Filtering Out Unreliable Language Model Outputs Using Contrast-Consistent Search

Stanford CS 224N Custom Project

Michael Byun
Department of Computer Science
Stanford University
mbyun@stanford.edu

Mauricio Baker
Department of Computer Science
Stanford University
maubaker@stanford.edu

Abstract

Unreliable outputs are a major barrier to the use of deep learning-based NLP systems, motivating methods which can evaluate and improve the truthfulness of generated text. One such proposed method is Contrast-Consistent Search (CCS). This unsupervised method aims to extract a language model's implicit knowledge of the truth value of a text sequence by searching for simple ways in which the model's activations represent consistent probabilities. We build on CCS to augment a pretrained large language model on zero-shot sentiment analysis tasks, improving accuracy. Then, we develop and test seven CCS-based methods for classifying a model's answers to questions as reliable or unreliable. We find that, in our two evaluation datasets, CCS-based methods enable substantially more precise identification of unreliable outputs.

1 Key Information

Project TA/Mentor: Christopher Cross

We collaborated with Peter Chatain, who switched to another project.

2 Introduction

The advances in capabilities and increasing deployment of deep learning-based natural language processing systems present many exciting opportunities, but a key concern is their unreliability. This issue is particularly salient for large language models (LLMs), which often output untruthful text by reproducing common misconceptions, agreeing with false prompts, "hallucinating" facts, or other means. The language modeling objective incentivizes such behavior; sometimes, a model can more accurately predict human text (or receive more positive feedback in RLHF (et al., 2022)) by generating untruthful outputs. In some situations, larger language models output untruthful answers more often than smaller models (Lin et al., 2022).

These issues have motivated the development of methods to improve truthfulness and reliability of LLMs, including popular methods such as reinforcement learning from human feedback (RLHF) (Christiano et al., 2017). In this paper, we study another such method, proposed by Burns et al. (2022): Contrast-Consistent Search (CCS). CCS assumes that the activations of a model's hidden states represent implicit knowledge of the truth value of a text sequence being evaluated by that model, and searches for a linear projection of hidden states which is consistent across negations. (We describe the method in more detail in Section 4.) Since this method is unsupervised, it is relatively scalable, and it may be able to detect truthfulness even when a model's training objective disincentivizes true outputs (e.g., in language modeling, reproducing a common misconception is rewarded).

Existing work on CCS has primarily studied its potential to produce true answers. While we consider that critical, in this paper we focus on another application: how metrics derived from CCS can be

used to filter answers from a language model (i.e. filtering out unreliable answers). In other words, we aim to use CCS to classify answers as reliable or unreliable. Such classification can help users know when it is safer to rely on some language model answer, and when it is not.

We first apply CCS to a large language model carrying out sentiment analysis tasks on the IMDB and AMAZON_POLARITY datasets, finding that this improves accuracy. Then, we construct and apply seven methods of filtering answers on this task, filtering answers based on functions of corresponding model outputs and CCS outputs. Evaluating these filters, we find that the best CCS-based filters (ones that incorporate CCS outputs) are much more successful than all baseline methods at precisely filtering out unreliable outputs; for a given level of accuracy, they can achieve that level of accuracy in filtered outputs, without filtering out as many answers.

3 Related Work

Burns et al. (2022) propose Contrast-Consistent Search (CCS), an unsupervised method for finding a model's implicit assignment of truth value to statements. They emphasize the usefulness of unsupervised methods to avoid the costs (and potential unreliability, for complicated questions) of human labeling. The authors describe experimental results which suggest CCS has some success in deriving models' representations of truth. CCS, applied to certain language models and datasets, achieves higher question-answering accuracy than using models' outputs directly.

Roger (2023) examines limitations of CCS as proposed by Burns et al. (2022). They find, among other results, that CCS only finds one of many orthogonal linear projections which get similar accuracy, and that it performs poorly on GPT models.

