Detecting Hallucinations in Natural Language Generation

Ahmet Emre Belge, Jonas Treplin

December 2023

Abstract

We implement, test, and compare multiple current approaches to hallucination detection on the SHROOM Task of SemEval-24. The two main classes of methods we investigate are methods that are based on semantic distance measures to an included reference solution as well as methods that try to leverage access to model information by trying to quantify the certainty the model has in the generated hypothesis. We also try to fine-tune the first class of models and use both classes of models in combination.

1 Introduction

There can be no doubt that, with the advent of Large Language Models (LLM), Natural Language Generation (NLG) is one of the fastest-growing technologies at the moment. Tools like ChatGPT have seen rapid adoption and user growth in the last year, and multiple companies plan on integrating chatbots based on LLMs into their products. However, there have been concerns about the factual reliability of the model outputs. While LLMs are relatively good at generating syntactically correct and semantically meaningful fluent output, they sometimes produce factually incorrect statements called hallucinations. The detection of hallucinations has become an important task for ensuring the security and reliability of the rapidly increasing applications of Large Language models.

In this paper, we work on the SHROOM challenge of the SemEval-24 conference [13]. The task is to detect generation mistakes across three different NLG tasks: Definition Models (DM), Paraphrase Generation (PG) and Machine Translation (MT). For this, we aim to develop a model that returns a score that will indicate how likely it is for a certain model output to be a hallucination. Additionally, we provide a threshold above which a score is considered a hallucination by the model.

For this, we are given a large dataset of model generations: each datapoint consists of a model input s, a model output h and a true reference solution r(this is just the model input for the PG task) that is supposed to be semantically close to the model output if the model is correct. Additionally, for a "model aware" portion, the LLM m that was used for each run is also annotated for a part of the dataset. The challenge has two tracks: one for methods that are model aware and one for methods that don't take into account the model that was used for generation. The largest portion of the dataset provided contains no information about whether a given datapoint is actually a hallucination or not. But there is also a small testing dataset where multiple human experts have annotated whether certain model outputs are hallucinations, referred to as the development dataset. This dataset is also split into a model agnostic part and a model aware part. Additionally, there is an even smaller trial dataset that is also annotated with human output but does not contain any model information at all. We use these humanly annotated datasets to test how well our methods measure up to human expert evaluations.

The challenge competitors will be evaluated on an evaluation dataset similarly annotated as the test set. The evaluation metrics of the competition are the accuracy of the model (the rate at which it agrees with the majority vote of the human annotators) and the Spearman's correlation of the model score to the average result of the annotators. These metrics are designed to assess the quality of our hallucination prediction as a binary classifier as well as a regression method.

2 Pre-trained Semantic Similarity Methods

Since we are given a reference solution, our first approach was to use methods for determining the semantic similarity between the reference solution r_i and the output hypothesis h_i proposed by the model. We use this

semantic similarity score $d(r_i, h_i)$ to compute $s_i = 1 - d(r_i, h_i)$ as a score predicting whether h_i is a hallucination or not, since the similarity score is close to 1 if r_i and h_i are similar, indicating that the model is likely correct if the similarity is high. There is a lot of research done on semantic similarity. So our main contribution here is to benchmark different existing and pretrained models on this specific dataset. We evaluate each method on all the tasks it is suited for as well as on the overall dataset if it is a general-purpose model. We use these approaches as a baseline for further exploration in Sections 3 and 4.

2.1 Embedding Methods

An embedding model provides a latent representation of their arbitrary-length input text in a vector space of a fixed size. Mathematically, we write for a text input s and model m:

$$m(s) \in \mathbb{R}^d$$

Most embeddings are trained such that cosine similarity is a sensible geometric measure for closeness in the embedding space. Our semantic similarity between reference solution r and hypothesis h then becomes:

$$d(r, h) = \frac{m(r)^T m(h)}{\|m(r)\| \cdot \|m(h)\|}$$

We used a selection of models from the sentence transformers library [8]. This library provides two types of models: Pure embedding models and CrossEncoders. CrossEncoders are described in Section 2.2. The embedding models evaluated here are different versions of MPnet [10], Distilroberta [9] and MiniLM [11]. We choose the most promising models according to the SentenceTransformers documentation, with some attention being paid to including different model families and architectures.

