
3.2	Methods and Related Work . . . . .	27
3.3	Experiment framework . . . . .	30
3.3.1	Three datasets . . . . .	30
3.3.2	Classifiers . . . . .	31
3.3.3	Six evaluation metrics . . . . .	31
3.4	Bias of Deletion metric and its variants . . . . .	33
3.5	No evidence that IM is better than LOO . . . . .	33
3.5.1	Under ROAR and $ROAR_{BERT}$ , IM is on-par with or worse than $LOO_{empty}$ . . . . .	34
3.5.2	$LOO_{empty}$ aligns significantly better with human annotations than IM . . . . .	35
3.5.3	IM is insensitive to model randomization . . . . .	37
3.5.4	Classification accuracy only drops marginally when one token is deleted . . . . .	38
3.6	Replacing (instead of deleting) <u>multiple</u> words can improve explanations . . . . .	39
3.6.1	$LIME_{BERT}$ attribution maps are <u>not</u> more aligned with human annotations . . . . .	39
3.6.2	$LIME_{BERT}$ consistently outperforms LIME under three ROAR metrics . . . . .	40
3.7	Discussion and Conclusion . . . . .	40
4	PiC: A Phrase-in-Context Dataset for Phrase Understanding and Semantic Search . . . . .	42
4.1	Overview . . . . .	42
4.2	Related Work . . . . .	43
4.3	PiC Dataset Construction . . . . .	46
4.3.1	Data Collection . . . . .	47
4.3.2	Data Annotation . . . . .	47
4.3.3	Annotation Verification . . . . .	48
4.4	Three Phrase Understanding Tasks . . . . .	48
4.4.1	Phrase Similarity (PS) . . . . .	48

4.4.2	Phrase Retrieval (PR)	49
4.4.3	Phrase Sense Disambiguation (PSD)	49
4.5	Experiments and Results	50
4.5.1	Phrase Similarity: Contextualized phrase embeddings improve accuracy	50
4.5.2	Human Baselines and Upperbound (95% Exact Match) on Phrase Retrieval	52
4.5.3	Phrase Retrieval: In ranking, context only helps BERT embeddings but not others	53
4.5.4	Phrase Retrieval: Span-selection models reach near-human accuracy	55
4.5.5	Phrase Sense Disambiguation: Best models also perform poorly	55
4.6	Discussion and Conclusion	57
4.7	Limitations	57
5	PEEB: Part-based Image Classifiers with an Explainable and Editable Language Bottleneck	58
5.1	Overview	58
5.2	Related Work	60
5.3	Datasets	62
5.3.1	Test classification benchmarks	62
5.3.2	Bird-11K dataset construction	63
5.3.3	Dataset splits for contrastive pre-training	63
5.4	Method	65
5.4.1	Backbone: OWL-ViT object-part detector	65
5.4.2	PEEB classifier	65
5.4.3	Training strategy	66
5.5	Experiments & Results	67
5.5.1	CLIP-based classifiers rely mostly on {class names} ( <u>not</u> descriptors)	67
5.5.2	Pre-trained PEEB outperforms CLIP-based classifiers in GZSL	69
5.5.3	PEEB is superior to text descriptor-based classifiers in GZSL on CUB-200	70



5.5.4	PEEB generalizes to traditional ZSL . . . . .	70
5.5.5	Finetuning the pre-trained PEEB on CUB-200 yields a competitive explainable classifier in supervised learning . . . . .	72
5.5.6	Applying PEEB to 🐕 dog identification . . . . .	74
5.6	Discussion and Conclusion . . . . .	75
5.7	Limitations . . . . .	76
6	Appendix . . . . .	77
6.1	Supplementary Materials for Out of Order (Chapter 2) . . . . .	77
6.1.1	Self-attention layers that match question-words to similar words in the answer	77
6.2	Supplementary Materials for Double Trouble (Chapter 3) . . . . .	85
6.2.1	IM explanations have smaller attribution magnitude per token and lower word coverage . . . . .	85
6.2.2	By design, IM always assigns near-zero attribution to high-likelihood words regardless of classifiers . . . . .	86
6.2.3	Train BERT as masked language model on SST-2 to help filling in missing words . . . . .	87
6.2.4	Comparison between original and modified version of Input Marginalization	88
6.2.5	Sanity check result . . . . .	88
6.3	Supplementary Materials for Phrase in Context (Chapter 4) . . . . .	98
6.3.1	Training models on Phrase Similarity . . . . .	98
6.3.2	Training SS models on Phrase Retrieval . . . . .	98
6.3.3	Data collection . . . . .	99
6.3.4	Statistics for search queries in Yahoo Search Query dataset . . . . .	102
6.3.5	Verification of Phrase Similarity . . . . .	102
6.3.6	Quantitative results on PR-page . . . . .	106
6.3.7	Finetuning on PSD does not substantially improve accuracy . . . . .	106

6.3.8	SS-style training improves <u>non</u> -contextualized but not contextualized phrase embeddings . . . . .	108
6.3.9	Qualitative examples for PS, PR-pass, PR-page and PSD . . . . .	110
6.3.10	Verifying annotations . . . . .	117
6.3.11	Data Sheet . . . . .	120
6.3.12	Evaluation with ChatGPT-4 . . . . .	122
6.4	Supplementary Materials for PEEB (Chapter 5) . . . . .	126
6.4.1	Architecture details . . . . .	126
6.4.2	Model and dataset notations . . . . .	136
6.4.3	Generating part-based descriptors . . . . .	136
6.4.4	Datasets . . . . .	138
6.4.5	Additional results . . . . .	140
6.4.6	Study on GPT-4 generated descriptors . . . . .	150
6.4.7	Qualitative Inspections . . . . .	153
	References . . . . .	158

## List of Tables

2.1	A real question on Quora (QQP dataset) and its three modified versions ( $Q_3$ to $Q_1$ ) created by randomly shuffling 3-grams, 2-grams, and 1-grams, respectively. $Q_s$ was created by swapping two random nouns. . . . .	12
2.2	All results (a–c) are reported on the GLUE dev-r sets i.e. 100% accuracy (a). Shuffling n-grams caused the accuracy to drop (b) the largest for CoLA and the least for QNLI. Each row is computed by averaging the results of 3 BERT-based models and 10 random shuffles. From top to bottom, the Word-Order Sensitivity (WOS) is sorted descendingly (c) and is consistent across three types of n-grams i.e. WOS scores decrease from top down and from left to right. In contrast, the StructBERT results (d), taken from Table 1 and 4 in [158], showed inconsistent improvements across different tasks. STS-B results are in scaled Spearman correlation. In addition to small accuracy drops, the mean confidence scores of all classifiers—reported in parentheses e.g. “(0.93)”—also changed marginally after words are shuffled (a vs. b). . . . .	17
2.3	Zero-ing out a set of 15 “word-matching” self-attention matrices (identified via the procedure in Sec. 2.4.4) caused a substantial drop of $\sim 25\%$ in accuracy (d) while the random baseline is 50%. These 15 matrices played an important role in QNLI because ablating 15 random matrices only caused a $\sim 1\text{-}3\%$ drop in accuracy. . . . .	21
2.4	With finetuning on synthetic tasks, all of our models (b) have a larger drop in accuracy on shuffled dev-s examples, compared to the standard RoBERTa-based classifiers (a). That is, our models are substantially more sensitive to word-order randomization (i.e. +148% to +194% in WOS scores). . . . .	22
2.5	Finetuning the pretrained RoBERTa on synthetic tasks (before finetuning on the downstream tasks) improved model dev-set performance on SQuAD 2.0 (b) and all the tested tasks in GLUE (a), except SST-2. . . . .	23
3.1	IM is the <b>best</b> method under $\text{Deletion}_{\text{BERT}}$ , as reported in [78], but the <b>worst</b> under Deletion. Both metrics measure AUC (lower is better). . . . .	34
3.2	Dev-set mean accuracy (%) of 5 models trained on the new SST-2 examples where $N\%$ of highest-attribution words per example are removed (i.e. ROAR) or replaced via BERT (i.e. $\text{ROAR}_{\text{BERT}}$ ). On average, under both metrics, $\text{LOO}_{\text{empty}}$ (a) is slightly better, i.e. lower mean accuracy, than IM (b). Notably, $\text{LOO}_{\text{empty}}$ statistically significantly outperforms IM under ROAR at $N = 10\%$ (2-sample $t$ -test; $p = 0.037$ ) (d). Both $\text{LOO}_{\text{empty}}$ and IM substantially outperform a random baseline (c) that considers $N\%$ random tokens important. . . . .	35

3.3	Compared to IM, $LOO_{empty}$ is substantially more consistent with human annotations over all three datasets. Note that the gap between $LOO_{empty}$ and IM is $\sim 3\times$ wider when comparing AMs with the e-SNLI tokens that at least three annotators label important (i.e. L3), compared to L2 (higher is better). $LIME_{BERT}$ explanations are slightly less consistent with human highlights than those of LIME (a) despite their counterfactuals are more realistic. . . . .	35
3.4	The dev-set accuracies on SST, e-SNLI and MultiRC (87.83%, 90.92%, and 69.10%, respectively) only drop marginally when a single token is deleted (a) or replaced using BERT (b). In contrast, LIME samples cause the classification accuracy to drop substantially (e.g. 16.38 points on SST). . . . .	39
4.1	Our Phrase Similarity (PS) dataset has a lower percent of lexical-overlap instances and is the <u>only</u> human-annotated dataset that provides phrases, each in a context sentence. . . . .	45
4.2	Our PR-pass, PR-page and PSD datasets are smaller in size compared to common QA datasets and contain shorter queries that are noun phrases instead of questions. However, our tasks require searching in much longer documents. . . . .	45
4.3	Accuracy (%) of state-of-the-art BERT-based models on the PS test set. Contextualized phrase embeddings (“Phrase + Ctx”) yield substantially higher performance on PS than non-contextualized embeddings (“Phrase”). The random baseline is 50%. . . . .	52
4.4	Best SS models reach near the Upperbound (95%) on PR-pass. Yet, ranking models based on phrase embeddings significantly <u>underperform</u> SS models. . . . .	53
4.5	<b>Ranking</b> accuracy (%) on <b>PR-pass</b> using the state-of-the-art pretrained phrase embeddings. See section 6.3.6 for the results on PR-page. $\Delta$ (e.g. <b>-3.62</b> ) denotes the differences between the Top-1 accuracy in the contextualized (“Phrase + Context”) vs. the non-contextualized (“Phrase”) setting. . . . .	54
4.6	<b>Test-set</b> performance (%) of <b>SS models</b> on PR-pass (a), PR-page (b), and PSD (c). When trained on PR-pass (a) and PR-page (b), SotA SS models perform well. However, testing the PR-pass-trained models on PSD shows a significant drop in accuracy (c). That is, SotA SS models tend to understand a <u>single sense</u> of a phrase in context well (high PR-pass, PR-page, and PSD EM scores). Yet, they are not able to differentiate two senses of the same phrase (e.g., here, PhraseBERT accuracy drops <b>-41.27</b> points between EM+loc vs. EM scores on PSD). . . . .	56
5.1	Top-1 test accuracy (%) on CUB-200 when using original, correct (a) vs. randomized, wrong descriptors (b). See fig. 5.4 for an example of the descriptors. . . . .	69
5.2	In the <b>GZSL</b> setting, PEEB outperforms CLIP and M&V by a large margin, from <b>+8</b> to <b>+29</b> pp in top-1 accuracy (see section 5.5.3). PEEB is also $\sim 10\times$ better than the other two models when class names are replaced by <b>scientific names</b> . As PEEB does not use class names, its accuracy remains unchanged when class names are changed into the scientific ones. . . . .	69

5.3	PEEB achieves SOTA CUB-200 accuracy among the <b>text descriptor-based</b> classifiers in GZSL. * <u>1-shot learning</u> . † <u>k-means with <math>k = 32</math></u> . . . . .	71
5.4	PEEB consistently outperforms other vision-language methods under Harmonic mean and especially in the hard split (SCE) by (+5 to +15) points, highlighting its generalization capability on ZSL. . . . .	72
5.5	PEEB is a state-of-the-art, explainable CUB-200 🐦 classifiers in the <b>supervised</b> learning. . . . .	73
5.6	In the <b>supervised</b> learning setting, PEEB is the state-of-the-art explainable, Stanford Dogs-120 🐕 classifiers and competitive w.r.t. SOTA black-box models. . . . .	75
6.1	The number of QNLI examples where we found $\geq$ one self-attention matrix that the most strongly attends to three pairs of <u>matching</u> words when given the dev-s examples i.e. (modified question, real answer) (a) or when given both the shuffled and real examples (b). In other words, <b>the numbers in (b) denote the number of examples where (1) there exist <math>\geq 3</math> words, regardless of its word order, in the question that can be found in the accompanying real answer; and (2) these correspondences are captured by at least one self-attention matrix.</b> The sum edit-distance for all 3 pairs of words are less than $N$ where $N = \{0, 1, 2, 3, 4, 15\}$ (left column). . . . .	79
6.2	All results (a–c) of RoBERTa <sub>base</sub> and RoBERTa <sub>large</sub> models finetuned on the combination of NLI datasets (SNLI, MNLI, FEVER and ANLI) are reported on the ANLI dev-r sets (i.e. 100% accuracy) which includes A1, A2 and A3 (a). The accuracies for RoBERTa <sub>base</sub> and RoBERTa <sub>large</sub> on ANLI are 51.19% and 56.98%, respectively. Each row is computed by averaging the results of 10 random shuffles. Word-Order Sensitivity (WOS) of ANLI and its subsets (c). Since ANLI is 3-way classification task, the baseline is 33.33% (as described in Sec 4.3). . . . .	80
6.3	<b>If the top-1 most important word in an SST-2 5/5 example has a positive meaning, then there is a 100% chance that the sentence is labeled positive in SST-2.</b> Similarly, the conditional probability of a sentence being labeled negative given a negative most important word (by LIME [134]) is 94.44%. . . . .	80
6.4	The number of examples per class before (a) and after each of the three filtering steps to produce dev-r sets (described in Sec. 2.3.3) for RoBERTa-based classifiers. For each task, we repeated the same procedure for three sets of classifiers, for BERT-, RoBERTa-, ALBERT-based classifiers, respectively. . . . .	83
6.5	Accuracy of all models on dev-s examples (created by shuffling n-grams and swapping 2 nouns) and their Word-Order Sensitivity scores ( $\in [0, 1]$ ) across seven GLUE tasks. STS-B is a regression task and thus not comparable in word-order sensitivity with the other tasks, which are binary classification. . . . .	84

6.6	The dev-set performance of models finetuned from three different BERT “base” variants (12 self-attention layers and 12 heads) and one RoBERTa “large” model (24 self-attention layers and 16 heads) on seven GLUE tasks. These results match either those reported by original papers, [67] or GLUE leaderboard. . . . .	84
6.7	The average absolute value of attribution scores per token of $LOO_{empty}$ is consistently higher than that of IM. . . . .	85
6.8	Compared to IM, the coverage of $LOO_{empty}$ is closer to the coverage of human explanations. . . . .	86
6.9	Top-1 likelihood scores (c) are the mean likelihood given by BERT for the top-1 predicted words that exactly match the original words (a). . . . .	87
6.10	The approximation in of IM-top10 compared to the original IM under two metrics on SST-2 task. Both metrics measure AUC (lower is better). . . . .	88
6.11	The percentage (%) of token (a) whose attribution scores change signs and (b) the average of absolute differences in attribution magnitude after classifier randomization (higher is better). IM is consistently more insensitive than $LOO_{empty}$ in both SST-2 and e-SNLI. . . . .	90
6.12	Dev-set mean accuracy (%) of 5 models trained on the new SST-2 examples where $N\%$ of highest-attribution words per example are removed (i.e. ROAR), replaced via BERT (i.e. $ROAR_{BERT}$ ) or BERT finetuned on SST-2 to fill in a [MASK] token (i.e. $ROAR_{BERT\_SST2}$ ). The original accuracy when no tokens are removed (i.e. $N = 0\%$ ) is $92.62 \pm 0.30$ . On average, under three metrics, $LIME_{BERT}$ (b) and $LIME_{BERT\_SST2}$ (c) are better, i.e. lower mean accuracy, than LIME (a). . . . .	90
6.14	Statistics of Yahoo queries across different query types. . . . .	102
6.13	Summary of our 3-stage data construction. $p, s, m, d, q, l$ denote target phrase, sentence, metadata, document, query, and label, respectively. . . . .	105
6.15	<b>Ranking</b> accuracy (%) on <b>PR-page</b> using the state-of-the-art pretrained phrase embeddings (a) and those finetuned on PR-pass via SS-style training (b). . . . .	106
6.16	Performance of <b>SS models</b> on 3,000 PSD <b>test</b> examples. (a) and (b) models are <b>finetuned</b> only on PR-pass and 1,938 PSD examples (PSD-2K), respectively. (c) models are finetuned on PR-pass first and then finetuned on PSD-2K. All models are “base” unless otherwise specified. The definitions of EM+loc and $F_1$ +loc are in Table 4.6’s caption. . . . .	107
6.17	<b>Ranking</b> accuracy (%) on <b>PR-pass</b> using the state-of-the-art pretrained phrase embeddings (a) and those finetuned on PR-pass via SS-style training (b). See Section 6.3.6 for the results on PR-page. $\Delta$ (e.g. <b>-3.62</b> ) denotes the differences between the Top-1 accuracy in the contextualized (“Phrase + Context”) vs. the non-contextualized (“Phrase”) setting. . . . .	109

6.18	<b>Zeroshot</b> accuracy (%) on <b>PS</b> using the state-of-the-art multimodal large language model ChatGPT-4. $\Delta$ (e.g. <b>-24.80</b> ) denotes the differences between the Top-1 accuracy in the contextualized (“Phrase + Context”) vs. the non-contextualized (“Phrase only”) setting. . . . .	124
6.19	<b>Zeroshot</b> accuracy (%) on <b>PSD</b> using the state-of-the-art multimodal large language model ChatGPT-4. $\Delta$ (e.g. <b>-24.80</b> ) denotes the differences between the Top-1 accuracy in the contextualized (“Phrase + Context”) vs. the non-contextualized (“Phrase only”) setting. . . . .	125
6.20	Pre-training details of our pre-trained models. . . . .	135
6.21	Details of our finetuned models. . . . .	135
6.22	Three pre-training splits for PEEB. . . . .	136
6.23	Number of images and species of different bird datasets. Our proposed dataset Bird-11K includes almost all avians on Earth. . . . .	138
6.24	Stress test results on CUB and NABirds datasets. Despite the ZSL challenge, our method consistently surpasses CLIP and M&V. This underscores the robust generalization of our approach, which leverages descriptors for classification. . . . .	141
6.25	The table showcases the classification accuracies on iNaturalist as we vary the noise levels. The data underscores that the performance disparity on iNaturalist is predominantly due to image noise. While all methods improve with cleaner images, our model exhibits the most substantial gains, particularly in the least noisy sets. . . . .	142
6.26	Classification accuracy on the CUB test set that uses a different number of parts. Performance dips significantly with just one part, especially for the least visible ones. Maximum accuracy is reached with all 12 parts. The last row of the table also shows the accuracy of [95] method which employs a different number of parts. It is evident that their method is insensitive to the number of parts used, which may not reflect a realistic scenario. . . . .	145
6.27	Impact of Training on Classification Accuracies: Untuned PEEB yields 0.55% on CUB and 0.31% on NABirds, almost mirroring random chance. With training (PEEB <sub>[-test]</sub> ), accuracy surges by <b>+63.78</b> points on CUB and <b>+68.72</b> points on NABirds. . . . .	146
6.28	Error rate of Box Prediction in Failure and Success Cases. We report the box prediction error rate, depending on whether the prediction box includes ground truth key points. No major difference is found between them, which means the failure is largely due to the part-descriptor mismatch. . . . .	147
6.29	Model evaluation on CUB and NABirds test sets. We evaluate the predicted boxes on two <i>ground-truth</i> sets; (1) predicted boxes from OWL-ViT <sub>Large</sub> as ground-truths, and (2) OWL-ViT <sub>Large</sub> ’s boxes that include the human-annotated key points. Our method has slightly lower performance in terms of mean IoU but comparable precision. . . . .	149

6.30	Prompt for GPT-4V evaluation on CUB where {list_of_200_classes} is the placeholder for the actual 200 CUB classes while {descriptors} (see an example in Section 6.4.3) is the placeholder for the actual descriptors associated with a given bird image from the CUB test set. . . . .	150
6.31	Summary of manual inspection results for 20 classes, highlighting the need for revision in GPT-4 generated descriptors. An average error rate of 45% indicates substantial room for improvement. . . . .	151
6.32	The revised descriptors result in +0.8 for PEEB <sub>[-test]</sub> in CUB. In particular, the average improvement among the 17 revised classes is +10.8, hinting at the large potential of our proposed model. . . . .	152



## List of Figures

- 2.1 A RoBERTa-based model achieving a 91.12% accuracy on QQP, here, correctly labeled a pair of Quora questions duplicate (a). Interestingly, the predictions remain unchanged when all words in question  $Q_2$  is randomly shuffled (b–c). QQP models also often incorrectly label a real sentence and its shuffled version to be duplicate (d). We found evidence that GLUE models rely heavily on words to make decisions e.g. here, “marijuana” and “cancer” (more important words are highlighted by LIME). Also, there exist self-attention matrices tasked explicitly with extracting word-correspondence between two input sentences regardless of the position of those words. Here, the top-3 pairs of words assigned the highest self-attention weights at (layer 0, head 7) are inside red, green, and blue rectangles, respectively. . . . . 6
- 2.2 The distribution of similarity scores over 6 ranges for the (real, shuffled) pairs in dev-s (green) is highly similar to that for (real, real) STS-B pairs in dev-r (red). The statistics in each range were computed over 3 models (BERT, RoBERTa, and ALBERT). . . . 16
- 2.3 An original SST-2 dev-set example (S) and its three shuffled versions ( $S_1$  to  $S_3$ ) were all correctly labeled positive by a RoBERTa-based classifier with high confidence scores (right column). . . . . 18

2.4 A RoBERTa-based model’s correct prediction of entailment on the original input pair (Q, A) remains unchanged when the question is randomly shuffled ( $Q_1$  &  $Q_2$ ) or when two random nouns in the question are swapped ( $Q_s$ ). The salient words in the questions e.g. **manage** and **missions** remain similarly important after their context has been shuffled. Also, the classifier harnessed self-attention to detect the correspondence between similar words that appear in both the question and the answer e.g. **manage** (Q) and **managed** (A). That is, the top-3 pairs of words that were assigned the largest question-to-answer weights in a self-attention matrix (layer 0, head 7) are inside in the **red**, **green**, and **blue** rectangles. . . . . 19

3.1 **By design, IM erroneously assigns near-zero attribution to highly-predictable words.** Color map: **negative -1**, neutral 0, **positive +1**. Many words labeled **important** by humans such as **“stepping”**, **“stone”** (a) or **“hot”**, **“air”** (b) are always given near-zero attribution by IM (because they are highly predictable by BERT, e.g. 0.9793 for **stepping**) regardless of the classifier. Even when randomizing the classifier’s weights three times, the IM attribution of these words remains unchanged at near zero ( $IM_1$  to  $IM_3$ ). Therefore, when marginalizing over the top- $k$  BERT candidates (e.g., “stepping”, “rolling”, “casting”), the IM attribution for low-entropy words tends to zero, leading to heatmaps that are biased, less accurate, and less plausible than  $LOO_{empty}$ . . . . . 26

3.2  $LOO_{empty}$  binarized **attribution maps** align better with **human** highlights than **IM maps**. 36

3.3  $LOO_{empty}$  important **words** are in a stronger agreement with **human** highlights than **IM important words**. Each e-SNLI example contains a pair of premise (P) and hypothesis (H). . . . . 37

3.4  $LIME_{BERT}$  slightly, but consistently outperforms LIME when evaluated under either ROAR or  $ROAR_{BERT}$ . The each point in the  $y$ -axis shows the mean accuracy of five different classifiers. See more results supporting the same conclusion in Table 6.12. . . 41

4.1	Given a phrase, two associated Wikipedia pages, and expert annotations, i.e. answers to Q1, Q2, and Q3 (a), we are able to construct <u>two</u> pairs of positive and negative examples for PS (b), a PR-pass example (c), a PR-page example (d), and a PSD example <u>only if</u> the answer to Q3 is No (e). . . . .	44
5.1	Existing explanations are either (a) textual but at the image level; or (b) part-level but not textual. Combining the best of both worlds, PEEB (c) first matches each detected object part to a text descriptor, then uses the part-level matching scores to classify the image. . . . .	59
5.2	Given an input image (a) from an unseen class of Eastern Bluebird, PEEB misclassifies it into <b>Indigo Bunting</b> (b), a visually similar blue bird in CUB-200 (d). To add a new class for Eastern Bluebird to the 200-class list that PEEB considers when classifying, we clone the <b>12 textual descriptors</b> of Indigo Bunting (b) and <b>edit (- -&gt;)</b> the descriptor of <b>thorat</b> and <b>wings</b> (c) to reflect their identification features described on AllAboutBirds.org (“Male Eastern Bluebirds are vivid, deep blue above and rusty or brick-red on the throat and breast”). After the edit, PEEB correctly predicts the input image into <b>Eastern Bluebird</b> (softmax: 0.0445) out of 201 classes (c). That is, the dot product between the <b>wings</b> text descriptor and the same orange region increases from <b>0.57</b> to <b>0.74</b> . . . . .	61
5.3	During inference, 12 visual part embeddings with the highest cosine similarity with encoded part names are selected ( <b>a</b> ). These visual part embeddings are then mapped ( <b>—&gt;</b> ) to bounding boxes via Box MLP. Simultaneously, the same embeddings are forwarded to the Part MLP and its outputs are then matched ( <b>b</b> ) with textual part descriptors to make classification predictions ( <b>—&gt;</b> ). fig. 6.35 shows a more detailed view of the same process. . . . .	64

- 5.4 With original descriptors, M&V [95] correctly classifies the input image into **Blue Jay** (a). Yet, interestingly, when **randomly** swapping the descriptors of this class with those of other classes (b), M&V’s top-1 prediction remains unchanged, suggesting that the class names in the prompt (e.g., “A photo of {class name}”) have the most influence over the prediction (not the expressive descriptors). In contrast, PEEB changes its top-1 prediction from **Blue Jay** (c) to **Least Tern** (d) when the descriptors are **randomized**. . . . . 68
  
- 5.5 PEEB classifies this Dogs-120 image into **Alaskan Malamute** (softmax: 0.199) due to the matching between the image regions and associated textual part descriptors. In contrast, the explanation shows that the input image is not classified into Cairn Terrier mostly because its ears and body regions do not match the text descriptors, i.e., dot products are 0.000 and 0.000, respectively. See Section 6.4.7 for more qualitative examples. . . . . 74
  
- 6.1 Among 144 self-attention matrices in the RoBERTa-based classifier finetuned for QNLI, there are 15 “word-matching” matrices (a) that explicitly attend to pairs of similar words that appear in both questions and answers regardless of the order of words in the question (see example pairs in Fig. 2.4). For each QNLI example, we identified one such matrix that exhibits the matching behavior the strongest (a). 92% of the task of attending to duplicate words is mostly handled in the first three layers (b). . . . . 78
  
- 6.2 Each CoLA example contains a single sentence. Here, we shuffled the words in the original sentence (S) five times to create five new sentences (S<sub>1</sub> to S<sub>5</sub>) and fed them to a RoBERTa-based classifier for predictions. Words that are important for or against the prediction (by LIME [134]) are in **orange** and **blue**, respectively. Most of the shuffled examples were classified into unacceptable label (i.e. grammatically incorrect) with even higher confidence score than the original ones. . . . . 81

6.3 Each MRPC example contains a pair of sentences i.e. (A, B). Here, we shuffled the words in the original sentence (A) to create modified sentences (A<sub>1</sub> & A<sub>2</sub>) and fed them together with the original second sentence (B) to a RoBERTa-based classifier for predictions. Also, the classifier harnessed self-attention to detect the correspondence between similar words that appear in both sentences. That is, the top-3 pairs of words that were assigned the largest cross-sentence weights in a self-attention matrix (layer 0, head 7) are inside in the red, green, and blue rectangles. . . . . 81

6.4 Each RTE example contains a pair of premises and hypotheses i.e. (P, H). We shuffled the words in the original hypothesis H to create modified hypotheses (H<sub>1</sub> & H<sub>2</sub>) and fed them together with the original premise (P) to a RoBERTa-based classifier for predictions. Also, the classifier harnessed self-attention to detect the correspondence between similar words that appear in both the premise and hypothesis. That is, the top-3 pairs of words that were assigned the largest premise-to-hypothesis weights in a self-attention matrix (layer 0, head 7) are inside in the red, green, and blue rectangles. 82

6.5 Color map: negative -1, neutral 0, positive +1. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.988$ . . . . . 89

6.6 Color map: negative -1, neutral 0, positive +1. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.917$ . . . . . 89

6.7 Color map: negative -1, neutral 0, positive +1. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.983$ . . . . . 89

6.8 Color map: negative -1, neutral 0, positive +1. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.802$ . . . . . 89

6.9 Color map: negative -1, neutral 0, positive +1. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.950$ . . . . . 89

6.10 BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled important by humans in the SST positive example. In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods. . . . . 90

6.11 BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled important by humans in the SST negative example. In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods. . . . . 91

6.12 BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled important by humans in the e-SNLI entailment example which contains a pair of premise (P) and hypothesis (H). In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods. . . . . 91

6.13 BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled important by humans in the e-SNLI neutral example which contains a pair of premise (P) and hypothesis (H). In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods. . . . . 91

6.14 BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled important by humans in the MultiRC True example which contains a triplet of paragraph (P), question (Q) and answer (A). In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods. . . . . 92

6.15 BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled important by humans in the MultiRC False example which contains a triplet of paragraph (P), question (Q) and answer (A). In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods. . . . . 93

6.16 The set of explanatory words given by LOO<sub>empty</sub> covers 75% of human highlights with higher precision and IoU in the SST negative example while there are a half of tokens highlighted by IM are in correlation with human explanations. . . . . 93

6.17 The set of explanatory words given by LOO<sub>empty</sub> covers all highlights (higher precision and IoU) that are important to human in the e-SNLI contradiction example which contains a pair of premise (P) and hypothesis (H) while there are a half of tokens highlighted by IM are in correlation with human explanations. . . . . 94

6.18 The set of explanatory words given by LOO<sub>empty</sub> covers all highlights (higher precision and IoU) that are important to human in the e-SNLI neutral example which contains a pair of premise (P) and hypothesis (H) while there are none tokens highlighted by IM are in correlation with human explanations. . . . . 94

6.19 The set of explanatory words given by LOO<sub>empty</sub> covers 95% of human highlights with higher precision and IoU in the MultiRC True example which contains a triplet of paragraph (P), question (Q) and answer (A) while there are only few tokens given by IM are in correlation with human explanations. . . . . 95

6.20 The set of explanatory words given by LOO<sub>empty</sub> covers two thirds of human highlights with higher precision and IoU in the MultiRC False example which contains a triplet of paragraph (P), question (Q) and answer (A) while there are two tokens given by IM are in correlation with human explanations. . . . . 96

6.21	When a word is <b>removed</b> , the predicted labels of all resulting sentences ( $S_1$ to $S_7$ ) are still positive with a confidence score of 1.0. . . . .	96
6.22	The removal of each token in both premise and hypothesis in e-SNLI example which contains a pair of premise (P) and hypothesis (H) <b>infrequently change the prediction</b> . Specifically, only the example of ( $P_2$ , $H_2$ ) shifted its prediction to neutral while the remaining partially-removed examples do not change their original prediction with high confidence score in parentheses. . . . .	97
6.23	A decision tree describing our verification process for PS that involves three experts. Red, green and blue cells represent Reject, Accept decision and Questions. The numbers of examples for each branch are shown in parentheses. . . . .	103
6.24	A decision tree describing our verification process for PS that involves three experts. Red, green and blue cells represent Reject, Accept decision and Questions. The numbers of examples for each branch are shown in parentheses. . . . .	104
6.25	PhraseBERT-based classifier <b>correctly</b> predicts positive given two phrases $P_1$ and $P_2$ with and without the presence of context $S_1$ and $S_2$ . Here, to humans, the phrases are non-polysemous and have the same meaning. . . . .	110
6.26	PhraseBERT-based classifier <b>correctly</b> predicts negative given two phrases $P_1$ and $P_2$ with and without the presence of context $S_1$ and $S_2$ . Here, to humans, the two phrases are non-ambiguously carrying different meanings. . . . .	110
6.27	PS case that requires context to determine similarity. Without context, a PhraseBERT-based classifier incorrectly thinks $P_1$ and $P_2$ are different. Yet, it changes the prediction to positive, i.e. thinking two phrases have the same meaning, when the context is taken into account. . . . .	110



- 6.28 PS case that requires context to determine similarity. Without context, PhraseBERT-based classifier incorrectly thinks  $P_1$  and  $P_2$  carry the same meaning. Yet, it correctly changes the prediction to negative when the context is taken into account. . . . . 111
  
- 6.29 Given document  $d$ , our Longformer<sub>Large</sub> SS model trained on PR-pass correctly retrieves **storage needs** in the second paragraph for the query  $q_1$  “storehouse purposes” but fails to retrieve the answer when the query  $q_2$  is “data caching”. The predicted answer for  $q_2$  should be **storage needs** (i.e. in the first passage) since this phrase relates to caching data digitally in computers while **storage needs** refers to physically storing objects. . . 112
  
- 6.30 Given document  $d$ , Longformer<sub>Large</sub> model trained with SS approach on PR-pass correctly retrieves **massive figure** in the second paragraph for the query  $q_2$  “giant number” but fails to retrieve the answer when the query  $q_1$  is “huge model”. The predicted answer for  $q_1$  should be **massive figure** in the first passage since this phrase relates to a physical shape instead of a number. . . . . 113
  
- 6.31 Given document  $d$ , Longformer<sub>Large</sub> model trained via the SS approach on PR-pass correctly retrieves **unrivalled power** in the first paragraph for the query  $q_2$  “indomitable strength” but fails to retrieve the answer when the query  $q_1$  is “incomparable energy”. The predicted answer for  $q_1$  should be **unrivalled power** in the second passage since the second passage changes “unrivalled power” meaning to a competition strength instead of military power. . . . . 114
  
- 6.32 A **ranking** model based on the phrase embeddings of the PR-pass-trained PhraseBERT SS model correctly ranks and retrieves the most semantically relevant answer “common thought” as the top-1 prediction in the retrieval list  $R$  for the query “prevalent theory” in a PR-pass example (which contains a document  $d$  and a query  $q$ ). . . . . 115

6.33 A **ranking** model based on the non-contextualized embeddings of USE-v5 fails to retrieve the correct answer “continued risk” for the query “sustained threat” in the PR-page example (which contains a document  $d$  and a query  $q$ ). The top-5 phrases retrieved (R) contains the word “threat” but have no identifier conveying the “continued” or ‘sustained’ sense. Here, the Wikipedia page is truncated to fit into a single manuscript page. . . . . 116

6.34 Gorilla layouts shown to MTurkers to verify annotations in the first round. . . . . 119

6.35 During the test time using PEEB, we perform 2 steps. **Step 1:** (a) Encode an input image and texts (i.e. 12 part names) by the image and text encoder to get patch embeddings  $p_i$  and text embeddings  $t'_i$ . (b) Feed  $p_i$  to Linear Projection to get  $p'_i$  in the same dimensional space with  $t'_i$  and compute dot product between  $\{p'_i\}$  and  $\{t'_i\}$ . (c)  $\arg \max$  over  $m$  embeddings to select 12 part embeddings. **Step 2:** (a) Encode input texts (i.e. N sets of 12-part descriptors) with the same text encoder to get  $t_i$ . (b) Feed the selected part embeddings to Box MLP to localize parts (in center format). (c) Also feed the selected part embeddings to Part MLP to get  $s_i$  in the same dimensional space with  $t_i$  (d) Compute a dot product between  $\{s_i\}$  and  $\{t_i\}$ , then diagonal sum for each class and  $\arg \max$  over logits to get predicted label  $\hat{y}$ . . . . . 128

6.36 In pre-training stage 1, the objective is to let the Image Encoder learn the general representation of different parts of the birds. Therefore, in pre-training stage 1, we train the *Image Encoder* and *Part MLP* contrastively. During the training, the **Step 1** utilizes a teacher model (OWL-ViT<sub>B/32</sub>) to help PEEB select 12 part embeddings. In **Step 2**, we update the model with symmetric Cross-Entropy loss. Here’s the flow of **Step 1**: (1a) We utilize the teacher model to encode 12 part names and the image to derive the text embedding  $t'_i$ , and the patch embedding  $p_i$ . (1b) Then the patch embeddings  $p$  is forwarded to Linear Projection to obtain  $p'$ , matching the dimension of  $t'$ . (1c) We compute the dot product between  $p$  and  $t'$  and apply *argmax* over  $p$  to derive 12 indices. In **Step 2**: (2a), We first encode the descriptors and the image with the *Text Encoder* and *Image Encoder* to obtain descriptor embeddings  $t$  and patch embeddings  $q$ . (2b), Then we select the 12 patch embeddings based on the 12 indices from (1c). (2c), The 12 patch embeddings then forwarded to *Part MLP* to derive  $s$ , which has the same dimension as  $t$ . Then, we compute the similarity matrix for the patch embedding and the descriptor embedding by computing the dot product between  $s$  and  $t$ . (2d), we construct a one-hot encoded matrix based on the descriptors as the ground truth label and minimize the Symmetric Cross-Entropy loss between the similarity matrix in (2c) and the ground truth label. . . . . 133

6.37 In pre-training stage 2, the goal is to eliminate the teacher model to obtain a standalone classifier. Therefore, the targeted components are **Linear Projection** and **Box MLP**. Since these two components are taking care of different functionalities for patch embedding selection and box prediction, respectively, stage 2 training is a multi-objective training. We employ Symmetric Cross-Entropy loss to learn the patch embedding selection and DETR losses to refine the box predictions. In **Step 1**: (1a), We first encode the 12 part names and the image with *Text Encoder* and *Image Encoder* to obtain the text embedding  $t'_i$  and patch embedding  $p_i$ . (1b) Then the patch embeddings  $p$  is projected by Linear Projection to obtain  $p'$ . (1c) We then compute dot product between  $p'$  and  $t'$  and one-hot encode the matrix via the dimension of  $p'$  to obtain the “teacher logits”. In **Step 2**: (2a), We encoder the image with *Image Encoder* to obtain patch embedding  $q_i$ . (2b) The patch embeddings are then being projected by **Linear Projection** to derive  $q'$ . (2c), We compute the dot product between projected patch embeddings  $q'$  and part name embeddings  $t'$  to obtain the similarity matrix. Then, we employ Symmetric Cross-Entropy loss between the similarity matrix and the “teacher logits” derived in (1c). (2d), Meanwhile, we select the 12 part embeddings by taking argmax over  $q'$ . Then, the selected part embeddings are forwarded to **Box MLP** to predict the coordinates of each part. We compute the DETR losses for the predicted coordinates and update the model. . . . . 134

6.38 The class distribution of Dog-140 dataset. The histogram indicates that most classes in Dog-140 have around 1,000 to 2,000 images. . . . . 140

6.39 The CDF plot (a), underscores significant imbalance of the Bird-11K dataset. While the dataset has abundant long-tailed classes, e.g., a striking 80% of the classes contribute to only 13.46% of the entire image count. The plot (b) showcases the correlation between the number of training figures/peeb/classes and the resulting classification accuracy. As the image count grows, there is a noticeable surge in accuracy, which nearly stabilizes upon surpassing 250K images. Additionally, a significant amount of long-tailed data contributes to a +1.5 points boost in accuracy. . . . . 144

6.40 Our predicted boxes (second column) often align closely with those of OWL-ViT<sub>B/32</sub> (third column). However, slight shifts can lead to significant IoU discrepancies. For instance, in the first row, both PEEB and OWL-ViT<sub>B/32</sub> accurately identify the tail. Yet, variations in focus yield a stark IoU contrast of 0.45 versus 0.81. . . . . 154

6.41 Qualitative example of original descriptors vs. randomized descriptors. Upon swapping descriptors randomly, the prediction outcomes from M&V exhibit minimal variations. 155

6.42 Qualitative example of original descriptors vs. randomized descriptors. Since PEEB’s decision is made by the descriptors, the model will try to find the descriptors that best match the image. e.g., in the random descriptors, most parts are blue. . . . . 155

6.43 An example of PEEB explanation. We can see that the descriptors of these two classes are largely similar, but PEEB makes the correct prediction based on the distinctive feature of the forehead in the two classes. . . . . 156

6.44 An example of PEEB explanation. M&V incorrectly classifies it as **red-legged kittiwake** where the **heermann gull** does not have red legs but a red beak. This example shows that CLIP is strongly biased towards some particular descriptors. . . . 156

6.45 An example of PEEB explanation. We can see that when the descriptor does not match the image, the matching score tends to be zero, e.g., *crown: yellowish-green*. The clear differences in scores provide us transparency of the model’s decision. . . . . 156

6.46 An example of PEEB explanation for dogs. Like birds, PEEB first identifies the predefined parts and then matches them to the descriptions. . . . . 157

6.47 An example of PEEB explanation for dogs. Like birds, PEEB first identifies the predefined parts and then matches them to the descriptions. . . . . 157

6.48 An example of PEEB explanation for dogs. Like birds, PEEB first identifies the predefined parts and then matches them to the descriptions. . . . . 157

## Chapter 1

### Introduction

In the era of machine learning (ML) and artificial intelligence (AI), improving interpretability and performance is essential for building trustworthy and efficient AI systems. Enhanced interpretability allows for better understanding and trust in model decisions, while improved performance ensures their effectiveness in real-world applications. Recent advancements in machine learning (ML) have led to significant improvements in natural language understanding (NLU) and image classification. However, several fundamental issues remain unresolved in advancing artificial general intelligence (AGI), particularly regarding the internal workings of models and more challenging evaluation benchmarks.

#### **1.1 Motivations**

First, while transformer-based models like BERT [30] and RoBERTa [93] have achieved superhuman performance on NLU benchmarks such as GLUE [155], it is unknown whether these models actually utilize word order information when performing NLU tasks. Previous studies have shown that BERT captures word-order information in the first three layers [90], but the practical implications of this for NLU tasks are not well understood. This motivates a deeper investigation into the role of word order in model performance.

Second, explaining model decisions through feature attribution remains a challenge, as many Leave-One-Out (LOO) methods often create out-of-distribution inputs that lead to erroneous attribution maps (AMs) [78, 56] or AMs inconsistent with human’s perception of causality [57]. Although Input Marginalization (IM) [78] method aims to generate more realistic counterfactuals by incorporating BERT, their effectiveness has only been tested on limited datasets and metrics.

This calls for a comprehensive evaluation of IM across diverse datasets and metrics to determine its impact on attribution accuracy.

Third, existing NLU benchmarks such as GLUE do not force models to adequately capture the context-dependent meanings of words or phrases, limiting the thorough evaluation of models as they often learn shortcuts to make predictions. Many phrase-similarity benchmarks [112, 150, 6, 180, 173] either lack human verification [157, 175] or contain a high percentage of phrase pairs with lexical overlap [112, 180], reducing their effectiveness in evaluating contextualized phrase embeddings [175]. There is a need for a benchmark that rigorously evaluates phrases in context.

Lastly, natural-language concepts as explanations are often incorporated to many image classifiers to make predictions and improve model interpretability [172, 108, 106, 176]. However, such textual concepts are often compared against *entire image* for classification and it is unknown what image details match a given descriptor [95, 172]. Moreover, current vision-language models [54, 124, 95] require prompts to have a known {class name} (like a special code instead of an expressive, natural description) that matches the input image [136], making the textual explanations redundant and unreliable. Therefore, a new approach is needed to classify images using part-based textual descriptions grounded to specific image regions to enhance model interpretability and potentially improve accuracy.

## 1.2 Contributions

In our investigation of word order sensitivity in NLU models [115], we find that 65% of the groundtruth labels of five out of seven GLUE tasks tested can be predicted even when the words in one sentence of each example are shuffled. Despite BERT embeddings being contextual, the contribution of an individual word to classification remains almost unchanged in some GLUE tasks, even after shuffling surrounding words. For instance, in sentiment analysis (SST-2), the polarity of a single salient word is approximately 60% predictive of the entire sentence’s label. Furthermore, BERT-based models trained on sequence-pair GLUE tasks utilize self-attention heads to find similar



tokens shared between the two inputs. Encouraging RoBERTa-based models to be more sensitive to word order improves performance on SQuAD 2.0 and most GLUE tasks tested, except for SST-2.

Our comprehensive evaluation of Input Marginalization (IM) in feature attribution [116] reveals that the  $\text{Deletion}_{\text{BERT}}$  metric is biased towards IM because both use BERT to replace words, whereas the vanilla Deletion metric favors the  $\text{LOO}_{\text{empty}}$  baseline. We find no evidence that IM outperforms a simple  $\text{LOO}_{\text{empty}}$  on four state-of-the-art AM evaluation metrics, excluding biased metrics like Deletion and  $\text{Deletion}_{\text{BERT}}$ . We argue that IM is not effective in practice because deleting a single word from an input has only a marginal effect on classification accuracy. Moreover, highly predictable words according to a perfect masked language model are always assigned near-zero attribution in IM, regardless of their importance to the classifier. To further test IM, we integrate BERT into LIME to replace multiple words in an input sequence, making LIME counterfactuals more realistic. This technique consistently improved LIME under multiple ROAR-based metrics but not under human annotation comparisons.

To address the limitations of existing benchmarks in contextualized phrase understanding, we create the Phrase-in-Context (PiC) dataset [117], the first human-annotated benchmark for evaluating and training contextualized phrase embeddings. PiC is composed of three tasks: (1) Phrase Similarity (PS), i.e. compare the semantic similarity of two phrases in the same context sentence; (2) Phrase Retrieval (PR), which is divided into PR-pass and PR-page, i.e. from a passage or a Wikipedia page, retrieve a phrase semantically-similar to a given query phrase; and (3) Phrase-Sense Disambiguation (PSD), i.e. find the target phrase  $p$  semantically similar to the query phrase from a 2-paragraph document where  $p$  appears twice, each time in a different context paragraph that provides a *unique* meaning to  $p$ . Unlike existing phrase similarity datasets, PiC requires models to rely on context. State-of-the-art models trained on PR-pass achieve near-human accuracy 92–94% vs. 95% Exact Match (EM) and high scores (84–89% EM) on PR-page, indicating the effectiveness of our training set and learned embeddings for real-world semantic search. However, on PR-pass, using phrase embeddings in a ranking approach yields poor accuracy (~59% EM),

presenting a challenge for future research into learning contextualized phrase embeddings. State-of-the-art models perform relatively well on PR-pass and PR-page but poorly on PSD, and below 70% accuracy on PS for binary classification of phrase similarity given a context sentence.

In the domain of image classification, we develop PEEB [118], a Part-based image classifier that is Explainable and Editable via a natural-language Bottleneck. PEEB classifies images by grounding the textual descriptor of object *parts* provided by humans or GPT-4 (no images needed) to detected parts in the image. We find that CLIP-based classifiers rely heavily on class names in prompts, with accuracy dropping drastically from 53.78% to 5.89% and 5.95% when class names are removed or replaced by scientific names. Our pre-trained PEEB outperforms CLIP-based classifiers by +8 to +29 percentage points in bird classification across CUB-200, NABirds-555, and iNaturalist-1486. PEEB allows for defining new classes in text at test time without further training, offering superior explainability and editability, and outperforms *text concept-based* methods in both generalized zero-shot and zero-shot settings. Compared with other explainable CUB classifiers, PEEB achieves 88.80% top-1 accuracy, on par with the best CUB-200 classifiers (81–87% accuracy) trained via supervised learning. Additionally, PEEB is applicable to multiple domains, achieving 92.20% accuracy on Stanford Dogs-120, outperforming other explainable models and matching state-of-the-art black-box models.

These contributions collectively advance our understanding of ML model behavior, from natural language processing to image classification, by addressing key limitations and proposing novel evaluation frameworks and methodologies.

### 1.3 Organization

This dissertation is structured as follows: Chapter 1 introduces the core challenges, motivations and contributions. Chapter 2 investigates the sensitivity of transformer-based models like BERT and RoBERTa to word order information. It details our experiments on GLUE tasks and discusses the implications of word order for natural language understanding. Chapter 3 comprehensively

evaluates feature attribution methods, specifically focusing on Input Marginalization (IM) and Leave-One-Out (LOO) techniques. This chapter presents our findings on the effectiveness of IM across diverse datasets and metrics, highlighting its impact on attribution accuracy. Chapter 4 addresses the limitations of current benchmarks in contextualized phrase understanding. We introduce the Phrase-in-Context (PiC) dataset, a human-annotated benchmark designed to rigorously evaluate the context-dependent meanings of phrases. This chapter explains the dataset’s construction, validation, and application. Chapter 5 presents PEEB, a part-based image classifier that uses natural-language descriptions for classification. It discusses the development of PEEB, its performance compared to existing models, and its applicability to multiple domains, including novel species classification.

## Chapter 2

### Out of Order: How Important is the Sequential Order of Words in a Sentence in Natural Language Understanding Tasks?

#### 2.1 Overview

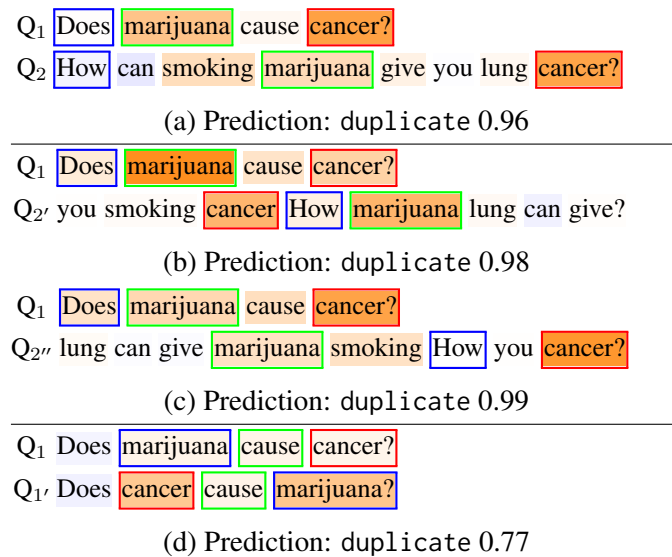


Figure 2.1: A RoBERTa-based model achieving a 91.12% accuracy on QQP, here, correctly labeled a pair of Quora questions duplicate (a). Interestingly, the predictions remain unchanged when all words in question Q<sub>2</sub> is randomly shuffled (b–c). QQP models also often incorrectly label a real sentence and its shuffled version to be duplicate (d). We found evidence that GLUE models rely heavily on words to make decisions e.g. here, “marijuana” and “cancer” (more important words are highlighted by LIME). Also, there exist self-attention matrices tasked explicitly with extracting word-correspondence between two input sentences regardless of the position of those words. Here, the top-3 pairs of words assigned the highest self-attention weights at (layer 0, head 7) are inside red, green, and blue rectangles, respectively.