Fine-tuning and reinforcement learning from human feedback are additional approaches that have been used in efforts to improve language models' truthfulness. In contrast to CCS, these approaches assume models will generalize truthfully from supervision, while naively prompting language models to elicit their knowledge often assumes that models will generalize truthfully from text prediction.

4 Approach

4.1 Contrast-Consistent Search (CCS)

We took as our starting point Contrast-Consistent Search (CCS), an unsupervised method for extracting latent knowledge from language models (Burns et al., 2022). CCS aims to extract a language model's latent knowledge by identifying simple representations of truth in a model's activations. To identify these representations, CCS leverages a simple property these representations should have. To illustrate this property by example, if a model's representation for (e.g.) the statement "Does the sun rise in the East? Yes." corresponds to 1 (true), then the model's representation for "Does the sun rise in the East? No." should correspond to 0 (false). More precisely, CCS involves the following steps:

1. For some dataset that consists of text questions, append "Yes" and "No" (or other binary answers) to a copy of each question.
   (a) The resulting pair of questions with answers is known as a "contrastive pair."
2. Map a model's hidden states in processing each question to a probability (intended to correspond to the model's truth assignments). The authors did this by taking a linear projection of hidden states and then applying a sigmoid function.
3. To make the above map correspond to the model's truth assignments, train the parameters of the projection so that the above "probabilities" score highly according to the following two loss function components:
   - Consistency: The model assigns opposite probabilities to opposite statements.
   - Confidence: The model assigns probabilities that are far from 0.5.

Our implementation of CCS is based on the code provided in Burns et al. (2022). We modified their code in a number of ways: among other changes, we modified the data loading and processing pipeline to correct inconsistencies, generalize to new datasets, and improve performance; we modularized the code; we added functionality that compares the model-assigned log probability of the statements in

each contrastive pair; and we added functionality to collect and write the ground truth, CCS label, and inferred model output associated with each dataset example.

4.2 Acquiring question features

For each question in (large subsets of) our (labeled) datasets, we identified the following features:

- The ground truth label: $y_i \in \{0, 1\}$, where 0 and 1 are the two possible binary answers to the question.
- The model's output in response to the question paired with a positive answer: $\hat{y}_{M,i,1} \in \mathbb{R}$.
- The model's output in response to the question paired with a negative answer: $\hat{y}_{M,i,0} \in \mathbb{R}$.
- The model's answer to the question, which corresponds to whichever of the above two outputs is greater: $\hat{y}_{M,i} = \text{argmax}_{j \in \{0,1\}} (\hat{y}_{M,i,j}) \in \{0, 1\}$.
- The output of CCS in response to the question paired with a positive answer: $\hat{y}_{C,i,1} \in \mathbb{R}$.
- The output of CCS in response to the question paired with a negative answer: $\hat{y}_{C,i,0} \in \mathbb{R}$.
- The CCS answer to the question, which corresponds to whichever of the above two outputs is greater: $\hat{y}_{C,i} = \text{argmax}_{j \in \{0,1\}} (\hat{y}_{C,i,j}) \in \{0, 1\}$.

4.3 Task definition

We propose and adopt the following framework. Our task is to identify functions that do well at distinguishing between reliable and unreliable answers, on the basis of the features defined above (except the ground truth). In other words, we seek a function $f : (\mathbb{R},\ \mathbb{R},\ \{0, 1\},\ \mathbb{R},\ \mathbb{R},\ \{0, 1\}) \to \{0, 1\}$ that maximizes the number of questions (i's) for which the following holds:

$$f(\hat{y}_{M,i,0},\ \hat{y}_{M,i,1},\ \hat{y}_{M,i},\ \hat{y}_{C,i,0},\ \hat{y}_{C,i,1},\ \hat{y}_{C,i}) = \mathbb{1}[\hat{y}_{A,i} = y_i],$$

where A is either C or M. That is, we may choose whether to look for accurate model answers or accurate CCS answers. This is appropriate because real-world, ML-based question-answering systems can report either of these answers to users; what matters here is that—however these systems reach the answer they report—they can determine (and thus communicate) its reliability.

