FunghiFunghi at SemEval-2025 Task 3: Mu-SHROOM, the Multilingual Shared-task on Hallucinations and Related Observable Overgeneration Mistakes

Tariq Ballout **Pieter Jansma** **Nander Koops** **Yong Hui Zhou**

University of Groningen, The Netherlands

{t.ballout, p.jansma.1, n.koops.1, y.h.zhou}@student.rug.nl

Abstract

Large Language Models (LLMs) often generate hallucinated content, which is factually incorrect or misleading, posing reliability challenges. The Mu-SHROOM shared task addresses hallucination detection in multilingual LLM-generated text. This study employs SpanBERT, a transformer model optimized for span-based predictions, to identify hallucinated spans across multiple languages. To address limited training data, we apply dataset augmentation through translation and synthetic generation. The model is evaluated using Intersection over Union (IoU) for span detection and Spearman's correlation for ranking consistency. While the model detects hallucinated spans with moderate accuracy, it struggles with ranking confidence scores. These findings highlight the need for improved probability calibration and multilingual robustness. Future work should refine ranking methods and explore ensemble models for better performance.

1 Introduction

Large Language Models (LLMs) are widely used in Natural Language Processing (NLP) applications, including text generation, summarization, and conversational AI (Fan et al., 2024). However, they often generate hallucinated content, information that appears plausible but is incorrect or misleading. These hallucinations pose challenges in ensuring the reliability and factual consistency of generated text (Ji et al., 2023).

Detecting hallucinations becomes more difficult in multilingual settings. Variations in grammar, vocabulary, and training data across languages affect a model's ability to identify and rank hallucinated spans consistently (Kang et al., 2024). Low-resource languages tend to exhibit higher hallucination rates due to limited training data, whereas high-resource languages, such as English, may benefit from more extensive supervision. Addressing these challenges requires models that generalize across languages while maintaining effective hallucination detection capabilities.

The Mu-SHROOM shared task[1] aims to advance research in multilingual hallucination detection. Participants are required to identify hallucinated spans in LLM-generated text across multiple languages by multiple models. The evaluation relies on two key metrics: Intersection over Union (IoU) to measure span detection accuracy and Spearman's correlation to assess the consistency of hallucination confidence scores. The task presents challenges, including variations in hallucination patterns across languages and the need for probability calibration to improve ranking reliability.

In this study, we present a SpanBERT-based approach for hallucination detection. SpanBERT (Joshi et al., 2020), a transformer model optimized for span-based predictions, is well-suited for identifying hallucinated text segments. Due to the limited availability of training data, we incorporate dataset augmentation through translation and synthetic data generation. Our results indicate that while our model detects hallucinated spans with reasonable accuracy, it struggles with ranking consistency, as reflected in low or negative Spearman's correlation scores. These findings highlight the need for improved probability calibration techniques and enhanced model robustness across languages.

The remainder of this paper is structured as follows: Section 2 reviews related work on hallucination detection and multilingual evaluation. Section 3 details the Mu-SHROOM task and dataset, including our data augmentation approach. Section 4 describes our methodology, covering model selection, preprocessing, and training setup. Section 5 presents the experimental setup. Section 5.3 provides a combined discussion and interpretation of the results. Finally, Section 6 summarizes key findings and outlines directions for future research.

[1] https://helsinki-nlp.github.io/shroom/

Proceedings of the The 19th International Workshop on Semantic Evaluation (SemEval-2025), pages 1602–1608

2 Related Work

Hallucination detection in LLMs is a well-known problem, where models generate factually incorrect or misleading information. Detecting these errors is important for improving the reliability of generated text (Luo et al., 2024).

In this section, we discuss prior work on evaluation metrics, hallucination detection, and multilingual challenges. Existing approaches range from sentence-level classification to span-level annotation, with multilingual settings introducing additional complexities.

Hallucination Detection in Large Language Models Kang et al. (2024) compare different hallucination detection metrics in multilingual settings. Their study finds that Natural Language Inference (NLI)-based methods often perform better than lexical overlap measures like ROUGE (Lin, 2004). However, these methods struggle with detecting fine-grained hallucinations at the span level, which is a key focus of Mu-SHROOM.

Shen et al. (2024) introduce a dataset for detecting hallucinations in news headlines across multiple languages. Their work includes fine-grained annotations, showing that different hallucination types require different detection strategies. This aligns with Mu-SHROOM's goal of identifying hallucination spans, though our task focuses on LLM-generated text rather than news headlines.

Multilingual Hallucination Detection and Evaluation Most hallucination detection research focuses on English, but hallucinations occur differently across languages. Detecting hallucinations in multilingual settings is more complex due to variations in grammar, entity representation, and knowledge availability.

