AlphaIntellect at SemEval-2024 Task 6: Detection of Hallucinations in Generated Text

Sohan Choudhury
KIIT, Bhubaneswar
sohan2004cc@gmail.com

Priyam Saha
Jadavpur University, Kolkata
priyam.saha2003@gmail.com

Subharthi Ray
Jadavpur University, Kolkata
subharthiray126@gmail.com

Shankha Shubhra Das
Jadavpur University, Kolkata
shankhasdas07@gmail.com

Dipankar Das
Jadavpur University, Kolkata
dipankar.dipnil2005@gmail.com

Abstract

One major issue in natural language generation (NLG) models is detecting hallucinations (semantically inaccurate outputs). This study investigates a hallucination detection system designed for three distinct NLG tasks: definition modeling, paraphrase generation, and machine translation. The system uses feedforward neural networks for classification and SentenceTransformer models for similarity scores and sentence embeddings. Even though the SemEval-2024 benchmark is showing good results, there is still room for improvement. Promising paths towards improving performance include considering multi-task learning methods, including strategies for handling out-of-domain data and minimizing bias, and investigating sophisticated architectures.

1 Introduction

AI hallucination refers to a phenomenon where a Large Language Model (LLM) - usually Generative AI or a computer vision tool produces nonsensical and inaccurate outputs (Maleki et al., 2024). Thus, this leads to fluent but inaccurate generations. The term 'hallucination' is usually associated with human or animal brains but from the standpoint of machines, hallucinations refer to these inaccurately produced outputs.

Hallucinations in generative models may arise due to multiple factors such as overfitting during model training, complexity of the model, and bias in training data. According to multiple surveys (Huang et al., 2023), Hallucinations in natural language models may arise primarily due to two reasons - Hallucinations due to data and hallucinations during modeling.

Hallucinations in AI models may prove to be a threat in multiple scenarios such as healthcare where a model may not be able to predict the existence of the exact condition that needs to be treated. Hallucinations in fluent over generations may also lead to the spread of misinformation.

The most pressing problem in the modern-day natural language generation landscape is that the existing metrics (Bandi et al., 2023) can mostly detect fluency in generation rather than accuracy.

To deal with this issue, several studies have explored different techniques, such as Knowledge Graph Integration, Bias Detection, and Mitigation (Rawte et al., 2023). Building upon this prior research, our work proposes training tailor-made deep learning models and using Transformer (Vaswani et al., 2017) based architectures to identify cases of hallucinations.

To deal with this scenario, a task (Mickus et al., 2024) was proposed to build a system to detect instances of hallucinations in generated text.

2 Task

The primary task was to build a hallucination detection system capable of detecting outputs that are grammatically sound but are semantically inaccurate concerning the provided source input - both with or without access to the model used to generate the outputs. This is essentially a binary classification task and we were provided with two tracks - model agnostic and model aware. Model agnostic refers to the track where one would have no access to the model used to generate the outputs and model aware refers to the track where the model was provided in the dataset.

3 Related Work

Recent advances in natural language processing (NLP) have resulted in the development of Transformer based models such as BERT and its specialized variations that have introduced efficiency and accuracy in several NLP tasks.

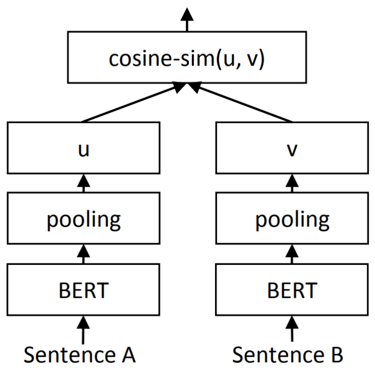

Figure 1: SBERT architecture to compute similarity scores

3.1 BERT

Bidirectional Encoder Representations from Transformers or BERT (Devlin et al., 2018), is an innovative machine learning technique for natural language processing (NLP). Researchers at Google AI Language created the adaptable BERT model in 2018, and it can perform more than 11 common NLP jobs, such as named entity recognition and sentiment analysis. Computers have never been very good at interpreting language. This requirement is attempted to be filled by NLP, a blend of languages, statistics, and machine learning. NLP activities required the usage of specialized models prior to BERT. With its cohesive approach and remarkable performance across a range of tasks, BERT transformed NLP.