Machine learning (ML) models recently achieved excellent performance on state-of-the-art benchmarks for evaluating natural language understanding (NLU). In July 2019, RoBERTa [93] was the first to surpass a human baseline on GLUE [155]. Since then, 13 more methods have also outperformed humans on the GLUE leaderboard [49]. Notably, at least 8 out of the 14 solutions

are based on BERT [29]—a transformer architecture that learns representations via a bidirectional encoder. Given their superhuman GLUE-scores, how do BERT-based models solve NLU tasks? How do their NLU capability differs from that of humans?

We shed light into these important questions by examining model sensitivity to the order of words. Word order is one of the key characteristics of a sequence and is tightly constrained by many linguistic factors including syntactic structures, subcategorization, and discourse [36]. Thus, arranging a set of words in a correct order is considered a key problem in language modeling [58, 181].

Therefore, a natural question is: **Do BERT-based models trained on GLUE care about the order of words in a sentence?** [90] found that pretrained BERT captures word-order information in the first three layers. However, it is unknown whether BERT-based classifiers actually use word order information when performing NLU tasks. Recently, [158] showed that incorporating additional word-ordering and sentence-ordering objectives into BERT pretraining could lead to text representations (StructBERT) that enabled improved GLUE scores. However, StructBERT findings are inconclusive across different GLUE tasks and models. For example, in textual entailment [155, RTE], StructBERT improved the performance for BERT<sub>large</sub> but hurt the performance for RoBERTa (Table 2.2d).

[158] motivated interesting questions: **Are state-of-the-art BERT-based models using word order information when solving NLU tasks? If not, what cues do they rely on?** To the best of our knowledge, our work is the first to study the above questions for an NLU benchmark (GLUE). We tested BERT-, RoBERTa-, and ALBERT-based [84] models on 7 GLUE tasks where the words of only one select sentence in the input text are shuffled at varying degrees. An ideal agent that truly understands language is expected to choose a reject option when asked to classify a sentence whose words are randomly shuffled. Alternatively, given shuffled input words, true NLU agents are expected to perform at random chance in multi-way classification that has no reject options (Fig. 2.1b). Our findings include:

1. 65% of the groundtruth labels of 5 GLUE tasks can be predicted when the words in one sentence in each example are shuffled (Sec. 2.4.1).
2. Although pretrained BERT embeddings are known to be contextual, in some GLUE tasks, the contribution of an individual word to classification is almost unchanged even after its surrounding words are shuffled (Sec. 2.4.3).
3. In sentiment analysis (SST-2), the polarity of a single salient word is  $\geq 60\%$  predictive of an entire sentence’s label (Sec. 2.4.4).
4. BERT-based models trained on sequence-pair GLUE tasks used a set of self-attention heads for finding similar tokens shared between the two inputs (Sec. 2.4.4).
5. Encouraging RoBERTa-based models to be more sensitive to word order improves the performance on SQuAD 2.0 and most GLUE tasks tested (i.e. except for SST-2) (Sec. 2.4.5).

Despite their superhuman scores, most GLUE-trained models behave similarly to Bag-of-Words (BOW) models, which are prone to naive mistakes (Fig. 2.1b–d). Our results also suggest that GLUE does not necessarily require syntactic information or complex reasoning.

## 2.2 Related Work

**Pretrained BERT** [90] found that positional information is encoded in the first three layers of BERT<sub>base</sub> and fades out starting layer 4. [39] found that BERT heavily relies on word order when predicting missing words in masked sentences from the CPRAG-102 dataset. That is, shuffling words in the context sentence caused the word-prediction accuracy to drop by  $\sim 1.3$  to  $2\times$ . While all above work studied the *pretrained* BERT, we instead study BERT-based models *finetuned* on downstream tasks.

**Word-ordering as an objective** In text generation, [36] found that recurrent neural networks were sensitive to regularities in word order in simple sentences. Language models [96] with long

short-term memory (LSTM) units [63] were able to recover the original word order of a sentence from randomly-shuffled words even without any explicit syntactic information [140]. [158] also observed an increase in GLUE performance after pretraining BERT with two additional objectives of word-ordering and sentence-ordering. Their work differs from ours in three points: (1) they did not study the importance of word order alone; (2) StructBERT improvements were inconsistent across tasks and models (Table 2.2d) and motivated us to compare the word-order importance between GLUE tasks; and (3) we proposed to improve model performance by finetuning not pretraining.

**Word-order insensitivity in other NLP tasks** ML models have been shown to be insensitive to word order in several NLP tasks such as reading comprehension [141, 145], dialog [139], natural language inference [110, 142], and essay scoring [109]. [177] found that for several text classification tasks, syntactic information was not always required. In word prediction, LSTMs and pre-trained BERT were found to exhibit a certain degree of insensitivity when the context words are randomly shuffled [73, 98, 39]. Compared to the prior work, we are the first to perform a word-order analysis on a NLU benchmark and to contrast this sensitivity across the tasks.

**Humans can also be word-order invariant** A recent human study interestingly showed that sentences with scrambled word orders elicit a response as high as that elicited by original sentences as long as the local mutual information among words is high enough [99]. [48] found that humans can also exhibit word-order-invariance effects, especially when one interpretation is much more semantically plausible. Our work therefore documents an important similarity between humans and advanced NLU models.

**Invariance to patch-order in computer vision** In computer vision, the accuracy of state-of-the-art image classifiers was found to only drop marginally when the patches in an image were randomly shuffled [24, 179].

## 2.3 Methods

### 2.3.1 Datasets

We chose GLUE because of three reasons: (1) GLUE is a common benchmark for NLU evaluation [155]; (2) there exist NLU models (e.g. RoBERTa) that outperform humans on GLUE, making an important case for studying their behaviors; (3) it is unknown how sensitive GLUE-trained models are to word order and whether GLUE requires them to be sensitive [158].

**Tasks** Out of 9 GLUE tasks, we chose all 6 binary-classification tasks because they share the same random baseline of 50% accuracy and enable us to compare models’ word-order sensitivity across tasks. Six tasks vary from acceptability (CoLA [159]), to natural language inference (QNLI [128]), RTE [155], paraphrase (MRPC [32], QQP [125]), and sentiment analysis (SST-2 [144]).

We also performed our tests on STS-B [20]—a regression task of predicting the semantic similarity of two sentences.<sup>1</sup> While CoLA and SST-2 require single-sentence inputs, all other tasks require sequence-pair inputs.

**Reject options** For all binary-classification tasks (except SST-2), the negative label is considered the reject option (e.g. QQP models can choose not duplicate in Fig. 2.1b to reject shuffled inputs).

**Metrics** We use accuracy scores to evaluate the binary classifiers (for ease of interpretation) and Spearman correlation to evaluate STS-B regressors, following [155].

### 2.3.2 Classifiers

We tested BERT-based models because (1) they outperformed humans on the [49]; and (2) the pretrained BERT was shown to capture word positional information [90].

**Pretrained BERT encoders** We tested three sets of classifiers finetuned from three different, pretrained BERT variants: BERT, RoBERTa, and ALBERT, downloaded from [67]. The pretrained models are the “base” versions i.e. bidirectional transformers with 12 layers and 12 self-attention

---

<sup>1</sup>We did not choose WNLI [88] as model performance is not substantially above random baseline.



heads. The pretraining corpus varies from uncased (BERT, ALBERT) to case-sensitive English (RoBERTa).

**Classifiers** For each of the seven GLUE tasks, we added one classification layer on top of each of the three pretrained BERT encoders and finetuned the entire model. Unless otherwise noted, the mean performance per GLUE task was averaged over three classifiers. Each model’s performance matches either those reported on [67] or the original papers (Table 6.6).

**Hyperparameters** Following [29], we finetuned classifiers for 3 epochs using Adam [79] with a learning rate of 0.00002,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$ . We used a batch size of 32, a max sequence length of 128, and dropout on all layers with a probability of 0.1.

### 2.3.3 Constructing sets of real and shuffled examples for experiments

**Modifying one sentence** As GLUE tasks vary in the number of inputs (one or two input sequences) and the sequence type per input (a sentence or a paragraph), we only re-ordered the words in one *sentence* from only one input while keeping the rest of the inputs unchanged. Constraining the modifications to a single sentence enables us to measure (1) the importance of word order in a single sentence; and (2) the interaction between the shuffled words and the unchanged, real context.

**Random shuffling methods** To understand model behaviors across varying degrees of word-order distortions, we experimented with three tests: randomly shuffling n-grams where  $n = \{1, 2, 3\}$ .

Shuffling 1-grams is a common technique for analyzing word-order sensitivity [139, 177]. We split a given sentence by whitespace into a list of n-grams, and re-combined them, in a random order, back into a “shuffled” sentence (see Table 2.1 for examples). The ending punctuation was kept intact. We re-sampled a new random permutation until the shuffled sentence was different from the original sentence.

As the label distributions, dev-set sizes, and the performance of models vary across GLUE tasks, to compare word-order sensitivity across tasks, we tested each model on two sets: (1) dev-r i.e. a subset of the original dev-set (Sec. 2.3.3); and (2) dev-s i.e. a clone of version of dev-r but that each example has one sentence with re-ordered words (Sec. 2.3.3).

How can smoking marijuana give you lung cancer?  
 Q<sub>3</sub> lung cancer marijuana give you How can smoking?  
 Q<sub>2</sub> smoking marijuana lung cancer give you How can?  
 Q<sub>1</sub> marijuana can cancer How you smoking give lung?  
 Q<sub>s</sub> How can smoking cancer give you lung marijuana?

Table 2.1: A real question on Quora (QQP dataset) and its three modified versions (Q<sub>3</sub> to Q<sub>1</sub>) created by randomly shuffling 3-grams, 2-grams, and 1-grams, respectively. Q<sub>s</sub> was created by swapping two random nouns.

### Selecting real examples

For each pair of (task, classifier), we selected a subset of dev-set examples via the following steps:

1. For tasks with either a single-sequence or a sequence-pair input, we used examples where the input sequence to be modified has only one sentence<sup>2</sup> that has more than 3 tokens (for shuffling 3-grams to produce a sentence different from the original sentence).
2. We only selected the examples that were correctly classified by the classifier (to study what features were important for high accuracy).
3. We balanced the numbers of positive and negative examples by removing random examples from the larger-sized class.

That is, on average, we filtered out ~34% of the original data. See Table 6.4 for the total number of examples remaining after each filtering step above.

### Creating shuffled sets

For each task, we cloned the dev-r sets above and modified each example to create a “shuffled” set (a.k.a. dev-s) per shuffling method.

Specifically, a CoLA and SST-2 example contains only a single sentence and we modified that sentence. Each QQP, MRPC and STS-B example has two sentences and we modified the first

<sup>2</sup>We used NLTK sentence splitter [12] to detect text that has more than one sentence.

sentence. An RTE example has a pair of (premise, hypothesis), and we modified the hypothesis since it is a single sentence while premises are paragraphs. Each QNLI example contains a pair of (question, answer) and we modified the question, which is a sentence, while an answer is often a paragraph.

## 2.4 Experiments and Results

### 2.4.1 How much is word order information required for solving GLUE tasks?

GLUE has been a common benchmark for evaluating NLU progress. But, do GLUE tasks require models to use word order and syntactic information? We shed light into this question by testing model performance when word order is increasingly randomized.

If a task strictly requires words to form a semantically meaningful sentence, then randomly re-positioning words in correctly-classified sentences will cause model accuracy to drop from 100% to 50% (i.e. the random baseline  $b$  for binary-classification tasks with two balanced classes). Thus, to compare model-sensitivity across tasks, we use a Word-Order Sensitivity score (WOS):

$$s = (100 - p)/(100 - b) \tag{2.1}$$

where  $p \in [50, 100]$  is the accuracy of a GLUE-trained model evaluated on a dev-s set (described in Sec. 2.3.3) and  $s \in [0, 1]$ . Here,  $b = 50$ .

**Experiments** For each GLUE task, we computed the mean accuracy and confidence score over three classifiers (BERT, RoBERTa, and ALBERT-based) on dev-s sets created by shuffling 1-grams, 2-grams, and 3-grams. The results reported in Table 2.2 were averaged over 10 random shuffling runs (i.e. 10 random seeds) per n-gram type, and then averaged over 3 models per task.

**Results** We found that for CoLA, i.e. detecting grammatically incorrect sentences, the model accuracy, on average, drops to near random chance i.e. between 50.69% and 56.36% (Table 2.2b)

when n-grams are shuffled. That is, most of examples were classified into unacceptable after n-gram shuffling, yielding  $\sim 50\%$  accuracy (see Fig. 6.2 for qualitative examples).

Surprisingly, for the rest of the 5 out of 6 binary-classification tasks (i.e. except CoLA), between 75% and 90% of the originally correct predictions remain constant after 1-grams are randomly re-ordered (Table 2.2b; 1-gram). These numbers increase as the shuffled n-grams are longer (i.e. as n increases from 1 $\rightarrow$ 3), up to 95.32% (Table 2.2b; QNLI). Importantly, given an average dev-set accuracy of 86.35% for these 5 tasks, **at least  $86.35\% \times 75\% \approx 65\%$  of the groundtruth labels of these 5 GLUE tasks can be predicted when all input words in one sentence are randomly shuffled.**

Additionally, on average over three n-gram types, models trained on these five GLUE tasks are from 2 to 10 times more *insensitive* to word-order randomization than CoLA models (Table 2.2c). That is, if not explicitly tasked with checking for grammatical errors, GLUE models mostly will not care about the order of words in a sentence (see qualitative examples in Figs. 2.1, 6.2–6.4). Consistently, the confidence scores of BERT-based models for five non-CoLA tasks only dropped  $\sim 2\%$  when 1-grams are shuffled (Table 2.2).

Consistently across three different BERT “base” variants and a RoBERTa “large” model (Table 6.5), our results suggest that word order and syntax, in general, are not necessarily required to solve GLUE.

**2-noun swaps** Besides shuffled n-grams, we also repeated all experiments with more syntactically-correct modified inputs where only two random nouns in a sentence were swapped (Table 2.1;  $Q_s$ ). This is a harder test for NLU models since the meaning of a sentence with two nouns swapped often changes while its syntax remains correct. We found the conclusions to generalize to this setting. That is, the models hardly changed predictions although the meanings of the original sentence and its swapped version are different (Table 2.2b; 2-noun swap vs. 1-gram).

## 2.4.2 How sensitive are models trained to predict the similarity of two sentences?

An interesting hypothesis is that models trained explicitly to evaluate the semantic similarity of two sentences should be able to tell apart real from shuffled examples. Intuitively, word order information is essential for understanding what an entire sentence means and, therefore, for predicting whether two sentences convey the same meaning.

We tested this hypothesis by analyzing the sensitivity of models trained on QQP and STS-B—two prominent GLUE tasks for predicting semantic similarity of a sentence pair. While QQP is a binary classification task, STS-B is a regression task where a pair of two sentences is given a score  $\in [0, 5]$  denoting their semantic similarity.

**Experiments** We tested the models on dev-r and dev-s sets (see Sec. 2.3.3) where in each pair, the word order of the first sentence was randomized while the second sentence was kept intact.

**QQP results** Above 83% of QQP models’ correct predictions on real pairs remained unchanged after word-order randomization (see Figs. 2.1a–c for examples).

**STS-B results** Similarly, STS-B model performance only drops marginally, i.e. less than 2 points from 89.67 to 87.80 in Spearman correlation (Table 2.2; STS-B). Since a STS-B model outputs a score  $\in [0, 5]$ , we binned the scores into 6 ranges. One might expect STS-B models to assign near-zero similarity scores to most modified pairs. However, the distributions of similarity scores for the modified and real pairs still closely match up (Fig. 2.2). In sum, **despite being trained explicitly on predicting semantic similarity of sentence pairs, QQP and STS-B are surprisingly insensitive to n-gram shuffling, exhibiting naive understanding of sentence meanings.**

## 2.4.3 How important are words to classification after their context is shuffled?

BERT representations for tokens are known to be highly contextual [38]. However, after finetuning on GLUE, would the importance of a word to classification drop after its context is shuffled?

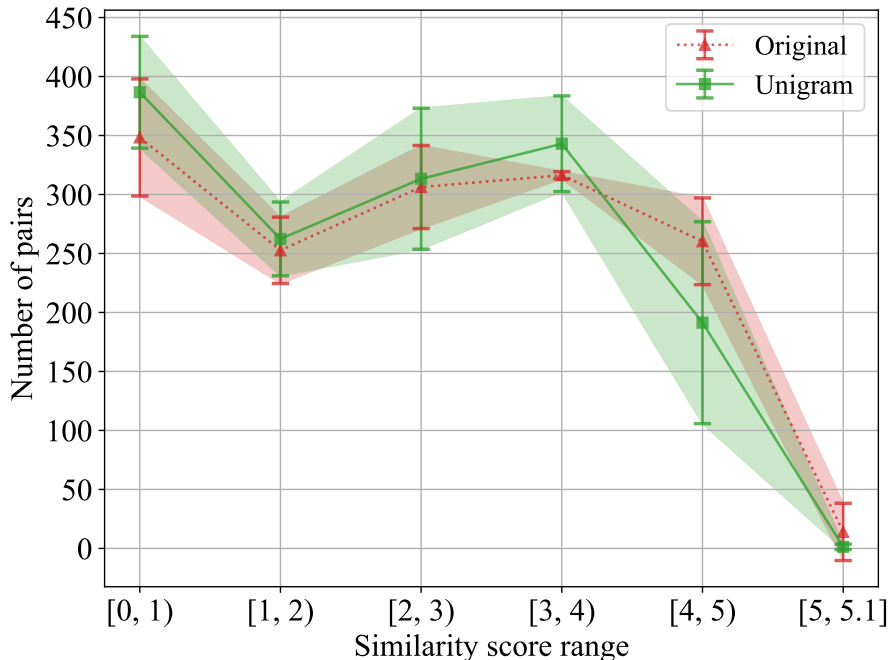


Figure 2.2: The distribution of similarity scores over 6 ranges for the (real, shuffled) pairs in dev-s (green) is highly similar to that for (real, real) STS-B pairs in dev-r (red). The statistics in each range were computed over 3 models (BERT, RoBERTa, and ALBERT).

To answer the above question, we used LIME [134] to compute word importance. **LIME** computes a score  $\in [-1, 1]$  for each token in the input denoting how much its presence contributes for or against the network’s predicted label (Fig. 2.1; highlights). The importance score per word  $w$  is intuitively the mean confidence-score drop over a set of randomly-masked versions of the input when  $w$  is masked out.

**Experiments** We chose to study RoBERTa-based classifiers here because they have the highest GLUE scores among the three BERT variants considered. We observed that **62.5% (RTE) to 79.6% (QNLI) of the dev-r examples were consistently, correctly classified into the same labels in all 5 different random shuffles** (i.e. 5 different random seeds). We randomly sampled 100 such examples per binary-classification task and computed their LIME attribution maps to compare the similarity between the LIME heatmaps before and after unigrams are randomly misplaced.

Task	(a) Perf. on dev-r		(b) Performance on dev-s				(c) Word-Order Sensitivity			(d) StructBERT improvements		
	Models	Baseline	2-noun swap	1-gram	2-gram	3-gram	1-gram	2-gram	3-gram	BERT <sub>base</sub>	BERT <sub>large</sub>	RoBERTa
CoLA	100 (0.93)	50	71.75 (0.91)	50.69 (0.95)	53.98 (0.94)	56.36 (0.92)	<b>0.99</b>	<b>0.92</b>	<b>0.87</b>	<b>+4.9</b>	+4.8	+1.4
RTE	100 (0.81)	50	85.86 (0.81)	75.69 (0.80)	81.89 (0.80)	85.18 (0.79)	0.49	0.36	0.30	N/A	<b>+13.0</b>	-0.9
QQP	100 (0.98)	50	86.90 (0.96)	83.19 (0.96)	88.02 (0.96)	89.04 (0.96)	0.34	0.24	0.22	+0.7	+1.2	+0.5
MRPC	100 (0.91)	50	96.51 (0.91)	83.89 (0.89)	87.1 (0.90)	89.38 (0.90)	0.32	0.26	0.21	N/A	+3.9	<b>+1.7</b>
SST-2	100 (0.99)	50	97.78 (0.98)	84.04 (0.96)	88.35 (0.97)	90.56 (0.97)	0.32	0.23	0.19	+0.2	+0.3	+0.4
QNLI	100 (0.98)	50	94.31 (0.97)	89.42 (0.96)	93.85 (0.97)	95.32 (0.98)	0.21	0.12	0.09	N/A	+3.0	+0.3
STS-B	89.67	N/A	88.93	87.80	88.66	88.95	N/A	N/A	N/A	N/A	N/A	N/A

Table 2.2: All results (a–c) are reported on the GLUE dev-r sets i.e. 100% accuracy (a). Shuffling n-grams caused the accuracy to drop (b) the largest for CoLA and the least for QNLI. Each row is computed by averaging the results of 3 BERT-based models and 10 random shuffles. From top to bottom, the Word-Order Sensitivity (WOS) is sorted descendingly (c) and is consistent across three types of n-grams i.e. WOS scores decrease from top down and from left to right. In contrast, the StructBERT results (d), taken from Table 1 and 4 in [158], showed inconsistent improvements across different tasks. STS-B results are in scaled Spearman correlation. In addition to small accuracy drops, the mean confidence scores of all classifiers—reported in parentheses e.g. “(0.93)” —also changed marginally after words are shuffled (a vs. b).

**Results** On CoLA and RTE, the importance of words (i.e. mean absolute value of LIME-attribution per word), *decreased* substantially by 0.036 and 0.019, respectively. That is, the individual words become *less important* after their context is distorted—a behavior expected when CoLA and RTE have the highest WOS scores (Table 2.2). In contrast, for the other 4 tasks, word importance only changed marginally (by 0.008, i.e. 4.5× smaller than the 0.036 change in CoLA). That is, **except for CoLA and RTE models, the contribution of a word to classification is almost unchanged even after the context of each word is randomly shuffled** (Fig. 2.1a–c). This result suggests that the word embeddings after finetuning on GLUE became much less contextual than the pretrained BERT embeddings [38].

#### 2.4.4 If not word order, then what do classifiers rely on to make correct predictions?

Given that all non-CoLA models are highly insensitive to word-order randomization, how did they arrive at correct decisions when words are shuffled?

We chose to answer this question for SST-2 and QNLI because they have the lowest WOS scores across all 6 GLUE tasks tested (Table 2.2) and they are representative of single-sentence and sequence-pair tasks, respectively.

#### SST-2: Salient words are highly predictive of sentence labels

As 84.04% of the SST-2 correct predictions did not change after word-shuffling (Table 2.2b), a common hypothesis is that the models might rely heavily on a few key words to classify an entire sentence.

S	the film 's performances are thrilling .	1.00
S <sub>1</sub>	the film thrilling performances are 's .	1.00
S <sub>2</sub>	's thrilling film are performances the .	1.00
S <sub>3</sub>	's thrilling are the performances film .	1.00

Figure 2.3: An original SST-2 dev-set example (S) and its three shuffled versions (S<sub>1</sub> to S<sub>3</sub>) were all correctly labeled positive by a RoBERTa-based classifier with high confidence scores (right column).

**Experiments** To test this hypothesis, we took all SST-2 dev-r examples whose all 5 randomly shuffled versions were all correctly labeled by a RoBERTa-based classifier (i.e. this “5/5” subset is ~65% of the dev-set). We used LIME to produce a heatmap of the importance of words in each example.

We identified the polarity of each top-1 most important word (i.e. the highest LIME-attribution score) per example by looking it up in the Opinion Lexicon [66] of 2,006 positive and 4,783 negative words. ~57% of these top-1 words were found in the dictionary and labeled either positive or negative (see Table 6.3).



QNLI sentence-pair inputs and their LIME attributions (negative -1, neutral 0, positive +1)		Confidence score
Q	How long did Phillips manage the Apollo missions?	1.00
A	Mueller agreed, and Phillips managed Apollo from January 1964, until it achieved the first manned landing in July 1969, after which he returned to Air Force duty.	
Q <sub>1</sub>	Apollo the Phillips How missions long did manage?	0.96
A	Mueller agreed, and Phillips managed Apollo from January 1964, until it achieved the first manned landing in July 1969, after which he returned to Air Force duty.	
Q <sub>2</sub>	Phillips long manage How missions the Apollo did?	0.97
A	Mueller agreed, and Phillips managed Apollo from January 1964, until it achieved the first manned landing in July 1969, after which he returned to Air Force duty.	
Q <sub>s</sub>	How long did Apollo manage the Phillips missions?	0.99
A	Mueller agreed, and Phillips managed Apollo from January 1964, until it achieved the first manned landing in July 1969, after which he returned to Air Force duty.	

Figure 2.4: A RoBERTa-based model’s correct prediction of entailment on the original input pair (Q, A) remains unchanged when the question is randomly shuffled (Q<sub>1</sub> & Q<sub>2</sub>) or when two random nouns in the question are swapped (Q<sub>s</sub>). The salient words in the questions e.g. **manage** and **missions** remain similarly important after their context has been shuffled. Also, the classifier harnessed self-attention to detect the correspondence between similar words that appear in both the question and the answer e.g. **manage** (Q) and **managed** (A). That is, the top-3 pairs of words that were assigned the largest question-to-answer weights in a self-attention matrix (layer 0, head 7) are inside in the red, green, and blue rectangles.

**Results** We found that if the top-1 word has a positive meaning, then there is a 100% probability that the sentence’s label is positive. For example, the word “thrilling” in a movie review indicates a positive sentence (see Fig. 2.3). Similarly, the conditional probability of a sentence being labeled negative given a negative top-1 word is 94.4%. That is, given this statistics, the SST-2 label distribution and model accuracy, **at least 60% of the SST-2 dev-set examples can be correctly predicted from only a single top-1 salient word.**

We also reached similar conclusions when experimenting with ALBERT classifiers and the SentiWords dictionary [46] (see Table 6.3).

**Self-attention layers matching similar words in both the question and the answer**

For sequence-pair tasks, e.g. QNLI, how can models correctly predict entailment when the question words are randomly shuffled (Fig. 2.4; Q<sub>1</sub>) or when the question syntax is correct but its

meaning changes entirely (Fig. 2.4;  $Q_s$ ). We hypothesize that inside the model, there might be a self-attention (SA) layer that extracts pairs of similar words that appear in both the question and the answer (e.g. “manage” vs. “managed” in Fig. 2.4).

**Experiments** To test this hypothesis, we analyzed the 5,000 QNLI dev-r examples (Table 6.4) of RoBERTa-based classifiers trained on QNLI. For each example, we identified one SA matrix (among all 144 as the base model has 12 layers & 12 heads per layer) that assigns the highest weights to pairs of similar words between the question and the answer, i.e. excluding intra-question and intra-answer attention weights (see the procedure in Sec. 6.1.1).

**Results** First, in ~58% of the examples, we found at least three pairs of words that *match* (i.e. the sum Levenshtein character-level edit-distance for all 3 pairs is  $\leq 4$ ). Second, we found, in total, 15 SA heads (out of the 144) which are explicitly tasked with capturing such question-to-answer word correspondence, *regardless* of word order (see Fig. 2.4).

Remarkably, **87% of the work of matching similar words that appear in both the QNLI question and the answer was handled by only 3 self-attention heads at (layer, head) of (0,7), (1,9), and (2,6).**

We found consistent results when repeating the same analysis for other three sequence-pair tasks. That is, interestingly, **the three SA heads at exactly the same location of (0, 7), (1, 9), and (2, 6) account for 76%, 89%, and 83% of the “word-matching” task on QQP, RTE, and MRPC, respectively.** This coincidence is likely due to the fact that these classifiers were finetuned for different downstream tasks starting from the same pretrained RoBERTa encoder. See Figs. 2.1, 2.4, 6.3–6.4 for qualitative examples of these three tasks.

**How important are the 15 word-matching attention heads to QNLI model performance?** We found that zero-ing out 15 random heads had almost no effect to correctly-classified predictions—i.e. accuracy dropped marginally ( $-1\%$  to  $-3\%$ , Table 2.3) across different groups of examples. However, ablating the 15 word-matching heads caused the performance to drop substantially i.e. (a) by 9.6% on the 1,453 positive examples identified in Sec. 6.1.1; (b) by 22.1% on a set of 2,906

QNLI dev-r examples	Full network	Zero-out 15 attention matrices	
		Random	Ours
a. 1,453 selected 0/5 (+) examples	100	99.31	90.43
b. 1,453 random 0/5 (+) examples	100	99.24	91.05
c. 1,453 random 0/5 (+) examples & 1,453 random 0/5 (-) examples	100	98.18	77.91
d. (+/-) All 5,000 examples	100	96.96	75.54

Table 2.3: Zero-ing out a set of 15 “word-matching” self-attention matrices (identified via the procedure in Sec. 2.4.4) caused a substantial drop of  $\sim 25\%$  in accuracy (d) while the random baseline is 50%. These 15 matrices played an important role in QNLI because ablating 15 random matrices only caused a  $\sim 1\text{-}3\%$  drop in accuracy.

random, examples including both positive and negative examples (at 50/50 ratio); and (c) by 24.5% on the entire QNLI 5,000-example dev-r set. That is, **the 15 SA heads that learned to detect similar words played an important role in solving QNLI, i.e. enabling at least  $\sim 50\%$  of the correct predictions** (Table 2.3d; accuracy dropped from 100% to 75.54% when the random chance is 50%). In sum, we found overlap between words in the question and answer of QNLI examples and strong evidence that QNLI models harnessed self-attention to exploit such overlap to make correct decisions in spite of a random word-order.

#### 2.4.5 Does increasing word-order sensitivity lead to higher model performance?

Here, we test whether encouraging BERT representations to be more sensitive to word order (i.e. more syntax-aware) would improve model performance on GLUE & SQuAD 2.0 [127]. We performed this test on the five GLUE binary-classification tasks (i.e. excluding CoLA because its WOS score is already at 0.99; Table 2.2).

**Experiments** Inspired by the fact that CoLA models are highly sensitive to word order, we finetuned the pretrained RoBERTa on a *synthetic*, CoLA-like task first, before finetuning the model on downstream tasks.

The synthetic task is to classify a single sentence into real vs. fake where the latter is formed by taking each real sentence and swapping two random words in it. For every downstream task (e.g. SST-2), we directly used its original training and dev sets to construct a balanced, 2-class, synthetic dataset. After finetuning the pretrained RoBERTa on this synthetic binary classification task, we re-initialized the classification layer (keeping the rest unchanged) and continued finetuning it on a downstream task.

For both finetuning steps, we trained 5 models per task and followed the standard BERT finetuning procedure (described in Sec. 2.3.2).

**Results** After the first finetuning on synthetic tasks, all models obtained a  $\sim 99\%$  training-set accuracy and a  $\sim 95\%$  dev-set accuracy. **After the second finetuning on downstream tasks, we observed that all models were substantially more sensitive to word order**, compared to the baseline models (which were only finetuned on the downstream tasks). That is, we repeated the 1-gram shuffling test (Sec. 2.4.1) and found a  $\sim 1.5$  to  $2\times$  increase in the WOS scores of all models (see Table 2.4a vs. b).

GLUE dev-s	(a) RoBERTa		(b) Ours	
	Accuracy	WOS	Accuracy	WOS
RTE	80.76	0.38	64.01	<b>0.72</b> (+189%)
MRPC	83.86	0.32	72.88	<b>0.54</b> (+169%)
SST-2	84.26	0.31	76.97	<b>0.46</b> (+148%)
QQP	87.66	0.25	77.11	<b>0.46</b> (+184%)
QNLI	91.09	0.18	82.44	<b>0.35</b> (+194%)

Table 2.4: With finetuning on synthetic tasks, all of our models (b) have a larger drop in accuracy on shuffled dev-s examples, compared to the standard RoBERTa-based classifiers (a). That is, our models are substantially more sensitive to word-order randomization (i.e. +148% to +194% in WOS scores).

**GLUE** On GLUE dev sets, on average over 5 runs, our models outperformed the RoBERTa baseline on all tasks except for SST-2 (Table 2.5). The highest improvement is in RTE (from 72.2%

to 73.21% on average, and to 74.73% for the best single model), which is consistent with the fact that RTE has the highest WOS score among non-CoLA tasks (Sec. 2.4.1).

**SQuAD 2.0** Our models also outperformed the RoBERTa baseline on the SQuAD 2.0 dev set, with the highest F1 gain from 80.62% to 81.08% (Table 2.5).

In sum, leveraging the insights that the original BERT-based models are largely word-order invariant, we showed that increasing model sensitivity via a simple extra finetuning step directly improves GLUE and SQuAD 2.0 performance.

	RTE	QQP	MRPC	SST-2	QNLI	SQuAD
	(Acc)	(Acc)	(Acc)	(Acc)	(Acc)	(F1)
RoBERTa	72.20	91.12	87.25	<b>94.50</b>	92.57	80.62
Our best model	<b>74.73</b>	<b>91.31</b>	<b>88.73</b>	94.50	<b>93.08</b>	<b>81.08</b>
	<b>+2.53</b>	<b>+0.19</b>	<b>+1.48</b>	+0	<b>+0.51</b>	<b>+0.46</b>
Average (5 runs)	<b>73.21</b>	<b>91.19</b>	<b>87.31</b>	94.22	<b>92.71</b>	<b>80.75</b>
	<b>+1.01</b>	<b>+0.07</b>	<b>+0.06</b>	-0.28	<b>+0.14</b>	<b>+0.13</b>

Table 2.5: Finetuning the pretrained RoBERTa on synthetic tasks (before finetuning on the downstream tasks) improved model dev-set performance on SQuAD 2.0 (b) and all the tested tasks in GLUE (a), except SST-2.

## 2.5 Discussion and Conclusion

Consistently across three BERT variants and two model sizes, we found that GLUE-trained BERT-based models are often word-order invariant unless explicitly asked for (e.g. in CoLA).

We present a reflection on the progress of NLU by studying GLUE—a benchmark where humans have been surpassed by many models in the past 18 months. As suggested by our work, these models; however, may neither use syntactic information nor complex reasoning. We revealed how self-attention, a key building block in modern NLP, is being used to extract superficial cues to solve sequence-pair GLUE tasks even when words are out of order.

**Adversarial NLI** We also replicated our shuffling experiments on ANLI [104], a task considered challenging to existing models, and where RoBERTa-based models only obtained a 56% accuracy. We found RoBERTa-based models to remain not always sensitive to word-order randomization on ANLI (Table 6.2; WOS of 0.63), suggesting a common issue in existing benchmarks.

## Chapter 3

### Double Trouble: How to not Explain a Text Classifier’s Decisions using Counterfactuals Synthesized by Masked Language Models?

#### 3.1 Overview

Feature attribution maps (AMs), i.e. highlights indicating the importance of each input token w.r.t. a classifier’s decision, can help improve *human accuracy* on downstream tasks including detecting fake movie reviews [83] or identifying biases in text classifiers [91].

Many Leave-One-Out (LOO) methods compute the attribution of an input token by measuring the prediction changes after substituting that token’s embedding with zeros [89, 71] or [UNK] [78]. That is, deleting or replacing features is the underlying principle of at least 25 attribution methods [26].

Based on the evidence in computer vision [7, 182], prior works in NLP *hypothesized* that removing a word from an input text forms out-of-distribution (OOD) inputs that yield erroneous AMs [78, 56] or AMs inconsistent with human’s perception of causality [57]. To generate plausible counterfactuals, two teams of researchers [78, 56] proposed Input Marginalization (IM), i.e. replace a word using BERT<sub>base</sub> [29] and compute an average prediction difference by marginalizing over all predicted words. [78] claimed that IM yields more accurate AMs than the baselines that replace words by [UNK] or zeros but their quantitative results were reported for only *one*<sup>1</sup> dataset and *one* evaluation metric.

In this paper, we re-assess their claim by, first, reproducing their IM results<sup>2</sup>, and then rigorously evaluate whether improving the realism of counterfactuals improves two attribution methods (LOO and LIME). On a diverse set of *three* datasets and *six* metrics, we find that:

---

<sup>1</sup>No *quantitative* results on SNLI, only SST-2.

<sup>2</sup>Code and pre-trained models are available at <https://github.com/anguyen8/im>.

(a) SST – Groundtruth & target class: positive	
S	The very definition of the ‘ small ’ movie , but it is a good <b>stepping</b> <b>stone</b> <b>for</b> director Sprecher . <b>0.9793</b> <b>stepping</b> <b>0.9760</b> <b>stone</b> <b>0.8712</b> <b>for</b> 0.0050 rolling    0.0048 stones    0.0860 to 0.0021 casting    0.0043 point    0.0059 ,
IM <sub>0</sub>	The very definition of the ‘ small ’ movie , <b>but</b> it is <b>a</b> good stepping stone for <b>director</b> Sprecher .
IM <sub>1</sub>	The <b>very</b> definition of the ‘ small ’ movie , but it is a good stepping stone for <b>director</b> Sprecher .
IM <sub>2</sub>	The very definition of the ‘ small ’ movie , <b>but</b> it <b>is</b> a good stepping stone for <b>director</b> Sprecher .
IM <sub>3</sub>	The <b>very</b> definition of the ‘ small ’ movie , but it is a <b>good</b> stepping stone for <b>director</b> Sprecher .
(b) e-SNLI – Groundtruth & target class: contradiction	
P	A group of people prepare <b>hot</b> <b>air</b> <b>balloons</b> for takeoff . <b>0.9997</b> <b>hot</b> <b>0.9877</b> <b>air</b> <b>0.9628</b> <b>balloons</b> 0.0001 compressed    0.0102 water    0.0282 balloon 0.0000 open    0.0008 helium    0.0019 engines
H	A group of people prepare <b>cars</b> for racing .
IM <sub>0</sub>	A group of people prepare hot air balloons for takeoff . A group of people prepare <b>cars</b> for <b>racing</b> .
IM <sub>1</sub>	A group of people prepare hot air balloons for takeoff . A group of people prepare <b>cars</b> for <b>racing</b> .
IM <sub>2</sub>	A group of people prepare hot air balloons for takeoff . A group of people prepare <b>cars</b> for <b>racing</b> .
IM <sub>3</sub>	A group of people prepare hot air balloons for takeoff . A group of people prepare cars for <b>racing</b> .

Figure 3.1: **By design, IM erroneously assigns near-zero attribution to highly-predictable words.** Color map: **negative -1**, neutral 0, **positive +1**. Many words labeled **important** by humans such as “stepping”, “stone” (a) or “hot”, “air” (b) are always given near-zero attribution by IM (because they are highly predictable by BERT, e.g. 0.9793 for **stepping**) regardless of the classifier. Even when randomizing the classifier’s weights three times, the IM attribution of these words remains unchanged at near zero (IM<sub>1</sub> to IM<sub>3</sub>). Therefore, when marginalizing over the top-*k* BERT candidates (e.g., “stepping”, “rolling”, “casting”), the IM attribution for low-entropy words tends to zero, leading to heatmaps that are biased, less accurate, and less plausible than LOO<sub>empty</sub>.

1. The Deletion<sub>BERT</sub> metric in [78] is biased towards IM as both use BERT to replace words (Sec. 3.4). In contrast, the vanilla Deletion metric [5] favors the LOO<sub>empty</sub> baseline as both delete words. This bias causes a **false conclusion** that IM is better than LOO baselines



in [78] and also **exists in other Deletion variants**, e.g., Insertion [5], Sufficiency, and Comprehensiveness [31].

2. We find **no evidence that IM is better** than a simple  $\text{LOO}_{\text{empty}}$  on any of the following four state-of-the-art AM evaluation metrics (which exclude the biased Deletion & Deletion<sub>BERT</sub>): ROAR, ROAR<sub>BERT</sub> [65] (Sec. 3.5.1), comparison against human annotations (Sec. 3.5.2), and sanity check [1] (Sec. 3.5.3).
3. We argue that IM is not effective in practice because: (1) deleting a single word from an input has only a marginal effect on classification accuracy (Sec. 3.5.4); and (2) given a *perfect*, masked language model  $G$ , IM would still be **unfaithful** because highly predictable words according to  $G$ , e.g. “hot”, “air” in Fig.3.1, are always assigned near-zero attribution in IM *regardless* of how important they are to the classifier (Sec. 6.2.2).
4. To further test the main idea of IM, we integrate BERT into LIME [134] to *replace* multiple words (instead of deleting) in an input sequence, making LIME counterfactuals more realistic. We find this technique to improve LIME consistently under multiple ROAR-based metrics, but not under comparison against human annotations (Sec. 3.6).

To our knowledge, our work is the first to thoroughly study the effectiveness of IM in NLP in both settings of replacing a single word (LOO) and multiple words (LIME). Importantly, we find improvement in the latter but not the former setting.

### 3.2 Methods and Related Work

Let  $f : \mathbb{R}^{n \times d} \rightarrow [0, 1]$  be a text classifier that maps a sequence  $\mathbf{x}$  of  $n$  token embeddings, each of size  $d$ , onto a confidence score of an output label. An attribution function  $A$  takes three inputs—a sequence  $\mathbf{x}$ , the model  $f$ , and a set of hyperparameters  $\mathcal{H}$ —and outputs a vector  $\mathbf{a} = A(f, \mathbf{x}, \mathcal{H}) \in [-1, 1]^n$ . Here, the explanation  $\mathbf{a}$  associates each input token  $x_i$  to a scalar  $a_i \in [-1, 1]$ , indicating how much  $x_i$  contributes for or against the target label.

**Leave-One-Out** (LOO) is a well-known method [89, 135, 71] for estimating the attribution  $a_i$  by computing the prediction-difference after a token  $x_i$  is left out of the input  $\mathbf{x}$ , creating a shorter sequence  $\mathbf{x}_{-i}$ :

$$a_i = f(\mathbf{x}) - f(\mathbf{x}_{-i}) \quad (3.1)$$

Under [114] causal framework, the attribution  $a_i$  in Eq. 3.1 relies on a single, unrealistic counterfactual  $\mathbf{x}_{-i}$  and thus is a biased estimate of the individual treatment effect (ITE):

$$\text{ITE} = f(\mathbf{x}) - \mathbb{E}[f(\mathbf{x}) \mid do(T = 0)] \quad (3.2)$$

where the binary treatment  $T$ , here, is to keep or “realistically remove” the token  $x_i$  (i.e.  $T = 1$  or  $0$ ) in the input  $\mathbf{x}$ , prior to the computation of  $f(\mathbf{x})$ .

**Perturbation techniques** In computer vision (CV), earlier attribution methods erase a feature by replacing it with (a) zeros [178, 134]; (b) random noise [27, 94]; or (c) blurred versions of the original content [42]. Yet, these perturbation methods produce unrealistic counterfactuals that make AMs more unstable and less accurate [7].

Recent works proposed to simulate the  $do(T = 0)$  operator using an image inpainter. However, they either generated unnatural counterfactuals [22, 50] or only a single, plausible counterfactual per example [2].

**Input marginalization (IM)** In NLP, IM offers the closest estimate of the ITE. IM computes the  $\mathbb{E}[\cdot]$  term in Eq. 3.2 by marginalizing over many plausible counterfactuals generated by BERT<sub>base</sub>:

$$\begin{aligned} \mathbb{E}[f(\mathbf{x}) \mid do(T = 0)] \\ = \sum_{\tilde{x}_i \in \mathcal{V}} p(\tilde{x}_i | \mathbf{x}_{-i}) \cdot f(\mathbf{x}_{-i}, \tilde{x}_i) \end{aligned} \quad (3.3)$$

where  $\tilde{x}_i$  is a token suggested by BERT<sub>base</sub> (e.g., “hot”, “compressed”, or “open” in Fig. 3.1) with a likelihood of  $p(\tilde{x}_i | \mathbf{x}_{-i})$  to replace the masked token  $x_i$ .  $\mathcal{V}$  is the BERT<sub>base</sub> vocabulary of 30,522 tokens.  $f(\mathbf{x}_{-i}, \tilde{x}_i)$  is the classification probability when token  $x_i$  in the original input is replaced with  $\tilde{x}_i$ .

IM attribution is in the log space:

$$\begin{aligned} a_{\text{IM}} = \text{log-odds}(f(\mathbf{x})) \\ - \text{log-odds}(\mathbb{E}[f(\mathbf{x}) \mid do(T = 0)]) \end{aligned} \quad (3.4)$$

where  $\text{log-odds}(p) = \log_2(p/(1-p))$ .

As computing the expectation in Eq. 3.3 over BERT’s ~30K-word vocabulary is prohibitively slow, IM authors only marginalized over the words that have a likelihood  $\geq 10^{-5}$ . We are *able to reproduce* the IM results of [78] by taking only the top-10 words. That is, using the top-10 words or all words of likelihood  $\geq 10^{-5}$  yields slightly different numbers but the same conclusions (Sec. 6.2.4). Thus, we marginalize over the top-10 for all experiments. Note that under BERT, the top-10 tokens, on average, already account for 81%, 90%, and 92% of the probability mass for SST-2, e-SNLI, & MultiRC, respectively.

**BERT** Like [78], we use a pre-trained BERT<sub>base</sub> “base”, uncased model [29], from [67], to fill in a [MASK] token to generate counterfactuals in IM.

**LIME** Based on the idea of IM, we also integrate  $\text{BERT}_{\text{base}}$  into LIME, which originally masks out multiple tokens at once to compute attribution. LIME generates a set of randomly masked versions of the input, and the attribution of a token  $x_i$ , is effectively the mean classification probability over all the masked inputs when  $x_i$  is not masked out. On average, each vanilla LIME counterfactual has 50% of tokens taken out, yielding text often with large syntactic and grammatical errors.

**LIME<sub>BERT</sub>** We use BERT to replace multiple masked tokens<sup>3</sup> in each masked sentence generated by LIME to construct more plausible counterfactuals. However, for each word, we only use the top-1 highest-likelihood token given by BERT instead of marginalizing over multiple tokens because (1) the full marginalization is prohibitively slow; and (2) the top-1 token already carries most of the weight ( $p \geq 0.81$ ; see Table 6.9).

### 3.3 Experiment framework

#### 3.3.1 Three datasets

We select a diverse set of three classification datasets that enable us to (1) compare with the results reported by [78]; and (2) assess AMs on six evaluation metrics (described in Sec. 3.3.3). These three tasks span from sentiment analysis (SST-2), natural language inference (e-SNLI) to question answering (MultiRC), covering a wide range of sequence length (~20, 24, and 299 tokens per example, respectively). SST-2 and e-SNLI were the two datasets where [78] found IM to be superior to LOO baselines.

**SST** Stanford Sentiment Treebank [144] is a dataset of ~12K RottenTomato movie-review *sentences*, which contain human-annotated sentiment annotations for phrases. Each phrase and sentence in SST is assigned a sentiment score  $\in [0, 1]$  (0 = negative, 0.5 = neutral, 1 = positive).

**SST-2** has ~70K SST examples (including both phrases and sentences) where the regression scores per example were binarized to form a binary classification task [144].

---

<sup>3</sup>We find replacing all tokens at once or one at a time to produce similar  $\text{LIME}_{\text{BERT}}$  results.

**e-SNLI** A 3-way classification task of detecting whether the relation between a premise and a hypothesis is entailment, neutral or contradiction [14]. e-SNLI has 569K instances of (input, label, explanation) where the explanations are crowd-sourced [16].

**MultiRC** Multi-sentence Reading Comprehension [74] is a multiple-choice question-answering task that provides multiple input sentences as well as a question and asks the model to select one or multiple correct answer sentences. MultiRC has ~6K examples with human-annotated highlights at the sentence level.

### 3.3.2 Classifiers

Following [78, 56, 57], we test IM and LOO baselines in explaining BERT-based classifiers.

For each task, we train a classifier by fine-tuning the entire model, which consists of a classification layer on top of the pre-trained BERT (described in Sec. 3.2). The dev-set top-1 accuracy scores of our SST-2, e-SNLI, & MultiRC classifiers are 92.66%, 90.92%, and 69.10%, respectively. On the SST binarized dev-set, which contains only sentences, the SST-2-trained classifier’s accuracy is 87.83%.

**Hyperparameters** Following the training scheme of HuggingFace, we fine-tune all classifiers for 3 epochs using Adam optimizer [79] with a learning rate of 0.00002,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$ . A batch size of 32 and a max sequence length of 128 are used for SST-2 and e-SNLI while these hyperparameters for MultiRC are 8 and 512, respectively. Dropout with a probability of 0.1 is applied to all layers. Each model was trained on an NVIDIA 1080Ti GPU.

### 3.3.3 Six evaluation metrics

As there are *no groundtruth* explanations in XAI, we use six common metrics to rigorously assess IM’s effectiveness. For each classifier, we evaluate the AMs generated for all dev-set examples.

**Deletion** is similar to “Comprehensiveness” [31] and is based on the idea that deleting a token of higher importance from the input should cause a larger drop in the output confidence score. We take the original input and delete one token at a time until 20% of the tokens in the input is deleted. A more accurate explanation is expected to have a lower Area Under the output-probability Curve (AUC) [5].

**Deletion<sub>BERT</sub>** a.k.a.  $AUC_{rep}$  in [78], is a Deletion variant where a given token is replaced by a BERT top-1 suggestion instead of an empty string. **Deletion<sub>BERT</sub>** was proposed to minimize the OOD-ness of samples (introduced by deleting words in the vanilla Deletion metric), i.e. akin to integrating BERT into LOO to create IM.

**Remove And Retrain (ROAR)** To avoid a potential OOD generalization issue caused by the Deletion metric, a common alternative is to retrain the classifier on these modified inputs (where  $N\%$  of the highest-attribution words are deleted) and measure its accuracy drop [65]. A more faithful attribution method is supposed to lead to a re-trained classifier of lower accuracy as the more important words have been deleted from training examples. For completeness, we also implement **ROAR<sub>BERT</sub>**, which uses BERT to replace the highest-attribution tokens<sup>4</sup> instead of deleting them without replacement in ROAR.

**Agreement with human-annotated highlights** In both CV and NLP, a common AM evaluation metric is to assess the agreement between AMs and human annotations [160]. The idea is that as text classifiers well predict the human labels of an input text, their explanations, i.e. AMs, should also highlight the tokens that humans deem indicative of the groundtruth label.

Because human annotators only label the tokens supportive of a label (e.g. Fig. 3.2), when comparing AMs with human annotations, we zero out the **negative** values in AMs. Following [184], we binarize a resulting AM at an optimal threshold  $\tau$  in order to compare it with human-annotated highlights under Precision@1.

**Sanity check** [1] is a well-known metric for testing insensitivity (i.e. bias) of attribution methods w.r.t. model parameters. For ease of interpretation, we compute the % change of per-word attribution

---

<sup>4</sup>The chance that a sentence remains unchanged after BERT replacement is low,  $\leq 1\%$ .

values in *sign* and *magnitude* as we randomize the classification layer’s weights. A better attribution method is expected to be more sensitive to the classifier’s weight randomization.

### 3.4 Bias of Deletion metric and its variants

In explaining SST-2 classifiers, we successfully reproduce the  $AUC_{rep}$  results reported in [78], i.e. IM outperformed  $LOO_{zero}$  and  $LOO_{unk}$ , which were implemented by replacing a word with the [PAD] and [UNK] token of BERT, respectively (Table 3.1). However, we hypothesize that  $Deletion_{BERT}$  is biased towards IM as both use BERT to replace words, yielding a false sense of IM effectiveness reported in [78].