We further compare those results with embeddings provided by OpenAI [1]. Not All these embedding models are trained to work in multilingual settings. Only some versions are specifically enhanced for multilingual contexts, so they will perform better at detecting hallucinations in the Machine Translation task. We evaluate the models on two metrics that will also be the metrics for the SHROOM competition:

- **Accuracy**: This metric assesses the model's performance on the binary classification. Since the embedding models produce continuous outputs, we have to choose a threshold above which we classify a datum as hallucination. To have a comparable metric for different models we search for an optimal threshold to produce the maximum accuracy using the Nelder-Mead method.
- **Spearman's Correlation**: To assess the regression performance we use Spearman's rank correlation to capture non-linear relations between our score s and the given p(Hallucination)

A considerable metric for real-world applications would also be the highly varying speed of these models. However, we just focus on the metrics that indicate the model quality not runtime speed.

Table 1 shows the results for embedding models on the overall dataset and table 2 shows performances of each task. From these results we can see that OpenAI's Ada is best under the embedding models for both definition and paraphrasing tasks but falls behind the multilingual version of MPnet on the translation task. These results seem to indicate that it might be best to use different models for each task rather than a monolithic model.
The stagnant but high accuracy and low Spearman's correlation for all evaluations in the PG task can be explained by target imbalance for this task. Only 23% of the test data for the PG task are classified as hallucinations by the majority vote. Whereas the DM and MT tasks have rather balanced test sets.

Model Tag	Maximum Accuracy	Spearman's Correlation
all-mpnet-base-v2	0.709	0.547
all-distilroberta-v1	0.702	0.562
multi-qa-mpnet-base-dot-v1	0.740	0.565
all-MiniLM-L12-v2	0.724	0.558
all-MiniLM-L6-v2	0.717	0.549
paraphrase-multilingual-mpnet-base-v2	0.734	0.612
ada-002 (openai)	0.740	0.597

Table 1: Results of different embedding models on the entire evaluation dataset

Model Tag	DM		PG		MT	
	Max. Acc.	Spearman	Max. Acc.	Spearman	Max. Acc.	Spearman
all-mpnet-base-v2	0.674	0.559	0.791	0.269	0.789	0.675
all-distilroberta-v1	0.736	0.631	0.783	0.253	0.775	0.650
multi-qa-mpnet-base-dot-v1	0.633	0.563	0.783	0.328	0.789	0.667
all-MiniLM-L12-v2	0.741	0.593	0.776	0.279	0.618	0.665
all-MiniLM-L6-v2	0.714	0.588	0.776	0.237	0.771	0.649
paraphrase-multilingual-mpnet-base-v2	0.785	0.670	0.783	0.289	0.802	0.717
ada-002 (openai)	0.763	0.684	0.776	0.311	0.699	0.696

Table 2: Results of different embedding methods on the specific tasks

2.2 Direct Scoring Methods

Beside pure embeddings, the SentenceTransformers library also provides so called CrossEncoders, these models directly take two sentences and compute their similarity. A CrossEncoder m thus directly provides a semantic similarity metric for a reference r_i and hypothesis h_i: $d(r_i, h_i) = m(r_i, h_i)$.
We use specialized variants of MiniLM [11], RoBERTa [5] and TinyBERT [3] for this benchmark. Our choice of models was again dependent on the reported performances in the SentenceTransformers documentation.

In addition, we also include the BERTScore metric [12], which uses token-level information for use in monolanguage settings (DM and PG) as well as the COMET Score [7], which takes both the source query and the reference solution into account. for the Machine Translation setting.

Table 3 shows the performance of the tested versions over the overall dataset and Table 4 shows the performance of the tasks that were suitable for the model. In comparison with the results of Section 2.1 we can see that the RoBERTa based model clearly outperforms all embedding-based methods. Especially in the PG task it is the only one that brings significant improvement.