3.2 SBERT

SBERT [1] or Sentence-Bert is a modified version of the BERT model which uses siamese [2] and triplet networks and is able to understand meaningful semantic embeddings in sentences. A common issue with BERT is, that the cross encoder setup of BERT takes up a lot of time and resources. To find the pair with the maximum similarity among n = 10,000 phrases, for example with BERT,

$$\frac{n(n-1)}{2} = 49{,}995{,}000$$

inference calculations are needed. This takes roughly sixty-five hours on a contemporary V100 GPU.

[1]http://arxiv.org/abs/1908.10084
[2]https://www.cs.cmu.edu/ rsalakhu/papers/oneshot1.pdf

4 Datasets

Since the task was divided into two tracks - model-aware and model-agnostic, we were provided with sets of two datasets. The model-aware dataset had a separate column for the generative model used to produce the outputs. The training dataset consisted of three natural language generation (NLG) tasks - Definition Modelling, Paraphrase Generation, and Machine Translation.

1. **Definition Modelling (DM)** - Clear and concise definition of concepts or terms generated by generative models.
2. **Paraphrase Generation (PG)** - Alternative wordings are generated that convey the same meaning as the input text.
3. **Machine Translation (MT)** - Translation of the text from one language to another while preserving fluency and meaning.

Each entry in the dataset comprises three text columns - hyp, src and tgt.

1. **hypothesis (hyp)** - Contains the generated text.
2. **source (src)** - The source text or the original text provided as input to the generative model for producing the hypothesis.
3. **target (tgt)** - Contains the correct generation output.

The trial dataset contains three columns dedicated to the labels. The 'labels' column contains the three most likely labels out of which the majority label is displayed in the 'label' column - which is either 'Hallucination' or 'Not Hallucination'. Finally, the 'p(Hallucination)' column comprises of probability values ranging from 0 to 1.

4.1 Trends in the dataset

The datasets provided include trial, validation (model-agnostic and model-aware), and test (model-agnostic and model-aware) sets, all represented by their respective figures (Figure 2, Figure 3, Figure 4, Figure 5, and Figure 6).

The datasets contain almost an even distribution of DM and PG tasks. However, the number of rows with the task 'MT' is considerably less.

Certain entries in the dataset had no probability values and to avoid difficulties in the training process, we have dropped the rows.

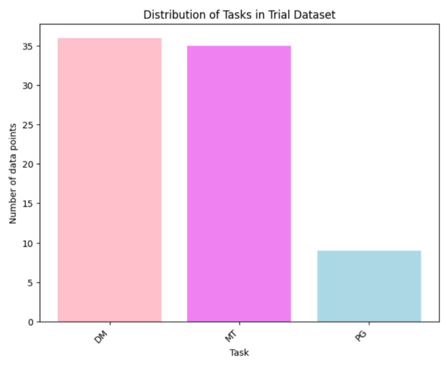

Figure 2: Trial Dataset

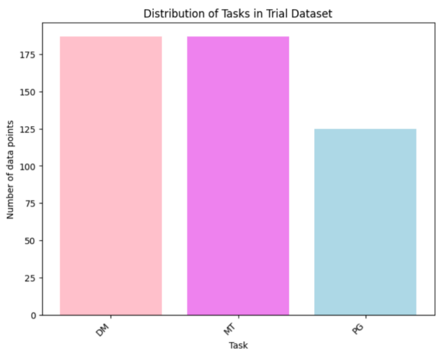

Figure 3: Validation Dataset (Model Agnostic)

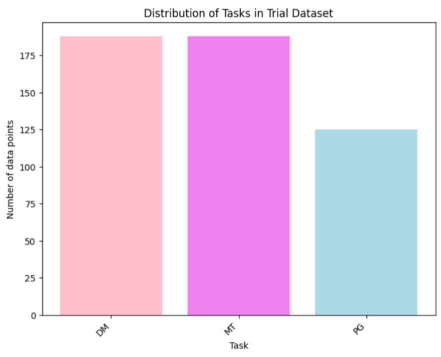

Figure 4: Validation Dataset (Model Aware)