Guerreiro et al. (2023) study hallucinations in multilingual translation models, showing that low-resource languages are more likely to produce hallucinated content. They highlight the need for language-specific approaches to hallucination detection, as models may behave differently depending on the training data. Mu-SHROOM builds on this by providing a multi-lingual dataset that evaluates hallucination detection across various languages and public LLMs.

3 Task Description and Datasets

This section outlines the Mu-SHROOM task and dataset. We describe the task of detecting hallucinated spans in multilingual LLM outputs and explain dataset augmentation through translation and synthetic data generation.

3.1 Task Description

The Mu-SHROOM task focuses on detecting hallucinated spans in text generated by LLMs. The organizers define hallucinations as content that contains or describes facts that are not supported by the provided reference. In other words, hallucinations occur when the answer text is more specific than it should be, given the information available in the provided context. Figure 1 shows an example of a hallucination.

ID	val-en-4
Language	English (EN)
Model Input	*When did Chance the Rapper debut?*
Model Output	*Chance the Rapper debuted in 2011.*
Model ID	tiiuae/falcon-7b-instruct
Soft Labels	
Start–End	Probability
18–29	0.0909
29–33	0.5455
Hard Labels	[29, 33]

Figure 1: Example of a hallucination in the English validation file.

The goal is to determine which parts of an LLM-generated output contain hallucinations. The task is multi-lingual and multi-model, with data provided by the organizers in multiple languages and from various open-weight LLMs. The provided data includes:

- **Language**: English, Chinese, Swedish, Spanish, German, Hindi, Finnish, Arabic, Italian, or French.
- **Model**: LLM model, such as Qwen, Llama, or Mistral.
- **Raw text**: a string of characters.
- **Tokenized representation**: a list of tokens.
- **Logits**: model confidence scores.

- **Soft Labels**: Probability-based (how likely it is a hallucination)
- **Hard Labels**: Binary (hallucination or not)

For each character in the output, the probability of it belonging to a hallucinated span must be computed. Any approach, including external resources, can be used, and there is flexibility in selecting which languages to focus on.

3.2 Dataset Augmentation

We received validation files containing 50 output sentences in various languages. To expand the training data, we translated the non-English validation files that had the most similar linguistic structure to English, specifically German, French, Swedish, Spanish, and Italian. The translations were generated using the `GPT-4o-mini` model via the OpenAI API with a prompt detailed in Appendix A.1.

In addition to translated data, we generated synthetic data using `GPT-4o-mini`. This was done by providing the model with a few examples from the validation set along with additional question-answer pairs. The prompt used for this process is described in Appendix A.2. Both prompts resulted in a total of 200 additional question-answer pairs.

4 Methodology

4.1 SpanBERT

For this task, we used the pre-trained SpanBERT model for span detection (Joshi et al., 2020). SpanBERT was chosen because of its unique training process compared to other BERT models. During training, SpanBERT masks entire spans of text rather than individual tokens, enabling it to better understand contextual spans. This makes it particularly useful for tasks like span prediction. Additionally, SpanBERT was trained on question-answering datasets, which closely resemble the structure of our data and task requirements.

4.2 Data

To fine-tune the model, several data files were used. The initial dataset only contained 50 samples per language, which was insufficient for effective fine-tuning. As mentioned in Section 3.2, we leveraged additional data from Germanic and Romance languages; French, Spanish, Swedish, and Italian, by translating these texts into English. This step enriched the dataset with moe data and was intended to help the model better understand these languages' structures, even if it had not been explicitly trained on them. Additionally, as described earlier, we utilized the GPT-4o-mini model to generate more question-answer samples. In total, the model was trained on 450 question-answer pairs.

To further enhance data quality, we removed special tokens from the text. These tokens, such as '<|endoftext|>', '<0x0A>', '<im_end|>', '</s>', and '<|eot_id|>', introduced noise without providing meaningful information. By stripping these tokens, we ensured cleaner input for the model.

4.3 Offset Mapping

The span annotations in the dataset were provided as character-level positions, indicating where hallucinated spans start and end in the text. However, since SpanBERT operates on tokenized inputs, these character-level spans had to be converted to token-level spans. To achieve this, we used the tokenizer's offset mapping.

During tokenization, SpanBERT records the character boundaries for each token in the text. For example, tokenizing the phrase *"An example"* produces the offset mapping shown in Figure 2.

Token	Offset Mapping
An	(0,2)
example	(3,10)

Figure 2: Offset mapping for the phrase "An example"

For each span in the hard labels, we matched the character positions to their corresponding tokens by checking whether a token's character boundaries included the start or end of the span. For example, if the span annotation is (0, 10), the offset mapping indicates that *"An"* marks the start and *"example"* marks the end of the hallucinated span.