We are particularly concerned with finding functions with low false positive rates (i.e. functions such that, when they imply some answer is correct, the answer is very rarely wrong). Because of this, we interpret these functions as classifiers of answer reliability; it is relatively acceptable if an answer classified as "unreliable" is right, but it is much more problematic if an answer classified as "reliable" is wrong. We call these functions "filters," since they can be applied to sets of answers to filter out unreliable answers.

4.4 Candidate filters

We tested seven (classes of) candidate filters, in order to identify our better candidates. Each filter was applied to a large set of questions from each dataset. We assumed these filters classify CCS answers (that is, we set $A = C$), since we found better performance in this setting. Most filters have a tunable parameter *t*; for each class of filters, we tested 100 different, uniformly distributed values of *t*. We tested the following filters:

1. **Max CCS:** Classify an answer as reliable if the largest CCS output is greater than *t*.
   - That is, let $f_{1,t}(\cdot) = \mathbb{1}[max\{\hat{y}_{C,i,0},\ \hat{y}_{C,i,1}\} > t]$.
2. **CCS & model agreement:** Classify an answer as reliable if the model answer is the same as the CCS answer.
   - That is, let $f_2(\cdot) = \mathbb{1}[\hat{y}_{M,i} = \hat{y}_{C,i}]$.
3. **Max CCS and CCS & model agreement:** Classify an answer as reliable if it passes both of the above filters (for some value of *t*).
   - That is, let $f_{3,t}(\cdot) = \mathbb{1}[f_{1,t}(\cdot) = 1 = f_2(\cdot)]$.

4. **CCS difference:** Classify an answer as reliable if the difference between CCS values is larger than some parameter *t*.
   - That is, let $f_{4,t}(\cdot) = \mathbb{1}[|\hat{y}_{C,i,0} - \hat{y}_{C,i,1}| > t]$.
5. **CCS difference and CCS & model agreement** Classify an answer as reliable if it passes both the "CCS difference filter" (for some *t*) and the "CCS & model agreement" filter.
   - That is, let $f_{5,t}(\cdot) = \mathbb{1}[f_{4,t}(\cdot) = 1 = f_2(\cdot)]$.
6. **Logits difference and CCS & model agreement:** Classify an answer as reliable if the difference of the logits is greater than some parameter *t* and it passes the "CCS & model agreement" filter.
   - That is, let $f_{6,t}(\cdot) = \mathbb{1}[|\hat{y}_{M,i,0} - \hat{y}_{M,i,1}| > t \ \wedge \ f_2(\cdot) = 1]$.
7. **CCS answer:** Simply classify all CCS answers as reliable.
   - That is, let $f_7(\cdot) = 1$.

4.5 Baseline filters

The above filters all rely (partially or completely) on CCS outputs. As baselines, we test three filters that do not rely at all on CCS outputs, in the baseline context of identifying questions that a model answers reliably:

- **Max logit:** Classify an answer as reliable if the largest logit output is greater than some parameter *t*.
  - That is, let $f_{8,t}(\cdot) = \mathbb{1}[max\{\hat{y}_{M,i,0}, \ \hat{y}_{M,i,1}\} > t]$.
- **Logits difference:** Classify an answer as reliable if the difference between CCS values is larger than some parameter *t*.
  - That is, let $f_{9,t}(\cdot) = \mathbb{1}[|\hat{y}_{M,i,0} - \hat{y}_{M,i,1}| > t]$.
- **Model answer:** Simply classify all model answers as reliable.
  - That is, let $f_{10}(\cdot) = 1$.

For the baselines, we assume CCS is not used, so we treat the baseline filters as classifiers of model outputs (i.e. we set $A = M$).