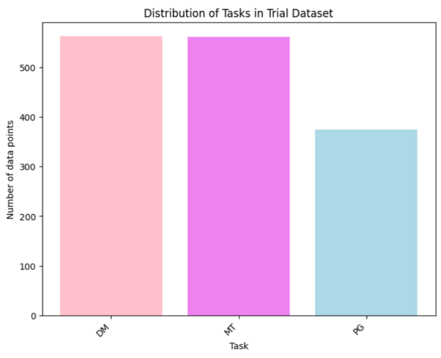

Figure 5: Test Dataset (Model Agnostic)

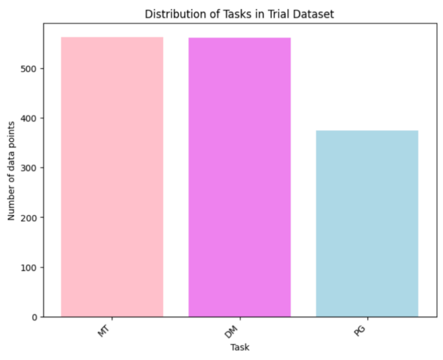

Figure 6: Test Dataset (Model Aware)

We determined that there was no viable utility for integrating the separate training dataset lacking annotations into our system. Furthermore, we were aware of the possible overfitting risk brought about by more unannotated data points. Consequently, we made the informed decision to refrain from its utilization.

5 Pre Processing

5.1 Data Cleaning

The text sections in the dataset have been processed by removing irrelevant elements in order to improve the efficiency of word embedding produced by the models used. The sentences under hyp, src, and tgt columns mainly contain prepositions and certain irrelevant expressions.

1. First the sentences are lowered by using Python's .lower() function.
2. In order to remove irrelevant expressions and prepositions, we have used Python's Regular Expression (re) library.
3. Finally, the sentences are stripped and split into individual words.

5.2 Labels

In order to make the training process more efficient, we converted the probability values in the 'p(Hallucination)' column into binary labels - if the probability value was more than or equal to 0.5, we converted it to 1 and otherwise it was labeled 0.

5.3 Tokenisation

We used 'all-mpnet-base-v2', a SentenceTransformer model to encode our sentences. The encoding process is comprised of three steps - tokenization, word embedding, and sentence embedding.

1. **Tokenization** - The sentence is split into individual tokens. This is done by the methods of stemming or lemmatization (Khyani and B S, 2021).

2. **Word Embedding** - Each token is then assigned a numerical vector. These are called word embeddings and they represent the semantic meaning of the word using information from a large text corpus.

3. **Sentence Embedding** - Finally the tokens are converted into a single vector representation for the entire sequence.

   Different models use different approaches to perform embedding.

   (a) **Mean pooling** - The average of all word embeddings is taken for the entire sentence. This method is useful in capturing the overall sentiment but may lead to a loss of semantic information.

   (b) **Weighted mean pooling** - Weights are assigned to the words using attention mechanisms to represent their importance in the sentence. This helps in prioritizing certain words and preserving semantic information.

   (c) **Transformers** - Transformer models consider the entire sentence and process the relationship between different words. Thus, contextualized embeddings capture the context more accurately.

6 Methodology

6.1 Experimental Setup

Since the generated texts have been divided into three tasks, we have divided the process into three branches.

For Definition modeling, we used 'all-mpnet-base-v2' [3] and for Paraphrase generation, we have used 'paraphrase-MiniLM-L6-v2' [4], both SentenceTransformer models to encode the sentences and calculate two sets of cosine similarity values. One represents the similarity score between the hypothesis (hyp) and source (src) text and the other represents the similarity score between the source (src) and target (tgt) text. These similarity scores are converted into a numpy array using numpy's column stack for input into two sequential models respectively.

For Machine translation, we used 'all-MiniLM-L6-v2' [5], also a SentenceTransformers model to encode the sentences and produce embeddings. The Spearman correlation between the two sentences in the hypothesis and target columns is calculated. We chose these columns specifically as 'hyp' contains the English translation produced by the generative model and 'tgt' contains the correct translation. We used the SciPy library of Python for this metric. Finally, the correlation coefficients are pushed into a numpy array. In this case, we have used a different sequential model for training.

The input array is split into an 80:20 ratio for training and validation respectively and the test dataset was entirely used for producing the outputs. We have used a common pipeline for processing the entire dataset and then branched the input array according to the task label - if the task is 'DM' it was fed into the model prepared for definition modeling.