Model Tag	Maximum Accuracy	Spearman's Correlation
stsb-TinyBERT-L-4	0.745	0.590
stsb-roberta-large	0.759	0.676
quora-roberta-base	0.728	0.621
ms-marco-MiniLM-L-12-v2	0.686	0.481

Table 3: Results of different direct scoring models on the entire evaluation dataset

Model Tag	DM		PG		MT	
	Max. Acc.	Spearman	Max. Acc.	Spearman	Max. Acc.	Spearman
stsb-TinyBERT-L-4	0.772	0.667	0.776	0.283	0.650	0.643
stsb-roberta-large	0.781	0.708	0.813	0.419	0.686	0.722
quora-roberta-base	0.727	0.685	0.761	0.371	0.721	0.627
ms-marco-MiniLM-L-12-v2	0.709	0.594	0.738	0.156	0.636	0.603
BERTScore	0.616	0.371	0.776	0.263	–	–
COMET	–	–	–	–	0.654	0.347

Table 4: Results of different direct scoring models on the specific tasks

3 Model Aware Methods

Since we are given the specific model that each datum was generated with, we can use this information. We explore two approaches here. Both aim to provide a measure of uncertainty in the model output. This works under the assumption that if a model hallucinates, it is hopefully not as certain of its outputs. The methods investigated here are:

- **SelfCheckGPT** (inspired from [6]): Here we use the model to generate multiple randomized outputs and check their similarity with the original output to discern how certain the model is in its hypothesis. We

also use two different variants, one where we analyze how far the different outputs diverge and one where we compare the randomly generated output with the reference solution.

- **SeqLogProb** (proposed in [2]): This method looks at the token output probabilities and is related to the model's perplexity.

We also investigate how well these methods can work in combination with the methods proposed in Section 2. Note that for this section, all models were obviously evaluated only on the model-aware test data. Also we only use the DM task for the tests in this section since it was hard to run the translation models and the PG task has unbalanced test data which hinders a good comparison of the results.

3.1 SelfCheck

For a datum of source prompt s_i, hypothesis h_i, reference solution r_i and the model used for generation of h_i g_i. We use the model in a randomized configuration to generate K additional hypotheses $\hat{h}_{ij}, j \in \{1, ..., K\}$. We remark that there is considerable fine-tuning potential in setting up the random generation. Now, using some similarity measure $d(.,.)$ from Section 2. We then propose the following scores:

- **Hypothesis-Check-Score**: We use the average similarity of the random hypothesis to the actual hypothesis:

$$S_i^{\text{hyp}} = \frac{1}{K} \sum_{j=1}^{K} d(h_i, \hat{h}_{ij})$$

  This is the proposed method most similar to the one originally used in the SelfCheckGPT paper [6]. The difference is that the original SelfCheckGPT works in a multisentence setting and it was fixed to using the BERTScore metric. Our dataset only contains single-sentence datapoints, so we do not have to worry about that and in Section 2 we found that we can improve upon the BERTScore distance by a large margin. This metric measures how certain the model is in its original output. A higher score means the generated outputs are all similar, so a lower indication of a hallucination.

- **Reference-Check-Score**: This is essentially the same as the precious score except that we compare to the reference solution r_i:

$$S_i^{\text{ref}} = \frac{1}{K} \sum_{j=1}^{K} d(r_i, \hat{h}_{ij})$$

  This quantifies how close the model usually hits the reference solution. Here, a higher score also means a lower indication of a hallucination, as the model usually hits close to the reference.

- **Confusion-Check-Score**: Here we use the difference between each of the generated outputs to obtain a measure of general confusion in the model output:

$$S_i^{\text{conf}} = \frac{2}{K \cdot (K+1)} \sum_{j=1}^{K} \sum_{k=1}^{j} d(\hat{h}_{ik}, \hat{h}_{ij})$$

  This should tell us how concentrated the model outputs are. If this score is low, we see that the model produces a lot of different outputs and has, in general, high variance. This could indicate that the hypothesis we have also strays far from the reference.

Note that the hypothesis score and the confusion score work without access to a reference solution, which is often the case in real-world applications.
We evaluate these approaches on the definition task only, because the models provided were hard to run on the translation task and the paraphrase task is too unbalanced for a good evaluation as we saw earlier. For the experiments, we use two different embedding similarity measures based on the `all-mpnet-base-v2` and `paraphrase-multilingual-mpnet-base-v2` embeddings. Table 5 shows the Spearman's rank correlation and the maximum accuracy of each of these methods. We see a clear benefit in using the better `paraphrase-mpnet` model for the semantic similarity. However, all the uncertainty metrics underperform in comparison to the semantic similarity methods.