To efficiently locate the start and end tokens based on character indices, we used list comprehension. In cases where a valid token span could not be found, such as when the span exceeded the text length, we assigned a default position of 0. This ensured that the model could process the input without errors during training.

4.4 Data Loader

A custom PyTorch dataset was developed to handle the complex structure of multiple spans per text. This dataset stored the tokenized inputs along with their span labels. A custom data collator was used to prepare batches during training. The collator

calculated the maximum number of spans across samples in a batch, padded the span positions, and stacked input tensors to ensure uniformity.

4.5 Training Setup

For this task, we used the `SpanBERT/spanbert-base-cased` model from the Hugging Face Transformers library [2]. We opted for the base version instead of the large model due to limited GPU computational resources. To balance training speed and resource usage, a batch size of 8 was selected. The learning rate was set to `3e-5`, and the number of epochs was capped at 20. However, early stopping was implemented, terminating training when the validation loss did not improve for two consecutive epochs. The model typically stopped training after 9 to 11 epochs.

We used the AdamW optimizer, which is standard for transformer-based models. Additionally, a custom loss function was implemented to handle multiple spans. This function filtered out invalid spans, which are predicted hallucination spans where the start or end position falls outside the actual text length, and computed the cross-entropy loss for both the start and end logits. The loss was averaged across all valid spans in each batch.

The training loop was structured as follows: for each batch, the model received input features including `input_ids`, `attention_mask`, and span annotations (`start_positions` and `end_positions`). The model performed a forward pass, generating probabilities for each token being the start or end of a hallucinated span. The custom loss function then compared the predicted logits with the ground truth spans. Invalid spans were filtered out, and the cross-entropy loss for both start and end logits was computed and averaged. This loss was backpropagated through the network, updating the model's parameters. The AdamW optimizer adjusted the model's weights accordingly. The total training loss was accumulated across all batches, and at the end of each epoch, the average training loss was calculated.

4.6 Validation

During validation, the model was switched to evaluation mode, which disabled gradient computation to reduce memory usage and improve performance. The validation data was processed in batches, similar to the training process. For each batch, the model received input features such as `input_ids` and `attention_mask` and generated predictions for start and end logits.

The custom loss function was applied to the validation data to compare the predicted logits with the ground truth spans. The total validation loss was accumulated across all batches. At the end of each epoch, the average validation loss was calculated by dividing the total loss by the number of batches.

To prevent overfitting, early stopping was implemented. If the validation loss did not improve for two consecutive epochs, training was terminated. If the validation loss improved, the best loss was updated, and the patience counter was reset.

4.7 Prediction and Evaluation Process

During the prediction phase, the model processed the test dataset by generating start and end logits for each token in the input text. These logits were aggregated across multiple runs to enhance the robustness of predictions. Adaptive thresholds were applied to dynamically determine high-confidence span boundaries. These thresholds were calculated based on the mean and standard deviation of the logits for each text, filtering out low-confidence predictions.

The decoded spans were mapped back to the original text using offset mappings from the tokenizer. Only spans that met specific criteria were retained: the start position had to precede the end position, and the span had to fall within the boundaries of the text. Overlapping spans were merged to reduce redundancy by averaging confidence scores and adjusting boundaries. This approach produced cleaner and more interpretable predictions.

5 Experiments

All experiments were conducted using Google Colab, utilizing the free GPU (NVIDIA Tesla T4 or P100, depending on session availability). The model training process was optimized for this environment to account for hardware limitations, including restricted memory and compute time. This setup facilitated efficient model testing and training without the need for additional infrastructure.

5.1 Data Splits

The dataset was split into training and validation with an 80/20 ratio. The training involved an iterative process across multiple epochs, during which

[2] https://huggingface.co/SpanBERT/spanbert-base-cased

the model generated logits for span start and end positions. A custom loss function was defined to compute the cross-entropy loss for both start and end logits, focusing on valid spans within each batch. The loss was averaged across spans and samples, and the model weights were updated using the AdamW optimizer.

The training loop incorporated early stopping based on validation loss to prevent overfitting. At the end of each epoch, the model's performance on the validation set was evaluated, and if the validation loss did not improve after a set number of epochs, training was halted early.

5.2 Evaluation Measures

The evaluation follows the official scorer used by the task organizers, as implemented in `scorer.py`. Two metrics are used: Intersection over Union (IoU) and Spearman's Rank Correlation.

IoU measures the overlap between predicted and ground truth hallucinated spans:

$$\text{IoU} = \frac{|S_{\text{pred}} \cap S_{\text{true}}|}{|S_{\text{pred}} \cup S_{\text{true}}|} \tag{1}$$

where S_{pred} and S_{true} are the predicted and actual spans, respectively.

Spearman's correlation (ρ) evaluates the rank correlation between predicted hallucination scores and ground truth:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{2}$$

where d_i is the rank difference for each character, and n is the total number of characters.