To test this hypothesis, we add another baseline of  $LOO_{empty}$ , which was *not* included in [78], i.e. erasing a token from the input without replacement (Eq. 3.1), mirroring the original Deletion metric. To compare with IM, all LOO methods in this paper are also in the log-odds space.

**Results** Interestingly, we find that, under Deletion, on both SST-2 and e-SNLI, IM *underperformed* all three LOO baselines and that  $LOO_{empty}$  is the highest-performing method (Table 3.1a). In contrast, IM is the best method under  $Deletion_{BERT}$ .

Re-running the same experiment but sampling replacement words from RoBERTa (instead of BERT), we find the same finding that  $LOO_{empty}$  is the best under Deletion while IM is the best under  $Deletion_{BERT}$  (Table 3.1b).

To our knowledge, our work is **the first to document this bias** of the Deletion metric **widely used in the literature** [57, 160, 5]. This bias, in principle, also **exists in other Deletion variants** including Insertion [5], Sufficiency, and Comprehensiveness [31].

### 3.5 No evidence that IM is better than LOO

To avoid the critical bias of Deletion and  $Deletion_{BERT}$ , we further compare IM and LOO on **four** common metrics that are not Deletion-based.

Task	Metrics ↓	IM	LOO <sub>zero</sub>	LOO <sub>unk</sub>	LOO <sub>empty</sub>
(a) BERT <sub>base</sub>					
SST-2	Deletion	<b>0.4732</b>	0.4374	0.4464	<b>0.4241</b>
	Deletion <sub>BERT</sub>	<b>0.4922</b>	0.4970	0.5047	<b>0.5065</b>
e-SNLI	Deletion	<b>0.3912</b>	0.2798	0.3742	<b>0.2506</b>
	Deletion <sub>BERT</sub>	<b>0.2816</b>	0.3240	<b>0.3636</b>	0.3328
(b) RoBERTa <sub>base</sub>					
SST-2	Deletion	<b>0.4981</b>	0.4524	0.4595	<b>0.4416</b>
	Deletion <sub>BERT</sub>	<b>0.4798</b>	0.5037	<b>0.5087</b>	0.4998

Table 3.1: IM is the **best** method under Deletion<sub>BERT</sub>, as reported in [78], but the **worst** under Deletion. Both metrics measure AUC (lower is better).

### 3.5.1 Under ROAR and ROAR<sub>BERT</sub>, IM is on-par with or worse than LOO<sub>empty</sub>

A lower AUC under Deletion may be the artifact of the classifier misbehaving under the distribution shift when one or multiple input words are deleted. ROAR [65] was designed to ameliorate this issue by re-training the classifier on a modified training-set (where the top  $N\%$  highest-attribution tokens in each example are deleted) before evaluating their accuracy.

To more objectively assess IM, we use ROAR and ROAR<sub>BERT</sub> metrics to compare IM vs. LOO<sub>empty</sub> (i.e. the best LOO variant in Table 3.1).

**Experiment** For both IM and LOO<sub>empty</sub>, we generate AMs for every example in the SST-2 train and dev sets, and remove  $N\%$  highest-attribution tokens per example to create new train and dev sets. We train 5 models on the new training set and evaluate them on the new dev set. We repeat ROAR and ROAR<sub>BERT</sub> with  $N \in \{10, 20, 30\}$ .<sup>5</sup>

**Results** As more tokens are removed (i.e.  $N$  increases), the mean accuracy of 5 models gradually decreases (Table 3.2; from 92.66% to ~67%). Under both ROAR and ROAR<sub>BERT</sub>, the models trained on the new training set derived from LOO<sub>empty</sub> AMs often obtain lower (i.e. better) mean accuracy than those of IM (Table 3.2a vs. b). At  $N = 10\%$  under ROAR, **LOO<sub>empty</sub> outperforms**

<sup>5</sup>We do not use  $N \geq 40$  because: (1) according to SST human annotations, only 37% of the tokens per example are labeled important (Table 6.8c); and (2) SST-2 examples are short and may contain as few as 4 tokens per example.



**IM** (Table 3.2; 74.59 vs. 76.22), which is statistically significant (2-sample  $t$ -test,  $p = 0.037$ ). In all other cases, the difference between IM vs.  $\text{LOO}_{\text{empty}}$  is not statistically significant.

In sum, under both ROAR and  $\text{ROAR}_{\text{BERT}}$ , IM is *not more faithful* than  $\text{LOO}_{\text{empty}}$ .

Accuracy in % (lower is better)	ROAR				$\text{ROAR}_{\text{BERT}}$		
	$N = 0\%$	10%	20%	30%	10%	20%	30%
(a) $\text{LOO}_{\text{empty}}$	$92.62 \pm 0.30$	<b><math>74.59 \pm 0.78</math></b>	<b><math>68.94 \pm 1.46</math></b>	$67.89 \pm 0.79$	<b><math>76.79 \pm 0.56</math></b>	$71.95 \pm 0.75$	<b><math>67.62 \pm 1.16</math></b>
(b) IM	$92.62 \pm 0.30$	$76.22 \pm 1.18$	$70.07 \pm 0.69$	<b><math>66.54 \pm 1.89</math></b>	$77.36 \pm 0.90$	<b><math>71.56 \pm 1.55</math></b>	$67.68 \pm 0.96$
(c) Random	$92.62 \pm 0.30$	$89.22 \pm 0.53$	$87.75 \pm 0.19$	$85.62 \pm 0.53$	$89.38 \pm 0.47$	$88.23 \pm 0.31$	$85.21 \pm 0.47$
(d) $t$ -test p-value	N/A	<b>0.0370</b>	0.1740	0.1974	0.2672	0.6312	0.9245

Table 3.2: Dev-set mean accuracy (%) of 5 models trained on the new SST-2 examples where  $N\%$  of highest-attribution words per example are removed (i.e. ROAR) or replaced via BERT (i.e.  $\text{ROAR}_{\text{BERT}}$ ). On average, under both metrics,  $\text{LOO}_{\text{empty}}$  (a) is slightly better, i.e. lower mean accuracy, than IM (b). Notably,  $\text{LOO}_{\text{empty}}$  statistically significantly outperforms IM under ROAR at  $N = 10\%$  (2-sample  $t$ -test;  $p = 0.037$ ) (d). Both  $\text{LOO}_{\text{empty}}$  and IM substantially outperform a random baseline (c) that considers  $N\%$  random tokens important.

Metric $\uparrow$	(a) SST					(b) e-SNLI L2		(c) e-SNLI L3		(d) MultiRC	
	IM	$\text{LOO}_{\text{empty}}$	LIME	$\text{LIME}_{\text{BERT}}$	$\text{LIME}_{\text{BERT\_SST2}}$	IM	$\text{LOO}_{\text{empty}}$	IM	$\text{LOO}_{\text{empty}}$	IM	$\text{LOO}_{\text{empty}}$
IoU	0.2377	<b>0.2756</b>	0.3193	0.3170	0.3127	0.3316	<b>0.3415</b>	0.2811	<b>0.3411</b>	0.0437	<b>0.0887</b>
precision	<b>0.5129</b>	0.4760	0.4831	0.4629	0.4671	0.4599	<b>0.4867</b>	0.3814	<b>0.4687</b>	0.1784	<b>0.1940</b>
recall	0.5245	<b>0.6077</b>	0.6882	0.7000	0.6886	0.6085	<b>0.6158</b>	0.5699	<b>0.5875</b>	0.0630	<b>0.2876</b>
F1	0.5186	<b>0.5338</b>	0.5677	0.5573	0.5566	0.5239	<b>0.5437</b>	0.4570	<b>0.5214</b>	0.0931	<b>0.2317</b>

Table 3.3: Compared to IM,  $\text{LOO}_{\text{empty}}$  is substantially more consistent with human annotations over all three datasets. Note that the gap between  $\text{LOO}_{\text{empty}}$  and IM is  $\sim 3\times$  wider when comparing AMs with the e-SNLI tokens that at least three annotators label important (i.e. L3), compared to L2 (higher is better).  $\text{LIME}_{\text{BERT}}$  explanations are slightly less consistent with human highlights than those of LIME (a) despite their counterfactuals are more realistic.

### 3.5.2 $\text{LOO}_{\text{empty}}$ aligns significantly better with human annotations than IM

Following [160], to increase our understanding of the differences between  $\text{LOO}_{\text{empty}}$  and IM, we compare the two methods against the human-annotated highlights for SST, e-SNLI, and MultiRC.

**Annotation preprocessing** To control for quality, we preprocess the human annotations in each dataset as the following. In SST, where each sentence has multiple phrases labeled with a sentiment score  $\in [0, 1]$  (0.5 being the neutral midpoint), we only use the phrases that have high-confidence sentiment scores, i.e.  $\leq 0.3$  (for negative) or  $\geq 0.7$  (for positive). Also, we do not use the annotated phrases that are too long, i.e., longer than 50% of the sentence length.

Each token in an e-SNLI example are labeled important by between 0–3 annotators. To filter out noise, we only use the tokens that are highlighted by *at least* two or three annotators (hereafter “L2” and “L3” subsets, respectively).

A MultiRC example contains a question and a paragraph where each sentence is labeled important or unimportant to the groundtruth answer (Fig. 6.14). We convert these sentence-level highlights into token-level highlights to compare them with the binarized AMs of IM and LOO<sub>empty</sub>.

**Experiment** We run IM and LOO<sub>empty</sub> on the BERT-based classifiers on the dev set of SST, e-SNLI, and MultiRC. All AMs generated are binarized using a threshold  $\tau \in \{0.05x \mid 0 < x < 20 \text{ and } x \in \mathbb{N}\}$ . We compute the average IoU, precision, recall, and F1 over pairs of (human binary map, binarized AM) and report the results at the optimal  $\tau$  of each explanation method. For both LOO<sub>empty</sub> and IM,  $\tau = 0.1$  on SNLI-L2 and 0.05 on both SST-2 and MultiRC. On SNLI-L3,  $\tau$  is 0.40 and 0.45 for LOO<sub>empty</sub> and IM, respectively.

**SST results** We found that LOO<sub>empty</sub> aligns better with human highlights than IM (Figs. 3.2 & 6.16). LOO<sub>empty</sub> outperforms IM in both F1 and IoU scores (Table 3.3a; 0.2756 vs 0.2377) with a notably large recall gap (0.6077 vs. 0.5245).

SST Groundtruth & Prediction: positive movie review	
Input	Mr. Tsai is a very original artist in his medium , and What Time Is It There ?
IM	Mr. Tsai is a very original artist in his medium , and What Time Is It There ? IoU: 0.17, precision: 0.33, recall: 0.25
LOO	Mr. Tsai is a very original artist in his medium , and What Time Is It There ? IoU: <b>0.80</b> , precision: <b>0.80</b> , recall: <b>1.00</b>

Figure 3.2: LOO<sub>empty</sub> binarized attribution maps align better with human highlights than IM maps.

**e-SNLI and MultiRC results** Similarly, in both tasks,  $LOO_{empty}$  explanations are more consistent with human highlights than IM explanations under all four metrics (see Table 3.3b–d and qualitative examples in Figs. 3.3 & 6.17–6.20).

Remarkably, in MultiRC where each example is substantially longer (~299 tokens per example) than those in the other tasks, the recall and F1 scores of  $LOO_{empty}$  is, respectively, 2× and 4× higher than those of IM (see Table 3.3).

e-SNLI example. Groundtruth & Prediction: entailment	
P	Two men dressed in black practicing martial arts on a gym floor .
H	Two men are doing martial arts .
IM	Two men dressed in black practicing martial arts on a gym floor . Two men are doing martial arts .
	IoU: 0.09, precision: 0.17, recall: 0.16
LOO	Two men dressed in black practicing martial arts on a gym floor . Two men are doing martial arts .
	IoU: <b>0.50</b> , precision: <b>0.56</b> , recall: <b>0.83</b>

Figure 3.3:  $LOO_{empty}$  important words are in a stronger agreement with human highlights than IM important words. Each e-SNLI example contains a pair of premise (P) and hypothesis (H).

### 3.5.3 IM is insensitive to model randomization

[1] found that many attribution methods can be surprisingly biased, i.e. *insensitive* to even randomization of the classifier’s parameters. Here, we test the degree of insensitivity of IM when the last classification layer of BERT-based classifiers is randomly re-initialized. We use three SST-2 classifiers and three e-SNLI classifiers.

Surprisingly, IM is consistently worse than  $LOO_{empty}$ , i.e. more insensitive to classifier randomization. That is, on average, the IM attribution of a word changes signs (from positive to negative or vice versa) less frequently, e.g. 62.27% of the time, compared to 71.41% for  $LOO_{empty}$  on SST-2 (Table 6.11a). The average change in attribution *magnitude* of IM is also ~1.5× smaller than that of  $LOO_{empty}$  (Table 6.11b).

For example, the IM attribution scores of **hot**, **air** or **balloons** in Fig. 3.1 remain consistently **unchanged near-zero even when the classifier is randomized three times**. That is, each of these three words is  $\sim 100\%$  predictable by BERT given the other two words (Fig. 3.1b;  $IM_1$  to  $IM_3$ ) and, hence, will be assigned a near-zero attribute by IM (by construction, via Eqn. 3.3 & 3.4) regardless of how important these words actually are to the classifier. Statistically, this is a major issue because across SST, e-SNLI, and MultiRC, we find BERT to correctly predict the missing word  $\sim 49, 60, 65\%$  of the time, respectively (Sec. 6.2.1). And that the average likelihood score of a top-1 exact-match token is high,  $\sim 0.81\text{--}0.86$  (Sec. 6.2.2), causing the highly predicted words (e.g., **hot**) to always be assigned low attribution regardless of their true importance to the classifier.

We find this insensitivity to be a major, **theoretical flaw of IM** in explaining a classifier’s decision at the *word* level. By analyzing the overlap between IM explanations and human highlights (generated in experiments in Sec. 3.5.2), we find consistent results that IM explanations have **significantly smaller attribution magnitude** per token (Sec. 6.2.1) and **substantially lower recall than LOO** (Sec. 6.2.2).

### 3.5.4 Classification accuracy only drops marginally when one token is deleted

Our previous results show that replacing *a single word* by BERT (instead of deleting) in IM creates more realistic inputs but actually hurts the AM quality w.r.t. LOO. This result interestingly contradicts the prior conclusions [78, 56] and assumptions [57] of the superiority of IM over LOO.

To understand why using more plausible counterfactuals did not improve AM explainability, we assess the  $\Delta$  drop in classification accuracy when a word is deleted (i.e.,  $LOO_{\text{empty}}$  samples; Fig. 6.21) and the  $\Delta$  when a word is replaced via BERT (i.e. IM samples).

**Results** Across SST, e-SNLI, and MultiRC, the accuracy scores of classifiers only drop marginally  $\sim 1\text{--}4$  points (Table 3.4) when a single token is deleted. See Figs. 6.21 & 6.22 for qualitative examples showing that deleting a single token hardly changes the predicted label. Whether a word is removed or replaced by BERT is almost unimportant in tasks with long examples such as MultiRC

(Table 3.4; 1.10 and 0.24). In sum, we do not find the unnaturalness of LOO samples to substantially hurt model performance, questioning the need raised in [57, 56, 78] for realistic counterfactuals.

$\Delta$ drop in accuracy (%)		SST	e-SNLI	MultiRC
(a) LOO	(1-token deleted)	3.52	4.92	1.10
(b) IM	(1-token replaced)	2.20	4.86	0.24
(c) LIME	(many tokens deleted)	16.38	25.74	17.85

Table 3.4: The dev-set accuracies on SST, e-SNLI and MultiRC (87.83%, 90.92%, and 69.10%, respectively) only drop marginally when a single token is deleted (a) or replaced using BERT (b). In contrast, LIME samples cause the classification accuracy to drop substantially (e.g. 16.38 points on SST).

### 3.6 Replacing (instead of deleting) *multiple* words can improve explanations

We find that deleting a single word only marginally affects classification accuracy. Yet, deleting  $\sim 50\%$  of words, i.e. following LIME’s counterfactual sampling scheme, actually substantially reduces classification accuracy, e.g.  $-16.38$  point on SST and  $-25.74$  point on e-SNLI (Table 3.4c). Therefore, it is interesting to test whether the core idea of harnessing BERT to replace words has merits in improving LIME whose counterfactuals are extremely OOD due to many missing words.

#### 3.6.1 LIME<sub>BERT</sub> attribution maps are *not* more aligned with human annotations

Similar to Sec. 3.5.2, here, we compare LIME and LIME<sub>BERT</sub> AMs with human SST annotations (avoiding the Deletion-derived metrics due to their bias described in Sec. 3.4).

**Experiment** We use the default hyperparameters of the original LIME [133] for both LIME and LIME<sub>BERT</sub>. The number of counterfactual samples was 1,000 per example.

**Results** Although LIME<sub>BERT</sub> counterfactuals are more natural, the derived AMs are surprisingly less plausible to human than those generated by the original LIME. That is, compared to human annotations in SST, LIME<sub>BERT</sub>’s IoU, precision and F1 scores are all slightly worse than those of LIME (Table 3.3a). Consistent with the IM vs. LOO<sub>empty</sub> comparison in Sec. 3.5.2, replacing one

or more words (instead of deleting them) using BERT in LIME generates AMs that are similarly or less aligned with humans.

To minimize the possibility that the pre-trained BERT is suboptimal in predicting missing words on SST-2, we also finetune BERT using the mask-language modeling objective on SST-2 (see details in Sec. 6.2.3) and repeat the experiment in this section. Yet, interestingly, we find the above conclusion to not change (Table 3.3a;  $\text{LIME}_{\text{BERT\_SST2}}$  is worse than LIME). In sum, for both LOO and LIME, we find **no evidence that using realistic counterfactuals from BERT causes AMs to be more consistent with words that are labeled important by humans.**

### 3.6.2 $\text{LIME}_{\text{BERT}}$ consistently outperforms LIME under three ROAR metrics

To thoroughly test the idea of using BERT-based counterfactuals in improving LIME explanations, we follow Sec. 3.5.1 and compare  $\text{LIME}_{\text{BERT}}$  and LIME under three ROAR metrics: (1) ROAR; (2)  $\text{ROAR}_{\text{BERT}}$ ; and (3)  $\text{ROAR}_{\text{BERT\_SST2}}$ , i.e. which uses the BERT finetuned on SST-2 to generate training data.

**Experiment** Similar to the previous section, we take the dev set of SST-2 and generate a LIME AM and a LIME-BERT AM for each SST-2 example. For  $\text{ROAR}_{\text{BERT\_SST2}}$ , we re-use the BERT finetuned on SST-2 described in Sec. 3.6.1.

**Results** Interestingly, we find that  $\text{LIME}_{\text{BERT}}$  slightly, but consistently outperforms LIME via all three ROAR metrics tested (Fig. 3.4; dotted lines are above solid lines). That is,  $\text{LIME}_{\text{BERT}}$  tends to highlight more discriminative tokens in the text than LIME, yielding a better ROAR performance (i.e. lower accuracy in Table 6.12). This result is consistent across all three settings of removing 10%, 20%, and 30% most important words, and when using either pre-trained BERT or BERT finetuned on SST-2.

## 3.7 Discussion and Conclusion

We find in Sec. 3.5.3 that IM is highly insensitive to classifier’s changes because, by design, it always assigns near-zero attribution to highly-predictable words  $x_i$  regardless of their true

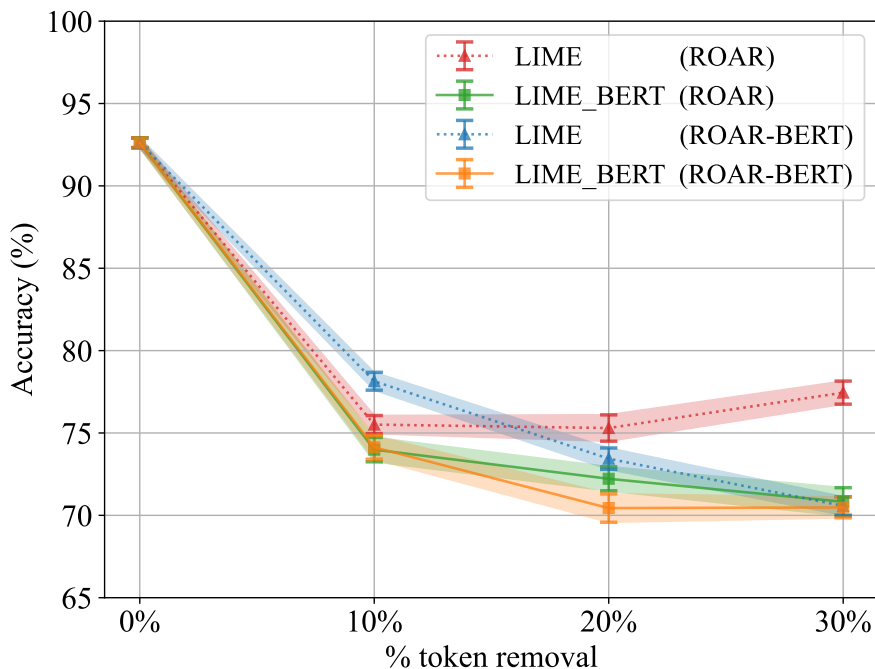


Figure 3.4: LIME<sub>BERT</sub> slightly, but consistently outperforms LIME when evaluated under either ROAR or ROAR<sub>BERT</sub>. The each point in the  $y$ -axis shows the mean accuracy of five different classifiers. See more results supporting the same conclusion in Table 6.12.

importance to a target classifier. A solution may be to leave such  $x_i$  token out of the marginalization (Eq. 3.3), i.e. only marginalizing over the other tokens suggested by BERT. However, these other replacement tokens altogether have a sum likelihood of 0. That is, replacing token  $x_i$  by zero-probability tokens (i.e. truly implausible) would effectively generate OOD text, which, in turn is not desired [57].

Our results in Sec. 3.6.2 suggests that IM might be more useful at the *phrase* level [71] instead of *word* level as deleting a set of contiguous words has a larger effect to the classifier predictions.

In sum, for the first time, we find that the popular idea of harnessing BERT to generate realistic counterfactuals [57, 56, 78] does not actually improve upon a simple LOO<sub>empty</sub> in practice as an LOO<sub>empty</sub> counterfactual only has a single word deleted. In contrast, we observe more expected benefits of this technique in improving methods like LIME that has counterfactuals that are extremely syntactically erroneous when multiple words are often deleted.

## Chapter 4

### PiC: A Phrase-in-Context Dataset for Phrase Understanding and Semantic Search

#### 4.1 Overview

Understanding phrases in context is a key to learning new vocabularies [100, 41], disambiguation [122], and many downstream tasks, including semantic search [40]. Yet, the contextualized *phrase* embeddings [175] in existing systems mostly capture the common meaning of a phrase, i.e. without strong dependence on its context [175]. While there are *word*-sense disambiguation datasets [34, 122], no such benchmarks exist for *phrases*. Existing phrase-similarity benchmarks [112, 150, 6, 180, 173] compare phrases alone (without context) and some of them [112, 180] contain a large, undesired amount ( $\sim 15\%$  to  $99\%$ ) of phrase pairs that have lexical overlap (Table 4.1).

Others generated the context for a phrase by querying GPT-2 [157] or by retrieving from Wikipedia [175]. Yet, there was no human verification of the realism of generated text [157] and no human annotation of how a phrase’s meaning changes w.r.t. the context [175]. All above drawbacks are limiting the evaluation of phrase understanding.

To advance the development of contextualized phrase embeddings, we propose Phrase-in-Context (PiC), a suite of three tasks: (1) Phrase Similarity (PS), i.e. compare the semantic similarity of two phrases in the same context sentence (fig. 4.1b); (2) Phrase Retrieval (PR), which is divided into PR-pass and PR-page (fig. 4.1c–d), i.e. from a passage or a Wikipedia page, retrieve a phrase semantically-similar to a given query phrase; and (3) Phrase-Sense Disambiguation (PSD), i.e. find the target phrase  $p$  semantically similar to the query phrase from a 2-paragraph document where  $p$  appears twice, each time in a different context paragraph that provides a *unique* meaning to  $p$  (fig. 4.1e). Our  $\sim 28\text{K}$ -example dataset is rigorously (a) *annotated* and *verified* by two groups of annotators: linguistics experts on Upwork.com and non-experts on Amazon Mechanical Turk (MTurk); and then (b) *tested* by models, linguists, and graduate students. Our contributions are:



1. We build PiC<sup>1</sup>, the first, human-annotated benchmark for evaluating and training contextualized phrase embeddings (section 4.4). Compared to existing phrase similarity datasets, PS is the first to require models to rely on context.
2. After training on PR-pass, i.e. finding a phrase from a passage, SS models perform at a near-human accuracy (92–94% vs. 95% EM). They also score high (84–89% EM) on PR-page, i.e. semantic phrase search in a Wikipedia page (Sec. 4.5.4), suggesting our training set and learned embeddings are useful for real-world semantic search.
3. Interestingly, on PR-pass, harnessing these SS models’ phrase embeddings in a ranking approach (i.e. comparing the similarity between the query and *all* candidate phrases) yields poor accuracy of  $\leq 59\%$  EM (Sec. 6.3.8), setting a challenge for future research into learning contextualized phrase embeddings.
4. After training on PR-pass, state-of-the-art (SotA) models perform relatively well on PR-pass and even PR-page but not on PSD (Sec. 4.5.5). On PS, SotA models perform poorly (below 70% accuracy) in binary classification of phrase similarity given a context sentence (section 4.5.1).

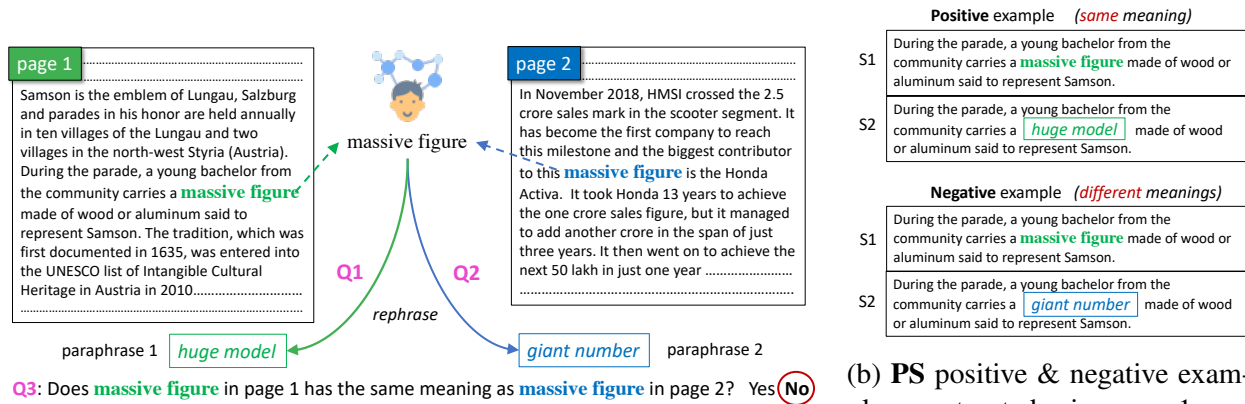
## 4.2 Related Work

Each of our tasks (PS—phrase similarity; PR—phrase retrieval; and PSD—disambiguation) is related to a separate research area discussed below.

**Phrase similarity** First, most existing phrase similarity datasets—e.g. PPDB-annotated [161], PPDB-filtered [157], BiRD [6], and PAWS-short [157, 180]—contain a large percent of instances with lexical overlap between two paired phrases while our PS contains the least percent (5.34%; table 4.1). Second, PS compares each pair of phrases in a context sentence while existing datasets only compare phrases alone (no context). Third, the phrases in PS are, on average, 2-token long,

---

<sup>1</sup>Dataset, code, and demos are available on <https://phrase-in-context.github.io>.



(a) Q1 & Q2 ask annotators to rephrase “massive figure” in page 1 and page 2. Q3 asks whether this phrase’s meaning is the same in both pages. (b) PS positive & negative examples constructed using page 1 context (similarly, we repeat for page 2).

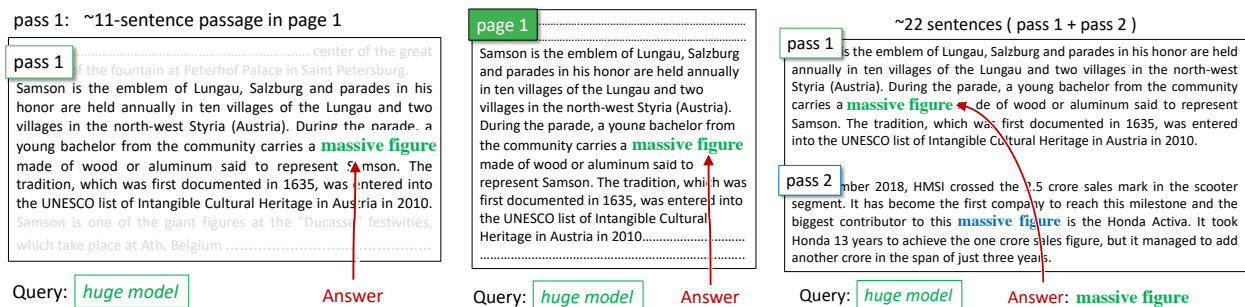


Figure 4.1: Given a phrase, two associated Wikipedia pages, and expert annotations, i.e. answers to Q1, Q2, and Q3 (a), we are able to construct *two* pairs of positive and negative examples for PS (b), a PR-pass example (c), a PR-page example (d), and a PSD example *only if* the answer to Q3 is No (e).

comparable to that of other datasets (table 4.1). Fourth, unlike other datasets, PS contains exclusively *noun-phrases*, the most common phrase type according to Yahoo’s search-query statistics [168] (79.54%; section 6.3.4) and Adobe (internal Acrobat Pro data not shown).

**Question answering (QA)** Our phrase-retrieval tasks—PR and PSD—follow the format of QA datasets except that our queries are *phrases* instead of questions and hence shorter (table 4.2). Like SQuAD 1.1 [129] and HotpotQA [174], our documents and queries are extracted from Wikipedia articles. While our PR dataset is  $\sim 3.5\times$  smaller than those two datasets, the paragraph document length in PR-pass and PSD is  $\sim 2\times$  longer than those of SQuAD 1.1 and HotpotQA (table 4.2).

Table 4.1: Our Phrase Similarity (PS) dataset has a lower percent of lexical-overlap instances and is the *only* human-annotated dataset that provides phrases, each in a context sentence.

	<b>PS</b> <b>(ours)</b>	WiC	PPDB- annotated	PPDB- filtered	BiRD	Turney	PAWS- short
# of All instances	10,004	7,466	3,000	15,532	3,345	2,180	1,214
# of Unique phrases	7,488	2,345	6,000	12,023	2,840	9,776	1,214
Lexical overlap (%)	5.34	100	70.10	97.93	14.98	0	99.42
Mean length (in tokens)							
• phrase <sub>1</sub>	2.06	1	3.67	2	2	2	9.52
• phrase <sub>2</sub>	2.46	1	3.73	2	1.49	1	9.42
• context sentence	22.53	8.40	0	0	0	0	0

For our task, intuitively, the longer the document, the harder the task since there would be more candidates a model must compare with the query.

Table 4.2: Our PR-pass, PR-page and PSD datasets are smaller in size compared to common QA datasets and contain shorter queries that are noun phrases instead of questions. However, our tasks require searching in much longer documents.

	<b>PR-pass</b>	<b>PR-page</b>	<b>PSD</b>	SQuAD 1.1	HotpotQA
All instances	28,147	28,098	4,858	98,169	105,257
Unique queries/questions	27,055	27,016	4,812	97,888	105,249
Unique answers	13,458	13,423	2,314	72,469	57,259
Mean length of					
query (tokens)	2.42	2.42	2.45	11.42	20.03
answer (tokens)	2.17	2.17	2.07	3.46	2.35
sentence (tokens)	23.22	24.08	23.00	27.62	26.77
document (sentences)	10.26	119.32	20.37	5.10	4.14
document (tokens)	238.34	2,872.73	468.48	140.92	110.72

**Sense disambiguation** While word-sense disambiguation (WSD) is a long-standing problem in NLP, recently, SotA models have reached super-human accuracy (80% F<sub>1</sub>) on the common English WSD [11]. Interestingly, these high-scoring models still struggle with rare senses that may be outside of the predefined sense inventories or have few training examples [13]. Without the need for predefined senses, WiC [122] poses disambiguation as a binary classification task where the goal

is to predict whether the same target word in two different sentences carries the same or different meanings.

**Compared to WiC** PS is also a binary classification task, but with two major differences: (1) in WiC, the same target word appears in two different sentences while in PS, two different phrases appear in the same context sentence; (2) PS compares phrases composed of  $\geq 2$  words instead of a single word as in WiC and WSD. While word senses are defined in WordNet and BabelNet dictionaries [11], there are no English dictionaries of senses for multi-word noun phrases (*mNPs*). Thus, it is more challenging to acquire and learn the senses of *mNPs*, hence the importance of our PiC dataset. Like WiC, PSD tests disambiguating the meanings of the same *n*-gram in two different contexts. Yet, PSD is a phrase search task, which involves many more phrase comparisons per example than PS or WiC.

Before the deep learning era, phrase-sense disambiguation was already proposed [18, 19] but only as an auxiliary task for training machine-translation models. And their phrase senses were not annotated by humans but inferred by performing word-alignment on a bilingual corpus. Here, our PSD is the first phrase-sense disambiguation task annotated by experts and requires understanding of phrase-senses in a passage.

### 4.3 PiC Dataset Construction

We first collect a set of phrases with context and human annotations. Then, we derive the examples and labels for three main tasks: PS, PR, and PSD (Fig. 4.1). Our idea is to mine a set of triplets ( $\mathbf{p}$ ,  $\text{page}_1$ ,  $\text{page}_2$ ) from Wikipedia where the phrase  $\mathbf{p}$  is a polysemous *mNP* that carries two different senses in two Wikipedia pages (e.g., “massive figure” means *a large number* in  $\text{page}_1$  but *a huge physical shape* in  $\text{page}_2$ ; fig. 4.1a). Then, we ask experts to rephrase  $\mathbf{p}$  into two paraphrases  $\mathbf{q}_1$  and  $\mathbf{q}_2$ , maintaining the two original senses of  $\mathbf{p}$  in  $\text{page}_1$  and  $\text{page}_2$ , respectively. The resultant set of 5-tuples ( $\mathbf{p}$ ,  $\mathbf{q}_1$ ,  $\mathbf{q}_2$ ,  $\text{page}_1$ ,  $\text{page}_2$ ) enables the tests for (1) comparing the semantic similarity of two phrases given the same context sentence (PS; fig. 4.1b); (2) finding a semantically similar

phrase in a document (PR-pass & PR-page; fig. 4.1c); (3) disambiguating the senses of the same target  $mNP$  given two context paragraphs (PSD; fig. 4.1e).

### 4.3.1 Data Collection

As there are no English dictionaries that contain sense inventories for  $mNPs$ , the key **challenge** to our data collection is to mine  $mNPs$  that have (1) multiple senses; and (2) a Wikipedia context page for each sense. To do that, we take a Wikipedia dump and perform a **6-step** procedure that essentially extracts all the  $mNPs$  that occur in more than one Wikipedia page and that contain at least one polysemous word defined in the WiC dataset. From the triplets of  $(p, page_1, page_2)$ , we programmatically narrow down to  $\sim 600K$  triplets where the context sentence of the  $mNP$  in  $page_1$  is the most semantically *dissimilar* to the context sentence in  $page_2$  (according to SimCSE [45]). We continue filtering down to the top 19,500 triplets where  $page_1$  and  $page_2$  have *the most* semantically dissimilar lists of Wikipedia categories. That is, 19,500 triplets are estimated to yield  $\sim 15K$  annotated triplets (the target size based on our budget) after the human annotation process where annotators are allowed to skip the cases they are not confident labeling. See section 6.3.3 for a detailed description of the data collection and **dataset biases**.

### 4.3.2 Data Annotation

Via Upwork, we hire 13 linguistics experts who are native English speakers at a rate of \$30/hour to annotate 15,021 out of 19,500 examples. For each phrase, we provide Upworkers with a triplet  $(p, passage_1, passage_2)$  where each  $passage_i$  consists of 5 sentences centered at the phrase-containing sentence in the corresponding  $page_i$ . We ask them to answer the three below questions (fig. 4.1a):

Q1 Rephrase the target phrase  $p$  to a paraphrase  $q_1$  such that its meaning is constant in  $passage_1$ .

Q2 Similarly, rephrase  $p$  w.r.t.  $passage_2$  to obtain a paraphrase  $q_2$ .

Q3 Answer Y/N if  $p$  has the same meaning in both contextual  $passage_1$  and  $passage_2$ .

Upworkers are asked to provide paraphrases that (1) have at least two words and (2) minimize lexical overlap with each other and the target  $p$ . See the annotation guidelines [119] and a sample annotation assignment [120] given to Upworkers. After receiving annotations, we use [85] to automatically find syntactical errors when the paraphrases are replaced by the original target phrase in the original passage and ask Upworkers to fix them. We also have annotators fix the remaining errors that we find via manual inspection.

### 4.3.3 Annotation Verification

To verify the annotations obtained in section 4.3.2 (i.e.  $2 \times 15,021 = 30,042$  paraphrases; and 15,021 Y/N labels), first, we present the same Q1, Q2, and Q3 questions to 1,000 qualified MTurkers and ask whether they agree with the answers by expert annotators in section 4.3.2. And then, for the cases that the MTurkers disagree with, we seek second opinions from 5 Upwork experts. After these two verification rounds, we discard all the examples where Upwork verifiers reject and arrive at the final 28,325 paraphrases and 13,413 Y/N labels (i.e. those annotations that *either* an MTurk or Upwork verifier endorses). See more details in section 6.3.10.

The total fee for both MTurk and Upwork combined is around USD 30,000.

## 4.4 Three Phrase Understanding Tasks

Using the human-annotated data, we construct three tasks of PS, PR, and PSD (as summarized in fig. 4.1) for evaluating contextualized phrase-embeddings and semantic-search models.

### 4.4.1 Phrase Similarity (PS)

PS is a binary classification task that asks whether two  $m$ NPs are semantically similar or not given *the same context* sentence. The unique challenge of PS is that, *without* context, the two given phrases can be easily interpreted as synonymous. Yet, in our PS context sentence, the two phrases *may* or *may not* carry distinct meanings (fig. 4.1b).

**Construction** From the annotated data, a positive example is a triplet of (an original phrase  $p$ , a paraphrase  $q_1$ , an original page<sub>1</sub>'s sentence that contains  $p$ ). To create a negative example, from the same triplets, we select only those where the paraphrase  $q_2$  holds a *different* meaning than  $q_1$  given the page<sub>1</sub> context of  $q_1$  (i.e., when the answer to Q3 is No; see fig. 4.1b). For quality assurance, we also hire three extra Upwork experts to double-check PS annotations (see section 6.3.5), keeping only examples that at least 2 out of 3 experts endorse. In total, we obtain 5,002 *negative* examples. Then, we randomly select 5,002 *positive* examples to form a class-balanced PS dataset.

#### 4.4.2 Phrase Retrieval (PR)

PR is a task of finding in a given document  $d$  a phrase  $p$  that is semantically similar to the given query phrase, which is the paraphrase  $q_1$  (the answer by annotators to Q1) or  $q_2$  (the answer to Q2). We release two versions of PR: **PR-pass** and **PR-page**, i.e. datasets of triplets (query  $q_1$ , target phrase  $p$ , document  $d$ ) where  $d$  is a random 11-sentence passage that contains  $p$  (fig. 4.1c) or an entire Wikipedia page (fig. 4.1d). While PR-pass contains 28,147 examples, PR-page contains slightly fewer examples (28,098) as we remove those examples whose Wikipedia pages coincidentally also contain exactly the query phrase (in addition to the target phrase). Both datasets are split into  $\sim 20\text{K}/3\text{K}/5\text{K}$  for train/dev/test, respectively.

#### 4.4.3 Phrase Sense Disambiguation (PSD)

The task is to find the location of the target phrase  $p$  where it has a similar meaning to that of the given query  $q$  in a 2-paragraph document where, by construction,  $p$  appears exactly twice but only one location is the correct answer (fig. 4.1e).

**Construction** From the verified annotations in section 4.3.3, there are in total 4,938 phrases that both annotators and verifiers agree to hold *different* meanings across the two context Wikipedia pages (i.e., “No” answer to Q3 in fig. 4.1a). To create a PSD example, given a phrase  $p$  from the above 4,938, we extract two corresponding  $\sim 11$ -sentence paragraphs (from its associated page<sub>1</sub> and page<sub>2</sub> as in PR-pass) and concatenate them (separated by an empty line) into a single document

(fig. 4.1e). Since a PSD example shares a pair of phrases (*query* and *answer*) with one PS *positive* example (*phrase<sub>1</sub>* and *phrase<sub>2</sub>*), we filter out that PSD example if the corresponding PS example is removed from the additional verification round (section 6.3.5). As the result, we exclude 80 examples and obtain 4,858 examples in total for PSD.

## 4.5 Experiments and Results

We test SotA models on PS, PR-pass, PR-page, and PSD to (1) assess how the models are able to leverage context to improve accuracy; and (2) quantify the headroom for future research.

**Phrase embeddings** Besides training and testing SotA BERT-based classifiers, we also test a ranking approach that involves computing the cosine similarity between the query’s and each candidate’s embedding. To compute a *contextualized* phrase embedding, following [175], we feed the entire phrase-containing sentence (e.g.  $S_1$  in fig. 4.1b) into a model, e.g. BERT, and then take the mean pooling of the last-layer embeddings over the words of the given phrase only. For non-contextualized phrase embeddings, we repeat the same process but input to the model only the phrase (instead of the entire sentence).

**Models** We choose SotA models in (a) phrase similarity: PhraseBERT [157]; (b) sentence similarity: USE-v5 [21], SentenceBERT [131], and SimCSE [45]); (c) question-answering: Longformer [8], DensePhrase [86]; and (d) contextualized embeddings: SpanBERT [72] and BERT [30].

For DensePhrase, we use their Phrase-Encoder (as opposed to the Query-Encoder) to compute phrase embeddings. USE-v5 is only available via public APIs [148] that do not support extraction of contextualized phrase embeddings.

### 4.5.1 Phrase Similarity: Contextualized phrase embeddings improve accuracy

**Q:** *Does incorporating context improve the phrase-similarity accuracy on PS?*

**Experiment** We split the PS dataset 70/10/20 for train/dev/test and test two approaches: (1) using the cosine similarity score between two pre-trained phrase-embeddings (with and without context)



to predict phrase similarity; (2) training BERT-based binary classifiers directly using PS training set. We use 6 backbone BERT models that are all “base” versions unless specified otherwise (table 4.3).

**Approach 1: Cosine similarity** First, we test how *pre-trained* phrase embeddings alone (without finetuning or extra weights) can be leveraged to solve PS. For each PS example of two phrases, we compute their non-contextualized phrase embeddings and compute their cosine similarity score. To evaluate the pre-trained embeddings on PS, we follow [171] and tune the binary-classification threshold  $T$  to maximize the training-set accuracy, and then use the same optimal  $T$  to report the test-set accuracy. We repeat the experiment for *contextualized* phrase embeddings.

**Approach 2: BERT-based classifiers** To complement Approach 1, we test Approach 2, i.e. building a binary classifier by adding two extra MLP layers on top of the pre-trained embeddings used in Approach 1. For a phrase pair, we concatenate the two 768-D phrase embeddings from BERT<sub>base</sub> into a 1,536-D vector, and then place one ReLU layer (256 units) and a 1-output linear classification layer with sigmoid on top. Following [157], we finetune these models for a maximum of 100 epochs (with early stopping and patience of 10 epochs) on the train set. See section 6.3.1 for more training details.

**Results** *Without* context, all models perform at  $\leq 50\%$  accuracy (i.e. the random chance; table 4.3a & c). Interestingly, incorporating context information into phrase embeddings substantially improves mean model-accuracy on PS for both Approach 1 (from 50.83% to 63.43%; table 4.3b vs. a) and Approach 2 (from 35.40% to 66.71%; table 4.3d vs. c), showing evidence that **PS requires models to rely on context**. While starting from the same backbone models, Approach 2 yields higher mean accuracy than Approach 1 (table 4.3; 66.71 vs. 63.43), which is expected as Approach 2 models have more capacity and the backbones are allowed to be finetuned on PS. See Figs. 6.25–6.28 for qualitative PS predictions from a PhraseBERT-based classifier.

Table 4.3: Accuracy (%) of state-of-the-art BERT-based models on the PS test set. Contextualized phrase embeddings (“Phrase + Ctx”) yield substantially higher performance on PS than non-contextualized embeddings (“Phrase”). The random baseline is 50%.

Model	<i>Approach 1:</i> <i>Cosine similarity</i>		<i>Approach 2:</i> <i>BERT-based classifiers</i>	
	(a) Phrase	(b) Phrase + Ctx	(c) Phrase	(d) Phrase + Ctx
PhraseBERT	51.75	63.40 (+11.65)	33.60	66.10 (+32.50)
BERT	51.05	64.10 (+13.05)	37.00	68.85 (+31.85)
SpanBERT	49.30	64.00 (+14.70)	40.15	66.85 (+26.70)
SpanBERT <sub>Large</sub>	50.40	<b>66.30 (+15.90)</b>	35.95	<b>69.25 (+33.30)</b>
SentenceBERT	50.35	60.30 (+9.95)	31.50	62.55 (+31.05)
SimCSE	52.15	62.50 (+10.35)	34.20	66.65 (+32.45)
mean $\pm$ std	50.83 $\pm$ 1.04	<b>63.43 <math>\pm</math> 1.98</b>	35.40 $\pm$ 3.01	<b>66.71 <math>\pm</math> 2.40</b>

#### 4.5.2 Human Baselines and Upperbound (95% Exact Match) on Phrase Retrieval

To interpret the progress of machine phrase-understanding on PR, here, we establish multiple human baselines for both non-experts and linguistics experts (with and without training them).

**Experiment** We recruit participants and have them perform one or two tests per person. A test consists of 20 PR-pass examples. That is, PR-pass documents are 11-sentence long and are feasible for a person to read in minutes (compared to reading an entire Wikipedia page). We test three groups: (1) 21 graduate students at our institution (1 test per person); (2) five Upwork experts (1 test per person); and (3) another five Upwork experts (2 tests per person, i.e., for a total of  $2 \times 5 = 10$  tests). The students in Group 1 volunteer to help our study unpaid while the Upworkers (Group 2 and 3) are hired using the same procedure as in section 4.3.2.

**Results** First, we find an unsurprising, large gap between non-experts and experts (table 4.4; 73.60% vs. 82.00%). Second, we train experts in Group 3 by having each do a preliminary test and giving them feedback before the real test. We find the training to substantially boost expert accuracy further (from 82.00% to 90.50%). Importantly, we find the Human Exact Match (EM) Upperbound to be 95%, i.e. the highest scores that 4 people (among all groups) make. Upon manual inspection

Table 4.4: Best SS models reach near the Upperbound (95%) on PR-pass. Yet, ranking models based on phrase embeddings significantly *underperform* SS models.

Accuracy of human groups and models	EM (%)
Group 1: 20 Non-experts (w/o training)	73.60 $\pm$ 7.90
Group 2: 05 Experts (w/o training)	82.00 $\pm$ 12.00
Group 3: 05 Experts (w/ training)	<b>90.50</b> $\pm$ 3.70
Best human accuracy (4 people)—Upperbound	<b>95.00</b> $\pm$ 0.00
Best untrained, ranking model (BERT)	47.44
Best PR-trained, ranking model (PhraseBERT)	59.02
Best PR-trained, SS model (Longformer <sub>Large</sub> )	<b>94.28</b>

of the submissions of these best performers, we find their incorrect answers sometimes partially overlap with the groundtruth or are sometimes reasonable. In other cases, the best performers find acceptable answers but that do not overlap at all with the groundtruth labels in PR. That is, we estimate a 5% of noise in the annotations of PR.

### 4.5.3 Phrase Retrieval: In ranking, context only helps BERT embeddings but not others

One way to evaluate the quality of SotA phrase embeddings is by testing:

**Q:** *How well do phrase embeddings perform in the **ranking** approach on PR?*

Ranking is a challenging and meaningful phrase-embedding test because the embedding of the query is compared against that of all phrase candidates (extracted by tokenizing the document), which can include syntactically-incorrect phrases, meaningless phrases or rare phrases. Such out-of-distribution challenge appears less often in PS or WiC, i.e. a binary classification setting.

**Experiment** As described in section 4.4.2, the PR train/dev/test splits are 20,147/3K/5K examples and we only use the 5K-example test set to test the models in this ranking experiment (no training). We follow [87] for span enumeration to construct a list of candidate phrases, we split each PR document into multiple sentences (using NLTK sentence splitter) and tokenize each sentence into tokens (using NLTK tokenizer) and build an exhaustive list of  $n$ -grams (here,  $n \in \{2, 3\}$  only for computational tractability). For every example, we add the groundtruth phrase (which can be longer

Table 4.5: **Ranking** accuracy (%) on **PR-pass** using the state-of-the-art pretrained phrase embeddings. See section 6.3.6 for the results on PR-page.  $\Delta$  (e.g. **-3.62**) denotes the differences between the Top-1 accuracy in the contextualized (“Phrase + Context”) vs. the non-contextualized (“Phrase”) setting.

Model	Phrase				Phrase + Context			
	Top-1	Top-3	Top-5	MRR@5	Top-1 ( $\Delta$ )	Top-3	Top-5	MRR@5
PhraseBERT [157]	36.62	66.96	75.90	52.20	33.00 (-3.62)	49.60	56.70	41.90
BERT [30]	29.80	47.90	55.40	39.50	<b>47.44 (+17.64)</b>	<b>65.78</b>	<b>73.30</b>	<b>57.30</b>
BERT <sub>Large</sub> [30]	23.76	38.52	45.40	31.70	<b>42.80 (+19.04)</b>	<b>58.90</b>	<b>64.90</b>	<b>51.30</b>
SpanBERT [72]	20.88	31.04	35.20	26.40	14.40 (-6.48)	30.46	39.80	23.40
SentenceBERT [131]	22.30	50.64	60.60	36.80	<b>25.14 (+2.84)</b>	39.52	46.20	32.90
SimCSE [45]	28.10	53.70	64.60	41.60	<b>32.40 (+4.30)</b>	53.44	62.80	<b>43.70</b>
USE-v5 [21]	<b>43.36</b>	<b>70.12</b>	<b>78.90</b>	<b>57.30</b>	n/a	n/a	n/a	n/a
DensePhrase [86]	32.24	51.30	60.50	42.60	31.50 (-0.74)	46.30	53.80	39.70

than 3 words) to the list of candidates (since we are only interested in testing phrase embeddings, not the phrase extractor).

**Results** We report top- $k$  accuracy (for  $k = 1, 3, 5$ ) and top-5 Mean Reciprocal Rank (MRR@5) on the PR-pass test set in table 4.5a. First, for most SotA embeddings, incorporating context sentence *hurts* the accuracy (*except for BERT* embeddings). That is, interestingly, for all BERT embeddings (base and large), the accuracy increases substantially (+17.64 and +19.04; table 4.5) when the one-sentence context is the input. In contrast, most models that started from BERT but were later finetuned lost the capability to leverage the context information (e.g., PhraseBERT, DensePhrase, and SpanBERT in table 4.5).