3.2 SeqLogProb

A second metric we investigate is proposed in [2] and found in their benchmark to be the best hallucination detection heuristic that is independent of a reference solution. This method uses token-level information, so we decompose our hypothesis $h_i = (y_1, ..., y_L)$. This tokenization is provided by the model. Now let $p(y_k|y_{<k}, s_i)$ be the probability that the model g_i chooses token y_k given all the previously chosen token $y_{<k}$ and the source prompt s_i. We can get these scores by running the model in forward mode like it is used for model training. The score we compute from this is:

$$S_i^{\text{unc}} = -\frac{1}{L}\sum_{k=1}^{L} \log p(y_k|y_{<k}, s_i)$$

Note that this is just the negative logarithm of the model perplexity. Also, like the scores in Section 3.1, this is an uncertainty score. If the probability of obtaining h_i is low then S_i^{unc} will be high, indicating that the model is not sure in its hypothesis.

In Table 5 we also see the result of this heuristic compared with the other model-aware methods. This SeqLogProb method is outperformed in terms of prediction quality by all the other methods presented in this section. It may, however, be noted that this method is by far the fastest of the compared ones.

Model Tag	Maximum Accuracy	Spearman's Correlation
Hypothesis-Check (mpnet)	0.540	0.264
Reference-Check (mpnet)	0.566	0.358
Confusion-Check (mpnet)	0.602	0.273
Hypothesis-Check (paraphrase-mpnet)	0.638	0.317
Reference-Check (paraphrase-mpnet)	0.665	0.438
Confusion-Check (paraphrase-mpnet)	0.660	0.308
SeqLogProb	0.553	0.229

Table 5: Results of different uncertainty quantifying methods restricted on the definition task

3.3 Uncertainty Measures as Auxiliary Metric

Even though the uncertainty metrics as standalone models are outperformed by large margins, they are not immediately useless. We might be able to use these uncertainty methods in combination with the semantic similarity methods so we can combine the information we get from the reference solution with the previously discussed uncertainty measures.
Mathematically, for a semantic similarity measure d, a model-aware score $S^k, k \in \{\text{hyp}, \text{ref}, \text{conf}, \text{unc}\}$ and a combination ratio r our combined score becomes:

$$S_i^{\text{comb}} = (1-r)(1-d(h_i, r_i)) + rS_i^k$$

For this, we combine the `stsb-roberta-large` semantic similarity model as the base score — this was the best model for the semantic similarity methods — with the models discussed in the last two sections, where we use the `paraphrase-mpnet` model for the SelfCheck approaches.

To make the ranking comparable, we have to optimize the ratio r of the base score and the additional score. As an optimization metric, we chose to maximize Spearman's correlation. We use this optimized ratio to compute an optimal threshold for the binary classification, as in Section 2.

Table 6 shows the results of these experiments on the Definition Task. We can observe that the additional uncertainty score brings a small but not insignificant boost to the accuracy but does not really increase the Spearman's correlation.

Model Tag	Maximum Accuracy	Maximum Spearman's Correlation
RoBERTa + Hypothesis-Check	0.790	0.709
RoBERTa + Reference-Check	0.794	0.710
RoBERTa + Confusion-Check	0.799	0.709
RoBERTa + SeqLogProb	0.794	0.713

Table 6: Results of different ensembles of uncertainty quantifying and semantic similarity methods restricted on the definition task

4 Semi-Supervised Fine-Tuning

Given the amount of unlabeled data available for the challenge and the remarkable performance of pre-trained models such as RoBERTa, we explore the possibility of fine-tuning a pre-trained CrossEncoder model. As fine-tuning these models is only possible with a labeled training set, we opt for a technique called Pseudo Labeling [4], a Semi-Supervised method. Semi-Supervised learning is a machine learning paradigm where labeled and unlabeled data are used together to improve the model. This paradigm is especially popular in the realm of Natural Language Processing models where large amounts of labeled data that are needed for training might not be available. In the case of Pseudo Labeling, these are the main steps:

1. A model is trained on a labeled dataset.
2. This model is used to make predictions on the unlabeled dataset.
3. The unlabeled dataset gets labeled with the predictions.
4. The model is re-trained using the pseudo labeled dataset.