5 Experiments

5.1 Data

We use the IMDB (Maas et al., 2011) and AMAZON_POLARITY (McAuley and Lescovec, 2013) datasets to construct contrastive pairs of prompts, each of which is a sentiment analysis question-answer pair. For each dataset, we use a number of different prompt templates. For example, two prompt templates for the IMDB dataset are:

- Consider the following example: ”’ <review> ”’ Between negative and positive, the sentiment of this example is [negative, positive]
- <review> Did the reviewer enjoy the movie? [No, Yes]

Each contrastive pair has one correct prompt and one incorrect prompt. For example, the following would be correct and incorrect prompts, respectively:

- I loved this movie! Did the reviewer enjoy the movie? [SEP] Yes
- I loved this movie! Did the reviewer enjoy the movie? [SEP] No

In a similar fashion, but with topic classification in place of sentiment analysis, we constructed contrastive pairs from the DBPEDIA_14 (Lehmann, 2015) and AG_NEWS (Zhang et al., 2015) datasets, which we used for training but not evaluating CCS. We used 13 different prompt templates for IMDB, 11 for AMAZON_POLARITY, 8 each for DBPEDIA and AG_NEWS, and 1000 examples per prompt template from the respective dataset, for a total of 40,000 examples.

5.2 Evaluation method

We evaluated each filter by examining its false positive rate (i.e. 1 minus the accuracy of the filtered examples), as well as the rate at which it outputs positives (i.e. accepts answers). The motivation for this is that a good filter ought to classify many right answers as "reliable" (i.e. have a high positive rate), without classifying too many wrong answers as "reliable."

Specifically, for each of our two evaluation datasets and each class of filters, we asked: "Across values of the filter's parameter t, what is the largest fraction of answers that this filter classifies as reliable, while maintaining an accuracy rate of $n\%$ (90%, 95%, or 99%) among the answers it labels as reliable (i.e. maintaining a false positive rate under 10%, 5%, or 1%)?" We determined accuracy by comparing the answer to the ground truth (i.e. $\mathbb{1}[\hat{y}_{A,i} = y_i]$). A few of our filters (e.g. "CCS answers") do not have an adjustable parameter. For these, we evaluated their (constant) false positive rate and positive rate.

For evaluating each filter, we used 4,000 examples from the IMDB and AMAZON_POLARITY datasets. To evaluate each class of filters with a parameter t, we evaluated the filter at 100 possible values of t. These values were uniformly distributed over the range of values that produce variation in the filter's behavior (usually 0 to 1).

5.3 Experimental details

As our base model, we used RoBERTa Large MNLI (Liu et al., 2019), which is the RoBERTa large model fine-tuned on the Multi-Genre Natural Language Inference (MNLI) corpus. For each contrastive pair:

- We apply the model to each prompt in the contrastive pair, and infer which prompt the model considers more likely by getting the logits. This is equivalent to a (zero-shot) fill-mask task, where the answer is masked and two possible answers are given as targets.
- We apply the model to each prompt in the contrastive pair, and save the activations of the last layer in order to either train or run CCS on that example.

We then do a randomized 60-40 train-test split for each prompt template and each dataset. We train CCS for 1000 epochs using the AdamW optimizer and a learning rate of 0.01, repeated with 10 random initializations and saving the best-performing CCS model. We then evaluate CCS on the test data, and compare it on a per-example basis with the zero-shot model outputs, as described in section 4.

Our code can be found at `https://github.com/michaelbyun/Exhaustive-CCS`.

5.4 Results

For each of our two evaluation datasets and each of three accuracy rates (90%, 95%, and 99%, defined as the proportion of the filtered examples which are correct, or equivalently the true positive rate of the filter), the following tables report the highest fraction of examples that each filter was able to accept, while maintaining the given accuracy. This can be interpreted as the frequency with which the filter can provide a given level of assurance. Additionally, the table lists the accuracy of our two filters that accept all inputs: the "Model answer" and "CCS answer" filters.

