Wrocław University of Science and Technology
Faculty of Information and Communication Technology

Field of study: Sztuczna Inteligencja SZT
Speciality: **-**

MASTER THESIS

Efficient hallucination detection in RAG system using LLMs internal states

Piotr Matys

Supervisor
dr inż. Jan Kocoń

Keywords:
Large Language Models, Natural Language Processing, Contextual Hallucinations, Hallucination Detection

WROCŁAW 2025

Contents

Streszczenie

Duże modele językowe często generują halucynacje, nawet gdy są wzbogacone o zewnętrzną wiedzę w systemach Retrieval-Augmented Generation (RAG), co znacząco ogranicza ich niezawodność w zastosowaniach praktycznych. Niniejsza praca magisterska odpowiada na to wyzwanie, prezentując AggTruth Sum - nowatorską metodę do wykrywania halucynacji kontekstowych w czasie rzeczywistym, opartą na technikach agregacji map uwagi w dużych modelach językowych. Proponowane podejście analizuje rozkład wartości uwagi w dostarczonym kontekście, aby skutecznie identyfikować sytuacje, gdy model LLM tworzy informacje nieoparte o dostarczony fragment tekstu. Kompleksowe eksperymenty przeprowadzone na wielu modelach LLM wykazują, że metody AggTruth konsekwentnie osiągają lepsze wyniki niż istniejące najnowocześniejsze metody, w tym podejścia oparte na stanach ukrytych i Lookback Lens, szczególnie w wymagających scenariuszach transferu wiedzy między zadaniami. AggTruth Sum wyłania się jako optymalny wariant implementacyjny, zapewniający równowagę między skutecznością detekcji halucynacji, a wydajnością obliczeniową. Co istotne, AggTruth Sum wykazuje znaczącą odporność w różnorodnych konfiguracjach testowych, utrzymując wydajność niezależnie od architektury modelu, domeny zadania czy zbioru danych. Proponowana metoda osiąga lepsze wyniki przy znacznie mniejszym zapotrzebowaniu pamięciowym i krótszym czasie obliczeniowym niż porównywane podejścia, co czyni ją praktycznym rozwiązaniem detekcji halucynacji, które można zintegrować w istniejące potoki przetwarzania LLM.

Abstract

Large Language Models (LLMs) often generate hallucinations, even when augmented with external knowledge in Retrieval-Augmented Generation (RAG) systems, significantly limiting their reliability in real-world applications. This thesis addresses this challenge by introducing AggTruth Sum, a novel method for online detection of contextual hallucinations based on attention map aggregation techniques within large language models. The proposed approach analyzes the distribution of attention scores within the provided context to effectively identify when an LLM is fabricating information not supported by the retrieved passage. Comprehensive experiments across multiple LLMs demonstrate that AggTruth methods consistently outperform existing state-of-the-art methods, including hidden states and Lookback Lens approaches, particularly in challenging cross-task transfer settings. AggTruth Sum emerges as the optimal implementation variant, providing the balance of detection performance and computational efficiency. Notably, AggTruth Sum exhibits significant robustness across diverse evaluation settings, maintaining consistent performance regardless of model architecture, task domain or dataset. The proposed method achieves superior performance while requiring significantly less memory and computation time than comparable approaches, making it a practical solution for integrating hallucination detection into existing LLM pipelines.

1. Introduction

1.1 Contextual Hallucinations in LLMs

Large language models (LLMs) have gained significant prominence over the past few years due to their remarkable capabilities across a wide spectrum of natural language processing tasks [16, 25, 33]. These models demonstrate impressive performance in question answering (QA), text summarization, named entity recognition (NER), sentiment analysis, machine translation, and conversational AI applications. Despite their impressive performance, these models consistently exhibit a critical drawback—their tendency to hallucinate. Although definitions of hallucination vary across academic literature, this work adopts the definition characterizing it as the phenomenon of producing responses that are nonsensical or unfaithful to the provided source content [14]. This phenomenon represents a substantial barrier to the deployment of LLMs in real-world applications where accuracy and reliability are paramount considerations, particularly in domains such as healthcare diagnostics, legal document analysis, and financial reporting, where factual precision is essential.