Second, the best top-1 accuracy scores on PR-pass for non-contextualized (USE-v5; 43.36%) and contextualized (BERT; 47.44%) embeddings are substantially lower than the non-expert baselines (73.60%; table 4.4) and Human Upperbound (95%). Future work is required to learn more robust, phrase embeddings for ranking. See Figs. 6.32–6.33 for qualitative examples.

#### 4.5.4 Phrase Retrieval: Span-selection models reach near-human accuracy

Consistent with [175], our ranking results in section 4.5.3 reveal that there exists a large headroom for improving both non-contextualized and contextualized phrase embeddings. Yet, because ranking is a naive approach and SS models [69, 30] are the SotA approach on many QA tasks [129], here we train SS models on the train set of PR-pass and PR-page in order to test:

**Q:** *How well do SotA semantic-search models perform on PR-pass and PR-page?*

**Experiment** We take the SotA embeddings tested in section 4.5.3 and add a linear classification layer on top and finetune each entire classifier on the train set of PR-pass or PR-page for 2 epochs using the default HuggingFace hyperparameters (see section 6.3.2 for finetuning details). Following the standard setup of BERT architectures for QA tasks [30], each SS model predicts the start and end index of the target phrase. Additionally, since PR-page documents are much longer than a typical QA paragraph (table 4.2), we also test training Longformer [8], which has a max sequence-length of 4,096, sufficient for an entire Wikipedia page. We take the models of the smallest dev loss and report their test-set performance in table 4.6.

**Results** On PR-pass, in contrast to the poor performance of ranking models (section 4.5.3), our PR-pass-trained SS models perform impressively at a near-upperbound level ( $\sim 93\text{--}94\%$  EM; table 4.6a) surpassing the accuracy of trained experts (90.50% EM). Surprisingly, on PR-page where the documents are substantially longer (around  $12\times$ ) than the documents of PR-pass, SS models’ accuracy only drops slightly (from  $\sim 94\%$  to  $\sim 85\text{--}89\%$  EM; table 4.6b). Note that in a full Wikipedia page of PR-page, there might be phrases that can be considered correct but are *not* labeled groundtruth according to our annotations. This remarkable result suggests that training on PR-pass can enable high-performing models on real-world semantic search.

#### 4.5.5 Phrase Sense Disambiguation: Best models also perform poorly

We find that SotA PR-pass-trained SS models reach superhuman accuracy on PR-pass, i.e. finding a phrase of the same meaning (section 4.5.4). Yet, PR-pass only tests models’ understanding of a *single sense* of the target phrase at a time. It is interesting to study:

Table 4.6: **Test-set performance (%) of SS models on PR-pass (a), PR-page (b), and PSD (c).** When trained on PR-pass (a) and PR-page (b), SotA SS models perform well. However, testing the PR-pass-trained models on PSD shows a significant drop in accuracy (c). That is, SotA SS models tend to understand a *single sense* of a phrase in context well (high PR-pass, PR-page, and PSD EM scores). Yet, they are not able to differentiate two senses of the same phrase (e.g., here, PhraseBERT accuracy drops **-41.27** points between EM+loc vs. EM scores on PSD).

Model	(a) PR-pass		(b) PR-page		(c) PSD			
	EM	F <sub>1</sub>	EM	F <sub>1</sub>	EM	F <sub>1</sub>	EM+loc	F <sub>1</sub> +loc
PhraseBERT [157]	93.42	94.97	85.24	87.19	92.98	94.08	51.67 (-41.31)	51.83
BERT [30]	93.26	94.65	85.64	87.77	93.50	94.57	54.84 (-38.66)	55.07
BERT <sub>Large</sub> [30]	93.64	95.16	87.36	89.52	94.67	95.57	55.43 (-39.24)	55.61
SpanBERT [72]	93.50	95.02	87.28	87.66	92.26	93.30	52.20 (-40.06)	52.34
SentenceBERT [131]	93.24	94.54	84.66	86.89	93.21	94.15	52.74 (-40.47)	52.85
SimCSE [45]	92.90	94.51	85.68	87.66	92.96	94.05	53.83 (-39.13)	53.94
Longformer [8]	94.26	<b>95.58</b>	<b>89.54</b>	<b>91.15</b>	96.17	96.88	<b>62.72 (-33.45)</b>	<b>62.83</b>
Longformer <sub>Large</sub> [8]	<b>94.28</b>	95.53	87.58	89.32	<b>96.32</b>	<b>96.91</b>	59.72 (-36.60)	59.82
mean	93.56	95.00	86.92	88.85	94.01	94.94	55.39 (-38.62)	55.54
± std	0.49	0.42	1.93	1.73	1.54	1.36	3.90	3.88

**Q:** Do PR-pass-trained SS models understand contextualized phrases sufficiently to separate two different senses of the same target phrase?

**Experiment** To do that, here we test the best PR-pass-trained SS models on PSD. Note that, PSD has the same task format as PR-pass (see fig. 4.1c–e) except that the document is twice as long and contains **two occurrences of the same target phrase**. We do not test the ranking models as they perform much worse than the SS models in section 4.5.3.

**Results** Although the PR-pass-trained SS models are never trained on PSD, they interestingly frequently find one occurrence of the target phrase (mean of 94.01% EM; table 4.6c). However, they mostly locate the **target phrase in the wrong context passage** with high confidence scores. That is, if we consider also the correctness of the location of the predicted phrase, their EM+loc<sup>2</sup>

<sup>2</sup>For a PSD example, if the predicted span does not intersect at all with the groundtruth span, the EM+loc and F<sub>1</sub>+loc scores would be 0. If they intersect, the two scores would be equal to EM and F<sub>1</sub>, respectively.

accuracy drops significantly to an average of 55.39%. Also, finetuning on a 2K-example train set of PSD only slightly improves the EM+loc to an average of 64.24% on a 3K-example PSD test set (section 6.3.7). Note that we estimate the Human Upperbound on PSD to be 95%, i.e. the same as that of PR-pass. See qualitative examples and predictions of Longformer (i.e. the best model tested) in Figs. 6.29–6.31.

In sum, there is a large headroom for future research on PSD. SS models are *not* yet capable of leveraging surrounding words to differentiate between two senses of the same phrase. Interestingly, after training on PR-pass, their contextualized phrase embeddings perform much worse in the ranking experiments on PR-pass (section 6.3.8).

## 4.6 Discussion and Conclusion

While WiC and English WSD rely exclusively on dictionaries [122] to obtain word senses and example sentences, our data collection depends on Wikipedia, WiC, & NLP models and our annotation depends on experts. In sum, we present PiC, the first 3-task suite for evaluating phrases in context. SS models can obtain high accuracy on semantic search after training on our PR-pass and PR-page datasets. Yet, their capability is limited to finding a semantically-similar phrase given a single context that contains the target phrase (in PR-pass). The results on PS and PSD show that SotA phrase embeddings are still limited in encoding contextualized phrases. It is interesting future work to improve these models for disambiguating the senses of a phrase in context (PS and PSD).

## 4.7 Limitations

Our dataset is currently limited to multi-word, English noun-phrases. Furthermore, it is expected to contain around a 5% error on PR-pass (i.e. the best human performance is 95% EM). On PR-page, there may be more than one correct target phrase; however, we only label one phrase as the correct answer per document. We use only phrases that contain at least one WiC word.

## Chapter 5

### PEEB: Part-based Image Classifiers with an Explainable and Editable Language Bottleneck

#### 5.1 Overview

Fine-grained classification [154, 152] is a long-standing computer-vision challenge. Furthermore, it is also important to explain how SOTA classifiers make a decision, e.g., which bird traits make a model think a given bird is Painted Bunting? (fig. 5.1)

First, many bird classifiers claim to be explainable [23, 33] by comparing the input image with a set of learned, part prototypes (Figure 5.1b) or natural-language concepts (Figure 5.1a). Yet, such prototypes are feature vectors and therefore not editable by users. Textual concepts are often compared against *entire image* for classification and it is unknown what image details match a given descriptor [95, 172]. Third, most vision-language classifiers need the prompt to have a known {class name} (like a special code instead of an expressive, natural description) that matches the input image [136]. Fourth, most classifiers require either training-set images in a supervised-learning setting or demonstration images in a zero-shot setting [163, 185]. This requirement is impractical when building a classifier for a novel species whose photos do not yet exist in the database.

To address the above four problems, we propose PEEB, a Part-based image classifier that is Explainable and Editable via a natural-language Bottleneck. PEEB classifies images by grounding the textual descriptor of object *parts* provided by humans or GPT-4 (no images needed) to detected parts in the image (fig. 5.1c). While PEEB leverages CLIP’s encoders [126], it uses no class names (e.g., Indigo Bunting) in the prompt. In contrast, CLIP-based models [126, 124, 95] rely so heavily on the *known* class names that their accuracy drops significantly when the names are removed or replaced by less-common ones (Section 5.5.1).



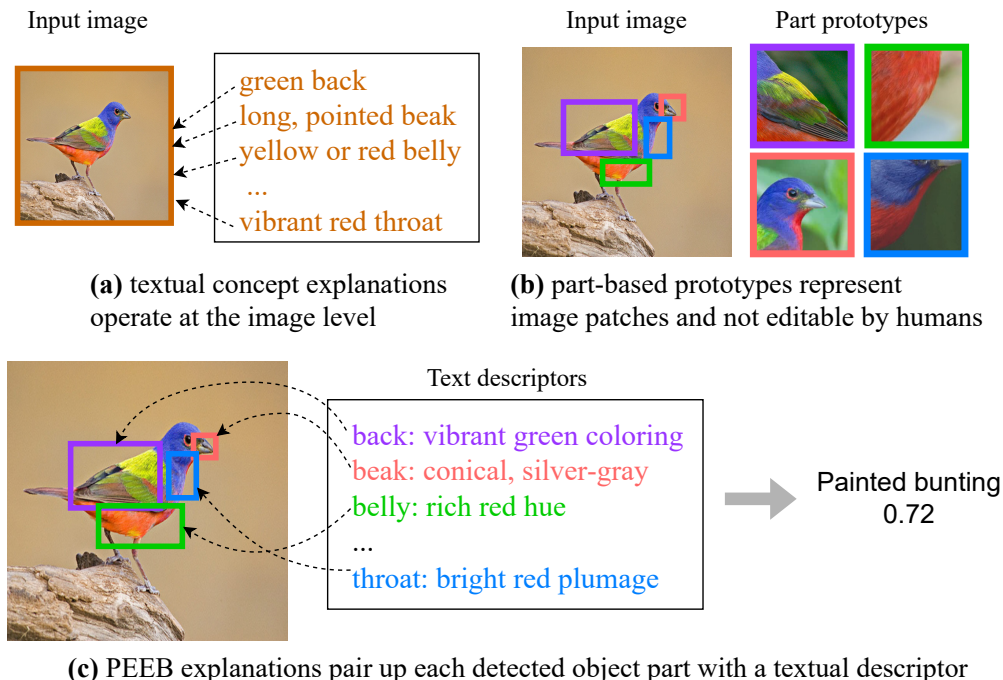


Figure 5.1: Existing explanations are either (a) textual but at the image level; or (b) part-level but not textual. Combining the best of both worlds, PEEB (c) first matches each detected object part to a text descriptor, then uses the part-level matching scores to classify the image.

For **birds** 🐦, we first define the parts of interest for identifying a bird. We take the 15 parts defined in CUB [154] and reduce them to 12 by merging similar parts, e.g. *left wing* and *right wing* are merged into *wings*. Using GPT-4 [107], we construct a *textual* descriptor to describe each bird part of every species (see Section 6.4.3). Next, PEEB localizes the 12 bird parts in the image and computes their matching scores with corresponding text descriptors (fig. 5.2). The sum of the 12 dot products between the paired visual and textual part embeddings would be the unnormalized distance (logits) between the input image and every class for classification (fig. 5.3). For **dogs** 🐕, we use a similar procedure.

To our knowledge, all existing public bird-image datasets (listed in table 6.23) are limited in size (less than 100K images per dataset) and in diversity (less than 1,500 species per dataset), impeding large-scale, vision-language, contrastive learning. Therefore, for our pre-training, we construct Bird-11K, an exceptionally large dataset of bird images, comprising ~290,000 images

spanning across ~11,000 species—essentially, *all* known bird species on Earth (section 5.3). Bird-11K is constructed from seven existing bird datasets and ~55,000 new images that we collect from the Macaulay Library. Similarly, we build Dog-140, a large-scale dataset of 206K dog images. Our main findings are:<sup>1</sup>

1. CLIP-based classifiers rely mostly on class names in the prompt: The CUB accuracy of M&V model [95] drops drastically from 53.78% to 5.89% and 5.95% after class names are removed or replaced by scientific names (section 5.5.1).
2. Our pre-trained PEEB outperforms CLIP-based classifiers by +8 to +29 percentage points (pp) in bird classification across CUB-200, NABirds-555, and iNaturalist-1486 (section 5.5.2).
3. PEEB allows defining new classes in text at test time (fig. 5.2) without any further training. Besides explainability and editability, PEEB outperforms *text concept-based* methods in both the generalized zero-shot (section 5.5.3) and zero-shot setting (section 5.5.4).
4. Compared with explainable CUB classifiers, PEEB scores an 88.80% top-1 accuracy, on par with the best CUB-200 classifiers (81–87% accuracy) that are trained via supervised learning and often *not* editable (section 5.5.5).
5. PEEB is applicable to multiple domains. On Stanford Dogs-120, PEEB scores 92.20%, substantially outperforming explainable models and on-par with SOTA black-box models (section 5.5.6).

## 5.2 Related Work

**Ante- vs. post-hoc explanations** It is common to build fine-grained classifiers based on CNNs [61] or ViTs [59]. Although high-performing, these models do not admit an *ante*-hoc explanation interface [52] and therefore rely on *post*-hoc interpretability methods, which tend to offer inaccurate

---

<sup>1</sup>Code & data: <https://github.com/anguyen8/peeb>

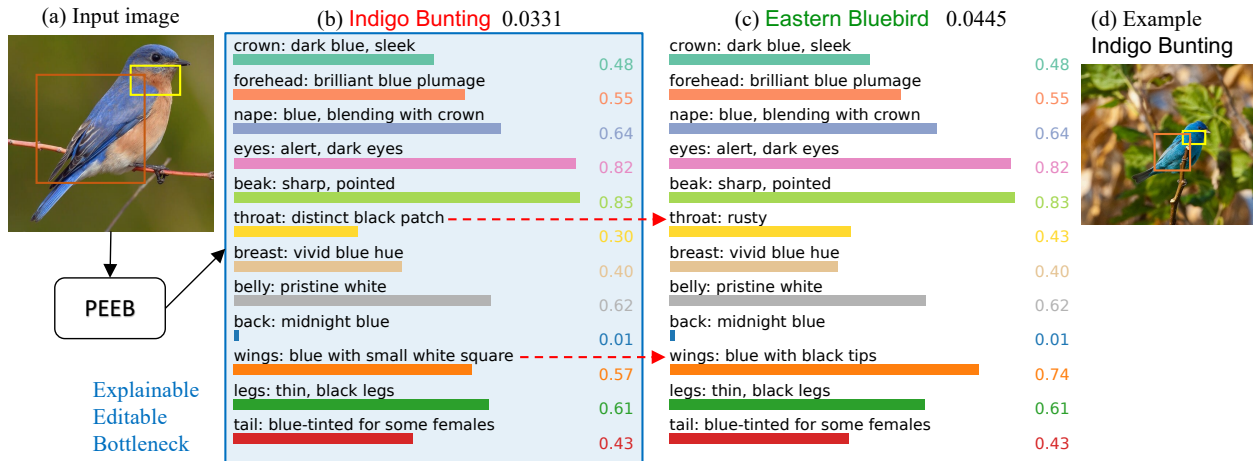


Figure 5.2: Given an input image (a) from an unseen class of Eastern Bluebird, PEEB misclassifies it into **Indigo Bunting** (b), a visually similar blue bird in CUB-200 (d). To add a new class for Eastern Bluebird to the 200-class list that PEEB considers when classifying, we clone the **12 textual descriptors** of Indigo Bunting (b) and **edit** ( - - > ) the descriptor of **throat** and **wings** (c) to reflect their identification features described on AllAboutBirds.org (“Male Eastern Bluebirds are vivid, deep blue above and rusty or brick-red on the throat and breast”). After the edit, PEEB correctly predicts the input image into **Eastern Bluebird** (softmax: 0.0445) out of 201 classes (c). That is, the dot product between the **wings** text descriptor and the same orange region increases from 0.57 to 0.74.

and unstable, after-the-fact explanations [137, 7]. PEEB’s textual part-descriptors form an ante-hoc, natural-language explanation bottleneck that enables users to observe and edit the object attributes that contribute to each final prediction. By editing text descriptors, users can re-program the model without any further re-training (fig. 5.2).

**Prototypical Part Networks** Like the explainable classifiers that learn part prototypes [102, 33, 101, 23], PEEB also operates at the object-part level. However, there are two major differences. First, the textual part descriptors in PEEB are human-understandable and editable. In contrast, a part prototype [23] is not directly editable or interpretable to users, and often interpreted by showing the nearest training-set image patches. Second, PEEB predicts a *contextualized* embedding for each object part and its spatial information can be viewed by inputting to the Box MLP (see fig. 5.3) for bounding-box visualization.

**Text-based Concept Bottlenecks** Like PEEB, [25, 185, 130, 113] also match visual part embeddings to text embeddings. Yet, they (1) do not use CLIP and instead rely on TF-IDF text features;

(2) require a trained bird-part detector to detect 7 bird parts. In contrast, PEEB relies on CLIP, which admits easy text editability, and OWL-ViT, which serves as an open-vocabulary object-part detector that generalizes to many domains.

Recent vision-language models (VLMs) claim to be interpretable as they use textual concepts in the prompt. Yet, some works that rely on class-wise differential captions [37] or learned concept weights [172, 108, 106, 176] do *not* generalize to unseen classes. The most recent, similar work to PEEB might be LaBo [172], which; however, operates at the *image* level instead of patch level, and does *not* generalize to unseen classes.

Many CLIP-based classifiers [54, 124, 95] rely heavily on having *seen* class names in the prompt and thus are neither explainable nor editable to users. Unlike CLIP-based models, PEEB reveals what image details are being used for classification by matching descriptors to corresponding visual object parts (e.g. a bird’s **beak** in fig. 5.3).

**Attribute-based Classifiers** Attribute-Label Embedding (ALE) approaches [3, 176] employ a fixed set of attributes and train an attribute-to-label weight matrix for zero-shot classification. Several studies [138, 166, 55] highlight its effectiveness on datasets like CUB, SUN [165], and AWA [164]. Yet, in practice, ALE requires tabular data annotations for every new class in the dataset (e.g., 312 attributes per CUB species), editing the weight matrix, and model re-training. In contrast, to add an unseen class to PEEB, users would only need to describe its 12 bird parts in natural language.

## 5.3 Datasets

### 5.3.1 Test classification benchmarks

We test PEEB on three 🦜 bird classification datasets: CUB-200 [154], NABirds-v1 of 555 classes [152], and iNaturalist [153] which has 1,486 bird classes. For 🐕 dog images, we test PEEB on Stanford Dogs-120 [75].

### 5.3.2 Bird-11K dataset construction

We combine *labeled* images from 7 distinct datasets and an extra ~55K images (10,534 classes) from Cornell’s Macaulay Library, to form a unified **Bird-11K** dataset<sup>2</sup> (Section 6.4.4) for large-scale pre-training. To the best of our knowledge, Bird-11K, comprising 440,934 images spanning 11,183 classes, is the first bird dataset to encompass almost all species on Earth. Since PEEB learns to match visual parts with textual descriptors, it requires that bird images be distinctly visible and sufficiently large for accurate part localization and matching (see Section 6.4.5 for ablation studies). However, small and “hard-to-see” bird images in Bird-11K make the dataset noisy and the training complex. Thus, we harness OWL-ViT<sub>Large</sub> [97] to detect a bird in all images using the prompt “bird” and filter out images where the detected bird’s bounding box is smaller than  $100 \times 100$  pixels. We find OWL-ViT’s bird detections to be fairly accurate—its mean Intersection over Union (IoU) between the predicted bird boxes and ground-truth boxes on CUB dataset is 0.91.

As class labels from different sources are either general (e.g. Cardinal) or fine-grained (e.g. Yellow vs. Northern Cardinal), we retain only the fine-grained species for more diverse training and exclude all general classes to avoid label ambiguity. Following these filtering steps, the refined Bird-11K dataset retains 294,528 images across 10,811 classes (table 6.23).

For each species in Bird-11K, we generate a set of part-based descriptors using GPT-4 (Section 6.4.3). These generated descriptors (see fig. 5.4) may not be 100% accurate but discriminative enough to help GPT-4V reach 69.40% accuracy on the CUB-200 test set (Table 5.3). That is, in the same prompt, we feed each test image  $x$  along with the 200 CUB classes’ part-based descriptors and ask GPT-4V to select a matching class label for  $x$  (details in Section 6.4.6).

### 5.3.3 Dataset splits for contrastive pre-training

There are two common settings in the zero-shot learning literature—standard zero-shot (**ZSL**) and generalized zero-shot (**GZSL**).

---

<sup>2</sup>We do not redistribute the published datasets but release a script for reconstructing Bird-11K on Github.

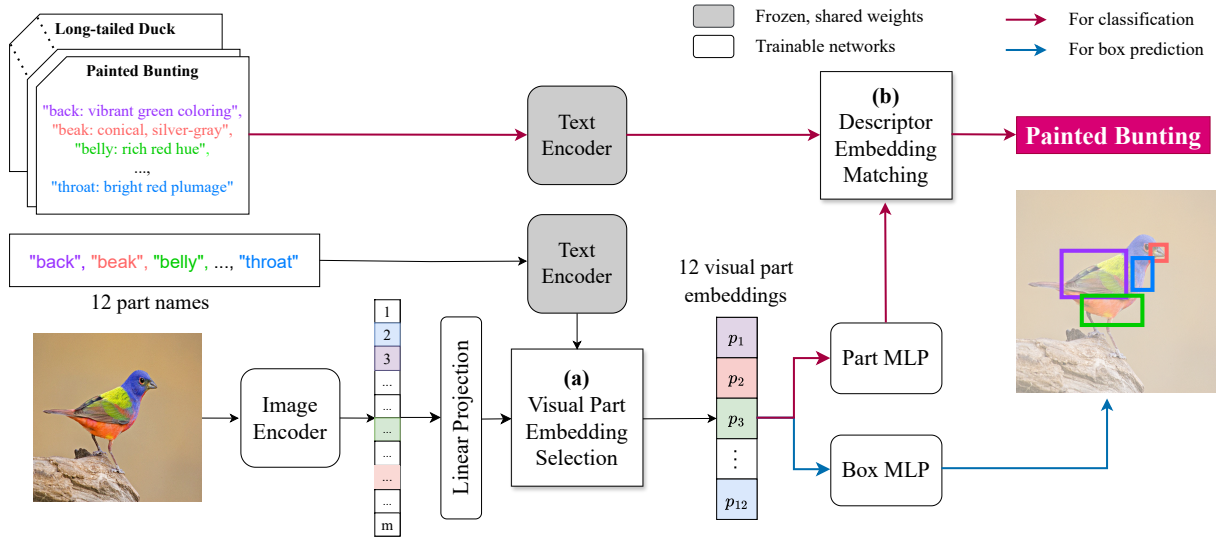


Figure 5.3: During inference, 12 visual part embeddings with the highest cosine similarity with encoded part names are selected (a). These visual part embeddings are then mapped ( $\rightarrow$ ) to bounding boxes via Box MLP. Simultaneously, the same embeddings are forwarded to the Part MLP and its outputs are then matched (b) with textual part descriptors to make classification predictions ( $\rightarrow$ ). fig. 6.35 shows a more detailed view of the same process.

**ZSL** is a stricter setup where a model is only tested on the *classes* unseen during any prior training. We ensure test-set classes from datasets (e.g., CUB-200 or NABirds-555) are not seen during pre-training. For example, to test on CUB under ZSL, we exclude all 200 CUB classes and their images from our pre-training on Bird-11K.

Following the ZSL literature, we use the CUB split proposed by [3] and two harder splits: Super-Category-Shared/Exclusive (SCS/SCE) by [35]. For example, in ZSL on CUB, we exclude all CUB classes in Bird-11K for pre-training and finetune only on the corresponding training set given by a ZSL split.

**GZSL** is closer to the real-world setup where models are tested on both seen & unseen classes. CLIP’s “zero-shot” tests technically fall under GZSL as its Internet-scale training set might actually have images from the test classes. To test PEEB under GZSL, we exclude the *test* sets of CUB, NABirds, and iNaturalist, and directly evaluate the Bird-11K-pretrained models without further finetuning.

## 5.4 Method

### 5.4.1 Backbone: OWL-ViT object-part detector

OWL-ViT is an open-vocabulary detector that detects objects and parts in an image given a text prompt, even if the model is not explicitly finetuned to detect those concepts. OWL-ViT consists of four networks (fig. 5.3): (1) a ViT-based image encoder, (2) an architecturally identical text encoder, (3) a bounding-box regression head called Box MLP, and (4) a Linear Projection. Box MLP is a three-layer Multilayer Perceptron (MLP) with GELU activations [62] after each of the first two layers. Linear Projection maps the visual and text embeddings to the same space (see Fig. 1 in [97]).

### 5.4.2 PEEB classifier

**Architecture** PEEB (fig. 5.3) has five networks: an image encoder, a text encoder, a Linear Projection, a Part MLP, and a Box MLP.

We introduce **Part MLP** to map the visual and textual part embeddings to the same space for computing dot products (logits) for classification ( $\rightarrow$  in Figure 5.3). This design allows PEEB to easily extend the number of classes without any re-training. Except for Part MLP, all components are adopted from the OWL-ViT framework. Details of all components are in Section 6.4.1.

**Inference** Given an input image, we first use the 12 generic part names to select the visual part embeddings based on cosine similarity. These selected visual part embeddings are then simultaneously fed into both Part MLP and Box MLP.

Box MLP predicts the bounding box from each part embedding. We compute a dot product to measure the similarity between each embedding output from Part MLP and a corresponding part-descriptor embedding. For classification, a class logit is the sum of the 12 dot products, which essentially computes the similarity between the 12 parts in the image and the 12 text descriptors of each class.

### 5.4.3 Training strategy

**Trainable networks** In preliminary experiments, we find training only Part MLP (while keeping all other networks frozen) to result in poor accuracy. Therefore, we train Part MLP from scratch and also finetune the image encoder, Linear Projection, and Box MLP. We finetune all OWL-ViT components from their original weights. In contrast, our proposed Part MLP starts from random weights. Our training has two phases: (a) 2-stage **pre-training** on the large-scale Bird-11K dataset and (b) **finetuning** on downstream tasks. More hyperparameter details are in Section 6.4.1.

**Objectives** We aim to train PEEB to classify images well while maintaining the ability to detect object parts. This translates into **three training objectives**: **(1)** Train the Part MLP contrastively using a symmetric cross-entropy (SCE) loss [126] to maximize the similarity between region-text pairs while minimizing the similarity for negative pairs; **(2)** Train the Linear Projection using a SCE loss to mimic OWL-ViT’s behaviors (i.e. the similarity matrix) for part selection; and **(3)** Train Box MLP to predict bounding boxes with DETR losses [183] i.e. a linear combination of  $\ell_1$  corner-to-corner distance loss and GIoU loss [132].

All three losses are described in Section 6.4.1.

**A challenge** when jointly minimizing all three losses above is that PEEB’s validation loss improves significantly slowly perhaps because of some tension between the two SCE losses and the DETR detection loss. To overcome this challenge, we split the pre-training phase into two stages: (1) first, train the image encoder and Part MLP for classification using the SCE loss; then (2) train the Linear Projection and Box MLP using the 2nd and 3rd loss so they can adapt their weights to the updated image encoder. We always keep the **text encoder** frozen since we want to preserve its generalizability to the descriptors of unseen objects.

#### 2-stage pre-training on Bird-11K

**Stage 1: Contrastive learning** The image encoder and Part MLP are jointly trained using a SCE loss, which allows PEEB to learn to map the visual parts to corresponding text descriptors. In this stage, we use a pre-trained OWL-ViT<sub>Large</sub> to *select* 12 part embeddings per input image (i.e.,



teacher forcing) to ensure the *selection* of part embeddings is meaningful and consistent while the embeddings themselves are updating (see fig. 6.36).

**Stage 2: Learning to detect from a teacher** After the image encoder is modified in Stage 1, we then train the Linear Projection and Box MLP jointly. We use the OWL-ViT<sub>Large</sub> as the teacher to train both components. Using SCE loss, we train the Linear Projection such that the similarity matrix between the part-names and visual parts matches those of the teacher (fig. 6.37, 1a–c, 2a–c). Given the absence of human-annotated boxes for object parts, we train Box MLP to predict the same boxes as the predictions by OWL-ViT<sub>Large</sub> using DETR losses (fig. 6.37, 2d). In this Stage 2, the image encoder is frozen while Part MLP is not involved. **After 2-stage training**, PEEB can perform zero-shot classification and generate explanations.

## Finetuning on classification tasks

We finetune the pre-trained PEEB on downstream tasks (CUB, NABirds and iNaturalist for birds and Dogs-120 for dogs) to further improve its accuracy. In this phase, to adapt to a downstream task, all components except the text encoder are trained jointly and the loss for Part MLP is changed from SCE (contrastive) to CE (classification) while the other two losses (DETR) are kept intact.

## 5.5 Experiments & Results

We conduct extensive experiments to evaluate PEEB on multiple 🦜 bird datasets (CUB, NABirds, iNaturalist) and on GZSL (sections 5.5.1 and 5.5.3), ZSL (section 5.5.4) and also supervised learning settings. We also find PEEB to perform well on 🐕 dog image classification on Dogs-120 (section 5.5.6).

### 5.5.1 CLIP-based classifiers rely mostly on {class names} (*not* descriptors)

M&V show that inserting extra GPT-3-generated descriptors into CLIP’s prompts increases its accuracy on many tasks [95]. Yet, it is unknown how important these expressive descriptors are



Figure 5.4: With original descriptors, M&V [95] correctly classifies the input image into **Blue Jay** (a). Yet, interestingly, when **randomly** swapping the descriptors of this class with those of other classes (b), M&V’s top-1 prediction remains unchanged, suggesting that the class names in the prompt (e.g., “A photo of {class name}”) have the most influence over the prediction (not the expressive descriptors). In contrast, PEEB changes its top-1 prediction from **Blue Jay** (c) to **Least Tern** (d) when the descriptors are **randomized**.

compared to the class names. To answer this question, we conduct two experiments on all three models: CLIP, M&V, and our PEEB.

**Experiment 1** We evaluate the role of expressive descriptors to CLIP-based models and to PEEB by measuring the drop in CUB-200 accuracy of each model when the descriptors are randomized.

For M&V and PEEB, we randomize the descriptors by swapping each descriptor with another from an arbitrary class (examples in fig. 5.4).

**Experiment 2** We test the dependence of models on class names by measuring the accuracy drop when they are replaced by scientific names (e.g., Painted Bunting → *Passerina ciris*) on CUB, NABirds, and iNaturalist.

**Results** When random descriptors are used, M&V accuracy drops marginally by **-0.9 pp** (table 5.1; 53.70% → 52.88%), showing that descriptors actually play a minimal role in model predictions. Instead, CLIP and M&V mostly rely on class names (e.g., 53.78% → 7.66%; table 5.2)—their

accuracy drops drastically when class names are replaced by scientific names, which are less common.

In contrast, the expressive part descriptors play a major role in PEEB whose accuracy decreases significantly to near random-chance (64.33%  $\rightarrow$  0.88%; table 5.1) when the descriptors are randomized. Indeed, in PEEB, the textual descriptors serve as editable and interpretable model parameters that can be refined and extended by humans to account for new classes (fig. 5.2).

Table 5.1: Top-1 test accuracy (%) on CUB-200 when using original, correct (a) vs. randomized, wrong descriptors (b). See fig. 5.4 for an example of the descriptors.

	CLIP [126]	M&V [95]	PEEB	
With class names	✓	✓	✗	
(a) Original descriptors	52.02	53.78	5.89	<b>64.33</b>
(b) Randomized descriptors	n/a	52.88	0.59	0.88

Table 5.2: In the **GZSL** setting, PEEB outperforms CLIP and M&V by a large margin, from +8 to +29 pp in top-1 accuracy (see section 5.5.3). PEEB is also  $\sim 10\times$  better than the other two models when class names are replaced by **scientific names**. As PEEB does not use class names, its accuracy remains unchanged when class names are changed into the scientific ones.

Acc (%)	CUB-200	NABirds-555	iNaturalist-1486
CLIP [126]	52.02 (5.95)	39.35 (4.73)	16.36 (2.03)
M&V [95]	53.78 (7.66)	41.01 (6.27)	17.57 (2.87)
PEEB (ours)	<b>64.33</b> (64.33)	<b>69.03</b> (69.03)	<b>25.74</b> (25.74)

### 5.5.2 Pre-trained PEEB outperforms CLIP-based classifiers in GZSL

The dependence on class names (section 5.5.1) suggests that CLIP was exposed to these names during training. Thus, for a fair comparison, we compare PEEB with CLIP-based classifiers in the GZSL setting.

**Experiment** We train PEEB on Bird-11K using the two-stage pre-training (described in section 5.4.3), and then test it on CUB, NABirds, and iNaturalist without any finetuning. That is,

PEEB’s contrastive pre-training is at the part level and therefore the model has not seen the species labels of images.

**Results** PEEB outperforms both CLIP and M&V on all three datasets by huge margins of around +10, +28, and +8 pp on CUB-200, NABirds-555 and iNaturalist-1486, respectively (see table 5.2).

### 5.5.3 PEEB is superior to text descriptor-based classifiers in GZSL on CUB-200

The advent of CLIP [126] by OpenAI enabled a class of image classifiers that match the input image with pre-defined textual prompts that may include class names or descriptors of the classes. Yet, in contrast to PEEB, these descriptors often describe the entire image and are also matched (via dot product) with the entire image instead of image regions. Here, we compare PEEB with these methods in the GZSL setting on CUB-200.

**Experiment** We repeat the same experiments in section 5.5.2. As these bird classifiers (listed in table 5.3) were reported on CUB only (not NABirds or iNaturalist), our comparison is on CUB.

**Results** PEEB exhibits superior GZSL performance, outperforming recent text concept-based approaches by +3 to +10 pp (table 5.3b). Compared to prior methods, PEEB is the only one to detect *visual* object parts and match them with text descriptors. Furthermore, attribute-based classifiers, e.g., [176] require re-training to adapt to new classes or datasets (e.g., NABirds or iNaturalist) in the same domain. In contrast, to apply PEEB to NABirds or a new class, no training is required—it is necessary to only edit its text descriptors (see fig. 5.2). Interestingly, PEEB is 2nd-best model, only after GPT-4V (64.33% vs. 69.40%), which is given the same textual part descriptors as PEEB for all 200 CUB classes and asked to select a matching class for each image.

### 5.5.4 PEEB generalizes to traditional ZSL

Since PEEB outperforms modern vision-language models in GZSL (section 5.5.3), we are motivated to further compare PEEB with SOTA approaches in the traditional ZSL setting (where the test classes are excluded from all prior training).

Table 5.3: PEEB achieves SOTA CUB-200 accuracy among the **text descriptor-based** classifiers in GZSL.

\* *1-shot learning*. † *k-means with  $k = 32$* .

Method	Acc (%)	{c}	Textual descriptors
<b>(a) Vision-language models with class names {c} in the prompt</b>			
CLIP [126]	52.02	✓	Image-level
M&V [95]	53.78	✓	Image-level
FuDD [37]	54.30	✓	Image-level
Han et al. [54]	56.13	✓	Image-level
<b>(b) Vision-language models with text bottlenecks and <b>no</b> class names {c}</b>			
LaBo [172]	54.19 <sup>†</sup>	✗	Image-level
[170]	60.27*	✗	Image-level, attribute-based
PEEB ( <b>ours</b> )	<b>64.33</b>	✗	Part-level
GPT-4V [107]	69.40	✓	Part-level
<b>(c) Concept-Bottleneck Models with attribute-based, non-textual bottlenecks</b>			
CBM [80]	62.90	✗	Attribute-based, tabular data
PCBM [176]	61.00	✗	Attribute-based, tabular data

**Experiment** We evaluate PEEB on **two common ZSL splits**: (a) the CUB split [3]; and (b) the Super-Category-Similar/Exclusive (SCS/SCE) splits [35] on CUB and NABirds. The SCS (Easy) and SCE (Hard) splits are designed to test two generalization levels (generalizing to close vs. distant unseen species).

Aligned with ZSL conventions, we exclude all species that exist in CUB or NABirds from the pre-training and then finetune PEEB using the train/test splits by [3] and [35]. We randomly take ~10% of the training set as the validation set and choose the checkpoints based on the lowest validation loss.

**Results** By a large margin, PEEB outperforms  $CLORE_{CLIP}$ , a SOTA CUB method in the [3] split, on both seen and unseen classes (table 5.4a). On the [35] splits, PEEB is the SOTA in the Hard set on both CUB and NABirds datasets (table 5.4b). That is, PEEB is better in generalizing to distant, unseen classes. This may be because PEEB decomposes both the image and the text descriptors into part-level features, which can re-combine to match an arbitrary unseen class (as illustrated in fig. 5.2).

Table 5.4: PEEB consistently outperforms other vision-language methods under Harmonic mean and especially in the hard split (SCE) by (+5 to +15) points, highlighting its generalization capability on ZSL.

Methods	CUB			NABirds		
	Seen	Unseen	Mean	Seen	Unseen	Mean
<b>(a) Data split by [3]</b>						
CLORE <sub>CLIP</sub> [53]	65.80	39.10	49.05	n/a		
PEEB (ours)	<b>80.78</b>	<b>41.74</b>	<b>55.04</b>			
<b>(b) SCS/SCE splits by [35]</b>						
	SCS (Easy)	SCE (Hard)	Mean	SCS (Easy)	SCE (Hard)	Mean
S <sup>2</sup> GA-DET [70]	42.90	10.90	17.38	39.40	9.70	15.56
GRZSL [185]	44.08	14.46	21.77	36.36	9.04	14.48
ZEST [113]	<b>48.57</b>	15.26	23.22	38.51	10.23	16.17
CANZSL [25]	45.80	14.30	21.12	38.10	8.90	14.43
DGRZSL [81]	45.48	14.29	21.75	37.62	8.91	14.41
DPZSL [130]	45.40	15.50	23.11	<b>40.80</b>	8.20	13.66
PEEB (ours)	44.66	<b>20.31</b>	<b>27.92</b>	28.26	<b>24.34</b>	<b>26.15</b>

Interestingly, on both CUB and NABirds, PEEB is competitive but *not* SOTA on the Easy sets (table 5.4b; Easy)—those classes that are close to the training-set classes and thus considered easier to identify. Overall, considering the harmonic mean over both Easy and Hard accuracy scores, PEEB is the SOTA on both CUB and NABirds.

### 5.5.5 Finetuning the pre-trained PEEB on CUB-200 yields a competitive explainable classifier in supervised learning

After finding that PEEB performs well in both GZSL (section 5.5.3) and ZSL settings (section 5.5.4), here we test finetuning the pre-trained PEEB on CUB-200. That is, we compare PEEB against SOTA explainable classifiers in the supervised learning setting to gain insights into our method’s adaptability to downstream tasks.

**Experiment** To understand the impact of pre-training and image resolution, we test finetuning three different PEEB variants: (1) PEEB initialized from OWL-ViT<sub>B/32</sub> **without pre-training** on Bird-11K; (2) PEEB initialized from OWL-ViT<sub>B/32</sub> with pre-training (described in section 5.5.2);

and (3) PEEB initialized from OWL-ViT<sub>B/16</sub> with pre-training. We take each PEEB model and finetune *all* components on CUB-200, for 30 epochs with a batch size of 30, a learning rate of  $2 \times 10^{-5}$ . Detailed hyperparameters are in table 6.21.

**Results** Without pre-training, PEEB reaches 77.80% top-1 accuracy on CUB-200. Yet, first pre-training on Bird-11K and then finetuning on CUB yields 86.73%, the best among all *explainable* classifiers (table 5.5b–c). Besides, pre-training PEEB from the higher-resolution OWL-ViT<sub>B/16</sub> results in a further gain of +2.07 (86.73%  $\rightarrow$  88.80%), which is intuitive since fine-grained classification is known to benefit from higher resolutions.

For a complete assessment, we compare and find PEEB to underperform SOTA standard, black-box classifiers by a few points (table 5.5a).

Table 5.5: PEEB is a state-of-the-art, explainable CUB-200 🦜 classifiers in the **supervised** learning.

Methods	Model size Backbone	Acc (%)
<b>(a) SOTA black-box classifiers</b>		
Base (ViT) [149]	22M DeiT-S [149]	84.28
ViT-Net [76]	26M DeiT-S	90.10
<b>(b) Concept-bottleneck classifiers</b>		
CBM [80]	11M ResNet-18	80.10
CPM [108]	155M ViT-B/16	72.00
CDM [106]	155M ViT-B/16	74.31
LaBo [172]	427M ViT-L/14	81.90
<b>(c) Part-based, explainable classifiers</b>		
ProtoPNet [23]	22M DeiT-S	84.04
ProtoTree [102]	92M ResNet-50	82.20
TesNet [156]	79M DenseNet-121	84.80
Deformable ProtoPNet [33]	23M ResNet-50	86.40
ProtoPFormer [167]	22M DeiT-S	84.85
PEEB ( <b>ours</b> )	155M	
<del>pre-training</del> + finetuning <i>only</i>	155M OWL-ViT <sub>B/32</sub>	77.80
pre-training + finetuning	155M OWL-ViT <sub>B/32</sub>	<b>86.73</b>
pre-training + finetuning	155M OWL-ViT <sub>B/16</sub>	<b>88.80</b>



Figure 5.5: PEEB classifies this Dogs-120 image into **Alaskan Malamute** (softmax: **0.199**) due to the matching between the image regions and associated textual part descriptors. In contrast, the explanation shows that the input image is not classified into **Cairn Terrier** mostly because its ears and body regions do *not* match the text descriptors, i.e., dot products are **0.000** and **0.000**, respectively. See Section 6.4.7 for more qualitative examples.

### 5.5.6 Applying PEEB to 🐕 dog identification

We have found that our pre-training dataset construction and PEEB performs well for bird identification. By design, our method is *not* specific to birds but is instead applicable to any fine-grained classification domains assuming that the object is decomposable into parts. Here, we show that our method performs well on dog image classification as well.

**Pre-training dataset construction** First, we define a set of six dog parts that humans use to identify dog species. We use all 4 dog parts defined by PartImageNet [60]—*head*, *body*, *legs*, and *tail*—and two more parts—*muzzle* and *ears*—based on our manual image examination.

We combine ImageNet-21K and Stanford Dogs-120 into Dog-140, our large-scale pre-training dataset spanning 140 dog species (details in section 6.4.4). For each class, we prompt GPT-4 to get the descriptors for 6 parts. For each image in Dog-140, we run OWL-ViT<sub>Large</sub> to detect the corresponding boxes for 6 pre-defined parts.

**Experiment** Following the supervised learning experiment in section 5.5.5, we first pre-train PEEB (initialized from OWL-ViT<sub>B/32</sub>) on Dog-140 and then further finetune it on Dogs-120.

**Results** Finetuning PEEB on Dogs-120 from OWL-ViT<sub>B/32</sub> without pre-training on Dog-140 results in a 74.17% top-1 accuracy on Dogs-120 (table 5.6b). In contrast, pre-training on Dog-140 only without finetuning results in much better Dogs-120 accuracy of 87.38%. That is, our contrastive



pre-training helps model generalize (in a GZSL setting) while directly finetuning on Dogs-120 perhaps yields an overfitting model. Yet, pre-training and then finetuning reaches the best supervised learning accuracy of 92.20%, which is SOTA among all explainable models reported on Dogs-120.

Besides, PEEB offers novel, editable image-text grounding explanations (see fig. 5.5).

Table 5.6: In the **supervised** learning setting, PEEB is the state-of-the-art explainable, Stanford Dogs-120 🐶 classifiers and competitive w.r.t. SOTA black-box models.

Methods	Model size Backbone	Acc (%)
<b>(a) SOTA black-box classifiers</b>		
TransFG [59]	86M ViT-B/16	92.30
ViT-Net [77]	86M DeiT-B	93.60
SR-GNN [9]	32M Xception	97.00
<b>(b) Explainable methods</b>		
FCAN [92]	50M ResNet-50	84.20
RA-CNN [43]	144M VGG-19	87.30
ProtoPNet [23]	22M DeiT-S	77.30
Deformable ProtoPNet [33]	23M ResNet-50	86.50
PEEB ( <b>ours</b> )	155M	
<del>pre-training</del> + finetuning <i>only</i>	155M OWL-ViT <sub>B/32</sub>	74.17
pre-training + <del>finetuning</del>	155M OWL-ViT <sub>B/32</sub>	87.37
pre-training + finetuning	155M OWL-ViT <sub>B/16</sub>	<b>92.20</b>

## 5.6 Discussion and Conclusion

We introduce PEEB, a unique, novel explainable classifier due to its editability (fig. 5.2) and operation at the part level on both image and text sides. The part-level operation makes PEEB applicable to fine-grained classification. Yet, it is also interesting to extend PEEB into an object-level model for multi-domain tasks like ImageNet or VQA.

Besides enabling users to edit PEEB’s text descriptors to re-program PEEB, it might also be promising to let users edit the bounding boxes while working with PEEB to improve the human-AI team accuracy [103]. On object detection, PEEB’s Box MLP performs on-par with OWL-ViT<sub>B/32</sub> based on quantitative (Section 6.4.5) and qualitative results (Section 6.4.7).

Finally, we contribute to the broader research community by curating the Bird-11K and Dog-140 datasets and showing that it is possible to leverage them for large-scale training.

## 5.7 Limitations

**Text encoder may not fully comprehend the bird descriptors** Our CLIP text encoder, pre-trained on an Internet-scale image-text dataset [126], may not fully capture the intricate details specific to birds. Furthermore, the CLIP text encoder is known to suffer from the *binding* problem and do not understand some logical operators such as “and”, “or”, or negation. PEEB accuracy depends heavily on the quality of the text encoder.

**Assumption that object parts mostly visible** PEEB operates based on the assumption that most if not all of the object parts are visible in the image. In cases where a part is missing or occluded, the model may still assign a non-zero similarity score (i.e. a non-zero dot product between the image-part embedding and its associated text descriptor), which makes it harder to separate classes. It might be beneficial to incorporate extra training samples and specifically encourages PEEB to assign *zero* image-text similarity score to the missing or occluded parts.

**Hallucinations in GPT-4 descriptors** The accuracy of PEEB is directly governed by the accuracy of descriptors, which are currently generated by GPT-4. Yet, our manual assessment over 20 bird classes reveals that, on average, 45% of these descriptors do not accurately reflect the birds’ features (Section 6.4.6). Also, we observe that revising certain descriptors in the CUB dataset led to a **significant** improvement of +10 points in classification accuracy for those classes (Section 6.4.6). This primitive observation suggests that PEEB can be further improved if trained with more accurate, human-labeled descriptors.

## 6.1 Supplementary Materials for Out of Order (Chapter 2)

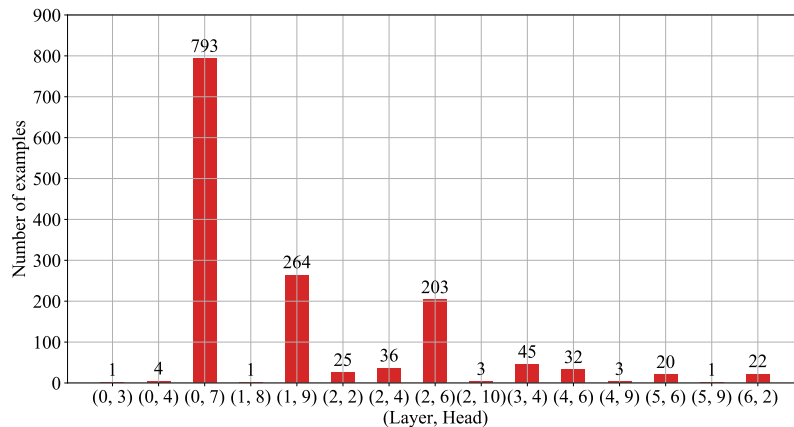
### 6.1.1 Self-attention layers that match question-words to similar words in the answer

QNLI models being so insensitive to word shuffling (i.e. 89.4% of the correct predictions remain correct) suggests that inside the finetuned BERT, there might be a self-attention (SA) layer that extract pairs of similar words that appear in both the question and answer.

We started by analyzing all 2,500 positive dev-r examples (Table 6.4) of RoBERTa-based classifiers trained on QNLI because there were fewer and more consistent ways for labeling a sentence positive than for the negative (shown in Sec. 2.4.3).

**Experiment** There were 1,776 (out of 2,500) examples whose predictions did not change in 5 random shufflings (a.k.a 5/5 subset). For each such example, we followed the following 4 steps to identify one SA matrix (among all 144 as the base model has 12 layers & 12 heads per layer) that captures the strongest attention connecting the question and answer words.

1. Per example  $x$ , we created its shuffled version  $\hat{x}$  by randomly shuffling words in the question and fed  $\hat{x}$  into the classifier.
2. For each SA matrix obtained, we identified the top-3 highest-attention weights that connect the shuffled question tokens and the real answer tokens (i.e. excluding attention weights between question tokens or answer tokens only).
3. For each shuffled example  $\hat{x}$ , we identified one matrix  $M$  whose the top-3 word pairs are the nearest in Levenshtein character-level edit-distance [105]. For instance, the distance is 1 between manage and managed (Fig. 2.4).



(a) Histogram of self-attention matrices

(Layer, Head)	# Examples	%
(0, 3), (0, 4), (0, 7)	798	54.9%
(1, 8), (1, 9)	265	18.2%
(2, 2), (2, 4), (2, 6), (2, 10)	267	18.4%
(3, 4)	45	3.1%
(4, 6), (4, 9)	35	2.4%
(5, 6), (5, 9)	21	1.5%
(6, 2)	22	1.5%
<b>Total</b>	<b>1,453</b>	<b>100%</b>

(b) Layer-wise comparison

Figure 6.1: Among 144 self-attention matrices in the RoBERTa-based classifier finetuned for QNLI, there are 15 “word-matching” matrices (a) that explicitly attend to pairs of similar words that appear in both questions and answers regardless of the order of words in the question (see example pairs in Fig. 2.4). For each QNLI example, we identified one such matrix that exhibits the matching behavior the strongest (a). 92% of the task of attending to duplicate words is mostly handled in the first three layers (b).

4. For each matrix  $M$  identified for  $\hat{x}$ , we fed the corresponding real example  $x$  through the network and re-computed the edit-distance for each of the top-3 word pairs.

**Results** At step 3, there were 1,590 SA matrices (out of 1,776) whose the top-3 SA weights connected three pairs of *matching* words (i.e. the total edit-distance for 3 pairs together is  $\leq 4$ )<sup>1</sup> that appear in both the shuffled question and original answer (see example top-3 pairs in Fig. 2.4). At step 4, this number only dropped slightly to 1,453 matrices when replacing the shuffled question by the original one (see Table 6.1 for detailed statistics).