In the below tables, "Agreement" is short for "CCS & model agreement," defined in the earlier section on candidate filters.

6 Analysis

Our best-performing filters have strictly (and significantly) better coverage than our baseline methods at the accuracy levels we examined; that is, they reject far fewer examples as "unreliable" while maintaining the same level of accuracy. Those best-performing filters are the ones based on CCS difference. We hypothesize that this is because the CCS difference metric is the best at encoding the "confidence" of CCS, since it directly compares the CCS outputs for the negative and positive prompts in a contrastive pair.

Accuracy	0.835	0.875	0.90	0.95	0.99
(Baseline) Model answer	1				
(Baseline) Max logit			0.591	0.274	0.026
(Baseline) Logits difference			0.771	0.369	0.076
Max CCS			0.897	0.691	0.249
CCS & model agreement			0.890	N/A	N/A
Max CCS and CCS & model agreement			0.889	0.645	0.245
CCS difference			**0.921**	0.628	**0.290**
CCS difference and CCS & model agreement			0.890	**0.699**	0.289
Logits difference and CCS & model agreement			0.890	0.368	0.076
CCS answer		1			

Table 1: The largest portion of examples that each filter can accept while maintaining a given accuracy level (IMDB dataset)

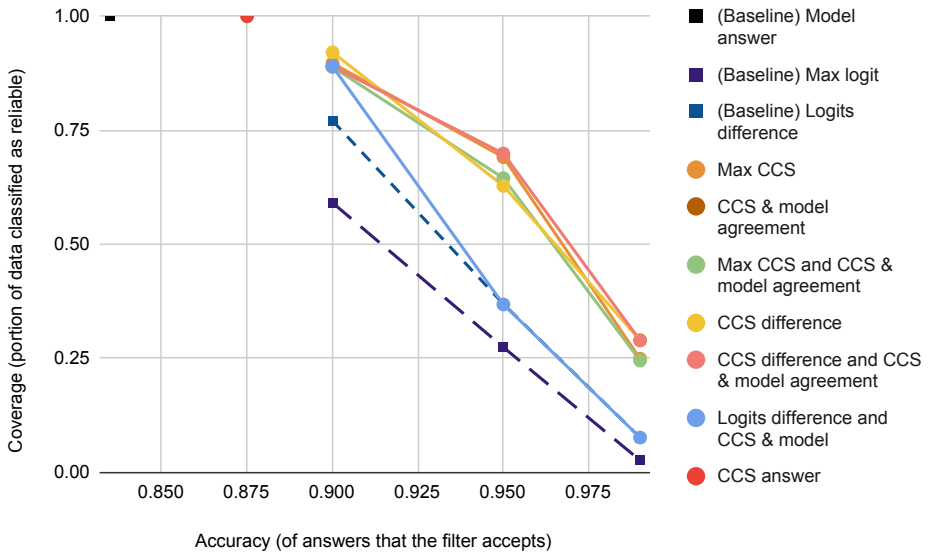

Figure 1: The largest portion of examples that each filter can accept while maintaining a given accuracy level (IMDB dataset).

Compared to baseline methods, the methods we use significantly expand the "accuracy-coverage frontier"—the portion of answers that a filter can accept while maintaining a given level of accuracy among the samples it accepts. (See Figures 1 and 2.) For example, in the AMAZON_POLARITY dataset, the baseline filters need to reject $\geq 50\%$ of examples for the remainder to be 90% accurate; in contrast, a CCS-based metric can achieve the same accuracy while only marking 3% of model outputs as unreliable. Similarly, in the IMDB dataset, the best baseline filter needs to accept just 8% of answers to achieve 99% accuracy among accepted answers, while the best CCS-based filter can accept 29% of answers while retaining 99% accuracy.