This work specifically focuses on detecting contextual hallucinations in the Retrieval-Augmented Generation (RAG) setup [20]. These hallucinations can be classified as intrinsic hallucinations, meaning that the model output directly conflicts with the provided context. Although RAG effectively enhances the factuality of LLMs by enriching their inputs with additional contextual information, it is not without limitations. The model may still generate hallucinations when presented with noisy or inconsistent context, or due to the inappropriate utilization of otherwise accurate contextual information [12].

1.2 Work based on accepted scientific paper

This work is heavily based on research conducted by our research group, with results published in a paper accepted for the International Conference on Computational Science (ICCS). As the first author of that paper, I ground my current research extensively on those findings, extending them by evaluating additional models, datasets, evaluation configurations, and computational efficiency experiments. My main contributions to the paper include proposing an attention aggregation method called AggTruth Sum, as well as introducing feature selection methods that can effectively identify appropriate transformer heads for detecting contextual hallucinations. My work has also resulted in a robust pipeline enabling quick and reproducible experiments for AggTruth methods. Other AggTruth methods introduced in the paper serve primarily as reference methods

against which AggTruth Sum is compared.

1.3 Objective of the work

The primary objective of this work is to propose and develop a novel method for hallucination detection in Retrieval-Augmented Generation systems based on aggregation techniques applied to attention maps within large language models. The proposed method aims to address several key challenges in current hallucination detection approaches by offering an approach that enables effective identification of contextual hallucinations while being highly robust and adaptable to specific use cases and domains.

Furthermore, this work seeks to develop a computationally efficient solution that can be integrated into existing LLM pipelines without introducing significant performance overhead. By leveraging attention map aggregations, the proposed approach aims to provide features that correlate with hallucination phenomena. Based on these features, the hallucination detection proves effective while maintaining practical implementation feasibility in resource-constrained environments and being computationally efficient. As for machine learning models for which feature importance varies significantly, this work also examines which methods are most suitable for selecting features produced by attention map aggregations.

1.4 Scope of work

The scope of this work revolves around white-box internal state hallucination detection methods in transformer-based large language models. The investigation specifically addresses contextual hallucinations related to the generated responses based on the provided reference texts that have been searched and selected in the retrieval process. However, it is important to note that this work does not examine the retrieval process itself, focusing instead on analyzing how models utilize the textual information once it has been retrieved.

This research emphasizes online hallucination detection, wherein the detector is integrated directly into the decoding process to identify potential hallucinations as they emerge. Nevertheless, the work does not extend to examining methods for modifying the decoding process or reducing hallucinations—the focus remains strictly on detection rather than mitigation strategies.

Chapter 2 presents a comprehensive review of related work in the domain of LLM hallucination detection, examining existing approaches including attention-based methods, hidden state analysis, and black-box methods analyzing model responses. Chapter 3 introduces basic concepts essential to understanding the problem space, including formal definitions of transformer, attention mechanism, and contextual hallucinations. In Chapter 4, both reference and proposed methods are thoroughly described, with detailed explanations of the AggTruth framework, particularly focusing on the novel AggTruth Sum method alongside other attention aggregation techniques. Feature selection methods

are also presented, describing approaches for identifying the most relevant transformer heads for hallucination detection. Chapter 5 outlines the experimental setup, detailing the datasets used for evaluation, the tested language models, approach for generating ground truth labels, and various evaluation configurations for both in-domain and cross-domain testing scenarios. Chapter 6 presents extensive experimental results, analyzing the detection capabilities of different methods across models and tasks. The chapter includes detailed comparisons between AggTruth methods and baseline approaches, highlighting the advantages of the AggTruth Sum method in both performance and computational efficiency. Finally, Chapter 7 concludes the work by summarizing key findings, discussing limitations of the current approach, and proposing directions for future research in the field of hallucination detection for large language models.

2. Related Work

2.1 Literature review

Research on hallucination detection in large language models (LLMs) has evolved along several paths, with some focusing on external response comparison methods, while others examine the model's internal states during generation.

One of the simplest external methods involves assessing the factuality of a model in a zero-resource fashion by comparing inconsistencies between multiple responses [22]. This approach, called SelfCheckGPT, operates under the premise that when an LLM has been properly trained on factual information, multiple sampled responses about the same topic will exhibit consistency, whereas hallucinated content tends to produce divergent or contradictory outputs across samples. The method measures information consistency between different stochastically sampled responses to determine whether statements are factual or hallucinated. SelfCheckGPT implemented several variants for measuring this consistency, including BERTScore for semantic similarity, question-answering techniques to extract key information, n-gram overlap analysis, natural language inference to detect contradictions, and even LLM prompting to evaluate consistency. The approach was particularly valuable for black-box models since it required no access to internal model parameters or external knowledge bases, making it a practical solution for systems where only the final output is accessible.