5.3 Results and Discussion

The evaluation results in Table 1 show variations in hallucination detection performance across languages. While the model achieves reasonable span detection accuracy, the correlation between predicted and actual hallucination scores remains low or negative in most cases. This suggests that the ranking of hallucination confidence scores does not consistently align with the ground truth.

A comparison with the baseline scores in Appendix A.3 highlights the advantages and limitations of our approach. Our model consistently outperforms the baseline in IoU across all languages, demonstrating better hallucination span localization. However, the baseline achieves higher Spearman's correlation in most cases.

The low or negative Spearman's correlation suggests that while the model can detect hallucinated spans, it struggles to rank them accurately. This may be due to over-prediction bias, inconsistencies in training labels, or suboptimal probability calibration.

5.4 Future Work

Future work should focus on improving probability calibration, enhancing multilingual robustness, and refining training data for more consistent cross-lingual performance. While SpanBERT was suitable for this task, exploring alternative models or ensemble approaches could further improve results, albeit at a higher computational cost. Similarly, larger models like SpanBERT-large may offer gains but exceed our current resource limits.

Language	IoU	Correlation
English	0.2943	**0.0116**
Spanish	0.1616	-0.0986
Italian	0.2111	-0.2116
French	0.3095	-0.1521
Swedish	**0.4156**	-0.1177

Table 1: Evaluation results per language. *IoU* represents Intersection over Union, and *Correlation* represents Spearman's correlation.

6 Conclusion

This study explores a SpanBERT-based approach for detecting hallucinated spans in multilingual LLM-generated text as part of the Mu-SHROOM shared task. The results indicate that while the model performs reasonably well in identifying hallucinated spans, particularly in high-resource languages, it struggles with ranking hallucination confidence scores accurately. This is reflected in low or negative Spearman's correlation values.

These findings suggest that while SpanBERT is effective for span detection, further improvements are needed for confidence ranking. Future work should focus on refining probability calibration techniques, improving robustness across multiple languages, and exploring alternative training objectives that incorporate ranking-aware learning. Additionally, ensemble approaches or fine-tuning architectures specifically designed for multilingual hallucination detection could further enhance performance.

References

Lizhou Fan, Lingyao Li, Zihui Ma, Sanggyu Lee, Huizi Yu, and Libby Hemphill. 2024. A bibliometric review of large language models research from 2017 to 2023. *ACM Transactions on Intelligent Systems and Technology*, 15(5):1–25.

Nuno M Guerreiro, Elena Voita, and André FT Martins. 2023. Hallucinations in large multilingual translation models. *arXiv preprint arXiv:2305.13016*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77.

Haoqiang Kang, Terra Blevins, and Luke Zettlemoyer. 2024. Comparing hallucination detection metrics for multilingual generation. *arXiv preprint arXiv:2402.10496*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Junliang Luo, Tianyu Li, Di Wu, Michael Jenkin, Steve Liu, and Gregory Dudek. 2024. Hallucination detection and hallucination mitigation: An investigation. *arXiv preprint arXiv:2401.08358*.

Jiaming Shen, Tianqi Liu, Jialu Liu, Zhen Qin, Jay Pavagadhi, Simon Baumgartner, and Michael Bendersky. 2024. Multilingual fine-grained news headline hallucination detection. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7862–7875.

A Appendix

A.1 Prompt for Translating Validation Data

Translate the <language> content of mushroom-<lang>-val.v2.jsonl to English using Google Translate. Provide the translated content in the same format as the original file (mushroom-<lang>-val.v2.jsonl), but in English.

A.2 Prompt for Synthetic Data Generation

Format:

```
{
  "model_input_text": <input text>,
  "model_output_text": <output text>,
 "hard_labels": [<start and end indices of hallucinated spans>]
}
```

Guidelines:

- Use hard_labels for fabricated spans (character-based indices).
- Leave empty ([]) for factual responses.
- Cover a variety of topics, including history, science, trivia, and personal advice.
- Include factual, partially hallucinated, and fully hallucinated responses.
- Maintain a ratio of **80% hallucinated** and **20% factual** responses.

A.3 Evaluation Comparison

Language	IoU (Ours)	IoU (Baseline)	Correlation (Ours)	Correlation (Baseline)
English	0.2943	**0.0310**	**0.0116**	**0.1190**
Spanish	0.1616	**0.0310**	-0.0986	**0.1190**
Italian	0.2111	0.0104	-0.2116	0.0800
French	0.3095	0.0022	-0.1521	0.0208
Swedish	**0.4156**	0.0308	-0.1177	0.0968

Table 2: Comparison of our hallucination detection model against the Baseline (Neural). *IoU* represents Intersection over Union, and *Correlation* represents Spearman's correlation.