Further research would be needed to determine exactly why CCS-based filters have the above success. We hypothesize that three contributing factors are:

- **Different policies toward ambiguous inputs $\rightarrow$ metrics that include agreement between CCS answers and model answers filter out unreliable answers.** If CCS and the model are using any independent information to calculate their implicit probability assignments to inputs, and both methods are better than chance, we should expect that combining the methods would strictly outperform either method on its own. Qualitative analysis of model answers and CCS answers provides preliminary support to this hypothesis; based on our subjective judgments of questions' ambiguity, we noted that model answers and CCS answers often disagreed on ambiguous questions (e.g. a sentiment analysis question about

Accuracy	0.794	0.889	0.90	0.95	0.99
(Baseline) Model answer	1				
(Baseline) Max logit			0.191	0.059	0.010
(Baseline) Logits difference			0.502	0.318	0.100
Max CCS			0.965	0.685	0.042
CCS & model agreement			0.824	N/A	N/A
Max CCS and CCS & model agreement			0.824	0.676	0.039
CCS difference			**0.969**	**0.802**	**0.391**
CCS difference and CCS & model agreement			0.824	0.738	0.377
Logits difference and CCS & model agreement			0.824	0.376	0.100
CCS answer		1			

Table 2: The largest portion of examples that each filter can accept while maintaining a given accuracy level (AMAZON-POLARITY dataset)

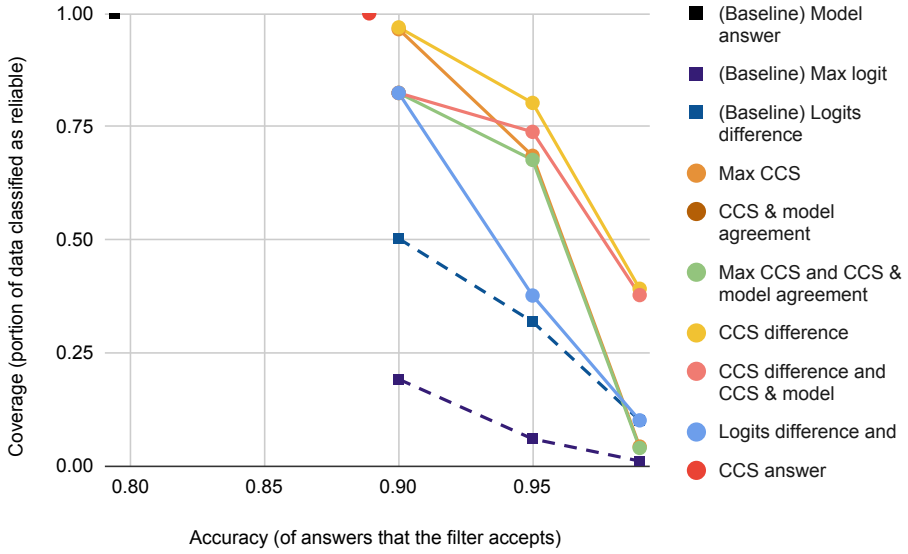

Figure 2: The largest portion of examples that each filter can accept while maintaining a given accuracy level (AMAZON-POLARITY Dataset)

a review that expressed mixed feelings), which appeared to be responsible for most of the model's false outputs.

- **CCS approximates fine-tuning for question-answering.** Our base model was not fine-tuned on question answering tasks, but our implementation of CCS takes the last layer's activations as inputs and trains a new layer using a loss that (for our datasets) indirectly represents question-answering accuracy. (The CCS loss function (described in section 4.1) does not represent anything per se, but is designed to search for truth value; in our data, each example's truth value was the accuracy of the question-answer pair.) As such, it is possible that our implementation of CCS simply approximated fine-tuning for question answering; future work could add a baseline which uses the model after fine-tuning on question-answering.
- **CCS outperforms model outputs.** Unfiltered CCS outperforms unfiltered zero-shot model outputs by 4-10% on our datasets, and our baseline methods only use model outputs, so some of the observed improvement over baseline methods may simply be due to the higher accuracy of CCS. However, a moderately higher starting point of accuracy appears insufficient to fully explain the large magnitude of the observed improvements—especially at the 99% accuracy level, which is well above the accuracy that we might expect could be achieved by starting with CCS answers and applying a mediocre filter.