However, there are only 15 unique, RoBERTa self-attention matrices in these 1,453 examples (see Fig. 6.1). Also at step 4, 83% of the same word pairs remained within the top-3 of the same SA matrices, after question replacement, i.e. 17% of attention changed to different pairs e.g. from (“managed”, “manage”) to (“it”, “it”).

First, our results showed that there is a set of 15 self-attention heads explicitly tasked with capturing question-to-answer word correspondence *regardless of word order*. Second, **for ~58%**

<sup>1</sup>4 is a tight budget to account for minor typos or punctuation differences e.g. “Amazon” vs. “Amazon’s”.

Sum distance	(a) dev-s alone		(b) dev-s & dev-r	
	# examples	%	# examples	%
$\leq 0$	749	42.17	392	24.65
$\leq 1$	1,253	70.55	1,071	67.36
$\leq 2$	1,440	81.08	1,283	80.69
$\leq 3$	1,543	86.88	1,391	87.48
$\leq 4$	<b>1,590</b>	<b>89.53</b>	<b>1,453</b>	<b>91.38</b>
$\leq 15$	1,776	100.00	1,574	98.99
Total	1,776	100.00	1,590	100.00

Table 6.1: The number of QNLI examples where we found  $\geq$  one self-attention matrix that the most strongly attends to three pairs of *matching* words when given the dev-s examples i.e. (modified question, real answer) (a) or when given both the shuffled and real examples (b). In other words, **the numbers in (b) denote the number of examples where (1) there exist  $\geq 3$  words, regardless of its word order, in the question that can be found in the accompanying real answer; and (2) these correspondences are captured by at least one self-attention matrix.** The sum edit-distance for all 3 pairs of words are less than  $N$  where  $N = \{0, 1, 2, 3, 4, 15\}$  (left column).

**(i.e. 1,453 / 2,500) of QNLI positive examples: (1) there exist  $\geq 3$  words in the question that can be found in the accompanying answer; and (2) these correspondences are captured by at least one of the 15 SA matrices.** We also found similar results for 2,500 negative dev-r examples (data not shown).

Model	Task	(a) Perf. on dev-r		(b) Perf. on dev-s			(c) Word-Order Sensitivity				
		Models	Baseline	2-noun swap	1-gram	2-gram	3-gram	2-noun swap	1-gram	2-gram	3-gram
RoBERTa <sub>base</sub>	ANLI	100	33.33	74.26	57.74	66.63	69.04	0.39	0.63	0.50	0.46
	A1	100	33.33	81.46	63.31	71.52	75.37	0.28	0.55	0.43	0.37
	A2	100	33.33	70.83	54.61	64.73	67.02	0.44	0.68	0.53	0.49
	A3	100	33.33	70.50	55.29	63.63	64.73	0.55	0.67	0.55	0.53
RoBERTa <sub>large</sub>	ANLI	100	33.33	70.41	54.87	64.11	68.76	0.44	0.68	0.54	0.47
	A1	100	33.33	78.06	60.31	70.57	75.86	0.33	0.6	0.44	0.36
	A2	100	33.33	67.88	51.44	60.64	66.31	0.48	0.73	0.59	0.51
	A3	100	33.33	65.30	52.85	61.11	64.10	0.52	0.71	0.58	0.54

Table 6.2: All results (a–c) of RoBERTa<sub>base</sub> and RoBERTa<sub>large</sub> models finetuned on the combination of NLI datasets (SNLI, MNLI, FEVER and ANLI) are reported on the ANLI dev-r sets (i.e. 100% accuracy) which includes A1, A2 and A3 (a). The accuracies for RoBERTa<sub>base</sub> and RoBERTa<sub>large</sub> on ANLI are 51.19% and 56.98%, respectively. Each row is computed by averaging the results of 10 random shuffles. Word-Order Sensitivity (WOS) of ANLI and its subsets (c). Since ANLI is 3-way classification task, the baseline is 33.33% (as described in Sec 4.3).

Dictionary	Opinion Lexicon [66]		SentiWords [46]	
	(a) RoBERTa	(b) ALBERT	(c) RoBERTa	(d) ALBERT
Total examples in subset 5/5 (positive / negative)	523 (278 / 245)	506 (228 / 278)	523 (278 / 245)	506 (228 / 278)
Not found in dictionary	223 / 523 (42.64%)	217 / 506 (42.89%)	110 / 523 (21.03%)	104 / 506 (20.55%)
Found in dictionary	300 / 523 (57.36%)	289 / 506 (57.11%)	413 / 523 (78.97%)	402 / 506 (79.45%)
P ( positive sentence   positive top-1 word )	174 / 174 (100.00%)	143 / 144 (99.31%)	222 / 258 (86.05%)	186 / 215 (86.51%)
P ( negative sentence   negative top-1 word )	119 / 126 (94.44%)	136 / 145 (93.79%)	145 / 155 (93.55%)	177 / 187 (94.65%)

Table 6.3: **If the top-1 most important word in an SST-2 5/5 example has a positive meaning, then there is a 100% chance that the sentence is labeled positive in SST-2.** Similarly, the conditional probability of a sentence being labeled negative given a negative most important word (by LIME [134]) is 94.44%.

LIME attributions (negative -1, neutral 0, positive +1)		
<b>CoLA example.</b> Groundtruth: acceptable		
S	Medea denied poisoning the phoenix.	acceptable 0.99
S <sub>1</sub>	poisoning the phoenix denied Medea.	acceptable 0.53
S <sub>2</sub>	phoenix Medea denied the poisoning.	acceptable 0.99
S <sub>3</sub>	Medea the poisoning phoenix denied.	unacceptable 0.95
S <sub>4</sub>	phoenix Medea denied the poisoning.	unacceptable 0.99
S <sub>5</sub>	Medea phoenix poisoning the denied.	unacceptable 0.96

Figure 6.2: Each CoLA example contains a single sentence. Here, we shuffled the words in the original sentence (S) five times to create five new sentences (S<sub>1</sub> to S<sub>5</sub>) and fed them to a RoBERTa-based classifier for predictions. Words that are important for or against the prediction (by LIME [134]) are in orange and blue, respectively. Most of the shuffled examples were classified into unacceptable label (i.e. grammatically incorrect) with even higher confidence score than the original ones.

<b>MRPC example.</b> Groundtruth: equivalent		
A	My decision today is not based on any one event . "	equivalent 0.99
B	Governor Rowland said his decision was " not based on any one event . "	
A <sub>1</sub>	event any is one decision based on My today not . "	equivalent 0.98
B	Governor Rowland said his decision was " not based on any one event . "	
A <sub>2</sub>	one based today not any My on event is decision . "	equivalent 0.98
B	Governor Rowland said his decision was " not based on any one event . "	

Figure 6.3: Each MRPC example contains a pair of sentences i.e. (A, B). Here, we shuffled the words in the original sentence (A) to create modified sentences (A<sub>1</sub> & A<sub>2</sub>) and fed them together with the original second sentence (B) to a RoBERTa-based classifier for predictions. Also, the classifier harnessed self-attention to detect the correspondence between similar words that appear in both sentences. That is, the top-3 pairs of words that were assigned the largest cross-sentence weights in a self-attention matrix (layer 0, head 7) are inside in the red, green, and blue rectangles.

RTE example. Groundtruth: entailment		
P	About 33.5 million people live in this massive conurbation. I would guess that 95% of the 5,000 officially foreign-capital firms in Japan are based in Tokyo.	entailment 0.90
H	About 33.5 million people live in Tokyo.	
P	About 33.5 million people live in this massive conurbation. I would guess that 95% of the 5,000 officially foreign-capital firms in Japan are based in Tokyo.	entailment 0.79
H <sub>1</sub>	people in million 33.5 live Tokyo About.	
P	About 33.5 million people live in this massive conurbation. I would guess that 95% of the 5,000 officially foreign-capital firms in Japan are based in Tokyo.	entailment 0.80
H <sub>2</sub>	33.5 in people About live Tokyo million.	

Figure 6.4: Each RTE example contains a pair of premises and hypotheses i.e. (P, H). We shuffled the words in the original hypothesis H to create modified hypotheses (H<sub>1</sub> & H<sub>2</sub>) and fed them together with the original premise (P) to a RoBERTa-based classifier for predictions. Also, the classifier harnessed self-attention to detect the correspondence between similar words that appear in both the premise and hypothesis. That is, the top-3 pairs of words that were assigned the largest premise-to-hypothesis weights in a self-attention matrix (layer 0, head 7) are inside in the red, green, and blue rectangles.



Task Name	Task Type	Label	GLUE dev-set processing				dev-r
			(a) dev set	(b) step 1	(c) step 2	(d) step 3	Total
CoLA	Acceptability	unacceptable	322	287	154	154	308
		acceptable	721	675	638	154	
RTE	NLI	not entailment	131	131	72	72	144
		entailment	146	145	127	72	
QQP	Paraphrase	not duplicate	25,545	22,907	20,943	12,683	25,366
		duplicate	14,885	14,000	12,683	12,683	
MRPC	Paraphrase	not equivalent	129	129	101	101	202
		equivalent	279	279	255	101	
SST-2	Sentiment	negative	428	427	402	402	804
		positive	444	443	420	402	
QNLI	NLI	not entailment	2,761	2,741	2,500	2,500	5,000
		entailment	2,702	2,690	2,527	2,500	
STS-B	Similarity	N/A	1,500	1,498	N/A	N/A	1,498

Table 6.4: The number of examples per class before (a) and after each of the three filtering steps to produce dev-r sets (described in Sec. 2.3.3) for RoBERTa-based classifiers. For each task, we repeated the same procedure for three sets of classifiers, for BERT-, RoBERTa-, ALBERT-based classifiers, respectively.

Model	Task	dev-r	dev-s	dev-s performance			Word-Order Sensitivity				
				performance	baseline	2-noun swap	1-gram	2-gram	3-gram	2-noun swap	1-gram
RoBERTa <sub>large</sub>	CoLA	100	50	70.80	51.40	55.62	57.98	<b>0.58</b>	<b>0.97</b>	<b>0.89</b>	<b>0.84</b>
	RTE	100	50	82.29	73.85	80.42	83.75	0.35	0.52	0.39	0.33
	SST-2	100	50	98.24	83.71	88.16	90.43	0.04	0.33	0.24	0.19
	MRPC	100	50	98.54	85.53	88.64	90.49	0.03	0.29	0.23	0.19
	QQP	100	50	87.13	86.84	90.65	92.60	0.26	0.26	0.19	0.15
	QNLI	100	50	95.26	91.12	95.20	96.46	0.09	0.18	0.10	0.07
	STS-B	90.43	N/A	88.95	85.47	87.20	87.98	N/A	N/A	N/A	N/A

Table 6.5: Accuracy of all models on dev-s examples (created by shuffling n-grams and swapping 2 nouns) and their Word-Order Sensitivity scores ( $\in [0, 1]$ ) across seven GLUE tasks. STS-B is a regression task and thus not comparable in word-order sensitivity with the other tasks, which are binary classification.

Task	GLUE dev set						
	CoLA (Acc)	RTE (Acc)	QQP (Acc)	MRPC (Acc)	SST-2 (Acc)	QNLI (Acc)	STS-B (Spearman Corr)
RoBERTa <sub>base</sub>	82.56	72.20	91.12	87.25	94.50	92.57	90.17
ALBERT <sub>base</sub>	81.21	72.20	90.25	87.99	91.40	91.78	90.82
BERT <sub>base</sub>	81.89	64.25	90.81	85.54	92.09	91.38	88.49
RoBERTa <sub>large</sub>	65.30	80.87	91.62	88.48	96.44	94.45	90.44
Average	82.78	72.38	90.95	87.32	93.61	92.55	89.98

Table 6.6: The dev-set performance of models finetuned from three different BERT “base” variants (12 self-attention layers and 12 heads) and one RoBERTa “large” model (24 self-attention layers and 16 heads) on seven GLUE tasks. These results match either those reported by original papers, [67] or GLUE leaderboard.

## 6.2 Supplementary Materials for Double Trouble (Chapter 3)

### 6.2.1 IM explanations have smaller attribution magnitude per token and lower word coverage

To further understand the impact of the fact that BERT tends to not change a to-remove token (Sec. 6.2.2), here, we quantify the magnitude of attribution given by IM and its coverage of important words in an example.

**Smaller attribution magnitude** Across three datasets, the average absolute values of attribution scores (which are  $\in [-1, 1]$ ) of IM are not higher than that of  $\text{LOO}_{\text{empty}}$  (Table 6.7). Especially in MultiRC, IM average attribution magnitude is  $4.5\times$  lower than that of  $\text{LOO}_{\text{empty}}$  (0.02 vs 0.09).

Method	SST	e-SNLI	MultiRC
$\text{LOO}_{\text{empty}}$	<b>0.22</b> $\pm$ 0.27	0.15 $\pm$ 0.24	<b>0.09</b> $\pm$ 0.09
IM	0.17 $\pm$ 0.27	0.15 $\pm$ 0.27	0.02 $\pm$ 0.09

Table 6.7: The average absolute value of attribution scores per token of  $\text{LOO}_{\text{empty}}$  is consistently higher than that of IM.

**Lower word coverage** We define *coverage* as the average number of highlighted tokens per example (e.g. Fig. 3.1) after binarizing a heatmap at the method’s optimal threshold.

The coverage of  $\text{LOO}_{\text{empty}}$  is much higher than that of IM on SST (40% vs 30%) and MultiRC examples (27% vs 6%), which is consistent with the higher *recall* of  $\text{LOO}_{\text{empty}}$  (Table 6.8; a vs. b). For e-SNLI, although IM has higher coverage than  $\text{LOO}_{\text{empty}}$  (14% vs. 10%), the coverage of  $\text{LOO}_{\text{empty}}$  is closer to the human coverage (9%). That is, IM assigns high attribution incorrectly to many words, resulting in a substantially lower *precision* than  $\text{LOO}_{\text{empty}}$ , according to e-SNLI L3 annotations (Table 3.3b; 0.3814 vs. 0.4687).

In sum, **chaining our results together**, we found BERT to often replace a token  $x_i$  by an exact-match with a high likelihood (Sec. 6.2.2), which sets a low empirical upper-bound on attribution

Explanations generated by	SST	e-SNLI		MultiRC
		L2	L3	
(a) LOO <sub>empty</sub>	40%	19%	10%	27%
(b) IM	30%	21%	14%	6%
(c) Human	37%	18%	9%	16%
# tokens per example	20	24		299

Table 6.8: Compared to IM, the coverage of LOO<sub>empty</sub> is closer to the coverage of human explanations.

values of IM, causing IM explanations to have smaller attribution magnitude. As the result, after binarization, fewer tokens remain highlighted in IM binary maps (e.g. Fig. 3.3).

## 6.2.2 By design, IM always assigns near-zero attribution to high-likelihood words regardless of classifiers

We observe that IM scores a substantially lower recall compared to LOO<sub>empty</sub> (e.g. 0.0630 vs. 0.2876; Table 3.3d). That is, IM tends to incorrectly assign too small of attribution to important tokens. Here, we test whether this low-recall issue is because BERT is highly accurate at predicting a single missing word from the remaining text and therefore assigns a high likelihood to such words in Eq. 3.3, causing low IM attribution in Eq. 3.2.

**Experiment** For each example in all three datasets, we replaced a single word by BERT’s top-1 highest-likelihood token and measured its likelihood and whether the replacement is the same as the original word.

**Results** Across SST, e-SNLI, and MultiRC, the top-1 BERT token matches exactly the original word ~49, 60, 65% of the time, respectively (Table 6.9a). This increasing trend of exact-match frequency (from SST, e-SNLI → MultiRC) is consistent with the example length in these three datasets, which is understandable as a word tends to be more predictable given a longer context. Among the tokens that human annotators label important, this exact-match frequency is similarly

high (Table 6.9b). Importantly, the average likelihood score of a top-1 exact-match token is high,  $\sim 0.81$ – $0.86$  (Table 6.9c). See Fig. 3.1 & Figs. 6.10–6.15 for qualitative examples.

% exact-match (uncased)	SST	e-SNLI	MultiRC
(a) over all tokens	48.94	59.43	64.78
(b) over human highlights	41.25	42.74	68.55
(c) Top-1 word’s likelihood	0.8229	0.8146	0.8556

Table 6.9: Top-1 likelihood scores (c) are the mean likelihood given by BERT for the top-1 predicted words that exactly match the original words (a).

Our findings are aligned with IM’s low recall. That is, if BERT fills in an exact-match  $\tilde{x}_i$  for an original word  $x_i$ , the prediction difference for this replacement  $\tilde{x}_i$  will be 0 in Eq. 3.4. Furthermore, a high likelihood of  $\sim 0.81$  for  $\tilde{x}_i$  sets an **empirical upper-bound of 0.19 for the attribution of the word  $x_i$** , which explains the insensitivity of IM to classifier randomization (Fig. 3.1; IM<sub>1</sub> to IM<sub>3</sub>).

The analysis here is also consistent with our additional findings that IM attribution tends to be smaller than that of  $\text{LOO}_{\text{empty}}$  and therefore leads to heatmaps of lower coverage of the words labeled important by humans (see Sec. 6.2.1).

### 6.2.3 Train BERT as masked language model on SST-2 to help filling in missing words

Integrating pre-trained BERT into LIME helps improve LIME explanations under two ROAR metrics (Sec. 3.6). However, the pre-trained BERT might be suboptimal for the cloze task on SST-2 sentences as it was pre-trained on Wikipedia and BookCorpus. Therefore, here, we take the pre-trained BERT, and finetune it on SST-2 training set using the masked language modeling objective. That is, we aim to test whether having a more specialized BERT would improve LIME results even further.

**Training details** We follow the hyperparameters by [67] and use Adam optimizer [79] with a learning rate of 0.00005,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$ , a batch size of 8, max sequence length of

512 and the ratio of tokens to mask of 0.15. We finetune the pre-trained BERT on SST-2 [143] train set and select the best model using the dev set.

**Results** On the SST-2 test set of 1,821 examples that contain 35,025 tokens in total, the cross-entropy loss of pre-trained BERT and BERT-SST2 are  $3.50 \pm 4.58$  and  $3.29 \pm 4.40$ , respectively. That is, our BERT finetuned on SST-2 is better than pre-trained BERT at predicting missing words in SST-2 sentences.

#### 6.2.4 Comparison between original and modified version of Input Marginalization

We follow [78] to reproduce results of the original Input Marginalization (IM) (Table 6.10a–b). To reduce the time complexity of Input Marginalization, we propose a modified version (IM-top10) by only marginalizing over the top-10 tokens sampled from BERT rather than using all tokens of likelihood  $\geq$  a threshold  $\sigma = 10^{-5}$ . We find that IM-top10 has comparable performance to that of the original IM (0.4732 vs. 0.4783; Table 6.10c). Our IM-top10 quantitative results are also close to the original numbers reported in [78] (0.4922 vs. 0.4972; Table 6.10).

Metrics ↓	a. IM ( <i>reported</i> in [78])	b. IM (Our reproduction)	c. IM-top10
Deletion	n/a	0.4783	0.4732
Deletion <sub>BERT</sub>	0.4972	0.4824	0.4922

Table 6.10: The approximation in of IM-top10 compared to the original IM under two metrics on SST-2 task. Both metrics measure AUC (lower is better).

We also find high qualitative similarity between heatmaps produced by two versions: IM vs. IM-top10 (Figs. 6.5–5). The average Pearson correlation score across the SST-2 8720-example test set is fairly high ( $\rho = 0.7224$ ). Thus, we use IM-top10 for all experiments in this paper.

#### 6.2.5 Sanity check result

SST-2 example. Groundtruth: positive & Prediction: positive (Confidence: 0.9996)										
IM	among	the	year	's	most	intriguing	explorations	of	alientation	.
	1.815	0.0118	0.54158	0.22394	1.03458	5.03105	1.94109	1.53783	-0.31367	-0.0026
IM <i>modified</i>	among	the	year	's	most	intriguing	explorations	of	alientation	.
	2.64685	0.03574	0.34608	0.51827	1.61421	5.74711	4.16886	2.30276	-0.35139	0.01431

Figure 6.5: Color map: **negative -1**, neutral 0, **positive +1**. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.988$ .

SST-2 example. Groundtruth: positive & Prediction: positive (Confidence: 0.9994)										
IM	a	solid	examination	of	the	male	midlife	crisis	.	
	1.07654	6.16288	2.91817	-0.01502	0.14328	-0.40143	0.1654	1.29851	1.2264	
IM <i>modified</i>	a	solid	examination	of	the	male	midlife	crisis	.	
	1.83532	5.85144	2.89864	0.00083	0.02024	-0.11491	0.06725	1.11138	0.05947	

Figure 6.6: Color map: **negative -1**, neutral 0, **positive +1**. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.917$ .

SST-2 example. Groundtruth: negative & Prediction: positive (Confidence: 0.9868)										
IM	rarely	has	leukemia	looked	so	shimmering	and	benign	.	
	6.62645	0.98643	-2.15698	-0.16744	0.59491	8.38053	3.50372	0.15773	0.05112	
IM <i>modified</i>	rarely	has	leukemia	looked	so	shimmering	and	benign	.	
	3.11005	0.58616	-3.29759	-0.20848	0.3003	8.72728	3.81542	0.26226	0.04914	

Figure 6.7: Color map: **negative -1**, neutral 0, **positive +1**. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.983$ .

SST-2 example. Groundtruth: negative & Prediction: negative (Confidence: 0.9950)														
IM	unfortunately	,	it	's	not	silly	fun	unless	you	enjoy	really	bad	movies	.
	0.97455	-0.00063	-0.00634	-0.15033	0.81403	-1.31111	0.76075	-0.03599	-0.00042	-0.22804	0.27508	1.36045	0.58812	-0.00371
IM <i>modified</i>	unfortunately	,	it	's	not	silly	fun	unless	you	enjoy	really	bad	movies	.
	1.6679	-0.00071	-0.00764	-0.35265	0.35085	-1.66804	-0.0029	0.37561	0.00036	-0.46997	0.35344	2.41716	0.78194	-0.00525

Figure 6.8: Color map: **negative -1**, neutral 0, **positive +1**. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.802$ .

SST-2 example. Groundtruth: positive & Prediction: negative (Confidence: 0.7999)																
IM	intriguing	documentary	which	is	emotionally	diluted	by	focusing	on	the	story	's	least	interesting	subject	.
	-7.28604	-2.3813	-4.68492	-0.11221	0.40301	8.17448	1.71521	0.06288	0.00117	0.06125	-0.64145	1.74269	9.00071	1.50607	-0.22335	-0.15134
IM <i>modified</i>	intriguing	documentary	which	is	emotionally	diluted	by	focusing	on	the	story	's	least	interesting	subject	.
	-3.96954	-1.1229	-2.38742	0.27984	4.07982	11.69405	0.68146	0.88004	-0.00308	0.04509	-0.43266	2.63444	9.97514	2.32102	-0.43297	0.03175

Figure 6.9: Color map: **negative -1**, neutral 0, **positive +1**. Attribution maps derived from both versions of IM have a high Pearson correlation  $\rho = 0.950$ .

Criteria	Method	SST-2	e-SNLI
(a) % tokens changing sign	LOO <sub>empty</sub>	<b>71.41</b> ± 17.12	<b>56.07</b> ± 21.82
	IM	62.27 ± 17.75	49.57 ± 20.35
(b) Average absolute of differences	LOO <sub>empty</sub>	<b>0.46</b> ± 0.18	<b>0.26</b> ± 0.14
	IM	0.31 ± 0.12	0.16 ± 0.12

Table 6.11: The percentage (%) of token (a) whose attribution scores change signs and (b) the average of absolute differences in attribution magnitude after classifier randomization (higher is better). IM is consistently more insensitive than LOO<sub>empty</sub> in both SST-2 and e-SNLI.

Accuracy ↓	ROAR			ROAR <sub>BERT</sub>			ROAR <sub>BERT_SST2</sub>		
	10%	20%	30%	10%	20%	30%	10%	20%	30%
(a) LIME	75.51 ± 0.55	75.30 ± 0.80	77.45 ± 0.70	78.14 ± 0.54	73.44 ± 0.65	70.57 ± 0.56	78.83 ± 1.28	74.47 ± 0.67	72.18 ± 1.02
(b) LIME <sub>BERT</sub>	<b>73.99</b> ± 0.74	72.22 ± 0.73	70.82 ± 0.86	<b>74.13</b> ± 0.72	70.44 ± 0.86	70.48 ± 0.63	<b>75.78</b> ± 0.22	71.33 ± 1.04	<b>68.76</b> ± 0.79
(c) LIME <sub>BERT_SST2</sub>	74.15 ± 1.26	<b>70.85</b> ± 0.89	<b>70.48</b> ± 0.98	76.19 ± 0.91	<b>69.77</b> ± 0.46	<b>67.61</b> ± 0.53	76.08 ± 0.46	<b>70.92</b> ± 0.64	71.08 ± 0.34

Table 6.12: Dev-set mean accuracy (%) of 5 models trained on the new SST-2 examples where  $N\%$  of highest-attribution words per example are removed (i.e. ROAR), replaced via BERT (i.e. ROAR<sub>BERT</sub>) or BERT finetuned on SST-2 to fill in a [MASK] token (i.e. ROAR<sub>BERT\_SST2</sub>). The original accuracy when no tokens are removed (i.e.  $N = 0\%$ ) is 92.62 ± 0.30. On average, under three metrics, LIME<sub>BERT</sub> (b) and LIME<sub>BERT\_SST2</sub> (c) are better, i.e. lower mean accuracy, than LIME (a).

SST example. Groundtruth: positive						
S	may not have generated many sparks , but with his affection for Astoria and its people <b>he has given his tale a warm glow</b> .					
S <sub>1</sub>	may not have generated many sparks , but with his affection for Astoria and its people <b>he</b> has <b>given</b> his tale <b>a</b> warm glow .					
	<b>0.9494</b>	<b>he</b>	<b>0.9105</b>	<b>given</b>	<b>0.9632</b>	<b>a</b>
	0.0103	it	0.0285	lent	0.0270	its
	0.0066	,	0.0143	gave	0.0033	another

Figure 6.10: BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in **red**, **green**, **blue** rectangles) and assigns a high likelihood to the tokens that are labeled **important** by humans in the SST positive example. In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods.



SST example. Groundtruth: negative						
S	Villeneuve spends too much time wallowing in Bibi 's generic angst ( there are a lot of shots of her gazing out windows ) .					
S <sub>1</sub>	Villeneuve spends too much time wallowing in Bibi 's generic angst ( there are a lot of shots of her gazing out windows ) .					
	<b>0.9987</b>	<b>much</b>	<b>0.9976</b>	<b>time</b>	<b>0.9675</b>	<b>in</b>
	0.0011	little	0.0005	money	0.0066	with
	0.0001	some	0.0003	space	0.0062	on

Figure 6.11: BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled important by humans in the SST negative example. In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods.

e-SNLI example. Groundtruth: entailment						
P	The two farmers are working on a piece of John Deere equipment .					
H	John Deere equipment is being worked on by two farmers					
P <sub>1</sub>	The two farmers are working on a piece of John Deere equipment					
H <sub>1</sub>	John Deere equipment is being worked on by two farmers					
	<b>0.9995</b>	<b>john</b>	<b>0.9877</b>	<b>equipment</b>	<b>0.9711</b>	<b>john</b>
	0.0000	johnny	0.0057	machinery	0.0243	the
	0.0000	henry	0.0024	hardware	0.0005	a

Figure 6.12: BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled important by humans in the e-SNLI entailment example which contains a pair of premise (P) and hypothesis (H). In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods.

e-SNLI example. Groundtruth: neutral						
P	A man uses a projector to give a presentation .					
H	A man is giving a presentation in front of a large crowd .					
P <sub>1</sub>	A man uses a projector to give a presentation .					
H <sub>1</sub>	A man is giving a presentation in front of a large crowd .					
	<b>1.0000</b>	<b>front</b>	<b>0.9999</b>	<b>of</b>	<b>0.9993</b>	<b>a</b>
	0.0000	view	0.0000	to	0.0005	the
	0.0000	presence	0.0000	with	0.0001	another

Figure 6.13: BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled important by humans in the e-SNLI neutral example which contains a pair of premise (P) and hypothesis (H). In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods.

MultiRC example. Groundtruth & Prediction: True (confidence: 0.98)																			
P	<p>What causes a change in motion ? The application of a force . Any time an object changes motion , a force has been applied . In what ways can this happen ? Force can cause an object at rest to start moving . Forces can cause objects to speed up or slow down . Forces can cause a moving object to stop . Forces can also cause a change in direction . <b>In short , forces cause changes in motion . The moving object may change its speed , its direction , or both .</b> We know that changes in motion require a force . We know that the size of the force determines the change in motion . How much an objects motion changes when a force is applied depends on two things . It depends on the strength of the force . It also depends on the objects mass . Think about some simple tasks you may regularly do . You may pick up a baseball . This requires only a very small force .</p>																		
Q	What factors cause changes in motion of a moving object ?																		
A	The object 's speed , direction , or both speed and direction																		
P <sub>1</sub>	<p>What causes a change in motion ? The application of a force . Any time an object changes motion , a force has been applied . In what ways can this happen ? Force can cause an object at rest to start moving . Forces can cause objects to speed up or slow down . Forces can cause a moving object to stop . Forces can also cause a change in direction . In short , forces cause changes in motion . The <b>moving</b> object may <b>change</b> its speed , its direction , <b>or</b> both . We know that changes in motion require a force . We know that the size of the force determines the change in motion . How much an objects motion changes when a force is applied depends on two things . It depends on the strength of the force . It also depends on the objects mass . Think about some simple tasks you may regularly do . You may pick up a baseball . This requires only a very small force .</p>																		
	<table border="0"> <tr> <td><b>0.9927</b></td> <td><b>moving</b></td> <td><b>0.9891</b></td> <td><b>change</b></td> <td><b>0.9995</b></td> <td><b>or</b></td> </tr> <tr> <td>0.0023</td> <td>moved</td> <td>0.0033</td> <td>alter</td> <td>0.0004</td> <td>and</td> </tr> <tr> <td>0.0016</td> <td>stationary</td> <td>0.0018</td> <td>affect</td> <td>0.0000</td> <td>etc</td> </tr> </table>	<b>0.9927</b>	<b>moving</b>	<b>0.9891</b>	<b>change</b>	<b>0.9995</b>	<b>or</b>	0.0023	moved	0.0033	alter	0.0004	and	0.0016	stationary	0.0018	affect	0.0000	etc
<b>0.9927</b>	<b>moving</b>	<b>0.9891</b>	<b>change</b>	<b>0.9995</b>	<b>or</b>														
0.0023	moved	0.0033	alter	0.0004	and														
0.0016	stationary	0.0018	affect	0.0000	etc														
Q <sub>1</sub>	John Deere equipment is being worked on by two farmers																		
A <sub>1</sub>	The object 's speed , direction , or both speed and direction																		

Figure 6.14: BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in **red**, **green**, **blue** rectangles) and assigns a high likelihood to the tokens that are labeled **important** by humans in the MultiRC True example which contains a triplet of paragraph (P), question (Q) and answer (A). In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods.

MultiRC example. Groundtruth & Prediction: False (confidence: 0.74)																			
P	There have been many organisms that have lived in Earths past . Only a tiny number of them became fossils . Still , scientists learn a lot from fossils . <b>Fossils are our best clues about the history of life on Earth . Fossils provide evidence about life on Earth . They tell us that life on Earth has changed over time .</b> Fossils in younger rocks look like animals and plants that are living today . Fossils in older rocks are less like living organisms . Fossils can tell us about where the organism lived . Was it land or marine ? Fossils can even tell us if the water was shallow or deep . Fossils can even provide clues to ancient climates .																		
Q	What are three things scientists learn from fossils ?																		
A	Who lived in prehistoric times																		
P <sub>1</sub>	There have been many organisms that have lived in Earths past . Only a tiny number of them became fossils . Still , scientists learn a lot from fossils . Fossils are our best clues about the history of <span style="border: 1px solid red; padding: 2px;">life</span> on Earth . Fossils provide evidence about life on Earth . They tell us that life on <span style="border: 1px solid green; padding: 2px;">Earth</span> has changed over <span style="border: 1px solid blue; padding: 2px;">time</span> . Fossils in younger rocks look like animals and plants that are living today . Fossils in older rocks are less like living organisms . Fossils can tell us about where the organism lived . Was it land or marine ? Fossils can even tell us if the water was shallow or deep . Fossils can even provide clues to ancient climates .																		
	<table border="0"> <tr> <td><b>0.9984</b></td> <td><span style="border: 1px solid red; padding: 2px;">life</span></td> <td><b>0.9982</b></td> <td><span style="border: 1px solid green; padding: 2px;">earth</span></td> <td><b>0.9980</b></td> <td><span style="border: 1px solid blue; padding: 2px;">time</span></td> </tr> <tr> <td>0.0004</td> <td>living</td> <td>0.0007</td> <td>mars</td> <td>0.0007</td> <td>millennia</td> </tr> <tr> <td>0.0002</td> <td>things</td> <td>0.0002</td> <td>land</td> <td>0.0003</td> <td>history</td> </tr> </table>	<b>0.9984</b>	<span style="border: 1px solid red; padding: 2px;">life</span>	<b>0.9982</b>	<span style="border: 1px solid green; padding: 2px;">earth</span>	<b>0.9980</b>	<span style="border: 1px solid blue; padding: 2px;">time</span>	0.0004	living	0.0007	mars	0.0007	millennia	0.0002	things	0.0002	land	0.0003	history
<b>0.9984</b>	<span style="border: 1px solid red; padding: 2px;">life</span>	<b>0.9982</b>	<span style="border: 1px solid green; padding: 2px;">earth</span>	<b>0.9980</b>	<span style="border: 1px solid blue; padding: 2px;">time</span>														
0.0004	living	0.0007	mars	0.0007	millennia														
0.0002	things	0.0002	land	0.0003	history														
Q <sub>1</sub>	What are three things scientists learn from fossils ?																		
A <sub>1</sub>	Who lived in prehistoric times																		

Figure 6.15: BERT<sub>base</sub> often correctly predicts the masked tokens (denoted in red, green, blue rectangles) and assigns a high likelihood to the tokens that are labeled **important** by humans in the MultiRC False example which contains a triplet of paragraph (P), question (Q) and answer (A). In each panel, we show the top-3 tokens suggested by BERT and their associated likelihoods.

SST example. Groundtruth & Prediction: negative (confidence: 1.00)	
S	For starters , the story <b>is just too slim</b> .
S <sub>IM</sub>	For <b>starters</b> , the <b>story</b> is <b>just too</b> slim .
	IoU: 0.33, precision: 0.50, recall: 0.50
S <sub>LOO</sub>	For starters , the story is <b>just too slim</b> .
	IoU: <b>0.75</b> , precision: <b>1.00</b> , recall: <b>0.75</b>

Figure 6.16: The set of **explanatory words** given by LOO<sub>empty</sub> covers 75% of **human** highlights with higher precision and IoU in the SST negative example while there are a half of **tokens highlighted by IM** are in correlation with human explanations.

e-SNLI example. Groundtruth & Prediction: contradiction (confidence: 1.00)	
P	Two men are <b>cooking</b> food together on the corner of the street .
H	The two men are <b>running</b> in a race .
P <sub>IM</sub>	Two men are cooking food together on the corner of the street .
H <sub>IM</sub>	<b>The</b> two men are <b>running</b> in a <b>race</b> .
	IoU: 0.25, precision: 0.33, recall: 0.50
P <sub>LOO</sub>	Two men are <b>cooking</b> food together on the corner of the street .
H <sub>LOO</sub>	The two <b>men</b> are <b>running</b> in a <b>race</b> .
	IoU: <b>0.50</b> , precision: <b>0.50</b> , recall: <b>1.00</b>

Figure 6.17: The set of **explanatory words** given by LOO<sub>empty</sub> covers **all** highlights (higher precision and IoU) that are important to **human** in the e-SNLI contradiction example which contains a pair of premise (P) and hypothesis (H) while there are **a half** of **tokens highlighted by IM** are in correlation with human explanations.

e-SNLI example. Groundtruth & Prediction: neutral (confidence: 1.00)	
P	Woman in a dress standing in front of a line of a clothing line , with clothes hanging on the line .
H	Her dress is <b>dark blue</b> .
P <sub>IM</sub>	<b>Woman</b> in <b>a</b> <b>dress</b> standing in front of a line of a clothing line , with clothes hanging on the line .
H <sub>IM</sub>	Her <b>dress</b> is dark blue .
	IoU: 0.00, precision: 0.00, recall: 0.00
P <sub>LOO</sub>	Woman in a <b>dress</b> standing in front of a line of a clothing line , with clothes hanging on the line .
H <sub>LOO</sub>	<b>Her dress is dark blue</b> .
	IoU: <b>0.33</b> , precision: <b>0.33</b> , recall: <b>1.00</b>

Figure 6.18: The set of **explanatory words** given by LOO<sub>empty</sub> covers **all** highlights (higher precision and IoU) that are important to **human** in the e-SNLI neutral example which contains a pair of premise (P) and hypothesis (H) while there are **none** **tokens highlighted by IM** are in correlation with human explanations.

MultiRC example. Groundtruth & Prediction: True (confidence: 0.90)	
P	There have been many organisms that have lived in Earth's past . Only a tiny number of them became fossils . Still , scientists learn a lot from fossils . Fossils are our best clues about the history of life on Earth . Fossils provide evidence about life on Earth . They tell us that life on Earth has changed over time . Fossils in younger rocks look like animals and plants that are living today . Fossils in older rocks are less like living organisms . Fossils can tell us about where the organism lived . Was it land or marine ? Fossils can even tell us if the water was shallow or deep . Fossils can even provide clues to ancient climates .
Q	What happened to some organisms that lived in Earth 's past ?
A	They became fossils . Others did not become fossils
P <sub>IM</sub>	There have been many organisms that have lived in Earth's past . Only a tiny number of them became fossils . Still , scientists learn a lot from fossils . Fossils are our best clues about the history of life on Earth . Fossils provide evidence about life on Earth . They tell us that life on Earth has changed over time . Fossils in younger rocks look like animals and plants that are living today . Fossils in older rocks are less like living organisms . Fossils can tell us about where the organism lived . Was it land or marine ? Fossils can even tell us if the water was shallow or deep . Fossils can even provide clues to ancient climates .
Q <sub>IM</sub>	What happened to some organisms that lived in Earth 's past ?
A <sub>IM</sub>	They became fossils . Others did not become fossils
	IoU: 0.16, precision: 0.50, recall: 0.19
P <sub>LOO</sub>	There have been many organisms that have lived in Earth's past . Only a tiny number of them became fossils . Still , scientists learn a lot from fossils . Fossils are our best clues about the history of life on Earth . Fossils provide evidence about life on Earth . They tell us that life on Earth has changed over time . Fossils in younger rocks look like animals and plants that are living today . Fossils in older rocks are less like living organisms . Fossils can tell us about where the organism lived . Was it land or marine ? Fossils can even tell us if the water was shallow or deep . Fossils can even provide clues to ancient climates .
Q <sub>LOO</sub>	What happened to some organisms that lived in Earth 's past ?
A <sub>LOO</sub>	They became fossils . Others did not become fossils
	IoU: <b>0.56</b> , precision: <b>0.57</b> , recall: <b>0.95</b>

Figure 6.19: The set of **explanatory words** given by LOO<sub>empty</sub> covers 95% of **human** highlights with higher precision and IoU in the MultiRC True example which contains a triplet of paragraph (P), question (Q) and answer (A) while there are only few tokens given by **IM** are in correlation with human explanations.

MultiRC example. Groundtruth & Prediction: False (confidence: 0.99)	
P	There have been many organisms that have lived in Earths past . Only a tiny number of them became fossils . Still , scientists learn a lot from fossils . Fossils are our best clues about the history of life on Earth . Fossils provide evidence about life on Earth . They tell us that life on Earth has changed over time . Fossils in younger rocks look like animals and plants that are living today . Fossils in older rocks are less like living organisms . Fossils can tell us about where the organism lived . Was it land or marine ? Fossils can even tell us if the water was shallow or deep . Fossils can even provide clues to ancient climates .
Q	What is a major difference between younger fossils and older fossils ?
A	Older rocks are rougher and thicker than younger fossils
P <sub>IM</sub>	There have been many organisms that have lived in Earths past . Only a tiny number of them became fossils . Still , <b>scientists</b> learn a lot from fossils . Fossils are our best <b>clues</b> about the history of life on Earth . Fossils provide evidence about life on Earth . They tell us that life on Earth has changed over time . Fossils in younger rocks look like animals and plants that are living today . Fossils in older rocks <b>are</b> less like <b>living</b> organisms . Fossils can tell us about where the organism lived . Was it <b>land</b> or <b>marine</b> ? Fossils can even tell us if the water was shallow or deep . Fossils can even provide clues to ancient climates .
Q <sub>IM</sub>	What is a major difference between younger fossils and older fossils ?
A <sub>IM</sub>	Older <b>rocks</b> are <b>rougher</b> and <b>thicker than younger</b> fossils
	IoU: 0.06, precision: 0.18, recall: 0.08
P <sub>LOO</sub>	There have been many <b>organisms</b> that have lived in <b>Earths</b> past . Only a tiny number of them became fossils . Still , scientists learn a lot from fossils . <b>Fossils</b> are our <b>best clues</b> about the history of <b>life</b> on <b>Earth</b> . Fossils provide evidence about life on Earth . They tell us that <b>life on</b> Earth has changed over time . <b>Fossils in</b> younger <b>rocks</b> look like <b>animals and plants that are living today</b> . <b>Fossils</b> in older rocks <b>are</b> less like <b>living organisms</b> . <b>Fossils can tell us about where the organism lived</b> . <b>Was it land or marine ? Fossils can even tell us if</b> the water was shallow or deep . <b>Fossils can even provide clues to ancient climates</b> .
Q <sub>LOO</sub>	What <b>is</b> a <b>major</b> difference between younger <b>fossils</b> and older <b>fossils</b> ?
A <sub>LOO</sub>	Older <b>rocks</b> are <b>rougher</b> and <b>thicker than younger</b> fossils
	IoU: <b>0.22</b> , precision: <b>0.25</b> , recall: <b>0.67</b>

Figure 6.20: The set of **explanatory words** given by LOO<sub>empty</sub> covers two thirds of **human** highlights with higher precision and IoU in the MultiRC False example which contains a triplet of paragraph (P), question (Q) and answer (A) while there are two tokens given by IM are in correlation with human explanations.

SST example. Groundtruth & Prediction: positive	
S	Enormously <b>entertaining</b> for moviegoers of <b>any age</b> .
S <sub>1</sub>	<del>Enormously</del> entertaining for moviegoers of any age .
S <sub>2</sub>	Enormously <del>entertaining</del> for moviegoers of any age .
S <sub>3</sub>	Enormously entertaining <del>for</del> moviegoers of any age .
S <sub>4</sub>	Enormously entertaining for <del>moviegoers</del> of any age .
S <sub>5</sub>	Enormously entertaining for moviegoers <del>of</del> any age .
S <sub>6</sub>	Enormously entertaining for moviegoers of <del>any</del> age .
S <sub>7</sub>	Enormously entertaining for moviegoers of any <del>age</del> .

Figure 6.21: When a word is **removed**, the predicted labels of all resulting sentences (S<sub>1</sub> to S<sub>7</sub>) are still positive with a confidence score of 1.0.

e-SNLI example. Groundtruth: entailment		Prediction
P	Two women having <b>drinks</b> and <b>smoking cigarettes</b> at the bar .	entailment (0.99)
H	Two women are at a <b>bar</b> .	
P <sub>1</sub>	<b>Two</b> women having drinks and smoking cigarettes at the bar .	entailment (0.98)
H <sub>1</sub>	Two women are at a bar .	
P <sub>2</sub>	Two <b>women</b> having drinks and smoking cigarettes at the bar .	<b>neutral</b> (0.93)
H <sub>2</sub>	Two women are at a bar .	
P <sub>3</sub>	Two women <b>having</b> drinks and smoking cigarettes at the bar .	entailment (0.99)
H <sub>3</sub>	Two women are at a bar .	
P <sub>4</sub>	Two women having <b>drinks</b> and smoking cigarettes at the bar .	entailment (0.99)
H <sub>5</sub>	Two women are at a bar .	
P <sub>5</sub>	Two women having drinks <b>and</b> smoking cigarettes at the bar .	entailment (0.99)
H <sub>5</sub>	Two women are at a bar .	
P <sub>6</sub>	Two women having drinks and <b>smoking</b> cigarettes at the bar .	entailment (0.99)
H <sub>6</sub>	Two women are at a bar .	
P <sub>7</sub>	Two women having drinks and smoking <b>cigarettes</b> at the bar .	entailment (0.99)
H <sub>7</sub>	Two women are at a bar .	
P <sub>8</sub>	Two women having drinks and smoking cigarettes <b>at</b> the bar .	entailment (0.98)
H <sub>8</sub>	Two women are at a bar .	
P <sub>9</sub>	Two women having drinks and smoking cigarettes <b>at the</b> bar .	entailment (0.98)
H <sub>9</sub>	Two women are at a bar .	
P <sub>10</sub>	Two women having drinks and smoking cigarettes at the <b>bar</b> .	entailment (0.97)
H <sub>10</sub>	Two women are at a bar .	
P <sub>11</sub>	Two women having drinks and smoking cigarettes at the bar <b>.</b>	entailment (0.99)
H <sub>11</sub>	Two women are at a bar .	
P <sub>12</sub>	Two women having drinks and smoking cigarettes at the bar .	entailment (0.99)
H <sub>12</sub>	<b>Two</b> women are at a bar .	
P <sub>13</sub>	Two women having drinks and smoking cigarettes at the bar .	entailment (0.98)
H <sub>13</sub>	Two <b>women</b> are at a bar .	
P <sub>14</sub>	Two women having drinks and smoking cigarettes at the bar .	entailment (0.99)
H <sub>14</sub>	Two women <b>are</b> at a bar .	
P <sub>15</sub>	Two women having drinks and smoking cigarettes at the bar .	entailment <b>(0.84)</b>
H <sub>15</sub>	Two women are <b>at</b> a bar .	
P <sub>16</sub>	Two women having drinks and smoking cigarettes at the bar .	entailment (0.97)
H <sub>16</sub>	Two women are at <b>a</b> bar .	
P <sub>17</sub>	Two women having drinks and smoking cigarettes at the bar .	entailment <b>(0.54)</b>
H <sub>17</sub>	Two women are at a <b>bar</b> .	
P <sub>18</sub>	Two women having drinks and smoking cigarettes at the bar .	entailment (0.95)
H <sub>18</sub>	Two women are at a bar <b>.</b>	

Figure 6.22: The removal of each token in both premise and hypothesis in e-SNLI example which contains a pair of premise (P) and hypothesis (H) **infrequently change the prediction**. Specifically, only the example of (P<sub>2</sub>, H<sub>2</sub>) shifted its prediction to neutral while the remaining partially-removed examples do not change their original prediction with high confidence score in parentheses.

## 6.3 Supplementary Materials for Phrase in Context (Chapter 4)

### 6.3.1 Training models on Phrase Similarity

**Hyperparameters** We train each *BERT-based classifier* for a maximum of 100 epochs with early stopping monitored on validation accuracy (patience of 10 epochs). We use a batch size of 200 and Adam optimizer with learning rate  $\alpha = 0.0001$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and  $\epsilon = 10^{-8}$ .

**Training time** On average, with early stopping, training a single model using one V100 GPU takes  $\sim 5$  and  $\sim 8$  mins for non-context and context settings, respectively.

### 6.3.2 Training SS models on Phrase Retrieval

We finetune each SS model that consists of a linear layer on top of a pretrained model selected in Sec. 4.5 to predict the start and end indices of answers (as the common setup in BERT SS models [30, 4]). The format of a tokenized input is “[CLS] query [SEP] document [SEP]” with maximum sequence length of 4,096 for  $\text{Longformer}_{\text{Base}}$  and  $\text{Longformer}_{\text{Large}}$  and 512 for the remaining models. If the document exceeds the maximum sequence length, it is split into smaller features for prediction and thus start and end indices with the highest confidence scores are selected.

**Hyperparameters** We follow HuggingFace scheme to finetune the SS models for 2 epochs using Adam optimizer with learning rate  $\alpha = 0.00003$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$ . The batch size varies from 1 to 8 for each model: On one V100 GPU, the “base” models can handle 8 examples while the “large” BERT models can only fit 2–4 examples into 16GB of memory. For  $\text{Longformer}_{\text{Large}}$ , we use an A100 GPU to feed one PR-page example into the model. We take the smallest dev-loss models from the training and report their test-set results.

**Training time** On average, training a single SS model for 2 epochs using one A100 GPU takes  $\sim 20$  mins for base models and  $\sim 9.5$  hours for  $\text{Longformer}_{\text{Large}}$ .



### 6.3.3 Data collection

From a Wikipedia dump, we perform a 6-step procedure (summarized in Table 6.13) for mining a list of *m*NPs sorted descendingly by their likelihood of containing multiple senses. The most polysemous 19,500 *m*NPs are then passed to experts for annotation (Section 4.3.2) and others for verification (Section 4.3.3).

**Step 1: Download Wiki articles** We download a Wikipedia dump file [147] that contains ~15.78M Wikipedia articles and filter out all empty pages to arrive at ~6.27M non-empty articles.

**Step 2: Extract phrases** We use NLTK sentence splitter [12] to split each Wikipedia article into multiple sentences. And then we use SpaCy [64] to extract noun phrases and proper nouns as we do not collect syntactically strict phrases. For each phrase, we remove all preceding and succeeding stopwords (those among the 179 stopwords in NLTK v3.6.5) and non-alphanumeric characters. We remove stopwords because they tend to create more pairs of phrases with lexical overlap, rendering the phrase similarity task easier. We then remove *unigram* phrases to arrive at ~286.78M *m*NPs. For example, from “a massive figure”, we changed to “massive figure”, which would be our final phrase after this step. For each *m*NP, we construct a 3-tuple (phrase, sentence, metadata), i.e. the phrase, its container sentence, and metadata for identifying the Wikipedia webpage (hereafter, page).

**Step 3: Remove phrases of a single context** We further remove all phrases that (1) contain non-ASCII characters (e.g. “phaenná nâsos”, which are non-English); and (2) appear only once, i.e. keeping those that occur in multiple sentences since we look for *polysemous m*NPs, which have multiple senses and contexts. After this step, ~17.96M phrases remain.

While some phrases with non-ASCII characters are also commonly used in English (e.g., “déjà vu”), we find only 2.48% of phrases at this stage contain non-ASCII characters, and 29% of them are common in English. In short, we are removing only 0.72% of the English phrases that contain non-ASCII characters in Step 3.

**Step 4: Find phrases of polysemous words** To increase the chance of collecting polysemous *m*NPs, we only keep *m*NPs that have at least one word in the list of 2,345 unique multiple-sense words of WiC [68], arriving at  $\sim 6.5$ M *m*NPs, each appearing in  $\geq 2$  sentences and in  $\geq 1$  Wikipedia pages. We empirically find that Step 4 is important and substantially increases our chance of finding polysemous *m*NPs (compared to skipping Step 4).