6.2 Model

We used three models for producing the outputs for the three tasks respectively.

6.2.1 Definition Modelling

Using the Tensorflow Keras framework, we created a deep-learning neural network. It has two densely concealed layers, each with 64 and 32 neurons. For the hidden layers, we employed ReLU activation functions, which give the model non-linearity (Kulathunga et al., 2021). This helps the model learn non-linear correlations and fortifies the neural network. To lessen overfitting, we have incorporated a dropout layer after each dense layer. Lastly, since this is a binary classification problem, we have utilized the sigmoid activation function for the output layer.

6.2.2 Paraphrase Generation

We designed a neural network architecture for the paraphrase generation type inputs comprising of an input layer accepting data with two features, followed by three hidden layers. The first layer consists of 128 neurons with ReLU activation, coupled with a dropout layer to mitigate overfitting. Subsequently, a 64-neuron layer employs ReLU

[3]https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[4]https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L6-v2

[5]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

Task Specific Model	Accuracy
Definition Modelling	0.67
Paraphrase Generation	0.74
Machine Translation	0.83

Table 1: Evaluation of Individual Models

activation, batch normalization, and dropout regularization. Similarly, the third hidden layer integrates 32 neurons, ReLU activation, batch normalization, and dropout regularization. The outputs from the second and third layers are concatenated before feeding into a single-neuron output layer with sigmoid activation, typical for binary classification. This architecture is optimized using the Adam optimizer with a learning rate of 1e-4 and binary cross-entropy loss, while early stopping is applied during training to prevent overfitting. The model's configuration demonstrates a structured approach to feature extraction and classification, tailored for paraphrase generation types input data.

6.2.3 Machine Translation

The model for machine translation harnesses the formidable capabilities of the all-MiniLM-L6-v2 Sentence Transformer, designed to adeptly encode semantic nuances within input sentences into dense embeddings. These embeddings undergo meticulous examination through Spearman correlation (halo), discerning their intrinsic similarities. Post-normalization, they serve as inputs to a meticulously designed neural network architecture, capitalizing on ReLU activation functions for intricate feature extraction. Culminating in a sigmoid activation layer, the network adeptly estimates the probability of sentence hallucinations, embodying a rigorously scientific approach to classification.

6.2.4 Evaluation of Task Specific Models

This section presents an evaluation of three task-specific models trained for Definition Modelling (DM), Paraphrase Generation (PG), and Machine Translation (MT) tasks, respectively. Each model is assessed based on its training accuracy, providing insights into its performance on the training data.

6.3 Loss

We use binary cross-entropy (BCE) (Ruby and Yendapalli, 2020) as our loss function as it measures the difference between the predicted probability and the true binary label (0 or 1). BCE is not affected by class imbalance, which occurs when one class has noticeably fewer samples than the other, in contrast to Mean Squared Error (MSE). This guarantees that the model concentrates on efficiently learning both classes.

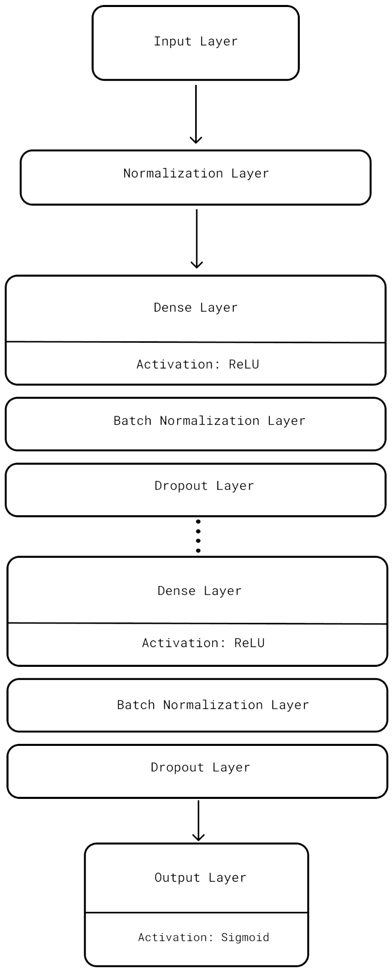

Figure 7: Generalised Architecture of the Model

6.4 Optimizer

We use the Adam (Kingma and Ba, 2017) optimizer for our neural networks as Adam is effective at traversing the loss landscape because it combines momentum and adjustable learning rates. It strikes a balance between exploration and exploitation, enabling the model to iteratively identify areas of high performance and improve its solutions.