To further examine what we can learn from our filters, we also graphed the calibration of (normalized) model outputs before and after applying a CCS difference filter. We find that the filter did not notably improve calibration (see graphs in the appendix). This is unsurprising, since we did not directly optimize for calibration, and greater accuracy does not entail better calibration; in fact, CCS involves optimizing for "confident" outputs. The lack of calibration (and the absence of clear, simple trends in calibration graphs) suggests that, while CCS-based filters can help identify reliable subsets of answers, they cannot robustly assess the reliability of individual answers.

7 Conclusion

We train Contrast-Consistent Search (CCS) on a dataset constructed from "contrastive pairs" of question-answer pairss. We then use CCS to create filters which reject unreliable answers from language models. Through these filters, we improve the accuracy of filtered outputs, while rejecting far fewer examples compared to baseline methods. We also find that, even when the question-answering accuracy of CCS does not significantly outperform zero-shot model accuracy, it extracts novel information which can be used to improve the combined accuracy of CCS and zero-shot generations.

Our work has a number of limitations. Firstly, CCS is a new, unrefined method with some known issues: for example, it only finds one of many orthogonal linear probes which get similar accuracy, and (for unknown reasons) performs poorly on GPT models (Roger, 2023). Secondly, our project is limited in scope: we only use one model, which is small relative to SOTA LLMs; we only train CCS on the last layer; and we only evaluate performance on 2 datasets (and 1 type of task). Additionally, our filters do not attempt to improve calibration; they only minimize inaccurate outputs. Finally, we do not address the question of what to do when a generation is rejected, which would be important for the deployment of any system based on our methods.

References

Ethan Perez et al. 2022. Discovering language model behaviors with model-written evaluations. Https://arxiv.org/abs/2212.09251.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in language models without supervision. Https://arxiv.org/abs/2212.03827.

Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Robert Isele Max Jakob Anja Jentzsch Dimitris Kontokostas Pablo N. Mendes Sebastian Hellmann et al. Lehmann, Jens. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. In *Semantic web 6*, volume 2, pages 167–195.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, volume 1, page 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

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

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Julian McAuley and Jure Lescovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

Fabien Roger. 2023. What discovering latent knowledge did and did not find. Https://www.alignmentforum.org/posts/bWxNPMy5MhPnQTzKz/what-discovering-latent-knowledge-did-and-did-not-find-4.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*.

A Appendix

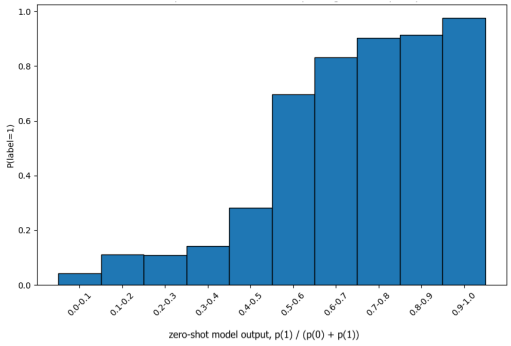

Figure 3: The calibration of the unfiltered zero-shot model outputs, calculated as $\frac{p(\text{label}=1)}{p(\text{label}=0)+p(\text{label}=1)}$, on the AMAZON_POLARITY dataset.

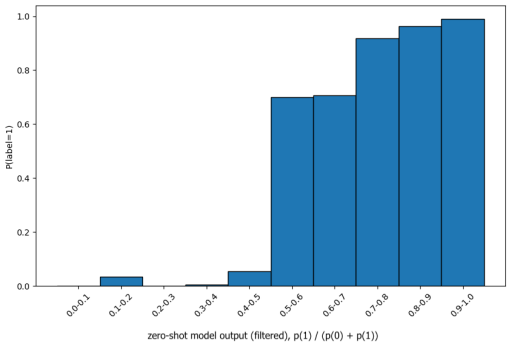

Figure 4: The calibration of the zero-shot model outputs, calculated as $\frac{p(\text{label}=1)}{p(\text{label}=0)+p(\text{label}=1)}$, on the AMAZON_POLARITY dataset, after filtering out examples for which the CCS difference was less than 0.5.