**Step 5: Find phrases in distinct contexts** We observe that a *m*NP is likely to be polysemous when (a) its context sentences are semantically different; and (b) its context Wikipedia pages are of dissimilar categories (e.g. “massive figure” in finance vs. history; Figure 6.30).

To implement this filter, we form all possible triplets (phrase, sentence<sub>1</sub>, sentence<sub>2</sub>) from the list of context sentences of each *m*NP<sup>2</sup>. We compute the cosine similarity of two sentences at the CLS embedding space of a SimCSE [45] provided on HuggingFace [51]. To find triplets where the two sentences are semantically dissimilar, we keep only the triplets where (sentence<sub>1</sub>, sentence<sub>2</sub>) has a low cosine similarity, i.e.  $\in [-0.3, 0.2]$  and the length difference of the two sentences is  $< 4$  words (as two sentences of substantially different lengths often have a low cosine similarity regardless of their semantic differences). As the result, there are  $\sim 600$ K triplets remaining after this step.

We further re-rank these  $\sim 600$ K descendingly by the dissimilarity of the lists of Wikipedia categories<sup>3</sup> of the context pages that contain sentence<sub>1</sub> and sentence<sub>2</sub>. That is, we treat each Wikipedia page’s comma-separated list of categories as an input text to SimCSE and sort the  $\sim 600$ K descendingly by the cosine similarity of the resultant embeddings.

**Step 6: Select data for annotation** Before asking annotators to label our sorted phrases we perform final filtering by removing proper nouns and phrases whose Wikipedia documents contain missing words.

---

<sup>2</sup>For computational tractability, we only keep at most 32 context sentences per *m*NP where each sentence’s length in words is  $\in [5, 25]$ .

<sup>3</sup>We use the provided Wikipedia API [146] to obtain the categories for each article as the dump file has no category-related information.

We perform final filtering to ensure the data given to annotators is in a proper format. That is, from ~600K phrases, we filter down to ~475K phrases by applying two filters: (1) Remove all phrases that are proper nouns (i.e. POS tagging returns PROPN) since proper nouns often refer to a single identity and thus unambiguous; (2) Remove all phrases that have a newline character and all phrases whose context Wikipedia page contains missing words (i.e. errors in the Wikipedia dump).

As the result, we obtain a list of ~475K phrases sorted by their estimate chance of carrying two different senses. After manual inspection, we take the top 19,500 triplets of the format (phrase, page<sub>1</sub>, page<sub>2</sub>)—i.e. a phrase  $p$  and its two context Wikipedia pages where  $p$  is the most likely to have two different senses (e.g., see “massive figure” in Figure 4.1a)—and hire linguistic experts to annotate them.

**Our manual inspection** involves taking 1,000 random triplets and manually reading them. We find that at least ~30% of the 1,000-triplet subset contain a polysemous target phrase  $p$  and two Wikipedia pages that give  $p$  two unique meanings. We perform this manual inspection repeatedly throughout the process of inventing and refining the data collection process in order to arrive at the final list of steps as presented in this paper.

### **Biases in the data collection**

While there are many filtering steps in our data collection above, most of them are data cleaning filters that are typically needed in a regular NLP dataset construction.

We recognize that there are *three key filters* in our system that impose strong biases:

1. In Step 4, we use only phrases that contain one word in the WiC. That is, we find Step 4 to substantially increase our chance of finding triplets with a polysemous target phrase. We have added this note in the Data Collection description. It is possible to remove Step 4, but that would require a larger human annotation effort to reach the same 15K labeled triplets.
2. In Step 5, we rely on SimCSE to find target phrases that are placed in two sentences of dissimilar meanings.

3. In Step 5, we rely on SimCSE to find target phrases that are placed in two Wikipedia pages of distinct topics.

### 6.3.4 Statistics for search queries in Yahoo Search Query dataset

We analyze 4,496 user queries released in the Yahoo Search Query Log To Entities dataset [169] and use SpaCy tokenizer [64] to classify them into 4 main categories: Noun phrases, verb phrases, URLs and others. As a result, noun phrases are the most common query type from users with 3,576 queries (~79.54%) followed by URLs with 675 queries (~15.01%) while verb phrases and other types are less preferred by users. Moreover, the average length of the real user queries is ~1.60 which is quite close to our PS task with ~2.27.

Table 6.14: Statistics of Yahoo queries across different query types.

Query type	# queries	Percentage (%)
Noun phrases	3,576	79.54
Verb phrases	148	3.29
URLs	675	15.01
Others	97	2.16
Total	4,496	100.00

### 6.3.5 Verification of Phrase Similarity

To enhance the quality of the proposed PiC benchmark, we hire three additional Upwork experts to verify the correctness of PS examples where two phrases are supposed to be *non-equivalent* for negative examples (e.g. massive figure and giant number in Figure 4.1b) or *equivalent* for positive examples (e.g. massive figure and huge model in Figure 4.1b), and keep an example if it is endorsed by at least two experts (the rest is discarded from PS).

Two Upwork verifiers  $A_1$  and  $A_2$  start checking 5,104 *negative* examples and the third verifier  $A_3$  is responsible for breaking the ties if  $A_1$  and  $A_2$  disagree with each other (see Figure 6.23). Both  $A_1$  and  $A_2$  are asked to provide corrections when they do not agree with the labels. As a result, 4,935 out of 5,104 examples are accepted by pairs of  $(A_1, A_2)$ ,  $(A_1, A_3)$  or  $(A_2, A_3)$ , 68 examples incorrect at first but are modified by either  $A_1$  or  $A_2$  and endorsed by  $A_3$ . In total, we reject 101 negative examples because there are not at least two experts agreeing with the annotations.

We repeat the same procedure to verify 5,104 *positive* examples. In sum, we retain 5,002 examples including 4,904 examples accepted by pairs of two verifiers and 98 examples incorrect at first but are modified by either  $A_1$  or  $A_2$  and endorsed by  $A_3$ . There are 102 positive examples rejected because there are not at least two experts agreeing with the annotations (Figure 6.24).

After this verification round, we collect 5,003 *negative* examples and 5,002 *positive* examples. and randomly exclude 1 negative example to make the dataset balance which results in 10,004 examples in total for PS.

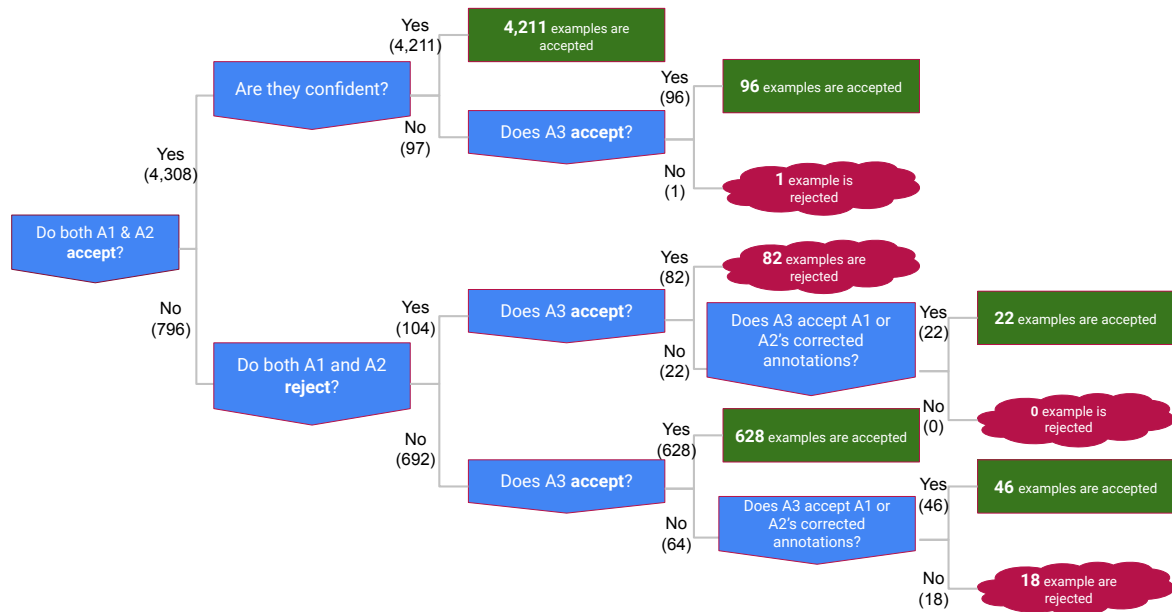


Figure 6.23: A decision tree describing our verification process for PS that involves three experts. Red, green and blue cells represent Reject, Accept decision and Questions. The numbers of examples for each branch are shown in parentheses.

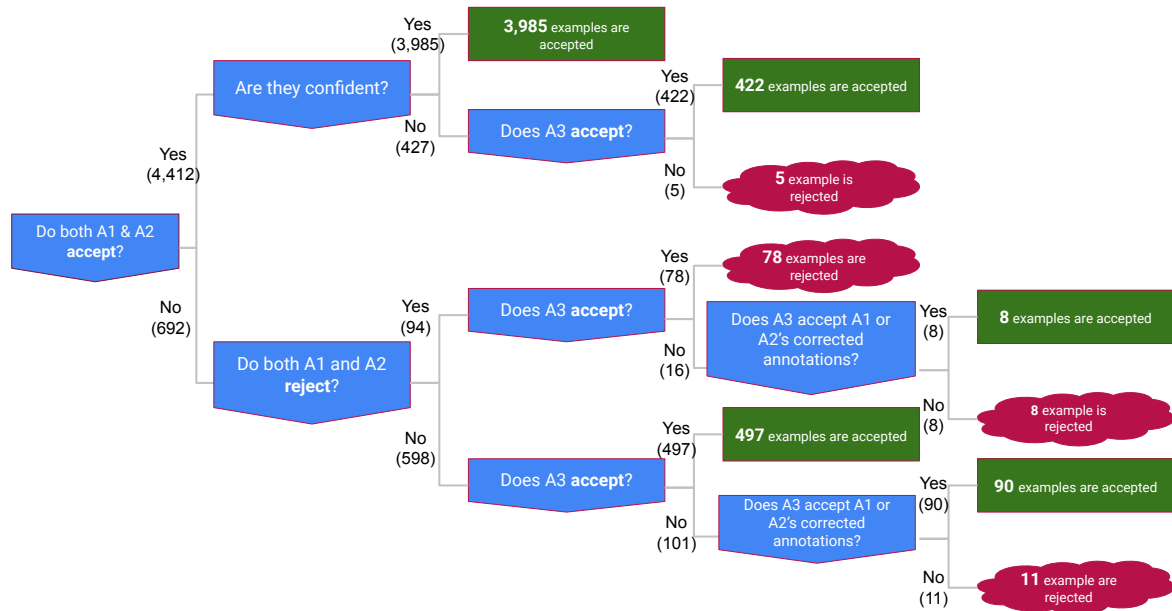


Figure 6.24: A decision tree describing our verification process for PS that involves three experts. Red, green and blue cells represent Reject, Accept decision and Questions. The numbers of examples for each branch are shown in parentheses.

Table 6.13: Summary of our 3-stage data construction.  $p, s, m, d, q, l$  denote target phrase, sentence, metadata, document, query, and label, respectively.

	Remaining #	Data type	Description
<b>Section 4.3.1 Data Collection</b>			
Step 1: Download Wiki articles	~6.27M	articles	Remove ~9.51M empty articles.
Step 2: Extract phrases	~286.78M	$(p, s, m)$	Extract noun phrases and proper nouns along with their context sentences from Wikipedia articles.
Step 3: Remove phrases of a single context	~17.96M	$(p, [s_1, \dots, s_n], m)$	For each phrase, gather all sentences where that phrase is used.
Step 4: Find phrases of polysemous words	~6.5M	$(p, [s_1, \dots, s_n], m)$	Filter those phrases that do not contain WiC words.
Step 5: Find phrases in distinct contexts			Sort by $X_i$ and apply filters to find pairs of sentences where their phrase potentially has different meanings.
- Sort and filter by semantic dissimilarity	~600K	$(p, s_1, s_2, m)$	$X_1$ : cosine similarity scores of sentences embeddings.
- Sort by domain dissimilarity	~600K	$(p, s_1, s_2, m)$	$X_2$ : cosine similarity scores of domain embeddings i.e., use categories of each article to get embeddings.
Step 6: Select data for expert annotation	19,500	$(p, d_1, d_2)$	Remove proper nouns and phrases with missing information and select top 19,500 examples for annotation.
<b>Section 4.3.2 Data Annotations</b>			
	30,042	$(p, d, q)$	Create a query i.e., paraphrase from the given phrase in each context document.
	15,021	$(p, d_1, d_2, l)$	Create a Yes/No label for each pair of documents.
<b>Section 4.3.3 Verifying Annotations</b>			
Round 1: MTurk verifier	22,496	$(p, d, q)$	Verify queries and Yes/No label by MTurkers.
	10,043	$(p, d_1, d_2, l)$	
Round 2: Upwork verifiers	28,325	$(p, d, q)$	Verify instances rejected in Round 1.
	13,413	$(p, d_1, d_2, l)$	

### 6.3.6 Quantitative results on PR-page

Table 6.15: **Ranking** accuracy (%) on **PR-page** using the state-of-the-art pretrained phrase embeddings (a) and those finetuned on PR-pass via SS-style training (b).

Model	Phrase				Phrase + Context			
	Top-1	Top-3	Top-5	MRR@5	Top-1	Top-3	Top-5	MRR@5
(a) Pre-trained embeddings								
BERT [30]	20.70	34.30	41.00	28.20	<b>35.40 (+14.70)</b>	<b>52.10</b>	<b>59.10</b>	<b>44.50</b>
USE-v5 [21]	<b>32.20</b>	<b>52.70</b>	<b>60.80</b>	<b>43.20</b>	n/a	n/a	n/a	n/a
(b) PR-pass-trained SS models' phrase embeddings								
PhraseBERT [157]	<b>49.40</b>	<b>69.40</b>	<b>76.70</b>	<b>60.10</b>	14.70	21.60	26.10	18.70
SimCSE [45]	44.20	66.60	73.50	55.70	24.60	37.80	43.20	31.70

### 6.3.7 Finetuning on PSD does not substantially improve accuracy

As PSD has only 4,858 examples, we use all examples for testing in Section 4.5.5 and find the best PR-trained SS models to perform poorly. To further understand the challenge of PSD, here, we ask:

**Q:** *How much does training on PR-pass and finetuning on PSD improve accuracy on PSD?*

**Experiment** We take the PR-pass-trained SS models and further finetune them on a subset of PSD to measure how training directly on PSD improves SS models. We split PSD into 1,438/500/3,000 examples for train/dev/test sets, respectively, and finetune the PR-pass-trained SS models on this PSD train set. For comparison with the results in Section 4.5.4, we use the same set of hyperparameters as when finetuning on PR-pass in Section 4.5.4. Below, we report the test-set results of the lowest dev-loss models.

**Results** On the PSD-3K test set, all models perform poorly at a mean EM score of 55.14% (Table 6.16a; mean). Interestingly, finetuning the original models using the 1,938 examples (hereafter, PSD-2K) instead of PR-pass decreases accuracy, on average by **-6.51** points. An explanation is that 1,438 PSD training examples are too few for the finetuning to be effective.



Indeed, finetuning the PR-pass-trained SS models further on PSD-2K increases the scores for all models by **+9.10** on average (Table 6.16c; mean). The best model is Longformer<sub>Base</sub> [8] (Table 6.16; 71.10 EM), which is still substantially lower than the human upperbound of 95%.

Table 6.16: Performance of **SS models** on 3,000 PSD **test** examples. (a) and (b) models are **finetuned** only on PR-pass and 1,938 PSD examples (PSD-2K), respectively. (c) models are finetuned on PR-pass first and then finetuned on PSD-2K. All models are “base” unless otherwise specified. The definitions of EM+loc and F<sub>1</sub>+loc are in Table 4.6’s caption.

Models finetuned on	(a) PR-pass		(b) PSD-2K		(c) PR-pass + PSD-2K	
	EM+loc	F <sub>1</sub> +loc	EM+loc	F <sub>1</sub> +loc	EM+loc	F <sub>1</sub> +loc
PhraseBERT [157]	51.00	51.15	35.43 (-15.57)	36.02	56.53 (+5.53)	56.81
BERT [30]	54.53	54.75	44.33 (-10.20)	45.28	63.83 (+9.30)	64.14
BERT <sub>Large</sub> [30]	54.77	54.99	54.07 (-0.70)	54.82	67.13 (+12.36)	67.36
SpanBERT [72]	52.27	52.37	44.67 (-7.60)	45.35	69.93 (+17.66)	70.14
SentenceBERT [131]	52.27	52.41	38.63 (-13.64)	39.31	58.93 (+6.66)	59.21
SimCSE [45]	53.47	53.59	43.67 (-9.80)	44.38	60.60 (+7.13)	60.80
Longformer [8]	<b>62.47</b>	<b>62.58</b>	61.97 (-0.50)	62.69	<b>71.10</b> (+8.63)	<b>71.30</b>
Longformer <sub>Large</sub> [8]	60.33	60.42	<b>66.27</b> (+5.94)	<b>67.10</b>	65.87 (+5.54)	66.10
mean	55.14	55.28	48.63 (-6.51)	49.37	64.24 (+9.10)	64.48
± std	4.10	4.08	11.03	11.08	5.23	4.13

### 6.3.8 SS-style training improves *non-contextualized* but not *contextualized* phrase embeddings

As the SS models trained on PR-pass and PR-page perform impressively (Section 4.5.4), almost  $1.5\times$  better than the ranking models that are based on pre-trained embeddings, an interesting question is:

**Q:** *Does SS training also improve contextualized phrase embeddings?*

This is important to understand because the impressive SS-models’ performance gain may come from the extra linear-classification layer (not necessarily from the finetuned embeddings).

**Experiment** We extract the phrase embeddings (both non-contextualized and contextualized) from the PR-pass-trained SS models from Section 4.5.4 (i.e. discarding the classification layer) and test them in the PR-pass ranking experiments (as in Section 4.5.3).

**Results** After finetuning on PR-pass, the *non-contextualized* phrase embeddings improve substantially for most models at an average gain of **+16.61** in top-1 accuracy (e.g., PhraseBERT top-1 accuracy increases from 36.62% to 59.02%; Table 6.17b). This result shows that training on PR-pass improves non-contextualized phrase embeddings. In stark contrast, the ranking scores of *contextualized* phrase embeddings drop significantly, **-11.95** points on average (Table 6.17c), compared to before finetuning on PR-pass.

In sum, we are observing a consistent trend that the contextualized phrase embeddings of the original pre-trained BERT (both “base” and “large”) are remarkably beneficial for retrieval (i.e. PR). However, after finetuning, e.g. on PR-pass or using other techniques (e.g. in PhraseBERT or SentenceBERT), such benefits of leveraging context disappear. Aligned with [175], we find that incorporating context effectively into phrase embeddings is an open research challenge.

Table 6.17: **Ranking** accuracy (%) on **PR-pass** using the state-of-the-art pretrained phrase embeddings (a) and those finetuned on PR-pass via SS-style training (b). See Section 6.3.6 for the results on PR-page.  $\Delta$  (e.g. **-3.62**) denotes the differences between the Top-1 accuracy in the contextualized (“Phrase + Context”) vs. the non-contextualized (“Phrase”) setting.

Model	Phrase				Phrase + Context			
	Top-1	Top-3	Top-5	MRR@5	Top-1 ( $\Delta$ )	Top-3	Top-5	MRR@5
(a) Pre-trained embeddings								
PhraseBERT [157]	36.62	66.96	75.90	52.20	33.00 ( <b>-3.62</b> )	49.60	56.70	41.90
BERT [30]	29.80	47.90	55.40	39.50	<b>47.44 (+17.64)</b>	<b>65.78</b>	<b>73.30</b>	<b>57.30</b>
BERT <sub>Large</sub> [30]	23.76	38.52	45.40	31.70	<b>42.80 (+19.04)</b>	<b>58.90</b>	<b>64.90</b>	<b>51.30</b>
SpanBERT [72]	20.88	31.04	35.20	26.40	14.40 ( <b>-6.48</b> )	30.46	39.80	23.40
SentenceBERT [131]	22.30	50.64	60.60	36.80	<b>25.14 (+2.84)</b>	39.52	46.20	32.90
SimCSE [45]	28.10	53.70	64.60	41.60	<b>32.40 (+4.30)</b>	53.44	62.80	<b>43.70</b>
USE-v5 [21]	<b>43.36</b>	<b>70.12</b>	<b>78.90</b>	<b>57.30</b>	n/a	n/a	n/a	n/a
DensePhrase [86]	32.24	51.30	60.50	42.60	31.50 ( <b>-0.74</b> )	46.30	53.80	39.70
(b) PR-pass-trained SS models’ phrase embeddings								
PhraseBERT [157]	<b>59.02</b>	<b>81.58</b>	<b>87.90</b>	<b>70.60</b>	24.98 ( <b>-34.04</b> )	37.78	43.90	32.00
BERT [30]	50.10	66.16	71.40	58.60	20.34 ( <b>-29.76</b> )	31.40	37.10	26.50
BERT <sub>Large</sub> [30]	32.70	42.40	45.90	37.80	11.40 ( <b>-21.30</b> )	17.00	20.50	14.60
SpanBERT [72]	15.22	22.88	26.60	19.40	8.92 ( <b>-6.30</b> )	13.56	16.60	11.60
SentenceBERT [131]	53.14	74.86	80.70	64.20	20.12 ( <b>-33.02</b> )	30.04	34.90	25.60
SimCSE [45]	50.96	76.70	83.40	64.00	37.70 ( <b>-13.26</b> )	52.38	58.90	45.60
(c) Differences between after vs. before finetuning, i.e. the 6 models in (b) vs. those in (a)								
mean differences	<b>+16.61</b>				<b>-11.95</b>			

### 6.3.9 Qualitative examples for PS, PR-pass, PR-page and PSD

<b>PS</b> example. Groundtruth: positive	
P <sub>1</sub>	moderate speed
P <sub>2</sub>	steady pace
S <sub>1</sub>	Deforestation due to logging and land conversion has likely caused the population to decline at a moderate speed.
S <sub>2</sub>	Deforestation due to logging and land conversion has likely caused the population to decline at a steady pace.

Figure 6.25: PhraseBERT-based classifier **correctly** predicts positive given two phrases P<sub>1</sub> and P<sub>2</sub> with and without the presence of context S<sub>1</sub> and S<sub>2</sub>. Here, to humans, the phrases are non-polysemous and have the same meaning.

<b>PS</b> example. Groundtruth: negative	
P <sub>1</sub>	greatest emphasis
P <sub>2</sub>	highest stress
S <sub>1</sub>	However, the rock art had the greatest emphasis on domesticated cattle.
S <sub>2</sub>	However, the rock art had the highest stress on domesticated cattle.

Figure 6.26: PhraseBERT-based classifier **correctly** predicts negative given two phrases P<sub>1</sub> and P<sub>2</sub> with and without the presence of context S<sub>1</sub> and S<sub>2</sub>. Here, to humans, the two phrases are non-ambiguously carrying different meanings.

<b>PS</b> example. Groundtruth: positive	
P <sub>1</sub>	unique image
P <sub>2</sub>	uncommon style
S <sub>1</sub>	Bayliss has been praised for her unique image and tendency to change up songs.
S <sub>2</sub>	Bayliss has been praised for her uncommon style and tendency to change up songs.

Figure 6.27: PS case that requires context to determine similarity. Without context, a PhraseBERT-based classifier incorrectly thinks P<sub>1</sub> and P<sub>2</sub> are different. Yet, it changes the prediction to positive, i.e. thinking two phrases have the same meaning, when the context is taken into account.

<b>PS</b> example. Groundtruth: <i>negative</i>	
P <sub>1</sub>	permanent post
P <sub>2</sub>	stable location
S <sub>1</sub>	His assistant, John Carver took over as caretaker manager, managing one win, but was not considered for the permanent post, and left in September 2004.
S <sub>2</sub>	His assistant, John Carver took over as caretaker manager, managing one win, but was not considered for the stable location, and left in September 2004.

Figure 6.28: PS case that requires context to determine similarity. Without context, PhraseBERT-based classifier incorrectly thinks P<sub>1</sub> and P<sub>2</sub> carry the same meaning. Yet, it correctly changes the prediction to *negative* when the context is taken into account.

PSD example.	
<i>d</i>	<p>Bubble memory is a type of non-volatile computer memory that uses a thin film of a magnetic material to hold small magnetized areas, known as "bubbles" or "domains", each storing one bit of data. The material is arranged to form a series of parallel tracks that the bubbles can move along under the action of an external magnetic field. The bubbles are read by moving them to the edge of the material where they can be read by a conventional magnetic pickup, and then rewritten on the far edge to keep the memory cycling through the material. In operation, bubble memories are similar to delay line memory systems. Bubble memory started out as a promising technology in the 1970s, offering memory density of an order similar to hard drives but performance more comparable to core memory while lacking any moving parts. This led many to consider it a contender for a "universal memory" that could be used for all <b>storage needs</b>. The introduction of dramatically faster semiconductor memory chips pushed bubble into the slow end of the scale, and equally dramatic improvements in hard drive capacity made it uncompetitive in price terms. Bubble memory was used for some time in the 1970s and 80s where its non-moving nature was desirable for maintenance or shock-proofing reasons. The introduction of Flash RAM and similar technologies rendered even this niche uncompetitive, and bubble disappeared entirely by the late 1980s. History. Precursors.</p> <p>The Inkerman stone, of which the building is made, was mined near Sevastopol and transported by barges. No convenient mooring facilities existed at that time, so the barges had to anchor in the harbor and the load was moved to the shore by boats and then transported to the construction site across the steppe. During the first year of construction, the builders concentrated on the basic structure at the expense of various facilities and decorations. At the end of 1816, the lighthouse looked like a conic 36-metre-high stone tower with a wooden 3.3-metre-high decagonal lantern. The lighthouse became operational in 1817 after its lighting system had been repaired. Three houses were built next to the tower to accommodate the lighthouse personnel and for <b>storage needs</b>. However, cold and humid winters of the Tarkhanut Peninsula, however, made these houses nearly unsuitable for living. In 1862, the lighting system was upgraded, and the spread of light reached 12.4 miles. In 1873, the construction resumed along with cleaning efforts of the surrounding areas. The building was finished and painted white. In 1876, an additional telegraph spot was built near the tower.</p>
<i>q</i> <sub>1</sub>	storehouse purposes    Groundtruth: <b>storage needs</b> & Prediction: <b>storage needs</b> (confidence: 0.99)
<i>q</i> <sub>2</sub>	data caching            Groundtruth: <b>storage needs</b> & Prediction: <b>storage needs</b> (confidence: 0.99)

Figure 6.29: Given document *d*, our Longformer<sub>Large</sub> SS model trained on PR-pass correctly retrieves **storage needs** in the second paragraph for the query *q*<sub>1</sub> “storehouse purposes” but *fails* to retrieve the answer when the query *q*<sub>2</sub> is “data caching”. The predicted answer for *q*<sub>2</sub> should be **storage needs** (i.e. in the first passage) since this phrase relates to caching data digitally in computers while **storage needs** refers to physically storing objects.

PSD example.	
<i>d</i>	<p>In the libretto, Delilah is portrayed as a seductive "femme fatale", but the music played during her parts invokes sympathy for her. The 1949 biblical drama "Samson and Delilah", directed by Cecil B. DeMille and starring Victor Mature and Hedy Lamarr in the titular roles, was widely praised by critics for its cinematography, lead performances, costumes, sets, and innovative special effects. It became the highest-grossing film of 1950, and was nominated for five Academy Awards, winning two. According to "Variety", the film portrays Samson as a stereotypical "handsome but dumb hulk of muscle". Samson has been especially honored in Russian artwork because the Russians defeated the Swedes in the Battle of Poltava on the feast day of St. Sampson, whose name is homophonous with Samson's. The lion slain by Samson was interpreted to represent Sweden, as a result of the lion's placement on the Swedish coat of arms. In 1735, C. B. Rastrelli's bronze statue of Samson slaying the lion was placed in the center of the great cascade of the fountain at Peterhof Palace in Saint Petersburg. Samson is the emblem of Lungau, Salzburg and parades in his honor are held annually in ten villages of the Lungau and two villages in the north-west Styria (Austria). During the parade, a young bachelor from the community carries a massive figure made of wood or aluminum said to represent Samson. The tradition, which was first documented in 1635, was entered into the UNESCO list of Intangible Cultural Heritage in Austria in 2010. Samson is one of the giant figures at the "Ducasse" festivities, which take place at Ath, Belgium.</p> <p>On September 22, 2015, Honda announced that they had sold over 1 million Activas in five months in the Indian market, from April to August. Honda launched their 5th generation of Honda Activa in 2018, and the sixth-generation Honda Activa 6G have been launched in India with prices starting at 63,912 (ex-showroom, Delhi). Milestones. In April, 2014, "The Economic Times" reported the Honda Activa to be the best selling two wheeler in India, outselling the Hero Splendor. During the month of September 2013, 141,996 Honda Activa scooters were sold, nearly equal to Honda's entire annual sales in North America. The 110cc Activa is the company's biggest seller, by far. It is responsible for over 2,00,000 sales units each month. In November 2018, HMSI crossed the 2.5 crore sales mark in the scooter segment. It has become the first company to reach this milestone and the biggest contributor to this massive figure is the Honda Activa. It took Honda 13 years to achieve the one crore sales figure, but it managed to add another crore in the span of just three years. It then went on to achieve the next 50 lakh in just one year.</p>
$q_1$	huge model      Groundtruth: massive figure & Prediction: massive figure (confidence: 0.99)
$q_2$	giant number      Groundtruth: massive figure & Prediction: massive figure (confidence: 0.99)

Figure 6.30: Given document *d*, Longformer<sub>Large</sub> model trained with SS approach on PR-pass correctly retrieves massive figure in the second paragraph for the query  $q_2$  "giant number" but fails to retrieve the answer when the query  $q_1$  is "huge model". The predicted answer for  $q_1$  should be massive figure in the first passage since this phrase relates to a physical shape instead of a number.

PSD example.	
<i>d</i>	<p>Eva held ambitions to replace Hortensio Quijano for the 1951 election, although her poor health kept her from this. Nonetheless many were concerned that her agenda would be pushed through. In march of 1951 the government arrested several retired army officers due to their dissent and disapproval of Perón’s administration. This raised tensions among the rest of the army, although action did not occur. By September tensions had risen among the military due to the <b>unrivalled power</b> of the Peronist regime. On September 28, 1951, during the election, Menéndez led the military uprising in an attempt to overthrow the government. He led a core of officers, commanding a division, and left Campo de Mayo bound for the Casa Rosada. Resolve for the uprising, especially among the non-commissioned officers and enlisted men, was not strong enough. They were not prepared to fight their own countrymen. The uprising was over as soon as opposition was encountered, almost completely bloodless. Perón admired the loyalty of the troops and pardoned all those involved.</p> <p>The design uses a similar standard to the JVX in terms of distortion reduction with crossbraces and 27 cells but that’s where the similarity ends. Petra was built from the ground up with entirely new panel shaping and trim. Petra has a highly elliptical planform and very high sweep. NZ Aerosports say she has a high roll rate, a long recovery arc and high maximal glide ratio. She is said to deliver <b>unrivalled power</b> in the turn, plane out and flare. Petra has a long list of World Records, National and International titles to back that up. She had an impressive debut at the PD Big Boy Pants event in July 2011, with Nick Batsch setting a new distance world record of 222.45m (729ft). One month later Nick took out the Pink Open in Klatovy and the FAI World Cup also; first in distance, speed and overall. He also won the 2011 US CP nationals on Petra. Patrick Boulongne came 2nd in the European Championships and 6th overall at the World Cup with Petra in his first competition with her. He went on to win the 2011 French Canopy Piloting Nationals.</p>
<i>q</i> <sub>1</sub>	incomparable energy    Groundtruth: <b>unrivalled power</b> & Prediction: <b>unrivalled power</b> (confidence: 0.99)
<i>q</i> <sub>2</sub>	indomitable strength    Groundtruth: <b>unrivalled power</b> & Prediction: <b>unrivalled power</b> (confidence: 0.99)

Figure 6.31: Given document *d*, Longformer<sub>Large</sub> model trained via the SS approach on PR-pass correctly retrieves **unrivalled power** in the first paragraph for the query *q*<sub>2</sub> “indomitable strength” but *fails* to retrieve the answer when the query *q*<sub>1</sub> is “incomparable energy”. The predicted answer for *q*<sub>1</sub> should be **unrivalled power** in the second passage since the second passage changes “unrivalled power” meaning to a competition strength instead of military power.



<b>PR-pass</b> example.	Groundtruth: <b>common thought</b>
<i>d</i>	As the medical corps grew in size there was also specialization evolving. Physicians surfaced that specialized in disease, surgery, wound dressing and even veterinary medicine. Veterinary physicians were there to tend to livestock for agricultural purposes as well as combat purposes. The Cavalry was known for their use of horses in combat and scouting purposes. Because of the type of injuries that would have been commonly seen, surgery was a somewhat common occurrence. Tools such as scissors, knives and arrow extractors have been found in remains. In fact, Roman surgery was quite intuitive, in contrast to <b>common thought</b> of ancient surgery. The Roman military surgeons used a cocktail of plants, which created a sedative similar to modern anesthesia. Written documentation also showed surgeons would use oxidation from a metal such as copper and scrape it into wounds, which provided an antibacterial effect; however, this method was most likely more toxic than providing an actual benefit. Doctors had the knowledge to clean their surgical instruments with hot water after each use. Wounds were dressed, and dead tissue was removed when bandages were changed.
<i>q</i>	prevalent theory
R	0.882 <b>common thought</b> 0.855 common thought of 0.702 fact 0.698 to common thought 0.675 common occurrence

Figure 6.32: A **ranking** model based on the phrase embeddings of the PR-pass-trained PhraseBERT SS model correctly ranks and retrieves the most semantically relevant answer “common thought” as the top-1 prediction in the retrieval list R for the query “prevalent theory” in a PR-pass example (which contains a document *d* and a query *q*).

PR-page example.	Groundtruth: <b>continued risk</b>
<p><i>d</i> ... Following a United Nations agreement between Indonesia and Portugal, a UN-supervised referendum held on 30 August 1999 offered a choice between autonomy within Indonesia and full independence. The people of East Timor voted overwhelmingly for independence. An Australian-led and Indonesian-sanctioned peacekeeping force, INTERFET, was sent into the territory to restore order following a violent 'scorched-earth' policy carried out by pro-integration militia and supported by elements of the Indonesian military. In response to Australia's involvement, Indonesia abrogated the 1995 security pact, asserting that Australia's actions in East Timor were inconsistent with 'both the letter and spirit of the agreement'. Official meetings were cancelled or delayed, including the Indonesia-Australia Ministerial Dialogue, which would not reconvene until March 2003. INTERFET was later replaced by a UN force of international police, UNTAET, which formed a detachment to investigate alleged atrocities. "Tampa" affair and the War on Terror. The relationship came under strain in August 2001 during the "Tampa" affair, when Australia refused permission for the Norwegian freighter ship MV "Tampa" to enter Australian waters while carrying Afghan asylum seekers that it had rescued from a distressed fishing vessel in international waters. The Indonesian Search and Rescue Agency did not immediately respond to requests from Australia to receive the vessel. When the ship entered Australian territorial waters after being refused permission, Australia attempted without success to persuade Indonesia to accept the asylum seekers. Norway also refused to accept the asylum seekers and reported Australia to international maritime authorities. The incident prompted closer coordination between Indonesian and Australian authorities, including regional conferences on people smuggling, trafficking in persons and other transnational crime. In 2002, a terrorist attack in Kuta, Bali killed 202 people, including 88 Australians, and injured a further 240. Jemaah Islamiyah, a violent Islamist group, claimed responsibility for the attack, allegedly in retaliation for Australia's support for East Timorese independence and the War on Terror. A subsequent attack in 2005 resulted in the deaths of a further 20 people, including 15 Indonesians and 4 Australians. The 2003 Marriott Hotel bombing was also perceived as targeted at Western interests in Indonesia; Al Qaeda claimed the attack was carried out by a Jemaah Islamiyah suicide bomber in response to actions of the United States and its allies, including Australia. A 2004 attack on the Australian embassy in Jakarta by Jemaah Islamiyah resulted in the deaths of nine Indonesians. The following year, Indonesian diplomatic and consular premises in Australia received a number of hoax and threat messages. Since then, both the United States and Australian governments have issued warnings against travel to Indonesia, advising their citizens of a <b>continued risk</b> of attacks. These incidents prompted greater cooperation between law enforcement agencies in the two countries, building on a 1999 agreement on drug trafficking and money laundering. The Australian Federal Police's Jakarta Regional Cooperation Team provided assistance to the Indonesian National Police, and has contributed to the Jakarta Centre for Law Enforcement Cooperation. This relationship has attracted criticism, particularly following the arrest and sentencing of the Bali Nine, a group of nine Australians arrested in Denpasar while attempting to smuggle heroin from Indonesia to Australia. The 2005 conviction of Schapelle Corby for attempting to smuggle drugs to Bali also attracted significant attention in the Australian media. The 2004 Indian Ocean earthquake prompted a significant humanitarian response from Australia, including a \$1 billion aid package from the federal government, a further \$17.45 million contribution from state and territory governments, and the commitment of 900 Australian Defence Force personnel to relief efforts in northern Sumatra and Aceh. A telethon broadcast on Australia's three major commercial television networks called "" generated pledges of more than \$10 million, contributing to total private aid of \$140 million. The Eighth "Australia-Indonesia Ministerial Forum" (AIMF) was held in Bali on 29 June 2006 and was attended by five Australian and eleven Indonesian ministers. A key outcome was support for the conclusion of a security agreement, later realised as the Lombok Agreement, providing a framework for the development of the security relationship by the end of 2006 on defence, law enforcement, counter-terrorism, intelligence, maritime security, aviation safety, WMD non-proliferation, and bilateral nuclear cooperation for peaceful purposes. Australia-Indonesia-East Timor Trilateral Ministerial Meetings occurred three times to September 2006. Recent relations. 2010 President Susilo Bambang Yudhoyono visited Australia in April 2010, and became the second Indonesian leader to address federal parliament: Finally, I look forward to a day in the near future. The day when policy makers, academicians, journalists and other opinion leaders all over the world take a good look at the things we are doing so well together. And they will say: these two used to be worlds apart. But they now have a fair dinkum of a partnership. ...</p>	
<i>q</i> sustained threat	
<p>R 0.830 threat . 0.802 potential threat 0.800 threat reached 0.787 threat as 0.787 threat to</p>	

Figure 6.33: A **ranking** model based on the non-contextualized embeddings of USE-v5 fails to retrieve the correct answer “continued risk” for the query “sustained threat” in the PR-page example (which contains a document *d* and a query *q*). The top-5 phrases retrieved (R) contains the word “threat” but have no identifier conveying the “continued” or ‘sustained’ sense. Here, the Wikipedia page is truncated to fit into a single manuscript page.

### 6.3.10 Verifying annotations

There are two common methods for evaluation of dataset quality: (1) Verify only a small, random subset [122] to estimate the quality of the full dataset or (2) verifying the entire dataset with multiple annotators and use the inter-annotator agreement (IAA) to control quality [15, 82]. The first approach for approximation is budget-friendly but it remains unknown whether the rest of examples are at high quality, while IAA is more desired but annotating thousands of instances can be prohibitively slow and costly.

We propose a **hybrid approach** to evaluate (leveraging both linguistic experts and non-experts) and ensure high quality for 30,042 queries and 15,021 Yes/No answers at lower cost compared to IAA via two rounds:

1. First, we ask around 1,000 highly qualified freelancers on Amazon Mechanical Turk (MTurk verifiers) to verify whether the *query* annotated by our Upwork annotators is interchangeable i.e. has the same meaning with the given *phrase* in *paragraph*. To verify Yes/No answers, MTurk verifiers need to read two short paragraphs containing the same phrase like Upwork annotators to make decisions. We do not show answers to the MTurk verifiers to avoid biases.
2. Second, we continue hiring 5 Upwork verifiers who are writing experts to double-check those instances rejected by MTurk verifiers from the previous round and only discard an example if the Upwork verifiers agree with MTurk verifiers.

#### Round 1: Verification by MTurk non-experts

We use AMT platform to recruit more than 1,000 MTurk verifiers. Also, we use Gorilla (gorilla.sc) to develop user interface to collect answers from participants because (1) Gorilla provides easy-to-use tools to build graphical interface, (2) it is straightforward to monitor and discard results from unqualified participants and (3) we can easily share the experiment with MTurk verifiers via a link. Per 30 verified answers in around ~20 minutes, the verification process costs us \$5.6 (AMT fees included) and 1 token to Gorilla to a single MTurk verifier.

Participants are given detailed instructions along with 5 practice samples to get familiar with the task (Fig. 6.34). They need to pass an evaluation checkpoint including 6 questions randomly sampled from our verified question bank in order to start working with sets of 30 questions. With this approach, all examples in the dataset are verified once and as a result, 22,496/30,042 queries (~74.88%) and 10,043/15,021 Yes/No answers (~66.86%) accepted by MTurkers are considered high quality since they are annotated by a writing expert and confirmed by a qualified English native speaker. The remaining 7,546 queries and 4,978 Yes/No answers rejected that are passed to another group of 5 writing experts for confirmation.

## **Round 2: Verification by Upwork experts**

We hired another set of 5 writing experts from Upwork (Upwork verifiers) with an hourly rate of \$25-40/hour to verify 12,524 examples rejected by MTurk verifiers, i.e., at an average cost of approximately \$0.26 per example. See a sample assignment given to an Upwork expert in [121].

We rely on IAA to decide whether to accept or reject an example. Specifically, we use the *same question types* as shown to MTurk verifiers in the previous step and see whether these Upwork verifiers agree with the Upwork annotators to keep this example or with MTurk verifiers to reject it. We find that the agreement between the first- and third-round annotators are 5,829 (out of 7,546) paraphrases and 3,370 (out of 4,978) Yes/No answers in total and thus the total high-quality queries and Yes/No answers we achieve are 28,325 and 13,413, respectively.

### Instructions

In this study, you will work with 2 types of Yes/No questions:

**Type 1: Same paragraphs, different phrases**

**P1:** 2020. The cancellation of the conference in 2020 due to the COVID-19 pandemic led to an online series of lectures entitled Skeptical Inquirer Presents. **These sessions included presentations by well known figures in the skeptical community and opportunities for viewers to ask questions.** Conference details.

**P2:** 2020. The cancellation of the conference in 2020 due to the COVID-19 pandemic led to an online series of lectures entitled Skeptical Inquirer Presents. **These sessions included presentations by Omicron variant in the skeptical community and opportunities for viewers to ask questions.** Conference details.

**Question:** In both passages, are **well known figures** and **Omicron variant** interchangeable (i.e., having the same meaning)?

**Type 2: Different paragraphs, same phrases**

**P1:** In 1949, a memorial for the 442 Regimental Combat Unit was incorporated and remembered for the Japanese-American soldiers who had fallen during World War II. Every year during the Obon festival, families gather to upkeep their relatives tombstones and to visit the spirits. Buddy Moon, nurse and philanthropist, was one of the **well known figures** to be buried at the cemetery in 1881. There is a section called the "Showmen's Rest" in which 400 carnival workers and circus performers are buried by a memorial that is decorated with a lion.

**P2:** 2020. The cancellation of the conference in 2020 due to the COVID-19 pandemic led to an online series of lectures entitled Skeptical Inquirer Presents. These sessions included presentations by **well known figures** in the skeptical community and opportunities for viewers to ask questions. Conference details.

**Question:** In both passages, does **well known figures** have the same meaning?

(a) Detailed instructions given to MTurkers

Your progress: 0/36 trials completed

**P1:** HotDocs transforms documents and graphical (PDF) forms into document-generation templates and deploys of these templates to various server environments. Document modeling in HotDocs can range from variable insertions to the formation and insertions of complex, computed variables. **Business logic consisting of IF/THEN statements and REPEAT loops can be built into the template to control the inclusion or exclusion of language blocks.** HotDocs includes a variety of other scripting instructions and sets of pre-packaged functions using boolean logic.

**P2:** HotDocs transforms documents and graphical (PDF) forms into document-generation templates and deploys of these templates to various server environments. Document modeling in HotDocs can range from variable insertions to the formation and insertions of complex, computed variables. **Business logic consisting of IF/THEN statements and REPEAT loops can be built into the template to control the inclusion or exclusion of linguistic sections.** HotDocs includes a variety of other scripting instructions and sets of pre-packaged functions using boolean logic.

**Q:** In both passages, are **language blocks** and **linguistic sections** interchangeable (i.e., having the same meaning)?

Note: Yes/No buttons will be displayed in 20 seconds. Please read the given contents carefully before answering question.

(b) Upon completion of training stage, MTurkers need to correctly answer the first 5 out of 6 questions to be invited to verify annotations from Upwork experts.

### Training

**P1:** This library was supposedly founded in 1945, but has started work in current object in 1947. During 1953-1956 it has played the role of the national library since the Kosovo National Library was closed. **Academy of Sciences and Art is a necessary institution for the education system that is placed in Pristina.** This institution was founded in 1975 as the Association of Science and Arts of Kosovo.

**P2:** This library was supposedly founded in 1945, but has started work in current object in 1947. During 1953-1956 it has played the role of the national library since the Kosovo National Library was closed. **Academy of Sciences and Art is a big supermarket for the education system that is placed in Pristina.** This institution was founded in 1975 as the Association of Science and Arts of Kosovo.

**Question 3:** In both passages, are **necessary institution** and **big supermarket** interchangeable (i.e., having the same meaning)?

(c) Verification of paraphrases via type-1 question.

### Training

**P1:** Mercury is poured over the dirt with bare hands. The method leaves much gold undetected, and therefore some miners are using metal detectors. The mercury pollution in the area is both an **environment problem**, and a health hazard. Most of the gold gets transported to the North of Paramaribo where the gold buyers are located.

**P2:** Tools such as KLEE, Cloud9, and Otter take this approach by implementing models for file system operations, sockets, IPC, etc. Forking the entire system state. Symbolic execution tools based on virtual machines solve the **environment problem** by forking the entire VM state. For example, in S2E each state is an independent VM snapshot that can be executed separately.

**Question 1:** In both passages, does **environment problem** have the same meaning?

(d) Verification of Yes/No labels via type-2 question.

### Training

**P1:** This library was supposedly founded in 1945, but has started work in current object in 1947. During 1953-1956 it has played the role of the national library since the Kosovo National Library was closed. **Academy of Sciences and Art is a necessary institution for the education system that is placed in Pristina.** This institution was founded in 1975 as the Association of Science and Arts of Kosovo.

**P2:** This library was supposedly founded in 1945, but has started work in current object in 1947. During 1953-1956 it has played the role of the national library since the Kosovo National Library was closed. **Academy of Sciences and Art is a big supermarket for the education system that is placed in Pristina.** This institution was founded in 1975 as the Association of Science and Arts of Kosovo.

**Question 3:** In both passages, are **necessary institution** and **big supermarket** interchangeable (i.e., having the same meaning)?

**NO.** The phrase **necessary institution** means a required establishment of the livery stable which does not totally refer to a **big supermarket.**

Yes

(e) Feedback is given when MTurkers give a wrong answer.

### Training

**P1:** Mercury is poured over the dirt with bare hands. The method leaves much gold undetected, and therefore some miners are using metal detectors. The mercury pollution in the area is both an **environment problem**, and a health hazard. Most of the gold gets transported to the North of Paramaribo where the gold buyers are located.

**P2:** Tools such as KLEE, Cloud9, and Otter take this approach by implementing models for file system operations, sockets, IPC, etc. Forking the entire system state. Symbolic execution tools based on virtual machines solve the **environment problem** by forking the entire VM state. For example, in S2E each state is an independent VM snapshot that can be executed separately.

**Question 1:** In both passages, does **environment problem** have the same meaning?

**NO.** The phrase **environment problem** in the first passage mentions an issue of the **physical** environment in which we are living while in the second passage, it means the issue of the **digital** environment of an operating system. Thus, it should have different meanings.

No

(f) or even a right answer.

Figure 6.34: Gorilla layouts shown to MTurkers to verify annotations in the first round.

### 6.3.11 Data Sheet

We follow the documentation template provided by Gebru et al. 2021 [47].

#### **Motivation**

**For what purpose was the dataset created?** Understanding phrases in context plays a vital role in solving many Natural Language Understanding (NLU) tasks such as question answering or reading comprehension. While there are *word-sense* disambiguation datasets like WiC, no such benchmarks exist for *phrases*. Existing phrase benchmarks compare only phrases without context and some of them contain numerous phrase pairs that have lexical overlap. The major drawback is no human annotation of how a phrase’s meaning changes w.r.t the context. This motivates us to construct a Phrase-in-Context benchmark to drive the development of contextualized phrase embeddings in NLU.

**Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?** Auburn University and Adobe Research.

#### **Composition/collection process/preprocessing/cleaning/labeling and uses**

We describe the data construction process, annotation and verification methods in our paper (See Sec. 2.3.3 and Sec. 4.4).

#### **Distribution**

**Will the dataset be distributed to third parties outside the entity (e.g., company, institution, organization) on behalf of which the dataset was created?** We release three datasets PS, PR (including PR-pass and PR-page) and PSD to the public.

**How will the dataset be distributed (e.g., tarball on website, API, GitHub)?** The datasets are released and can be viewed and downloaded on HuggingFace <https://huggingface.co/PiC> or on our website <https://phrase-in-context.github.io>.

**When will the dataset be distributed?** It has been released in July 2022.

**What is the dataset format and how it can be read?** We use JSON - a widely used data format for PiC dataset and follow a scheme of HuggingFace datasets to host it. Three datasets PS, PR and PSD in the PiC dataset are loaded as follows:

```
from datasets import load_dataset

ps      = load_dataset("PiC/phrase_similarity")
pr_pass = load_dataset("PiC/phrase_retrieval", "PR-pass")
pr_page = load_dataset("PiC/phrase_retrieval", "PR-page")
psd     = load_dataset("PiC/phrase_sense_disambiguation")
```

**Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?** Our dataset is distributed under the CC-BY-NC 4.0 license.

## Maintenance

**How can the owner/curator/manager of the dataset be contacted (e.g., email address)?** Thang Pham ([thangpham@auburn.edu](mailto:thangpham@auburn.edu)) and Anh Nguyen ([anh.ng8@gmail.com](mailto:anh.ng8@gmail.com)) will be responsible for maintenance.

**Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?** Yes. If we include more tasks or find any errors, we will correct the dataset. It will be updated on our website and also HuggingFace.

**If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?** They can contact us via email for the contribution.

### 6.3.12 Evaluation with ChatGPT-4

ChatGPT is an artificial intelligence (AI) chatbot developed by OpenAI and released in November 2022. It is built on top of OpenAI's Generative Pre-trained Transformer 4 (i.e. GPT-4), a member of GPT family in large language models (LLMs). Since GPT-4 achieves impressive results and on many real-world tasks, we benchmark this model to see how it performs on our proposed tasks.

#### Phrase Similarity (PS)

Since PS is the binary classification task, we prompt ChatGPT-4 in different ways to get Yes/No answer as its prediction. Prompts and their accuracy are reported in Table 6.18.