A generalized representation of the model is visualized in Figure 7.

7 Evaluation

The predicted probability values generated by the model were translated into binary labels using a

Track	Accuracy	Rho
Model-Agnostic	0.654	0.294608108
Model-Aware	0.7113333333	0.4264291384

Table 2: Evaluation results

threshold approach. Data points with a predicted probability of 0.5 or higher were assigned the label "Hallucination," while those below 0.5 were labeled "Not Hallucination." These labels were then saved in a JSON file conforming to the specified format.

The SemEval-2024 task had two measures to evaluate the performance:

1. the accuracy that the system reached on the binary classification.

2. the Spearman correlation of the systems' output probabilities with the proportion of the annotators marking the item as overgenerating.

   It is given by:

$$\rho_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}$$

   where d_i is the difference in ranks for item i and n is the total number of items.

Also, the submissions were divided into model agnostic and model aware tracks. Our model placed 31st out of all entries in the model-aware track and 41st out of all entries in the model-agnostic track.

8 Conclusion and Future Work

This paper explored detecting hallucinations in natural language generation (NLG) outputs using specialized models for definition modeling, paraphrase generation, and machine translation tasks - both while having access to the models used to generate the sentences and without. We used transformer-based models for calculating the similarity scores as they outperform other models such as Universal Sentence Encoder (USE) (Cer et al., 2018) and Doc2Vec (Lau and Baldwin, 2016).

While there have been previous studies on hallucination detection, our approach offers several key novelties that have contributed to its effectiveness.

- **Task-specific Models**: Instead of the one-size-fits-all approach, we built specific models for each task to better capture their unique characteristics. This customization aids in efficiently extracting features crucial for identifying hallucinations.

- **Transformer-based Similarity Scores**: To compute sentence similarity scores, we made use of SentenceTransformer, a Transformer based model. These models do better at capturing contextual information and fine-grained semantic relationships inside phrases than other models.

Several avenues exist for further development of our hallucination detection system. To enhance the performance of our model, we advise investigating data augmentation techniques, as transformer-based models have a large thirst for data. To increase the model's robustness and durability, we also suggest using adversarial training and exploring more advanced deep learning architectures.

This project has been possible due to the contributions of Sohan Choudhury, who developed the architecture for the definition modelling task, Priyam Saha, who created the paraphrase generation model, and Subharthi Ray, who built the machine translation model. We are also deeply grateful for the insightful guidance and mentorship provided by Shankha Shubhra Das and Dr. Dipankar Das throughout this journey.

References

Ajay Bandi, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. 2023. The power of generative ai: A review of requirements, models, inputndash;output formats, evaluation metrics, and challenges. *Future Internet*, 15(8).

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

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

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting

Liu. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *Preprint*, arXiv:2311.05232.

Divya Khyani and Siddhartha B S. 2021. An interpretation of lemmatization and stemming in natural language processing. *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology*, 22:350–357.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Nalinda Kulathunga, Nishath Rajiv Ranasinghe, Daniel Vrinceanu, Zackary Kinsman, Lei Huang, and Yunjiao Wang. 2021. Effects of nonlinearity and network architecture on the performance of supervised neural networks. *Algorithms*, 14(2).

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *CoRR*, abs/1607.05368.

Negar Maleki, Balaji Padmanabhan, and Kaushik Dutta. 2024. Ai hallucinations: A misnomer worth clarifying.

Timothee Mickus, Elaine Zosa, Raúl Vázquez, Teemu Vahtola, Jörg Tiedemann, Vincent Segonne, Alessandro Raganato, and Marianna Apidianaki. 2024. Semeval-2024 shared task 6: Shroom, a shared-task on hallucinations and related observable overgeneration mistakes. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1980–1994, Mexico City, Mexico. Association for Computational Linguistics.

Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *Preprint*, arXiv:2309.05922.

Usha Ruby and Vamsidhar Yendapalli. 2020. Binary cross entropy with deep learning technique for image classification. *International Journal of Advanced Trends in Computer Science and Engineering*, 9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.