Table 3: IMDB prompt templates, in descending order of average agreement between ground truth, CCS output, and zero-shot model output.

idx	prompt template	0-shot/CCS	CCS/GT	0-shot/GT
7	"<review>How does the viewer feel about the movie? [negative, positive]"	0.98	0.9125	0.9025
12	"<review>How does the reviewer feel about the movie? [They didn't like it!, They loved it]"	0.965	0.9075	0.8975
5	"<review>How does the viewer feel about the movie? [negative, positive]"	0.965	0.9075	0.8925
8	"<review>What is the sentiment expressed in this text? [negative, positive]"	0.95	0.915	0.88
3	"The following movie review expresses what sentiment? <review>[negative, positive]"	0.9675	0.8925	0.87
0	"Consider the following example: ''' <review>''' Between negative and positive, the sentiment of this example is [negative, positive]"	0.9425	0.9	0.8825
10	"<review>What sentiment does the writer express for the movie? [negative, positive]"	0.955	0.89	0.865
9	"<review>What is the sentiment expressed by the reviewer for the movie? [negative, positive]"	0.9525	0.89	0.8525
6	"Consider the following example: ''' <review>''' Between negative and positive, which is the sentiment of this example? [negative, positive]"	0.8175	0.8925	0.805
2	"<review>Is this review positive or negative? [negative, positive]"	0.825	0.8225	0.8175
11	"<review>Did the reviewer find this movie good or bad? [bad, good]" 0.7775	0.86	0.8125	
4	"<review>This is definitely not a [positive review., negative review.]"	0.855	0.7975	0.7625
1	"<review>Did the reviewer enjoy the movie? [No, Yes]"	0.645	0.82	0.695

Table 4: AMAZON_POLARITY prompt templates, in descending order of average agreement between ground truth, CCS output, and zero-shot model output.

idx	prompt template	0-shot/CCS	CCS/GT	0-shot/GT
4	"Title: <title> Review: <review> Does this product review convey a negative or positive sentiment? [Negative, Positive]"	0.9425	0.925	0.9425
7	"Is there a negative or positive tone to this product review? === Title: <title> Review: <review> Answer: [No, Yes]"	0.95	0.9325	0.9225
2	"You are considering whether to buy a product. You look at the reviews. Would the following review decrease or increase the chances of you buying the product? Review title: <title> Product review: <review>[decrease, increase]"	0.9125	0.9075	0.9
0	"Here is a review left by a customer on a product. Would you say he was satisfied or dissatisfied? <review>[dissatisfied, satisfied]"	0.9275	0.91	0.8775
10	"Consider the following example: ”’ <review>”’ Between negative and positive, the sentiment of this example is [negative, positive]"	0.9075	0.8975	0.875
1	"Title: <title> Review: <review> Is this product review negative? [Yes, No]"	0.87	0.8325	0.8025
8	"Title: <title> Product review: <review> Would you say this review depicts the product in a flattering or unflattering light? [unflattering, flattering]"	0.8175	0.885	0.7725
5	"Consider the following example: ”’ <review>”’ Between negative and positive, which is the sentiment of this example? [Negative, Positive]"	0.8375	0.8325	0.78
3	"Title: <title> Review: <review> Is the review positive or negative? [Negative, Positive]"	0.6975	0.9125	0.69
9	"Is this product review positive? Title: <title> Review: <review> Answer: [No, Yes]"	0.675	0.8675	0.6475
6	"Based on this review, would the user recommend this product? === Review: <review> Answer: [No, Yes]"	0.6125	0.8825	0.605