#### Phrase Sense Disambiguation (PSD)

For PSD, since it is hard to prompt ChatGPT-4 for the span and its location, we turn the task to be binary classification for the first prompt (Table 6.19) and ask the model to output both span prediction and the passage number (i.e. 1 or 2) to measure the metrics EM, EM+loc, F1 and F1+loc.

#### Answers from ChatGPT-4

Question

**Sentence 1:** Portugal has a stable wolf population of 200–300, which is afforded full protection.

**Sentence 2:** Portugal has a stable wolf population of 200–300, which is afforded complete defense.

Is "full protection" in Sentence 1 semantically equivalent to "complete defense" in Sentence 2?

Answer: This is answer



GPT-4 Response

Yes, "full protection" and "complete defense" are semantically equivalent in this context. Both phrases indicate that the wolf population is given the highest level of protection or defense from threats, such as hunting or habitat destruction. ✘

Table 6.18: **Zeroshot** accuracy (%) on **PS** using the state-of-the-art multimodal large language model ChatGPT-4.  $\Delta$  (e.g. **-24.80**) denotes the differences between the Top-1 accuracy in the contextualized (“Phrase + Context”) vs. the non-contextualized (“Phrase only”) setting.

Prompt	Accuracy
(a) With context sentences (Phrase + Context)	
Given two context sentences "<sentence <sub>1</sub> >" and "<sentence <sub>2</sub> >" containing two phrases "<phrase <sub>1</sub> >" and "<phrase <sub>2</sub> >", respectively. Are the phrases <b>semantically equivalent</b> ? Please answer Yes or No along with an explanation.	66.40
Sentence 1: <sentence <sub>1</sub> > Sentence 2: <sentence <sub>2</sub> >	74.70
Does "<phrase <sub>1</sub> >" in Sentence 1 <b>have the same meaning</b> as "<phrase <sub>2</sub> >" in Sentence 2? Please answer Yes or No along with an explanation.	
Sentence 1: <sentence <sub>1</sub> > Sentence 2: <sentence <sub>2</sub> >	71.05
Does "<phrase <sub>1</sub> >" in Sentence 1 <b>have a similar meaning</b> as "<phrase <sub>2</sub> >" in Sentence 2? Please answer Yes or No along with an explanation.	
Sentence 1: <sentence <sub>1</sub> > Sentence 2: <sentence <sub>2</sub> >	<b>75.00</b>
Is "<phrase <sub>1</sub> >" in Sentence 1 <b>semantically equivalent</b> to "<phrase <sub>2</sub> >" in Sentence 2? Please answer Yes or No along with an explanation.	
(b) Without context sentences (Phrase only)	
Is "<phrase <sub>1</sub> >" in Sentence 1 semantically equivalent to "<phrase <sub>2</sub> >" in Sentence 2? Please answer Yes or No along with an explanation.	50.20 <b>(-24.80)</b>

Table 6.19: **Zeroshot** accuracy (%) on **PSD** using the state-of-the-art multimodal large language model ChatGPT-4.  $\Delta$  (e.g. **-24.80**) denotes the differences between the Top-1 accuracy in the contextualized (“Phrase + Context”) vs. the non-contextualized (“Phrase only”) setting.

Prompt
Passage 1: <passage <sub>1</sub> > Passage 2: <passage <sub>2</sub> >
Question: Does "<query phrase>" have a similar meaning to "<target phrase>" in Passage 1 or Passage 2? Please answer 1 or 2 and then explain.
<b>Accuracy: 70.68</b>
Passage 1: <passage <sub>1</sub> > Passage 2: <passage <sub>2</sub> >
Question: Output a phrase that has a similar meaning to "<query phrase>" along with 1 if that phrase is in Passage 1 or 2 if that phrase is in Passage 2 using the format <phrase>:<number> and then explain.
<b>EM: 64.97   F<sub>1</sub>: 82.54   EM+loc: 43.64   F<sub>1</sub>+loc: 56.70</b>

## 6.4 Supplementary Materials for PEEB (Chapter 5)

### 6.4.1 Architecture details

#### Image encoder and text encoder

We employ the image encoder and text encoder from OWL-ViT. In order to maintain a general understanding of natural languages and avoid overfitting our training samples, we keep the text encoder frozen for all training and experiments. This setup allows our design to be flexible about the choice of text encoder, e.g., one can easily replace the text encoder without changing other architecture.

#### Linear projection (for part embedding selection)

The image embedding will be forwarded to a **Linear Projection** layer (see detail implementation here), which is composed of a learnable logit scale, a learnable logit shift, and an Exponential Linear Unit (ELU) activation function. These processed image embeddings then have the same dimension as the text embeddings. For OWL-ViT<sub>B/32</sub>, the image embeddings are projected from 768 to 512. We select a single image embedding for each text query. In this context, the text queries correspond to the component names of the target object, which includes twelve distinct parts. This selection is based on the cosine similarity between the projected image embeddings and the text embeddings. Finally, the chosen images embeddings (before projection) will be sent to the **Part MLP** for classification and **Box MLP** for box prediction (Figure 6.35, Step 1).

#### Part MLP

We introduce **Part MLP** to enable part-based classification (see implementation detail here). It comprises a three-layer MLP with GELU activations [62]. **Part MLP** takes in the selected part embeddings (i.e. output of step 1 in Figure 6.35) and outputs a vector of size  $\mathbb{R}^d$  for each part, where  $d$  is the dimension of descriptor embeddings (for OWL-ViT<sub>B/32</sub>, the input dimension is 768, and

$d = 512$ ). Part MLP is trained to map the selected part embeddings to the same dimensional space with descriptor embeddings to compute final logits for classification.

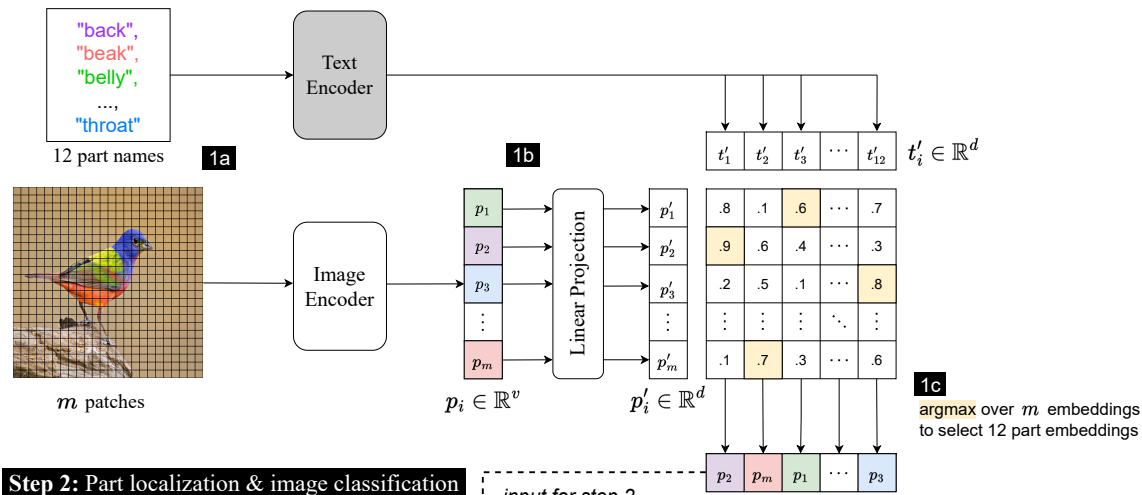
### **Box MLP**

The **Box MLP** retained from OWL-ViT consists of a three-layer MLP (see here for implementation detail). It takes the visual embedding as input and generates a four-element vector corresponding to the center coordinates and size of a bounding box (e.g.,  $[x, y, \text{width}, \text{height}]$ ). It is important to note that the image embedding inputs of **Box MLP** and **Part MLP** layers are the same, as shown in Figure 6.35, Step 2.

### **Visual part embedding selection**

As shown in Figure 6.35 step 1, 1c, the image embeddings are first projected by a Linear Projection layer and compute the dot product with the encoded part names. The image embeddings (before Linear Projection) are chosen as visual part embeddings by selecting the embedding that has the highest similarity scores with the corresponding part after the Linear Projection.

**Step 1: Part embeddings selection**



**Step 2: Part localization & image classification**

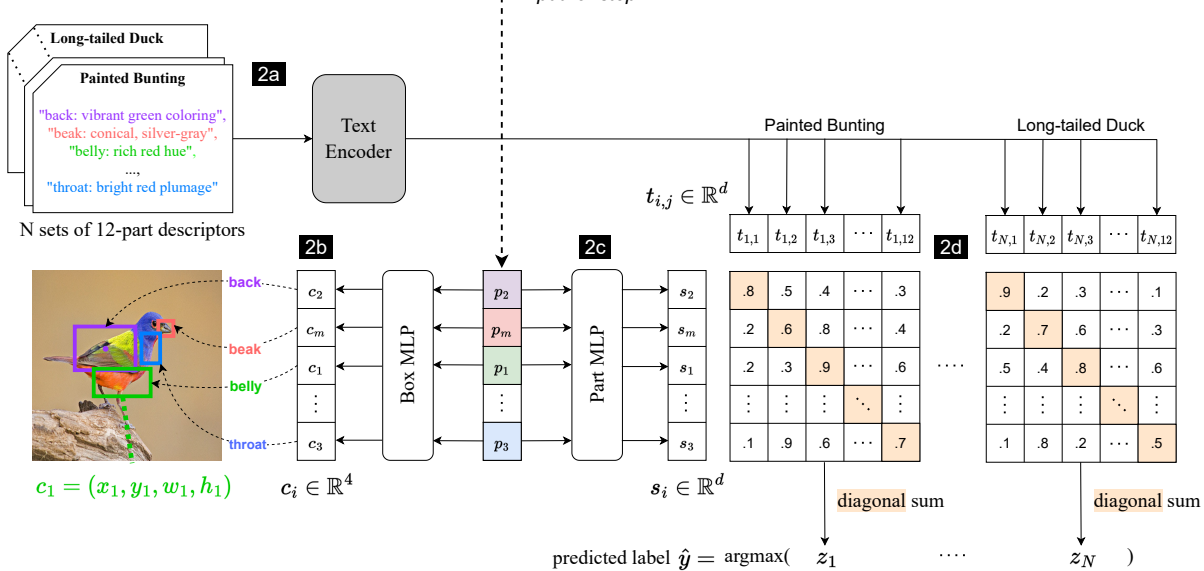


Figure 6.35: During the test time using PEEB, we perform 2 steps.

**Step 1:** (a) Encode an input image and texts (i.e. 12 part names) by the image and text encoder to get patch embeddings  $p_i$  and text embeddings  $t'_i$ . (b) Feed  $p_i$  to Linear Projection to get  $p'_i$  in the same dimensional space with  $t'_i$  and compute dot product between  $\{p'_i\}$  and  $\{t'_i\}$ . (c) arg max over  $m$  embeddings to select 12 part embeddings.

**Step 2:** (a) Encode input texts (i.e. N sets of 12-part descriptors) with the same text encoder to get  $t_i$ . (b) Feed the selected part embeddings to Box MLP to localize parts (in center format). (c) Also feed the selected part embeddings to Part MLP to get  $s_i$  in the same dimensional space with  $t_i$  (d) Compute a dot product between  $\{s_i\}$  and  $\{t_i\}$ , then diagonal sum for each class and arg max over logits to get predicted label  $\hat{y}$ .

## Descriptor embedding matching

To enhance the model’s flexibility, we do not use a linear layer for classification. Instead, we adopt a strategy similar to CLIP: we compute the similarity matrix of the projected visual embeddings (image embeddings after processing by the **Part MLP**) and the text embeddings. Then, we sum the corresponding similarities of each part in the class; the class with the highest score is considered the predicted class as shown in Figure 6.35, step 2, 2d. This design enables our proposed method to perform arbitrary ways of classification.

## Implementation details

Our experiments are conducted under PyTorch [111]. We employ HuggingFace’s [162] implementation of OWL-ViT and use their pre-trained models. The DETR losses implementation [17] is employed directly from their official implementation.

## Training hyperparameters

We provide the hyperparameters of all models trained in this work. Table 6.20 shows the details of the pre-training models. Table 6.21 presents the details of the finetuned models. All trainings utilize optimizer AdamW with Plateau Scheduler.

## Computational budget and infrastructures

We use 8 Nvidia RTX A100 GPUs for our experiments. The pertaining approximate takes ~24 hours on Bird-11K. The finetuning takes 2 to 4 hours with one single GPU.

## Pre-training and Finetuning objectives

As discussed in Section 5.4.3, we have three objectives during the **Pre-training** phase:

1. Pre-training **Stage 1:** (Figure 6.36) During the pre-training stage one, we contrastively pre-train the model to maximize the similarity between related part-descriptor pairs while minimizing the unrelated pairs using *symmetric cross-entropy (SCE)* loss [126].
2. Pre-training **Stage 2:** (Figure 6.37) We try to remove the dependence on the OWL-ViT<sub>Large</sub> teacher model by training PEEB to mimic OWL-ViT<sub>Large</sub>'s box predictions using the SCE loss.
3. Pre-training **Stage 2:** (Figure 6.37) We simultaneously train PEEB to improve box prediction with DERT losses [183].

During the **Finetuning** phase where we finetune on a downstream task (e.g. Dogs-120 or CUB-200), we also employ the same three losses. However, we change the first loss from SCE into CE since on the downstream classification task, the classifier is tasked with selecting one class that matches the single input image from a set of classes.

### **Pre-training stage one: Symmetric cross-entropy loss for contrastive pre-training**

We first define the embeddings derived from the image and text encoders:

$$I'_f = \text{image\_encoder}(I) \tag{6.1}$$

where  $I$  is the input image, and  $I'_f \in \mathbb{R}^{n \times d_i}$  is output image embeddings. Here,  $d_i$  is the feature dimension of the image encoder. The text embedding  $T_f$  is given by

$$T_f = \text{text\_encoder}(T) \tag{6.2}$$

where  $T$  represents the text input, and  $T_f \in \mathbb{R}^{m \times d_t}$ . In this case,  $d_t$  is the feature dimension of the text encoder. The image embedding  $I'_f$  is then transformed by *Part MLP* layer (Figure 6.35, 1b) to



align its dimensions with the text embedding. This transformation is denoted as

$$I_f = \text{Part MLP}(I'_f) \quad (6.3)$$

where  $I_f \in \mathbb{R}^{n \times d_t}$ . The similarity matrix  $S$  between the image and text embeddings is computed as the dot product of  $I_f$  and the transpose of  $T_f$ , expressed as

$$S = I_f \cdot T_f^\top \quad (6.4)$$

where  $S \in \mathbb{R}^{n \times m}$ . The image logits ( $S^i$ ) and text logits ( $S^t$ ) are then defined as

$$S^i = \text{softmax}(S, \text{axis}=0) \quad (6.5)$$

and

$$S^t = \text{softmax}(S, \text{axis}=1) \quad (6.6)$$

Next, we define the symmetric cross-entropy loss for the multi-modal embeddings.

$$L_{sce} = -\frac{(\sum_i y_i^i \log(S_i^i) + \sum_m y_m^t \log(S_m^t))}{2} \quad (6.7)$$

where  $y^i \in \mathbb{R}^n$  is the label for image and  $y^t \in \mathbb{R}^m$  is the label for text.

## **Pre-training stage 2: Symmetric cross-entropy loss to mimic the teacher OWL-ViT<sub>Large</sub> detector**

To mimic the object detection capability of the OWL-ViT<sub>Large</sub> teacher, we train PEEB to mimic the image-text similarity prediction between image embedding and textual part-name embeddings (as shown in Figure 6.35, 1c). We first binary the teacher logits and consider it as the ground truth label. Then, apply the same symmetric cross-entropy loss as described in Equation (6.7) with two

minor differences: (1) The text input is part names rather than descriptions. (2) The *Part MLP* is replaced by *Linear Projection* (Figure 6.35, 2c).

### Pre-training stage 2: DETR losses to mimic the teacher OWL-ViT<sub>Large</sub> detector

DETR losses are designed to optimize the box detection performance. We employ partial losses in our training for box predictions. Specifically, we employ  $\ell_1$  corner-to-corner distance loss and GIoU loss. For the selected embeddings, we predict the boxes with *Box MLP* (Figure 6.35, 2b)

$$B = \text{Box MLP}(I'_f) \quad (6.8)$$

where  $I'_f$  is the image selected image embeddings from Equation (6.1),  $B \in \mathbb{R}^{n \times 4}$  is the predicted bounding boxes. Let  $Y^{GT} \in \mathbb{R}^{n \times 4}$  be the ground truth boxes. The  $\ell_1$  corner-to-corner distance loss is defined as

$$L_{\ell_1} = \sum_i \|Y_i^{GT} - B_i\| \quad (6.9)$$

The GIoU loss  $L_{GIoU}$  is defined in Section 6.4.1, and the total box loss is defined as

$$L_{Box} = \frac{L_{\ell_1} + L_{GIoU}}{2} \quad (6.10)$$

---

#### Algorithm 1 Generalized Intersection over Union

---

**Require:** Two arbitrary convex shapes:  $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$

**Ensure:**  $GIoU$

- 1: For  $A$  and  $B$ , find the smallest enclosing convex object  $C$ , where  $C \subseteq \mathbb{S} \in \mathbb{R}^n$
  - 2:  $IoU = \frac{|A \cap B|}{|A \cup B|}$
  - 3:  $GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|}$
-

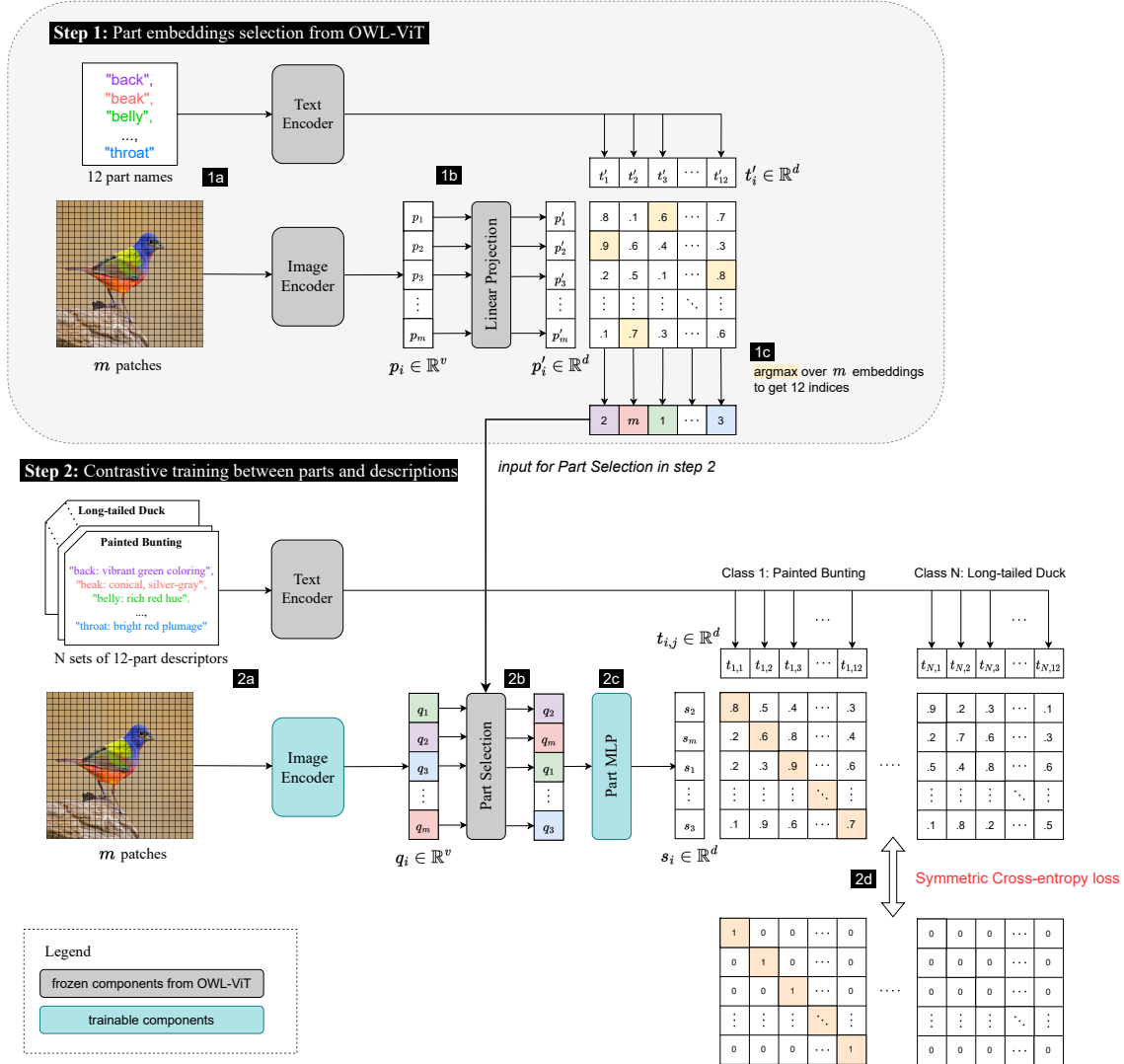


Figure 6.36: In pre-training stage 1, the objective is to let the Image Encoder learn the general representation of different parts of the birds. Therefore, in pre-training stage 1, we train the *Image Encoder* and *Part MLP* contrastively. During the training, the **Step 1** utilizes a teacher model (OWL-ViT<sub>B/32</sub>) to help PEEB select 12 part embeddings. In **Step 2**, we update the model with symmetric Cross-Entropy loss. Here's the flow of **Step 1**: (1a) We utilize the teacher model to encode 12 part names and the image to derive the text embedding  $t'_i$ , and the patch embedding  $p_i$ . (1b) Then the patch embeddings  $p$  is forwarded to Linear Projection to obtain  $p'$ , matching the dimension of  $t'$ . (1c) We compute the dot product between  $p$  and  $t'$  and apply  $\text{argmax}$  over  $p$  to derive 12 indices. In **Step 2**: (2a), We first encode the descriptors and the image with the *Text Encoder* and *Image Encoder* to obtain descriptor embeddings  $t$  and patch embeddings  $q$ . (2b), Then we select the 12 patch embeddings based on the 12 indices from (1c). (2c), The 12 patch embeddings then forwarded to *Part MLP* to derive  $s$ , which has the same dimension as  $t$ . Then, we compute the similarity matrix for the patch embedding and the descriptor embedding by computing the dot product between  $s$  and  $t$ . (2d), we construct a one-hot encoded matrix based on the descriptors as the ground truth label and minimize the Symmetric Cross-Entropy loss between the similarity matrix in (2c) and the ground truth label.

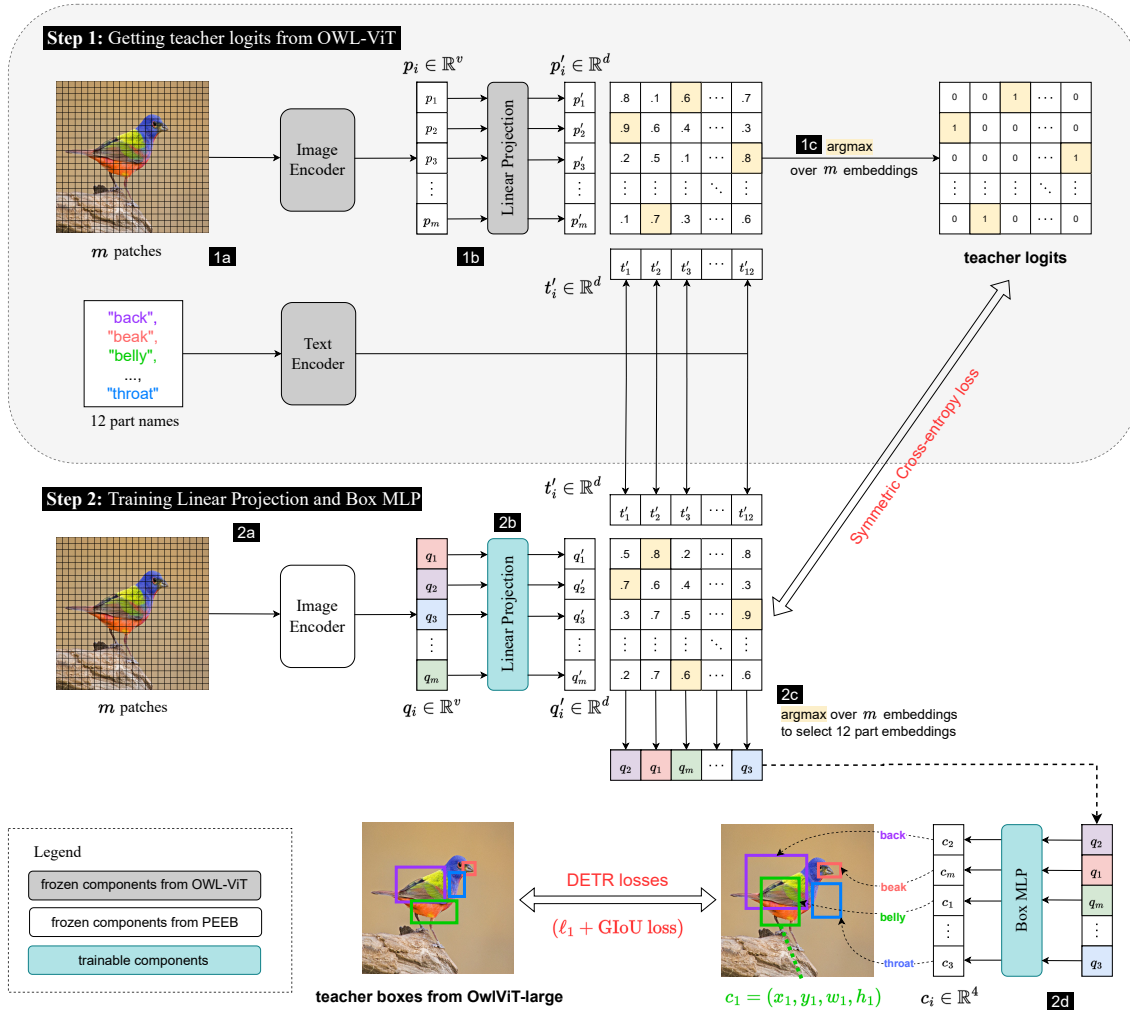


Figure 6.37: In pre-training stage 2, the goal is to eliminate the teacher model to obtain a standalone classifier. Therefore, the targeted components are **Linear Projection** and **Box MLP**. Since these two components are taking care of different functionalities for patch embedding selection and box prediction, respectively, stage 2 training is a multi-objective training. We employ Symmetric Cross-Entropy loss to learn the patch embedding selection and DETR losses to refine the box predictions. In **Step 1**: (1a), We first encode the 12 part names and the image with *Text Encoder* and *Image Encoder* to obtain the text embedding  $t'_i$  and patch embedding  $p_i$ . (1b) Then the patch embeddings  $p$  is projected by Linear Projection to obtain  $p'$ . (1c) We then compute dot product between  $p'$  and  $t'$  and one-hot encode the matrix via the dimension of  $p'$  to obtain the “teacher logits”. In **Step 2**: (2a), We encode the image with *Image Encoder* to obtain patch embedding  $q_i$ . (2b) The patch embeddings are then being projected by **Linear Projection** to derive  $q'$ . (2c), We compute the dot product between projected patch embeddings  $q'$  and part name embeddings  $t'$  to obtain the similarity matrix. Then, we employ Symmetric Cross-Entropy loss between the similarity matrix and the “teacher logits” derived in (1c). (2d), Meanwhile, we select the 12 part embeddings by taking  $\text{argmax}$  over  $q'$ . Then, the selected part embeddings are forwarded to **Box MLP** to predict the coordinates of each part. We compute the DETR losses for the predicted coordinates and update the model.

Table 6.20: Pre-training details of our pre-trained models.

Model	Epoch	Batch size		LR	Weight decay	# in-batch classes		Early stop	Training set
		Train	Val			Train	Val		
Pre-training stage 1									
PEEB <sub>[-test]</sub>	32	32	50	$2e^{-4}$	0.01	48	50	5	Bird-11K <sub>[-test]</sub>
PEEB <sub>[-CUB]</sub>	32	32	50	$2e^{-4}$	0.001	48	50	10	Bird-11K <sub>[-CUB]</sub>
PEEB <sub>[-NAB]</sub>	32	32	50	$2e^{-4}$	0.001	48	50	10	Bird-11K <sub>[-NAB]</sub>
Pre-training stage 2									
PEEB <sub>[-test]</sub>	32	32	50	$2e^{-5}$	0.01	48	50	5	Bird-11K <sub>[-test]</sub>
PEEB <sub>[-CUB]</sub>	32	32	50	$2e^{-5}$	0.001	48	50	5	Bird-11K <sub>[-CUB]</sub>
PEEB <sub>[-NAB]</sub>	32	32	50	$2e^{-5}$	0.001	48	50	5	Bird-11K <sub>[-NAB]</sub>

Table 6.21: Details of our finetuned models.

Model	Fine-tune from	Epoch	Batch size	LR	Weight decay	Early stop	Training set
PEEB <sub>[-test]</sub> <sup>CUB</sup>	PEEB <sub>[-test]</sub>	30	32	$2e^{-5}$	0.001	5	CUB
PEEB <sub>[-cub]</sub> <sup>Akata</sup>	PEEB <sub>[-CUB]</sub>	5	32	$2e^{-5}$	0.001	5	CUB ZSL [3]
PEEB <sub>[-cub]</sub> <sup>SCS</sup>	PEEB <sub>[-CUB]</sub>	5	32	$2e^{-5}$	0.001	5	CUB-SCS
PEEB <sub>[-cub]</sub> <sup>SCE</sup>	PEEB <sub>[-CUB]</sub>	5	32	$2e^{-5}$	0.001	5	CUB-SCE
PEEB <sub>[-nab]</sub> <sup>SCS</sup>	PEEB <sub>[-NAB]</sub>	5	32	$2e^{-5}$	0.001	5	NABirds-SCS
PEEB <sub>[-nab]</sub> <sup>SCE</sup>	PEEB <sub>[-NAB]</sub>	5	32	$2e^{-5}$	0.001	5	NABirds-SCE

## 6.4.2 Model and dataset notations

### Dataset notations

Following the conventional setup of ZSL, we execute certain exclusions to make sure none of the test classes or descriptors are exposed during pre-training. That is,  $\text{Bird-11K}_{[-\text{CUB}]}$  and  $\text{Bird-11K}_{[-\text{NAB}]}$  exclude all CUB and NABirds classes, respectively. For GZSL, we exclude all test sets in CUB, NABirds, and iNaturalist, denoted as  $\text{Bird-11K}_{[-\text{test}]}$ . We provide detailed statistics for the three pre-training sets in Table 6.22.

Table 6.22: Three pre-training splits for PEEB.

Training set	Number of images		Number of classes	
	Train	Val	Train	Val
$\text{Bird-11K}_{[-\text{test}]}$	234,693	29,234	10,740	9,746
$\text{Bird-11K}_{[-\text{CUB}]}$	244,182	28,824	10,602	9,608
$\text{Bird-11K}_{[-\text{NAB}]}$	216,588	27,996	10,326	9,332

### Model notations

We adopt a strategy based on the datasets excluded during training to simplify our model naming convention. Specifically:

- $\text{PEEB}_{[-\text{test}]}$  is pre-trained model using  $\text{Bird-11K}_{[-\text{test}]}$  dataset.
- $\text{PEEB}_{[-\text{CUB}]}$  is pre-trained model using the  $\text{Bird-11K}_{[-\text{CUB}]}$  dataset.
- $\text{PEEB}_{[-\text{NAB}]}$  is pre-trained model using the  $\text{Bird-11K}_{[-\text{NAB}]}$  dataset.

We named finetuned models after the pre-trained model and the finetuned training set. For example,  $\text{PEEB}_{[-\text{test}]}^{\text{CUB}}$  is finetuned from  $\text{PEEB}_{[-\text{test}]}$ , on CUB training set.

## 6.4.3 Generating part-based descriptors

CUB annotations initially comprise 15 bird parts. However, distinctions between the left and right part are not essential to our method, we merge them into a single part (i.e., “left-wing” and

“right-wing” are merged into “wings”) Hence, we distilled the original setup into 12 definitive parts: *back, beak, belly, breast, crown, forehead, eyes, legs, wings, nape, tail, throat*. To compile visual part-based descriptors for all bird species within Bird-11K, we prompted GPT-4 [107] with the following input template:

```
A bird has 12 parts: back, beak, belly, breast, crown, forehead, eyes, legs, wings, nape, tail and throat. Visually describe all parts of {class name} bird in a short phrase in bullet points using the format 'part: short phrase'
```

Where {class name} is substituted for a given bird name (e.g., Painted Bunting).

The output is a set of twelve descriptors corresponding to twelve parts of the query species. e.g. The response for Cardinal is:

```
Cardinal: {  
  back: vibrant red feathers,  
  beak: stout, conical, and orange,  
  belly: light red to grayish-white,  
  breast: bright red plumage,  
  crown: distinctive red crest,  
  forehead: vibrant red feathers,  
  eyes: small, black, and alert,  
  legs: slender, grayish-brown,  
  wings: red with black and white accents,  
  nape: red feather transition to grayish-white,  
  tail: long, red, and wedge-shaped,  
  throat: bright red with sharp delineation from white belly  
}
```

## 6.4.4 Datasets

### Bird-11K

We provide a brief statistic of Bird-11K in Table 6.23. Bird-11K is a diverse and long-tailed bird-image dataset. The descriptors generated by GPT-4 are in English and only describe the visual features of the corresponding class. We propose Bird-11K for academic research only.

Table 6.23: Number of images and species of different bird datasets. Our proposed dataset Bird-11K includes almost all avians on Earth.

Dataset	# of Images	# of Species
CUB-200-2011 [154]	12,000	200
Indian Birds [151]	37,000	25
NABirds v1 [152]	48,000	400
Birdsnap v7 [10]	49,829	500
iNaturalist 2021-birds [153]	74,300	1,320
ImageNet-birds [28]	76,700	59
BIRDS 525 [123]	89,885	525
Macaulay Library at the Cornell Lab of Ornithology	55,283	10,534
Bird-11K (Raw Data)	440,934	11,097
<b>Bird-11K (pre-training set)</b>	294,528	10,811

**Data splits** We provide data splits and metadata, e.g., file names, image size, and bounding boxes, along with the instruction of Bird-11K construction in our repository. Note that the Bird-11K dataset is for pre-training purposes; it is important to execute exclusion based on the test set.

### License and terms

- CUB [154]: The dataset can be freely used for academic and research purposes; commercial use is restricted.
- Indian Birds [151]: CC0: Public Domain.



- NABirds-v1 [152]: For non-commercial research purposes, other use is restricted <sup>4</sup> here for detail: .
- Birdsnap-v7 [10]: The dataset creator provides no specific license or terms of use. We only use this dataset for academic research until more specific details can be obtained.
- iNaturalist 2021-birds [153]: CC0: Public Domain. We use the `train_mini` subset on Github, which has 1,486 classes. After filtering out images (as described in Section 5.3.2), we end up with 1,320 classes and 74,300 images for including in Bird-11K.
- ImageNet-birds [28]: BSD-3-Clause license.
- BIRDS 525 [123]: CC0: Public Domain
- Cornell eBird: We used the following 55,384 recordings from the Macaulay Library at the Cornell Lab of Ornithology. The data is for academic and research purposes only, not publicly accessible unless requested. (Please refer to our Supplementary Material for the full list):

ML187387391, ML187387411, ML187387421, ML187387431, ML262407521, ML262407481, ML262407531, ML262407491, ML262407511, ML257194111 ML257194071, ML257194081, ML257194061, ML495670791, ML495670781, ML495670801, ML495670771, ML183436431, ML183436451, ML183436441 ML183436411, ML183436421, ML256545901, ML256545891, ML256545841, ML256545851, ML256545831, ML169637941, ML238083081, ML169637881 ML169637911, ML238083111, ML238083051, ML169637971, ML299670841, ML64989231, ML299670831, ML64989241, ML299670791, ML64989251 ML246866001, ML246865941, ML246866011, ML246865961, ML246865971, ML333411961, ML240835531, ML240835541, ML240835701, ML240835591 ML245260391, ML245260341, ML245260371, ML245260411, ML245260421, ML245260431, ML245260441, ML240866351, ML240866331, ML240866321 ML240866341, ML240866371, ML248318661, ML248318571, ML248318591, ML248318581, ML248318631, ML245204281, ML245204311, ML245204371 ML245204381, ML245204291, ML245603571, ML245603521, ML245603511, ML245603491, ML245603501, ML245603601, ML245257771, ML245257651 ML245257631, ML245257661, ML245257761, ML247221051, ML247221061, ML247221071, ML247221081, ML240365811, ML240365751, ML240365781 ML240365761, ML300579541, ML247298551, ML247298541, ML247298561, ML247298611, ML247298571, ML247298591, ML247298601, ML247298631...

## Dog-140

To pre-train PEEB on dogs, we construct Dog-140 by combining dog images from ImageNet-21K and Stanford Dogs-120. Specifically, we selected 189 dog classes from ImageNet-21K, and based on Fédération Cynologique Internationale (FCI) [44], we merged them with 120 classes from Stanford Dogs, ending up with 140 classes. After merging, Dog-140 has 206,076 images in total. We provide a class distribution analysis in Figure 6.38, where we can find that Dog-140 is roughly class-balanced.

---

<sup>4</sup>See Terms of Use

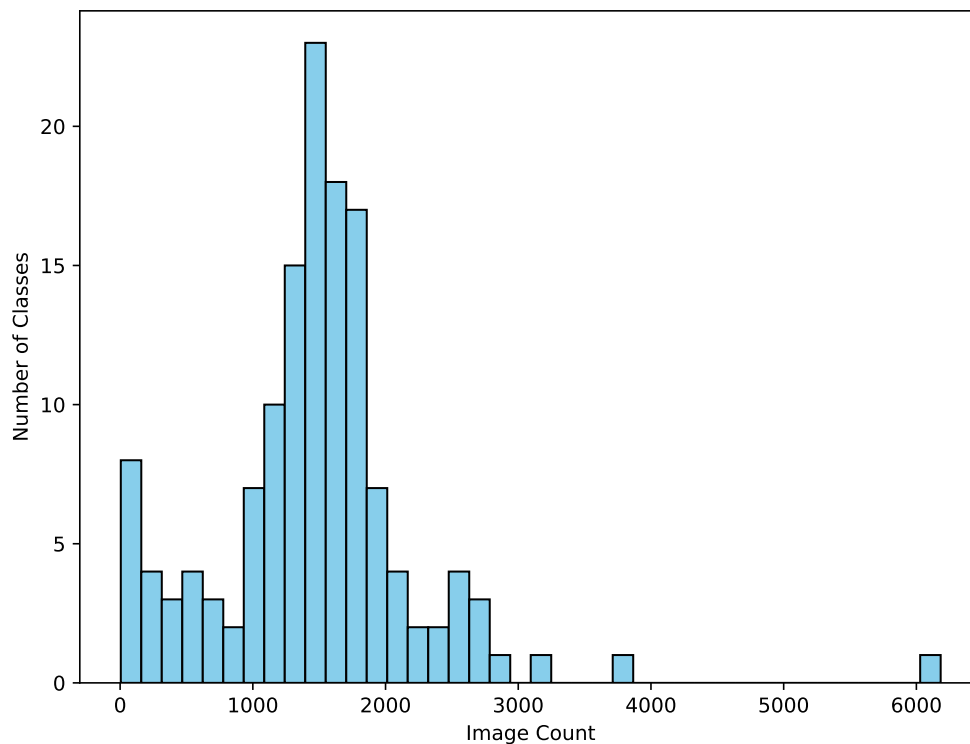


Figure 6.38: The class distribution of Dog-140 dataset. The histogram indicates that most classes in Dog-140 have around 1,000 to 2,000 images.

**Data splits** Similar to Bird-11K, we provide data splits and metadata, e.g., file names, image size, and bounding boxes, along with the instruction of Dog-140 construction in our repository.

### License and terms

- Stanford Dogs [75]: The dataset was constructed using images and annotations from ImageNet. Therefore, all the images (including those presented in the paper) follow the ImageNet license.
- ImageNet-21K [28]: BSD-3-Clause license, non-commercial.

## 6.4.5 Additional results

### PEEB outperforms M&V in CUB and NABirds in ZSL setting

To rigorously evaluate the ZSL capabilities of our pre-trained models, we introduce a stress test on the CUB and NABirds datasets. The crux of this test involves excluding all classes from

the target dataset (CUB or NABirds) during the pre-training. The exclusion ensures that the model has no prior exposure to these classes. Subsequently, we measure the classification accuracy on the target dataset, comparing our results against benchmarks set by CLIP and M&V in the scientific name test. In this experiment, we consider the scientific name test a ZSL test for CLIP and use them as the baseline because the frequencies of scientific names are much lower than common ones.

**Experiment** To conduct this test, we pre-train our model on Bird-11K<sub>[-CUB]</sub> and Bird-11K<sub>[-NAB]</sub>, which deliberately exclude images bearing the same class label as the target dataset. Specifically, we test on our pre-train model PEEB<sub>[-CUB]</sub> and PEEB<sub>[-NAB]</sub> (see Table 6.20 for details), respectively.

**Results** The primary objective is to ascertain the superiority of our pre-trained model, PEEB, against benchmarks like CLIP and M&V. For CUB, our method reported a classification accuracy of 17.9%, contrasting the 5.95% and 7.66% achieved by CLIP and M&V, respectively, as shown in Table 6.24. The PEEB score, which is substantially higher (+10) than M&V, highlights the advantages of our part-based classification. On NABirds, our method surpasses CLIP and M&V by +1 point. The performance disparity between CUB and NABirds can be attributed to two factors: The elevated complexity of the task (555-way classification for NABirds versus 200-way for CUB) and the marked reduction in training data. An auxiliary observation, detailed in Section 6.4.5, indicates that our pre-trained model necessitates at least 250k images to achieve admirable classification accuracy on CUB, but we only have 210k images training images in Bird-11K<sub>[-NAB]</sub> (the variants of Bird-11K with classes excluded for ZSL testing are described in Table 6.22).

Table 6.24: Stress test results on CUB and NABirds datasets. Despite the ZSL challenge, our method consistently surpasses CLIP and M&V. This underscores the robust generalization of our approach, which leverages descriptors for classification.

Method	CLIP	M&V	PEEB (ours)
CUB	5.95	7.66	<b>17.90</b>
NABirds	4.73	6.27	<b>7.47</b>

## Performance measurement on different noisy levels

In our evaluations, as indicated in Table 5.2, we discerned a marked performance disparity between the iNaturalist dataset and others. Probing this further, we identified image noise as a principal contributor to these discrepancies.

**Experiment** A qualitative assessment of the iNaturalist test images revealed a significantly higher noise level than CUB or NABirds. To systematically study this, we utilize the object detector OWL-ViT<sub>Large</sub> to measure the size of the bird within the images. We formulated two filtered test sets based on the detector’s output, categorizing them by the bird’s size, specifically, the detected bounding box. Images were filtered out if the bird’s size did not exceed predetermined thresholds (areas of  $100^2$  or  $200^2$  pixels). Larger birds naturally reduced other content by occupying more image space, thus serving as a proxy for reduced noise. All three test sets, including the original, were evaluated using our pre-trained model PEEB<sub>[-test]</sub>.

**Results** The results presented in Table 6.25 reveal a clear trend: as the image noise level decreases, the classification accuracy consistently improves, with gains ranging from (+6 to +17) points across the various methods. Notably, cleaner images consistently yield better results. At each noise level, our method outperforms the alternatives. While our method exhibits an impressive (+17 points) accuracy boost on the cleanest test set, this substantial gain also indicates that our model is sensitive to image noise.

Table 6.25: The table showcases the classification accuracies on iNaturalist as we vary the noise levels. The data underscores that the performance disparity on iNaturalist is predominantly due to image noise. While all methods improve with cleaner images, our model exhibits the most substantial gains, particularly in the least noisy sets.

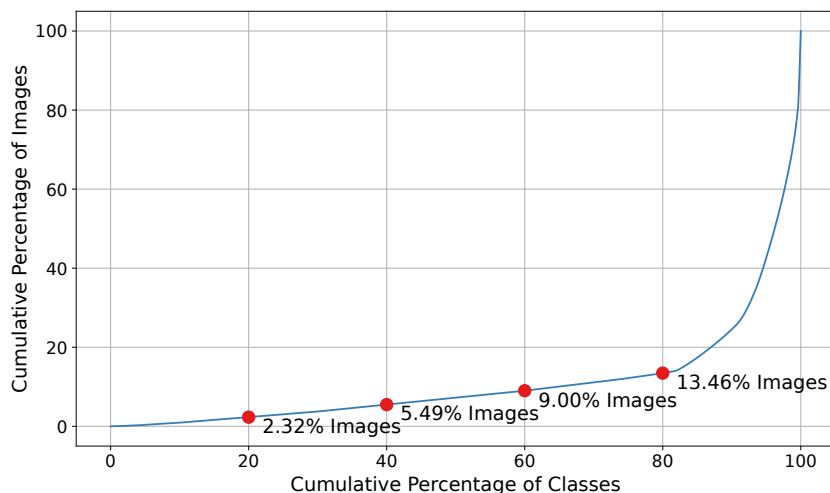
Splits	CLIP	M&V	PEEB (ours)
Original	16.36	17.57	<b>25.74</b>
> $100^2$ pixels	20.18	21.66	<b>35.32</b>
> $200^2$ pixels	22.88	24.90	<b>42.55</b>

## **Number of training images is the most critical factor towards classification accuracy**

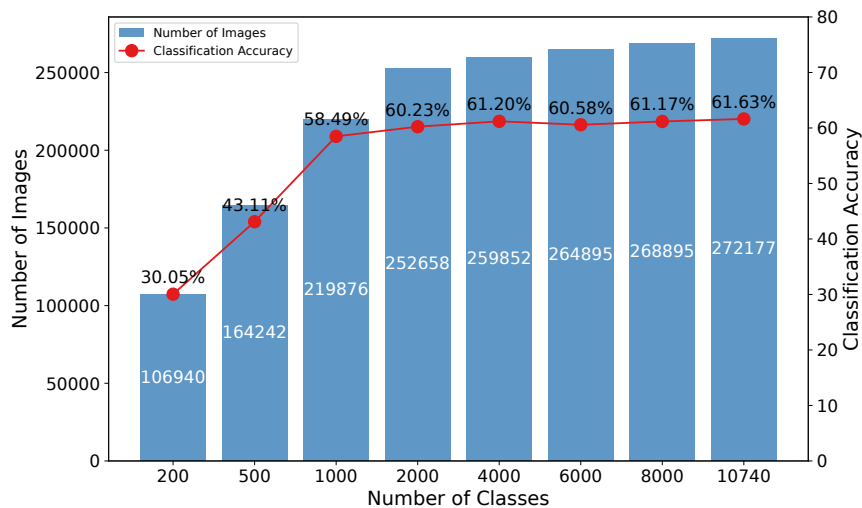
Bird-11K, as shown in Figure 6.39a, is a highly imbalanced dataset characterized by a large amount of long-tailed classes. We conduct a comprehensive study to discern how variations in the number of classes and images affect the classification accuracy of our pre-trained models. Predictably, the volume of training images occurred as the most influential factor. However, a noteworthy observation was that the abundance of long-tailed data enhanced the model's accuracy by approximately **+1.5** points.

**Experiment** We curated eight training sets based on varying class counts: 200, 500, 1,000, 2,000, 4,000, 6,000, 8,000, and 10,740. For each set, we maximized the number of training images. It is important to note that a set with a lesser class count is inherently a subset of one with a higher count. For instance, the 500-class set is a subset of the 2,000-class set. For each split, we apply the same training strategy as in Section 5.4.3, and choose the checkpoint with the best validation accuracy. We consider the CUB test set as a generic testing benchmark for all variants.

**Results** As illustrated in Figure Figure 6.39b, there is a pronounced correlation between the increase in the number of images and the corresponding surge in accuracy. For instance, an increment from 106K to 164K images led to a rise in classification accuracy from 30.05% to 43.11%. The accuracy appears to stabilize around 60% when the image count approaches 250K. This trend strongly suggests that the volume of training images is the most critical factor for the pre-trained model. We believe that the accuracy of the pre-trained model could be further enhanced if enough data is provided. Interestingly, a substantial amount of long-tailed data bolsters the model's performance, evident from **+1.5** points accuracy improvement when comparing models trained on 2,000 classes to those on 10,740 classes. Note that the additional classes in the latter set averaged merely 2.2 images per class.



(a) The Cumulative Distribution Function (CDF) plot for the Bird-11K dataset.



(b) Correlation between the number of training figures/peeb/classes and accuracy.

Figure 6.39: The CDF plot (a), underscores significant imbalance of the Bird-11K dataset. While the dataset has abundant long-tailed classes, e.g., a striking 80% of the classes contribute to only 13.46% of the entire image count. The plot (b) showcases the correlation between the number of training figures/peeb/classes and the resulting classification accuracy. As the image count grows, there is a noticeable surge in accuracy, which nearly stabilizes upon surpassing 250K images. Additionally, a significant amount of long-tailed data contributes to a +1.5 points boost in accuracy.

### Ablation study on the influence of parts utilized

In this ablation study, we aimed to measure the impact of varying the number of distinct “parts” (back, beak, belly, breast, crown, forehead, eyes, legs, wings, nape, tail, and throat) used in our

model. We experiment with a range from a single part to all 12 identifiable parts. Interestingly, even with a solitary part, the model could make correct predictions, though there was an evident decline in performance, approximately **-20** points.

**Experiment** Our testing ground is the pre-trained model PEEB<sub>[-test]</sub>, evaluated against the CUB test set. We assessed the model’s prowess utilizing various subsets of parts: 1, 3, 5, 8, and all 12. These subsets were derived based on the frequency of visibility of the parts within the CUB dataset, enabling us to compare the model’s performance when relying on the most frequently visible parts versus the least. For comparison, we also conduct a similar experiment on M&V, where we only use 1, 3, 5, 8, and 12 descriptors (if possible).

**Results** Relying solely on the most frequent part led to a decline in classification accuracy by around **-20** points, registering at 45.44% (Table 6.26). In contrast, utilizing the least frequent part resulted in a sharper drop of around **-27**, with an accuracy of 37.02%. As the model was furnished with increasing parts, its accuracy improved incrementally. The data underscores that optimal performance, an accuracy of 64.33%, is attained when all 12 parts are included. For M&V, the accuracy keeps increasing homogeneously from 5 to 12 descriptors, hinting that accuracy may increase further by increasing the number of descriptors.

Table 6.26: Classification accuracy on the CUB test set that uses a different number of parts. Performance dips significantly with just one part, especially for the least visible ones. Maximum accuracy is reached with all 12 parts. The last row of the table also shows the accuracy of [95] method which employs a different number of parts. It is evident that their method is insensitive to the number of parts used, which may not reflect a realistic scenario.

Number of Parts (descriptors)	1	3	5	8	12
Accuracy (most frequent parts)	45.44	56.48	59.89	61.32	<b>64.33</b>
Accuracy (least frequent parts)	37.02	55.51	60.04	61.13	<b>64.33</b>
Accuracy of [95]	51.93	52.87	52.83	53.33	53.92

## Training is essential for PEEB’s classification efficacy

In this ablation study, we highlight the pivotal role of training in the performance of PEEB on bird classification tasks. We demonstrate that without adequate tuning, the results are indistinguishable from random chance.