In our case, the first step is covered by the pre-trained RoBERTa model, in particular its `stsb-roberta-large` version specific for sentence similarity tasks, with "stsb" standing for "Semantic Textual Similarity Benchmark". As it was shown in Section 2, this model is the best-performing readily available model for our task. As a Cross Encoder model, it takes a sentence pair, as opposed to the case of Bi-Encoders (referred to above as pure embedding models) where the sentences are encoded separately, and assigns a similarity score between 0 and 1 to them. CrossEncoders are shown to perform better for tasks similar to ours where outputting sentence embeddings is not needed [8].

To construct our sentence pairs for the second step, the reference solution r_i is "tgt" (target, what is the expected output of the model given the source sentence) for Machine Translation and Definition Modelling tasks and "src" (source, input given to the model) for Paraphrase Generation tasks. The (r_i, h_i), i.e., (reference, hypothesis), pairs are then fed into the model to obtain their similarity scores.

Usually in the third step, we would only consider the unlabeled data points whose predictions have high confidence and iteratively train the model this way, where in each iteration we consider the most confident predictions of the latest model. However, in our case, considering only predictions with high confidence, for example, labeling and using for training only sentence pairs whose predicted similarity score is below 0.2 (hallucination with high confidence) and above 0.8 (not hallucination with high confidence), resulted in lower performance in the fine-tuned model compared to the base model. Changing these thresholds and considering only < 0.4 and > 0.6 resulted in a similar drop in performance. This drop is likely caused by the reduction in the size of the dataset and the loss of diversity in the data that happens when we filter to only include high-confidence examples, which can result in the model missing out on nuanced, less obvious examples of sentence relationships necessary to effectively generalize across a variety of scenarios and tasks. This result led us to include the entirety of the pseudo labeled data, which improved on the base model. Three other datasets, grouped by the task, were also created from the pseudo labeled data in order to evaluate the results of fine-tuning on specific tasks.

For the training, we closely followed the training examples provided by SentenceTransformers [8] on how to train Cross Encoders for Semantic Textual Similarity tasks with slight variations in hyperparameters. The best results were achieved with only 1 epoch of training, likely due to the learning capabilities of RoBERTa, which is able to capture the nuances of our data with a single epoch of training, after which it starts to overfit. We had 4 variations of the fine-tuned model: one trained on the entirety of the pseudo labeled dataset and one model for each task, which was trained only on the respective pseudo labeled task-specific sentence pairs. The test set used to evaluate the fine-tuned model was built by adding together the provided test set and the "model aware" and "model agnostic" development sets. From this set, three sub-test sets were derived by separating according to the task.

It is also important to note that we were provided with two separate unlabeled training sets, one "model aware" and one "model agnostic", as was the case for the development sets. It was observed that training only with the pseudo labeled aware set or training with the pseudo labeled aware and agnostic sets together consistently performed under the base model, whereas training only with the pseudo labeled agnostic set resulted in an increase in maximum accuracy that can be seen in Table 7. An observation that can explain these results is the fact that adding the aware development set to the test set decreases test accuracy and Spearman's correlation in both the base and fine-tuned models, with similar results in a test set constructed solely with the aware

development set. This might lead to the intuition that it is harder for our pre-trained model to make accurate predictions about the aware data, which can be caused by specific patterns in the data, and therefore produce less accurate pseudo labels for the aware training set, which in turn decreases the performance of the model fine-tuned with such a training set.