**Experiment** We conduct the experiment based on OWL-ViT<sub>B/32</sub>. We retain all components as illustrated in Figure 6.35, with one exception: we substitute the Part MLP with the MLP layer present in the box prediction head of OWL-ViT because the proposed layers require training. The MLP layers in the box prediction head project the part embeddings to match the dimensionality of the text embeddings. Our focus is on assessing the classification accuracy of the untuned PEEB on two datasets: CUB and NABirds.

**Results** Table 6.27 reveals the outcomes of our experiment. Without training, PEEB yields classification accuracies of 0.55% for CUB and 0.31% for NABirds, both of which are proximate to random chance (0.5% for CUB and 0.1% for NABirds). However, with training, the model’s performance dramatically transforms: 64.33% for CUB (an increase of +63.78 points) and 69.03% for NABirds (a leap of +68.72 points) for PEEB<sub>[-test]</sub>. These pronounced disparities underscore the vital role of training in PEEB.

Table 6.27: Impact of Training on Classification Accuracies: Untuned PEEB yields 0.55% on CUB and 0.31% on NABirds, almost mirroring random chance. With training (PEEB<sub>[-test]</sub>), accuracy surges by +63.78 points on CUB and +68.72 points on NABirds.

	CUB	NABirds
PEEB (no training)	0.55	0.31
PEEB <sub>[-test]</sub> pre-trained	64.33	69.03
PEEB <sub>[-test]</sub> <sup>CUB</sup> finetuned	86.73	-



## Failure analysis

Since PEEB has two branches, box detection, and descriptor matching, we would like to find out, in the failure case, what is the main cause. i.e., is it because of the mismatch in the descriptor to the part embeddings? Or is it because the box detection is wrong? From our ablation study, it turns out that most errors come from the descriptor-part matching.

**Experiment** We conduct the experiment with PEEB<sub>[-test]</sub> on CUB test set. Specifically, we measure the box detection accuracy based on the key point annotation in CUB dataset, i.e., We consider the box prediction as **correct** if the prediction includes the human-annotated key point. We report the box prediction error rate (in %) based on parts.

**Results** As shown in Table 6.28, the average error rate difference between success and failure cases is merely 0.38. That is, in terms of box prediction, the accuracy is almost the same, disregarding the correctness of bird identification. It indicates that the prediction error is predominantly due to the mismatch between descriptors and part embeddings. We also noted that some parts, like Nape and Throat, have a very high average error rate, which may greatly increase the matching difficulties between descriptors and part embeddings.

Table 6.28: Error rate of Box Prediction in Failure and Success Cases. We report the box prediction error rate, depending on whether the prediction box includes ground truth key points. No major difference is found between them, which means the failure is largely due to the part-descriptor mismatch.

Body Part	Average	Back	Beak	Belly	Breast	Crown	Forehead	Eyes	Legs	Wings	Nape	Tail	Throat
Failure Cases	16.52	23.38	3.28	8.06	15.96	7.41	24.72	7.29	5.63	3.36	64.79	7.25	27.07
Success Cases	16.14	23.03	2.96	7.44	18.64	7.13	21.53	3.93	6.85	2.68	68.66	6.40	24.38
Difference	<b>0.38</b>	0.35	0.33	0.62	-2.68	0.28	3.19	3.36	-1.22	0.68	-3.87	0.85	2.68

## Evaluation of predicted boxes from PEEB

Our proposed method primarily aims to facilitate part-based classification. While the core objective is not object detection, retaining the box prediction component is paramount for ensuring

model explainability. This section delves into an evaluation of the box prediction performance of our method against the OWL-ViT<sub>B/32</sub> model.

**Experiment** Given our focus on part-based classification, we aimed to ascertain the quality of our model’s box predictions. To this end, we employed two metrics: mean Intersection over Union (IoU) and precision based on key points. We opted for mean IoU over the conventional mAP because: (1) Ground-truth boxes for bird parts are absent, and (2) our model is constrained to predict a single box per part, ensuring a recall of one. Thus, we treat OWL-ViT<sub>Large</sub>’s boxes as the ground truth and evaluate the box overlap through mean IoU. Furthermore, leveraging human-annotated key points for bird parts, we measure the precision of predicted boxes by determining if they contain the corresponding key points. We evaluate our finetuned models on their corresponding test sets. For instance, PEEB<sub>[-cub]</sub><sup>Akata</sup>, finetuned based on the CUB split [3], is evaluated on the CUB test set.

**Results** Our evaluation, as presented in Table 6.29, shows that PEEB’s box predictions do not match those of OWL-ViT<sub>B/32</sub>. Specifically, on average, there is a **-5 to -10** points reduction in mean IoU for CUB and NABirds datasets, respectively. The disparity is less distinct when examining precision based on human-annotated key points; our method records about **-0.14** points lower precision for CUB and **-3.17** points for NABirds compared to those for OWL-ViT<sub>B/32</sub>. These observations reinforce that while PEEB’s box predictions might not rival these dedicated object detection models, they consistently highlight the same parts identified by such models as shown in Figure 6.40. It is important to note that our approach utilized the same visual embeddings for both classification and box prediction tasks. This alignment emphasizes the part-based nature of our model’s predictions.

Table 6.29: Model evaluation on CUB and NABirds test sets. We evaluate the predicted boxes on two *ground-truth* sets; (1) predicted boxes from OWL-ViT<sub>Large</sub> as ground-truths, and (2) OWL-ViT<sub>Large</sub>’s boxes that include the human-annotated key points. Our method has slightly lower performance in terms of mean IoU but comparable precision.

Models		Mean IoU		
		(1) All	(2) w/ Keypoints	Precision
CUB	OWL-ViT <sub>Large</sub>	<b>100.00</b>	<b>100.00</b>	<b>83.83</b>
	OWL-ViT <sub>B/32</sub>	44.41	49.65	83.53
	PEEB (Average)	35.98	40.14	83.39
	PEEB <sub>[-test]</sub> <sup>CUB</sup>	37.45	41.79	81.55
	PEEB <sub>[-cub]</sub> <sup>Akata</sup>	35.11	39.14	82.72
	PEEB <sub>[-cub]</sub> <sup>SCS</sup>	35.77	39.96	84.89
	PEEB <sub>[-cub]</sub> <sup>SCE</sup>	35.58	39.67	84.38
	NABirds	OWL-ViT <sub>Large</sub>	<b>100.00</b>	<b>100.00</b>
OWL-ViT <sub>B/32</sub>		40.14	47.63	83.89
PEEB (Average)		36.47	42.01	80.72
PEEB <sub>[-nab]</sub> <sup>SCS</sup>		36.45	42.03	80.09
PEEB <sub>[-nab]</sub> <sup>SCE</sup>		36.49	41.99	81.34

## 6.4.6 Study on GPT-4 generated descriptors

### Assessment of the generated part-based descriptors

We test GPT-4V on the CUB test set using the generated descriptors of 200 classes to assess their usability. Specifically, we feed GPT-4V with each test image encoded in the payload and 200 sets of part-based descriptors through a carefully designed prompt (Table 6.30). GPT-4V is asked to output one of 200 provided class names to compute the classification accuracy. As a result, GPT-4V achieves 69.4% accuracy which is slightly higher than PEEB’s generalized zero-shot accuracy (64.33%) and significantly lower than PEEB results after finetuning (86-88%).

Table 6.30: Prompt for GPT-4V evaluation on CUB where {list\_of\_200\_classes} is the placeholder for the actual 200 CUB classes while {descriptors} (see an example in Section 6.4.3) is the placeholder for the actual descriptors associated with a given bird image from the CUB test set.

You are an image classifier which can tell what type of a bird is from the given image and its associated part descriptors describing 12 parts of the bird. Your answer should be strictly formatted as {"prediction": "bird\_class"}.

where "bird\_class" is one of the following 200 bird classes: {list\_of\_200\_classes}

Given the bird image and the following descriptors: {descriptors}

What kind of bird is this? Let’s think step by step.

### Noise measurement in GPT-4 generated descriptors

In this section, we conduct an empirical analysis to quantify the noise in descriptors generated by GPT-4 for 20 different classes within the CUB dataset. To achieve this, we manually inspect each descriptor and tally the instances where at least one factual error is present. Our findings reveal that every one of the 20 classes contains descriptors with errors, and on average, 45% of the descriptors necessitate corrections. This substantial noise level underscores the need for further refinement in our work, particularly in text descriptors.

We observe a notably high error rate in descriptors on the *back* and *wings*, with approximately 60% of these containing inaccurate information (refer to Table 6.31). This could be attributed to the challenges in distinguishing between the *back* and *wings*, given that the *back* is typically positioned behind the *wings*, yet exhibits considerable variability in size and shape. Addressing all descriptor issues by revising all 11,000 fine-grained descriptors would demand a significant investment of time and resources, which is beyond the scope of the current work. As such, we identify this as an area for future research and development, aiming to enhance the quality of the Bird-11K dataset.

Table 6.31: Summary of manual inspection results for 20 classes, highlighting the need for revision in GPT-4 generated descriptors. An average error rate of 45% indicates substantial room for improvement.

	Back	Beak	Belly	Breast	Crown	Forehead	Eyes	Legs	Wings	Nape	Tail	Throat	Average
Error Rate	60	30	50	40	50	55	50	20	60	50	35	40	45

### Revising descriptors improves classification accuracy

As mentioned in the limitation section, the descriptors are generated from GPT-4 and therefore noisy and incorrect. Given that PEEB accepts open vocabulary inputs for classification, a natural way to improve classification accuracy is to improve the correctness of the descriptors.

**Experiment** We first collect descriptors of 183 CUB classes from AllAboutBirds. We then prompt GPT-4 to revise our original descriptors by providing the collected descriptor. We revise the descriptors with the following prompt:

Given the following descriptors of {class name}: {AllAboutBirds descriptors}. Can you revise the incorrect items below (if any) of this bird, return them as a Python dictionary, and use the key as the part name for each item? If a part's descriptor is not specifically described or cannot be inferred from the definition, use your own knowledge. Otherwise, leave as is. Note: please use a double quotation mark for each item such that it works with JSON format.

{Original descriptors}

Where `{class name}` the placeholder for the class name, `{AllAboutBirds descriptors}` is the description collected from AllAboutBirds, `{Original descriptors}` is the descriptors we used for training.

Due to the errors in the descriptors we used to train PEEB, simply replacing the descriptors with their revised version does not lead to better performance. Because the incorrect descriptors in training change the meaning of some of the phrases. For example, the belly of Blue bunting is pure blue, but the descriptors from GPT-4 is *soft, creamy white*. In addition, the GPT-4 uses the exact same descriptor in the belly for other classes, e.g., Blue breasted quail, which should be cinnamon. Blue Fronted Flycatcher, which should be yellow. Training the same descriptors with different colors confuses the model, and the model will convey the phrase “creamy white” with a different meaning to humans. Therefore, simply changing the descriptors to their’ revised version will not work. We empirically inspect the descriptors that PEEB can correctly respond to and replace the class descriptors with the revised version. Specifically, we replace the descriptors of 17 classes in CUB and test the classification accuracy on PEEB<sub>[-test]</sub>.

**Results** As shown in Table 6.32, the overall accuracy increases by **+0.8** points.

The average improvement of the revised class is around **+10.8**, hitting that if we have correct descriptors of all classes, we may significantly improve the classification accuracy of the pre-trained model. However, correcting all 11k class descriptors is too expensive and out of the scope of this work. We leave it as a further direction of improving the part-based bird classification.

Table 6.32: The revised descriptors result in **+0.8** for PEEB<sub>[-test]</sub> in CUB. In particular, the average improvement among the 17 revised classes is **+10.8**, hinting at the large potential of our proposed model.

Descriptors	Original	Partially Revised	Avg. Improvement
PEEB <sub>[-test]</sub>	64.33	<b>65.14</b>	10.80

## 6.4.7 Qualitative Inspections

### Visual comparison of predicted boxes

We provide a visual comparison of the box prediction from OWL-ViT<sub>Large</sub>, OWL-ViT<sub>B/32</sub>, and PEEB in Figure 6.40. We find that despite the fact that our predicted boxes have lower mean IoU compared to OWL-ViT<sub>Large</sub>, they are visually similar to the boxes as OWL-ViT<sub>B/32</sub>.

### Qualitative examples of using randomized descriptors

We visually compare M&V and PEEB based on their utilization of descriptors. (Figures 6.41 and 6.42). Specifically, we randomly swap the descriptors of the classes and then use these randomized descriptors as textual inputs to the tested models to see how they perform. We observe that the scores from M&V tend to cluster closely together. Surprisingly, M&V’s prediction remains unchanged despite the inaccurate descriptors. In contrast, PEEB, when presented with randomized descriptors, attempts to identify the best match grounded on the given descriptors.

### Examples of PEEB explanations for birds

Figures 6.43 to 6.45 are examples of how PEEB makes classification based on the descriptors and how it can reject the predictions made by M&V. Since we aggregate all descriptors for the final decision, even if some of them are similar in two classes, our method can still differentiate them from other descriptors. For instance, in Figure 6.43, while other descriptors are similar, PEEB can still reject chesnut-sided warbler thanks to the distinct features of *forehead*, *throat* and *belly*.

### Examples of PEEB explanations for dogs

Figures 6.46 to 6.48 are examples of how PEEB makes classification based on the descriptors in Stanford Dogs dataset. We demonstrate that our model works well on dogs, which indicates that our proposed method is transferable to other domains while maintaining high-quality explainability as in birds.

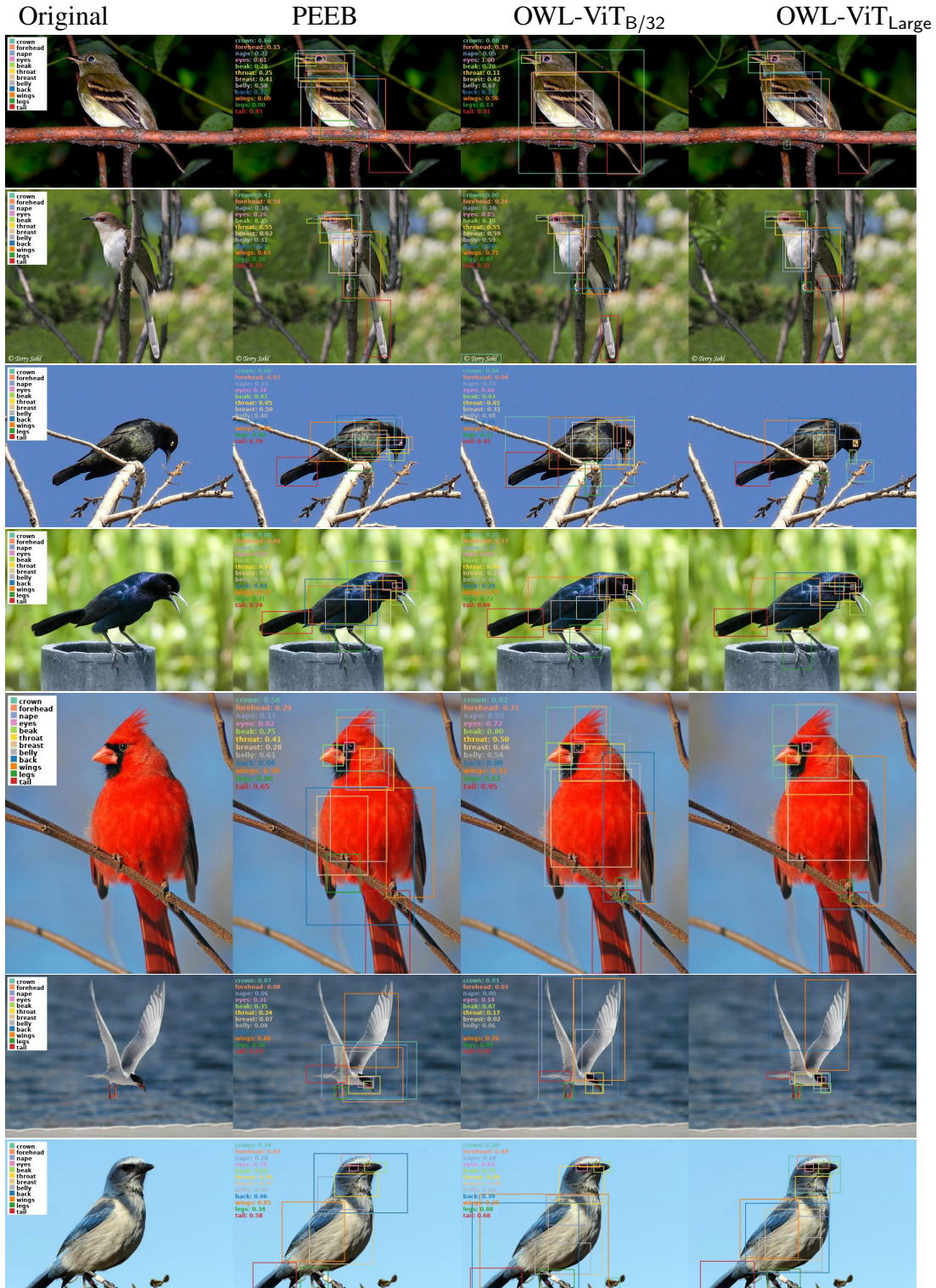


Figure 6.40: Our predicted boxes (second column) often align closely with those of OWL-ViT<sub>B/32</sub> (third column). However, slight shifts can lead to significant IoU discrepancies. For instance, in the first row, both PEEB and OWL-ViT<sub>B/32</sub> accurately identify the tail. Yet, variations in focus yield a stark IoU contrast of 0.45 versus 0.81.



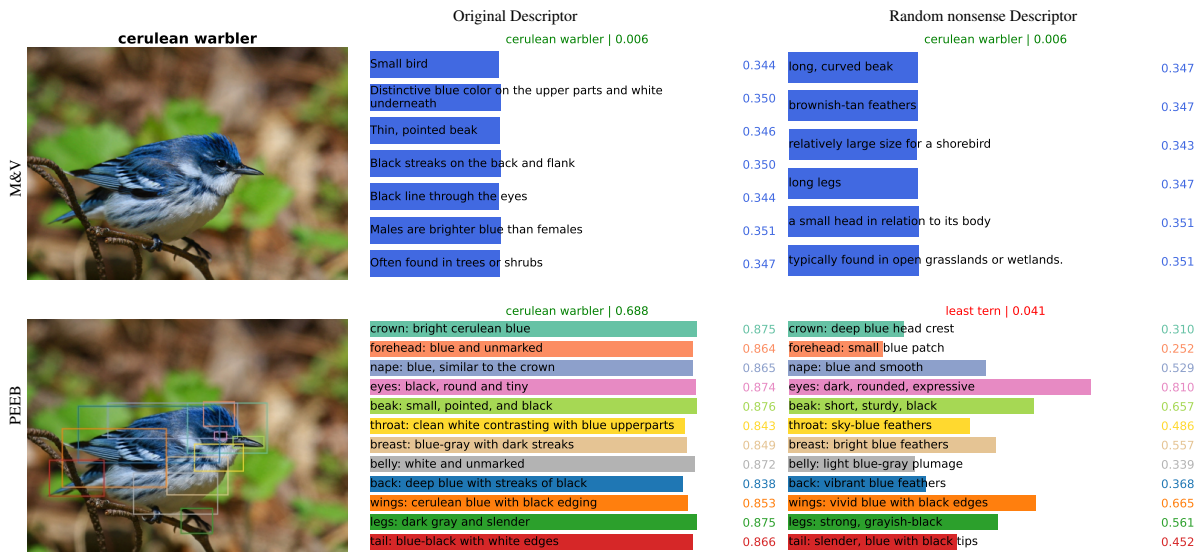


Figure 6.41: Qualitative example of original descriptors vs. randomized descriptors. Upon swapping descriptors randomly, the prediction outcomes from M&V exhibit minimal variations.



Figure 6.42: Qualitative example of original descriptors vs. randomized descriptors. Since PEEB's decision is made by the descriptors, the model will try to find the descriptors that best match the image. e.g., in the random descriptors, most parts are blue.



Figure 6.43: An example of PEEB explanation. We can see that the descriptors of these two classes are largely similar, but PEEB makes the correct prediction based on the distinctive feature of the forehead in the two classes.

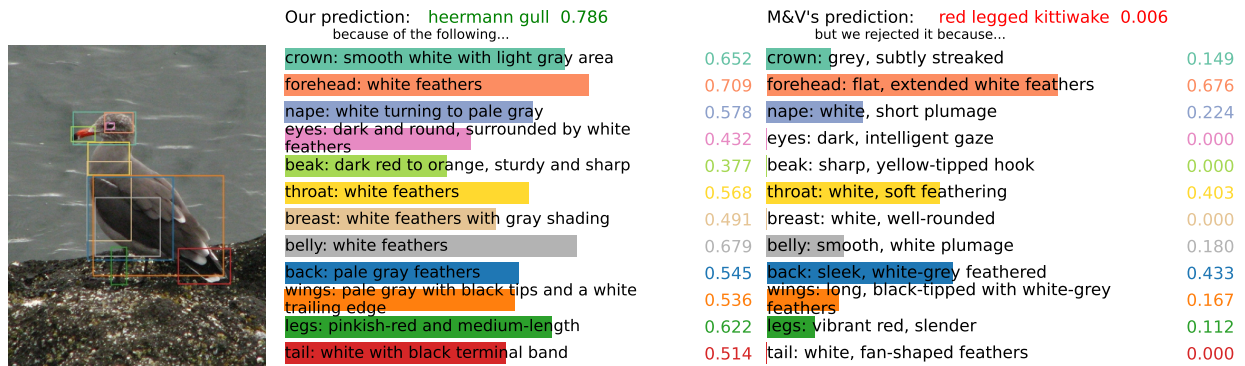


Figure 6.44: An example of PEEB explanation. M&V incorrectly classifies it as red-legged kittiwake where the heermann gull does not have red legs but a red beak. This example shows that CLIP is strongly biased towards some particular descriptors.

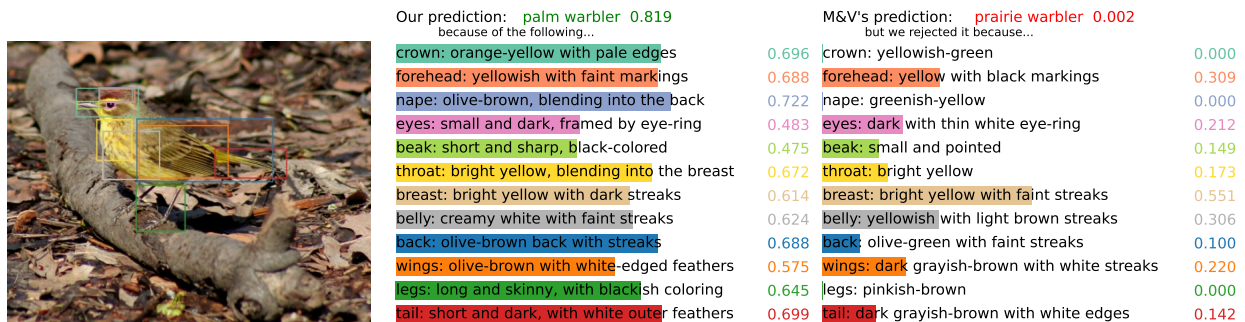


Figure 6.45: An example of PEEB explanation. We can see that when the descriptor does not match the image, the matching score tends to be zero, e.g., crown: yellowish-green. The clear differences in scores provide us transparency of the model's decision.

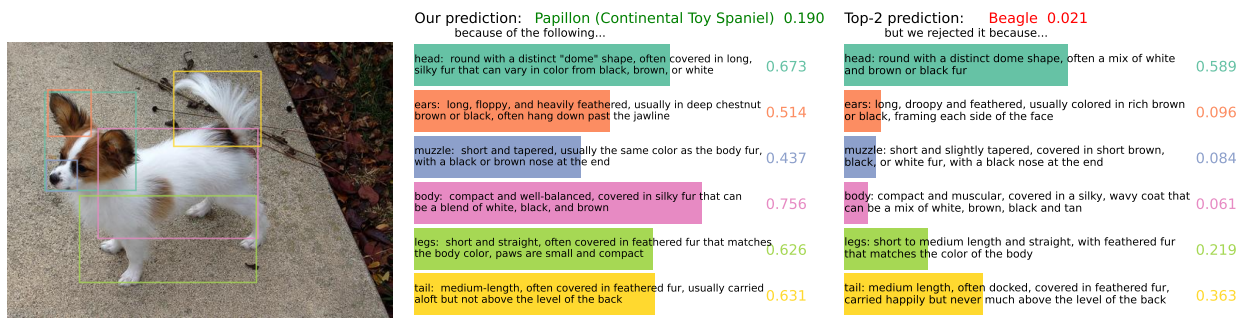


Figure 6.46: An example of PEEB explanation for dogs. Like birds, PEEB first identifies the predefined parts and then matches them to the descriptions.

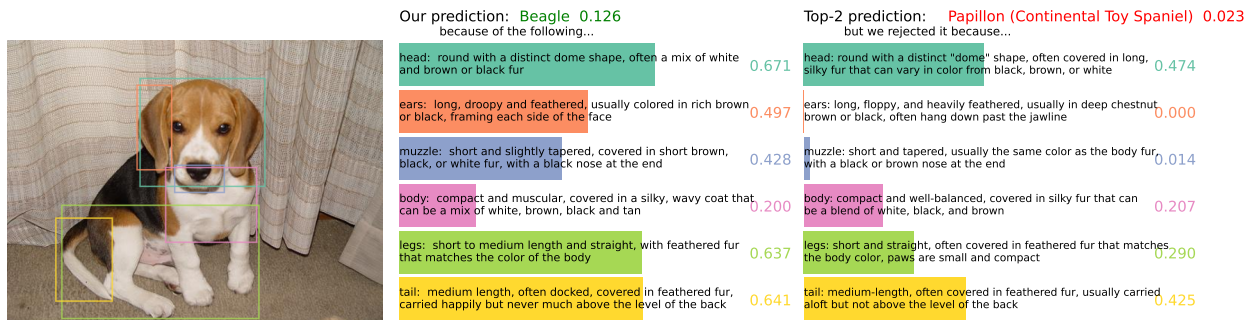


Figure 6.47: An example of PEEB explanation for dogs. Like birds, PEEB first identifies the predefined parts and then matches them to the descriptions.

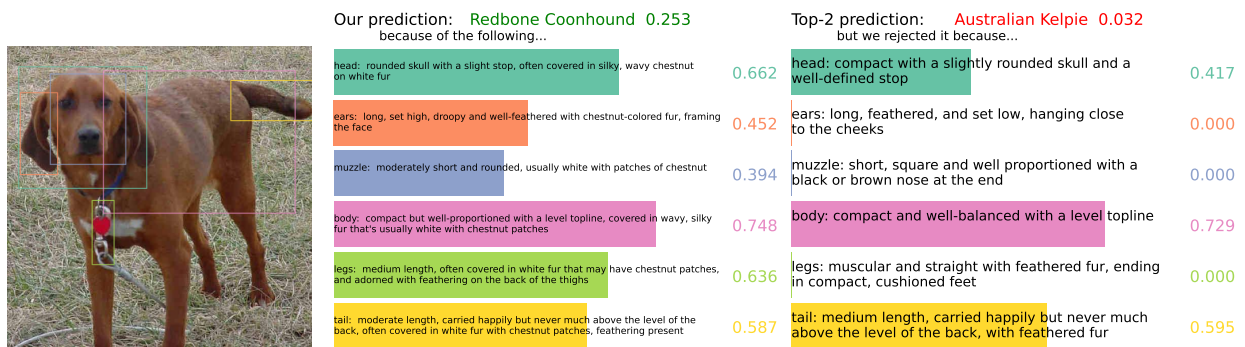


Figure 6.48: An example of PEEB explanation for dogs. Like birds, PEEB first identifies the predefined parts and then matches them to the descriptions.

## References

- [1] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31:9505–9515, 2018.
- [2] C. Agarwal and A. Nguyen. Explaining image classifiers by removing input features using generative models. In *Proceedings of the Asian Conference on Computer Vision*, 2020.
- [3] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1425–1438, 2015.
- [4] T. Arici, H. Ceker, and I. B. Tutar. Multi-span question answering using span-image network. *preprint*, 2020.
- [5] L. Arras, G. Montavon, K.-R. Müller, and W. Samek. Explaining recurrent neural network predictions in sentiment analysis. *EMNLP 2017*, page 159, 2017.
- [6] S. Asaadi, S. Mohammad, and S. Kiritchenko. Big BiRD: A large, fine-grained, bigram relatedness dataset for examining semantic composition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 505–516, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.
- [7] N. Bansal, C. Agarwal, and A. Nguyen. Sam: The sensitivity of attribution methods to hyperparameters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8673–8683, 2020.
- [8] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *ArXiv preprint*, abs/2004.05150, 2020.

- [9] A. Bera, Z. Wharton, Y. Liu, N. Bessis, and A. Behera. Sr-gnn: Spatial relation-aware graph neural network for fine-grained image categorization. *IEEE Transactions on Image Processing*, 31:6017–6031, 2022.
- [10] T. Berg, J. Liu, S. Woo Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur. Bird-snap: Large-scale fine-grained visual categorization of birds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2018, 2014.
- [11] M. Bevilacqua, T. Pasini, A. Raganato, R. Navigli, et al. Recent trends in word sense disambiguation: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. International Joint Conference on Artificial Intelligence, Inc, 2021.
- [12] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O’Reilly, Beijing, 2009.
- [13] T. Blevins, M. Joshi, and L. Zettlemoyer. FEWS: Large-scale, low-shot word sense disambiguation with the dictionary. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 455–465, Online, 2021. Association for Computational Linguistics.
- [14] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.
- [15] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, 2015. Association for Computational Linguistics.
- [16] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom. e-snli: Natural language inference with natural language explanations. In S. Bengio, H. Wallach, H. Larochelle,

- K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [17] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.
- [18] M. Carpuat and D. Wu. How phrase sense disambiguation outperforms word sense disambiguation for statistical machine translation. In *Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages: Papers*, Skövde, Sweden, 2007.
- [19] M. Carpuat and D. Wu. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72, Prague, Czech Republic, 2007. Association for Computational Linguistics.
- [20] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, Aug. 2017. Association for Computational Linguistics.
- [21] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *ArXiv preprint*, abs/1803.11175, 2018.
- [22] C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud. Explaining image classifiers by counterfactual generation. In *International Conference on Learning Representations*, 2019.
- [23] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019.

- [24] P. Chen, C. Agarwal, and A. Nguyen. The shape and simplicity biases of adversarially robust imagenet-trained cnns. *arXiv e-prints*, pages arXiv–2006, 2020.
- [25] Z. Chen, J. Li, Y. Luo, Z. Huang, and Y. Yang. Canzsl: Cycle-consistent adversarial networks for zero-shot learning from natural language. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 874–883, 2020.
- [26] I. Covert, S. Lundberg, and S.-I. Lee. Feature removal is a unifying principle for model explanation methods. *arXiv preprint arXiv:2011.03623*, 2020.
- [27] P. Dabkowski and Y. Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, pages 6967–6976, 2017.
- [28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.
- [31] J. DeYoung, S. Jain, N. F. Rajani, E. Lehman, C. Xiong, R. Socher, and B. C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th*

- Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online, July 2020. Association for Computational Linguistics.
- [32] W. B. Dolan and C. Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.
- [33] J. Donnelly, A. J. Barnett, and C. Chen. Deformable protopnet: An interpretable image classifier using deformable prototypes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10265–10275, 2022.
- [34] P. Edmonds and S. Cotton. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France, 2001. Association for Computational Linguistics.
- [35] M. Elhoseiny, Y. Zhu, H. Zhang, and A. Elgammal. Link the head to the "beak": Zero shot learning from noisy text description at part precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5640–5649, 2017.
- [36] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
- [37] R. Esfandiarpour and S. H. Bach. Follow-up differential descriptions: Language models resolve ambiguities for image classification. *arXiv preprint arXiv:2311.07593*, 2023.
- [38] K. Ethayarajh. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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 55–65, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.
- [39] A. Ettinger. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48, 2020.



- [40] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: the concept revisited. In V. Y. Shen, N. Saito, M. R. Lyu, and M. E. Zurko, editors, *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 406–414. ACM, 2001.
- [41] U. Fischer. Learning words from context and dictionaries: An experimental comparison. *Applied Psycholinguistics*, 15(4):551–574, 1994.
- [42] R. Fong, M. Patrick, and A. Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2950–2958, 2019.
- [43] J. Fu, H. Zheng, and T. Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [44] Fédération Cynologique Internationale (FCI). Nomenclature of the breeds recognised by the fci, 2023. Accessed: 2014-02-25.
- [45] T. Gao, X. Yao, and D. Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics.
- [46] L. Gatti, M. Guerini, and M. Turchi. Sentiwords: Deriving a high precision and high coverage lexicon for sentiment analysis. *IEEE Transactions on Affective Computing*, 7(4):409–421, 2015.
- [47] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. Iii, and K. Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

- [48] E. Gibson, L. Bergen, and S. T. Piantadosi. Rational integration of noisy evidence and prior semantic expectations in sentence interpretation. *Proceedings of the National Academy of Sciences*, 110(20):8051–8056, 2013.
- [49] GLUE leaderboard. Glue benchmark. <https://gluebenchmark.com/leaderboard>. (Accessed on 01/29/2021).
- [50] Y. Goyal, A. Feder, U. Shalit, and B. Kim. Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165*, 2019.
- [51] P. N. Group. princeton-nlp/sup-simcse-roberta-large · hugging face. <https://huggingface.co/princeton-nlp/sup-simcse-roberta-large>, 2022. (Accessed on 06/08/2022).
- [52] D. Gunning, E. Vorm, J. Y. Wang, and M. Turek. Darpa’s explainable ai (xai) program: A retrospective. *Applied AI Letters*, 2(4):e61, 2021.
- [53] C. Han, H. Pei, X. Du, and H. Ji. Zero-shot classification by logical reasoning on natural language explanations. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8967–8981, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [54] S. Han, L. Zhuo, Y. Liao, and S. Liu. Llms as visual explainers: Advancing image classification with evolving visual descriptions. *arXiv preprint arXiv:2311.11904*, 2023.
- [55] C. Hanouti and H. Le Borgne. Learning semantic ambiguities for zero-shot learning. *Multi-media Tools and Applications*, pages 1–15, 2023.
- [56] D. Harbecke and C. Alt. Considering likelihood in NLP classification explanations with occlusion and language modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 111–117, Online, July 2020. Association for Computational Linguistics.

- [57] P. Hase, H. Xie, and M. Bansal. The out-of-distribution problem in explainability and search methods for feature importance explanations. *Advances in Neural Information Processing Systems*, 34, 2021.
- [58] E. Hasler, F. Stahlberg, M. Tomalin, A. de Gispert, and B. Byrne. A comparison of neural models for word ordering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 208–212, Santiago de Compostela, Spain, Sept. 2017. Association for Computational Linguistics.
- [59] J. He, J.-N. Chen, S. Liu, A. Kortylewski, C. Yang, Y. Bai, and C. Wang. Transfg: A transformer architecture for fine-grained recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 852–860, 2022.
- [60] J. He, S. Yang, S. Yang, A. Kortylewski, X. Yuan, J.-N. Chen, S. Liu, C. Yang, Q. Yu, and A. Yuille. Partimagenet: A large, high-quality dataset of parts. In *European Conference on Computer Vision*, pages 128–145. Springer, 2022.
- [61] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [62] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- [63] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [64] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spaCy: Industrial-strength Natural Language Processing in Python. <https://spacy.io/>, 2020.

- [65] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim. A benchmark for interpretability methods in deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [66] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 168–177, New York, NY, USA, 2004. Association for Computing Machinery.
- [67] Huggingface. Pretrained models — transformers 3.3.0 documentation. [huggingface.co/transformers/pretrained\\_models.html](https://huggingface.co/transformers/pretrained_models.html), 2020. (Accessed on 09/30/2020).
- [68] Huggingface. super\_glue · datasets at hugging face. [https://huggingface.co/datasets/super\\_glue/viewer/wic](https://huggingface.co/datasets/super_glue/viewer/wic), 2022. (Accessed on 06/08/2022).
- [69] Huggingface. transformers/examples/pytorch/question-answering at main · huggingface/transformers. <https://github.com/huggingface/transformers/tree/main/examples/pytorch/question-answering>, 2022. (Accessed on 06/09/2022).
- [70] Z. Ji, Y. Fu, J. Guo, Y. Pang, Z. M. Zhang, et al. Stacked semantics-guided attention model for fine-grained zero-shot learning. *Advances in neural information processing systems*, 31, 2018.
- [71] X. Jin, Z. Wei, J. Du, X. Xue, and X. Ren. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. In *International Conference on Learning Representations*, 2020.
- [72] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

- [73] U. Khandelwal, H. He, P. Qi, and D. Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Melbourne, Australia, July 2018. Association for Computational Linguistics.
- [74] D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.
- [75] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*, volume 2. Citeseer, 2011.
- [76] S. Kim, J. Nam, and B. C. Ko. Vit-net: Interpretable vision transformers with neural tree decoder. In *International Conference on Machine Learning*, pages 11162–11172. PMLR, 2022.
- [77] S. Kim, J. Nam, and B. C. Ko. ViT-NeT: Interpretable vision transformers with neural tree decoder. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11162–11172. PMLR, 17–23 Jul 2022.
- [78] S. Kim, J. Yi, E. Kim, and S. Yoon. Interpretation of NLP models through input marginalization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3154–3167, Online, Nov. 2020. Association for Computational Linguistics.

- [79] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.*
- [80] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang. Concept bottleneck models. In *International conference on machine learning*, pages 5338–5348. PMLR, 2020.
- [81] S. Kousha and M. A. Brubaker. Zero-shot learning with class description regularization. *arXiv preprint arXiv:2106.16108*, 2021.
- [82] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.
- [83] V. Lai and C. Tan. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 29–38, 2019.
- [84] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.
- [85] LanguageTool. Languagetool - online grammar, style & spell checker. <https://languagetool.org/>, 2022. (Accessed on 06/08/2022).
- [86] J. Lee, M. Sung, J. Kang, and D. Chen. Learning dense representations of phrases at scale. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6634–6647, Online, 2021. Association for Computational Linguistics.

- [87] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark, 2017. Association for Computational Linguistics.
- [88] H. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer, 2012.
- [89] J. Li, W. Monroe, and D. Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.
- [90] Y. Lin, Y. C. Tan, and R. Frank. Open sesame: Getting inside bert’s linguistic knowledge. *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019.
- [91] F. Liu and B. Avci. Incorporating priors with feature attribution on text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6274–6283, Florence, Italy, July 2019. Association for Computational Linguistics.
- [92] X. Liu, T. Xia, J. Wang, Y. Yang, F. Zhou, and Y. Lin. Fully convolutional attention networks for fine-grained recognition. *arXiv: Computer Vision and Pattern Recognition*, 2016.
- [93] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [94] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.
- [95] S. Menon and C. Vondrick. Visual classification via description from large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

- [96] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.
- [97] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, and N. Houlsby. Simple open-vocabulary object detection with vision transformers. *ECCV*, 2022.
- [98] J. Mitchell and J. Bowers. Priorless recurrent networks learn curiously. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5147–5158, 2020.
- [99] F. Mollica, M. Siegelman, E. Diachek, S. T. Piantadosi, Z. Mineroff, R. Futrell, H. Kean, P. Qian, and E. Fedorenko. Composition is the core driver of the language-selective network. *Neurobiology of Language*, 1(1):104–134, 2020.
- [100] W. E. Nagy, P. A. Herman, and R. C. Anderson. Learning words from context. *Reading research quarterly*, pages 233–253, 1985.
- [101] M. Nauta, A. Jutte, J. Provoost, and C. Seifert. This looks like that, because... explaining prototypes for interpretable image recognition. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2021, Virtual Event, September 13-17, 2021, Proceedings, Part I*, pages 441–456. Springer, 2022.
- [102] M. Nauta, R. Van Bree, and C. Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14933–14943, 2021.
- [103] G. Nguyen, M. R. Taesiri, S. S. Kim, and A. Nguyen. Allowing humans to interactively guide machines where to look does not always improve a human-ai team’s classification accuracy. *arXiv preprint arXiv:2404.05238*, 2024.



- [104] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online, July 2020. Association for Computational Linguistics.
- [105] NLTK. `nltk.metrics.distance` — nltk 3.5 documentation. [https://www.nltk.org/\\_modules/nltk/metrics/distance.html](https://www.nltk.org/_modules/nltk/metrics/distance.html). (Accessed on 12/25/2020).
- [106] T. Oikarinen, S. Das, L. M. Nguyen, and T.-W. Weng. Label-free concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023.
- [107] OpenAI. Gpt-4 technical report, 2023.
- [108] K. P. Panousis, D. Ienco, and D. Marcos. Hierarchical concept discovery models: A concept pyramid scheme. *arXiv preprint arXiv:2310.02116*, 2023.
- [109] S. Parekh, Y. K. Singla, C. Chen, J. J. Li, and R. R. Shah. My teacher thinks the world is flat! interpreting automatic essay scoring mechanism. *arXiv preprint arXiv:2012.13872*, 2020.
- [110] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, Nov. 2016. Association for Computational Linguistics.
- [111] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

- [112] E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China, 2015. Association for Computational Linguistics.
- [113] T. Paz-Argaman, R. Tsarfaty, G. Chechik, and Y. Atzmon. ZEST: Zero-shot learning from text descriptions using textual similarity and visual summarization. In T. Cohn, Y. He, and Y. Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 569–579, Online, Nov. 2020. Association for Computational Linguistics.
- [114] J. Pearl. *Causality*. Cambridge university press, 2009.
- [115] T. Pham, T. Bui, L. Mai, and A. Nguyen. Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1145–1160, Online, Aug. 2021. Association for Computational Linguistics.
- [116] T. Pham, T. Bui, L. Mai, and A. Nguyen. Double trouble: How to not explain a text classifier’s decisions using counterfactuals synthesized by masked language models? In Y. He, H. Ji, S. Li, Y. Liu, and C.-H. Chang, editors, *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 (Volume 1: Long Papers)*, pages 12–31, Online only, Nov. 2022. Association for Computational Linguistics.
- [117] T. Pham, S. Yoon, T. Bui, and A. Nguyen. PiC: A phrase-in-context dataset for phrase understanding and semantic search. In A. Vlachos and I. Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational*

- Linguistics*, pages 1–26, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.
- [118] T. M. Pham, P. Chen, T. Nguyen, S. Yoon, T. Bui, and A. Nguyen. Peeb: Part-based image classifiers with an explainable and editable language bottleneck. June 2024.
- [119] PiC. upwork\_annotation\_guidelines.pdf. <https://drive.google.com/file/d/1UsWXvJLzWzuhuRuaCsla3Ljcxp-vM2KK/view?usp=sharing>, 2021. (Accessed on 06/08/2022).
- [120] PiC. upwork\_samples.pdf. [https://drive.google.com/file/d/1ume4QeQvEPlwajyeNIQjdH\\_cCI9w3P6f/view?usp=sharing](https://drive.google.com/file/d/1ume4QeQvEPlwajyeNIQjdH_cCI9w3P6f/view?usp=sharing), 2021. (Accessed on 06/08/2022).
- [121] PiC. upwork\_samples.pdf. [https://drive.google.com/file/d/1Lplc\\_cRla\\_hbDFzLiJ7Zn06G0GnRq1gc/view?usp=sharing](https://drive.google.com/file/d/1Lplc_cRla_hbDFzLiJ7Zn06G0GnRq1gc/view?usp=sharing), 2022. (Accessed on 06/08/2022).
- [122] M. T. Pilehvar and J. Camacho-Collados. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.
- [123] G. Piosenka. Birds 525 - species image classification. 02 2022.
- [124] S. Pratt, I. Covert, R. Liu, and A. Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15691–15701, 2023.
- [125] Quora. (1) first quora dataset release: Question pairs - data @ quora - quora. <https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs>, 2017. (Accessed on 09/30/2020).

- [126] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [127] P. Rajpurkar, R. Jia, and P. Liang. Know what you don’t know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics.
- [128] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, Nov. 2016. Association for Computational Linguistics.
- [129] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics.
- [130] Y. Rao, Z. Yang, S. Zeng, Q. Wang, and J. Pu. Dual projective zero-shot learning using text descriptions. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(1):1–17, 2023.
- [131] N. Reimers and I. Gurevych. 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, Hong Kong, China, 2019. Association for Computational Linguistics.

- [132] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [133] Ribeiro. marcotcr/lime: Lime: Explaining the predictions of any machine learning classifier. <https://github.com/marcotcr/lime>, 2021. (Accessed on 05/17/2021).
- [134] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.
- [135] M. Robnik-Šikonja and I. Kononenko. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):589–600, 2008.
- [136] K. Roth, J. M. Kim, A. Koepke, O. Vinyals, C. Schmid, and Z. Akata. Waffling around for performance: Visual classification with random words and broad concepts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15746–15757, 2023.
- [137] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.
- [138] D. Samuel, Y. Atzmon, and G. Chechik. From generalized zero-shot learning to long-tail with class descriptors. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 286–295, 2021.
- [139] C. Sankar, S. Subramanian, C. Pal, S. Chandar, and Y. Bengio. Do neural dialog systems use the conversation history effectively? an empirical study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 32–37, Florence, Italy, July 2019. Association for Computational Linguistics.

- [140] A. Schmaltz, A. M. Rush, and S. Shieber. Word ordering without syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2319–2324, Austin, Texas, nov 2016. Association for Computational Linguistics.
- [141] C. Si, S. Wang, M.-Y. Kan, and J. Jiang. What does bert learn from multiple-choice reading comprehension datasets? *arXiv preprint arXiv:1910.12391*, 2019.
- [142] K. Sinha, P. Parthasarathi, J. Pineau, and A. Williams. Unnatural language inference. *arXiv preprint arXiv:2101.00010*, 2020.
- [143] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics.
- [144] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.
- [145] S. Sugawara, P. Stenetorp, K. Inui, and A. Aizawa. Assessing the benchmarking capacity of machine reading comprehension datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8918–8927, 2020.
- [146] W. Team. Api:categories - mediawiki. <https://www.mediawiki.org/wiki/API:Categories>, 2021. (Accessed on 06/08/2022).
- [147] W. Team. Wikimedia downloads. <https://dumps.wikimedia.org/>, 2021. (Downloaded on November 1st, 2021).

- [148] TensorFlow. Universal sentence encoder | tensorflow hub. [https://www.tensorflow.org/hub/tutorials/semantic\\_similarity\\_with\\_tf\\_hub\\_universal\\_encoder](https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder), 2022. (Accessed on 08/11/2022).
- [149] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.
- [150] P. D. Turney. Domain and function: A dual-space model of semantic relations and compositions. *Journal of artificial intelligence research*, 44:533–585, 2012.
- [151] P. Vaibhav Rokde, Matthew Jansen. Indian birds species image classification, 2023. Dataset originally sourced from eBird, Cornell Lab of Ornithology. <https://media.ebird.org/>.
- [152] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 595–604, 2015.
- [153] G. Van Horn, E. Cole, S. Beery, K. Wilber, S. Belongie, and O. Mac Aodha. Benchmarking representation learning for natural world image collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12884–12893, 2021.
- [154] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.
- [155] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

- [156] J. Wang, H. Liu, X. Wang, and L. Jing. Interpretable image recognition by constructing transparent embedding space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 895–904, 2021.
- [157] S. Wang, L. Thompson, and M. Iyyer. Phrase-BERT: Improved phrase embeddings from BERT with an application to corpus exploration. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10837–10851, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics.
- [158] W. Wang, B. Bi, M. Yan, C. Wu, J. Xia, Z. Bao, L. Peng, and L. Si. Structbert: Incorporating language structures into pre-training for deep language understanding. In *International Conference on Learning Representations*, 2020.
- [159] A. Warstadt, A. Singh, and S. R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, Mar. 2019.
- [160] S. Wiegrefe and A. Marasović. Teach me to explain: A review of datasets for explainable nlp. *arXiv preprint arXiv:2102.12060*, 2021.
- [161] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358, 2015.
- [162] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics.



- [163] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.
- [164] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, 2019.
- [165] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010.
- [166] W. Xu, Y. Xian, J. Wang, B. Schiele, and Z. Akata. Attribute prototype network for zero-shot learning. *Advances in Neural Information Processing Systems*, 33:21969–21980, 2020.
- [167] M. Xue, Q. Huang, H. Zhang, L. Cheng, J. Song, M. Wu, and M. Song. Protopformer: Concentrating on prototypical parts in vision transformers for interpretable image recognition. *arXiv preprint arXiv:2208.10431*, 2022.
- [168] Yahoo. Webscope | yahoo labs. <https://webscope.sandbox.yahoo.com/catalog.php?datatype=l&did=66>, 2022. (Accessed on 08/10/2022).
- [169] Yahoo. Webscope | yahoo labs. <https://webscope.sandbox.yahoo.com/catalog.php?datatype=l&did=66>, 2022. (Accessed on 08/10/2022).
- [170] A. Yan, Y. Wang, Y. Zhong, C. Dong, Z. He, Y. Lu, W. Y. Wang, J. Shang, and J. McAuley. Learning concise and descriptive attributes for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3090–3100, 2023.
- [171] R. Yang. `arcface-pytorch/test.py` at master · ronghuaiyang/arcface-pytorch. <https://github.com/ronghuaiyang/arcface-pytorch/blob/master/test.py>, 2022. (Accessed on 06/09/2022).

- [172] Y. Yang, A. Panagopoulou, S. Zhou, D. Jin, C. Callison-Burch, and M. Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19187–19197, 2023.
- [173] Y. Yang, Y. Zhang, C. Tar, and J. Baldridge. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. 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 3687–3692, Hong Kong, China, 2019. Association for Computational Linguistics.
- [174] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics.
- [175] L. Yu and A. Ettinger. Assessing phrasal representation and composition in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4896–4907, Online, 2020. Association for Computational Linguistics.
- [176] M. Yuksekgonul, M. Wang, and J. Zou. Post-hoc concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023.
- [177] F. M. Zanzotto, A. Santilli, L. Ranaldi, D. Onorati, P. Tommasino, and F. Fallucchi. KERMIT: Complementing transformer architectures with encoders of explicit syntactic interpretations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 256–267, Online, Nov. 2020. Association for Computational Linguistics.
- [178] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

- [179] T. Zhang and Z. Zhu. Interpreting adversarially trained convolutional neural networks. In *International Conference in Machine Learning*, 2019.
- [180] Y. Zhang, J. Baldridge, and L. He. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.
- [181] Y. Zhang and S. Clark. Discriminative syntax-based word ordering for text generation. *Computational linguistics*, 41(3):503–538, 2015.
- [182] Y. Zhang, K. Song, Y. Sun, S. Tan, and M. Udell. " why should you trust my explanation?" understanding uncertainty in lime explanations. *arXiv preprint arXiv:1904.12991*, 2019.
- [183] M. Zheng, P. Gao, R. Zhang, K. Li, X. Wang, H. Li, and H. Dong. End-to-end object detection with adaptive clustering transformer. 2021.
- [184] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.
- [185] Y. Zhu, M. Elhoseiny, B. Liu, X. Peng, and A. Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1004–1013, 2018.