Model	Maximum Accuracy	Spearman's Correlation
Base stsb-roberta-large	0.767	0.678
Fine-tuned general stsb-roberta-large	0.786	0.683

Table 7: Results of fine-tuning on the pseudo labeled agnostic training set, tested on the union of the test and agnostic + aware development sets

Model Tag	PG		MT		DM	
	Max. Acc.	Spearman	Max. Acc.	Spearman	Max. Acc.	Spearman
Base stsb-roberta-large	0.768	0.489	0.817	0.764	0.769	0.699
Fine-tuned task-specific	0.764	0.464	0.805	0.754	0.769	0.702
Fine-tuned general stsb-roberta-large	0.780	0.482	0.810	0.760	0.793	0.701

Table 8: Task-specific model, general fine-tuned model and the base model, tested on the task-specific union of the test and agnostic + aware development sets

We observe that by fine-tuning on the pseudo labels, we successfully increase test maximum accuracy and Spearman's correlation of the `stsb-roberta-large` model on a mixed-task dataset. Our general fine-tuned model has a higher maximum accuracy and a similar, if not better, Spearman's correlation compared to the task-specific fine-tuned models. The base model has a higher maximum accuracy in Machine Translation, however when the general fine-tuned model's superior results in the other tasks are taken into account, we can see that it is preferable in general to the base model.

As mentioned above, this model outputs a similarity score between 0 and 1 given a sentence pair, an increasing measure indicating how similar these sentences are to each other. In order to translate this into a binary hallucination detection model, one would only have to calculate, denoting with $m_i(r_i, h_i)$ the similarity score of the pair, $1 - m_i(r_i, h_i)$ to obtain some kind of confidence score about h_i being a hallucination, assuming a direct inverse relationship between h_i being similar to r_i and h_i being a hallucination. After this calculation, the last step would be deciding on a binary classification threshold that would maximize the accuracy, similar to our method, considering the fact that this threshold may vary depending on the model and the distribution of the data.

5 Conclusions

The results that we obtained with the Semi-Supervised paradigm go on to show the value of this approach in Natural Language Processing. It makes it possible to improve on a pre-existing model and obtain a fine-tuned model that performs better in most of the tasks and is comparable in the rest without needing access to large amounts of labeled data that are usually costly and time-consuming to obtain. The Pseudo Labeling technique is effective, but it naturally comes with some limitations that we were also faced with. Researchers using this technique should remain vigilant for pseudo labels with low accuracies that can hinder the fine-tuning of their model, similar to what happened in our case when we added the pseudo labeled aware dataset to our training dataset. Another limitation is the decrease in the number of data points when proceeding in an iterative manner using only high-confidence predictions, which can also affect fine-tuning performance, as happened in our model. These limitations, however, do not limit the great potential of this method, and we see various opportunities for refinement. In a case where more data is available, the iterative approach might yield even better results by carefully adjusting the confidence threshold.

There is some further work required for the final model we will submit for the competition. The quality of the uncertainty methods in Section 3 should be tested on the PG and MT tasks. The effect of combining these uncertainty measures with the fine-tuned model also remains to be investigated, but we have a strong intuition that this will be slightly better than the pure fine-tuned model. Also, there are methods that can look deeper into the generating model, such as detecting anomalous attention patterns in the encoder and/or decoder of the

generating transformer model. A way to further refine the SelfCheck scores of Section 3.1 may be to use the fine-tuned similarity scores of Section 4 as their distance measure.

All in all, we note that we were able to improve on the naive semantic similarity approach by a small but not insignificant margin. This was achieved using two approaches: using information from the model output or generating new hypotheses to quantify the uncertainty in a given hypothesis, and fine-tuning the pre-trained semantic similarity models on the pseudo-labeled dataset.

References

[1] Ryan Greene, , Ted Sanders, Lilian Weng, and Arvind Neelakantan. New and improved embedding model, 2022.

[2] Nuno M. Guerreiro, Elena Voita, and André F. T. Martins. Looking for a needle in a haystack: A comprehensive study of hallucinations in neural machine translation, 2023.

[3] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351, 2019.

[4] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 07 2013.

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

[6] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models, 2023.

[7] Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. COMET: A neural framework for MT evaluation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online, November 2020. Association for Computational Linguistics.

[8] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[9] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[10] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*, 2020.

[11] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[12] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

[13] Elaine Zosa, Raúl Vázquez, Jörg Tiedemann, Vincent Segonne, Teemu Vahtola, Alessandro Raganato, Timothee Mickus, and Marianna Apidianaki. Semeval-2024 task-6 - shroom, a shared-task on hallucinations and related observable overgeneration mistakes.