A similar but more statistically grounded idea, based on semantic entropy, has also been presented [6]. Their approach measures the 'semantic' entropy of LLM generations—an entropy computed over meanings rather than tokens—to detect uncertainties and potential hallucinations. Unlike conventional entropy measures that might be misleadingly high when semantically equivalent answers are expressed differently, their method clusters responses that have similar meanings by determining whether they entail each other bidirectionally. Using both general-purpose LLMs and specialized natural language inference tools, they identify when sentences in a cluster mutually entail one another, indicating semantic equivalence despite different wording. This semantic consistency check across multiple samples provides a more reliable measurement of the model's uncertainty about factual information, with higher semantic entropy potentially indicating areas where the model is prone to hallucination.

The main advantage of these approaches is that they do not require access to the model's internal parameters or training data, making them applicable to black-box models. However, both are limited by the requirement to generate multiple answers, which compromises online efficiency and increases computational cost. Additionally, these methods may not generalize well to tasks where generating diverse outputs is

challenging or where the model's responses are highly deterministic.

In [2] authors developed FactuaL Error detection and correction with Evidence Retrieved from external Knowledge (FLEEK), a model-agnostic approach that tackles hallucination detection through external knowledge verification. FLEEK functions by first identifying potentially verifiable factual claims within generated text, then automatically transforming these claims into structured queries. These queries are used to retrieve supporting evidence from both curated knowledge graphs and open web sources. The system evaluates the retrieved evidence against each claim to determine factual accuracy, highlighting verified, uncertain, and incorrect statements with different colors in an interactive interface. When inaccuracies are detected, FLEEK generates factually corrected alternatives based on the retrieved evidence. The solution employs a transparent verification process where users can inspect the extracted facts, generated queries, and supporting evidence that contributed to each verification decision. While FLEEK demonstrates strong capability in identifying factual inconsistencies in model outputs, its major limitation is that it waits for the whole response to be generated and only then verifies it based on available knowledge resources.

Fine-tuning has also been proposed as a strategy to mitigate hallucinations [7]. This approach can be highly effective for specific tasks, as it allows the model to adapt to domain-specific data and reduce the likelihood of hallucinations in targeted scenarios. However, a significant drawback is the risk of catastrophic forgetting [15], where the model loses performance on tasks not included in the fine-tuning process. Moreover, fine-tuning requires access to labeled data and computational resources, which may not always be feasible.

The internal state analysis approach offers an alternative perspective. For example, analyses of hidden states [1] can provide insights into the model's reasoning process and help identify hallucinations. The main advantage of this approach is its potential to detect hallucinations without relying on external references or multiple generations. However, authors [1] have only considered binary true-false datasets, which may not capture the full complexity of contextual hallucination phenomena. Moreover, as the authors stated, there is no general rule based on which layer's hidden states should be leveraged.

Building on this foundation, [4] demonstrated that attention maps can also serve as effective indicators for hallucination detection. This approach leverages the interpretability of attention mechanisms, allowing for the identification of anomalous attention patterns associated with hallucinated content. Notably, attention-based methods have shown better performance in transferring between different tasks while requiring a smaller number of input features. However, the effectiveness of the attention map aggregation approach proposed by [4] is constrained by its reliance on aggregating attention over all prompt tokens, making it highly sensitive to variations in the input. Furthermore, this method struggles with longer generated responses, as the lookback ratio used in their approach tends to diminish to zero, limiting its applicability in such cases. As authors also evaluated their approach on 3 datasets using one large language model (Llama-2), which enabled them to establish a benchmark for hallucination detection performance across different text generation tasks, this evaluation scope remains limited. Testing on such a small number of datasets and a single model architecture constrains

the generalizability of their findings to diverse real-world applications and emerging model architectures.

In our work [23], we introduced attention weights aggregation methods for contextual hallucination detection. Building on foundational observations that attention map analysis provides valuable insights for detecting hallucinations in large language models during context-based retrieval-augmented generation (RAG) tasks, we proposed a more focused approach. Rather than examining the entire attention matrix, our method concentrates specifically on attention patterns related to the provided context or passage. This targeted analysis creates attention-based features by examining how newly generated tokens attend to the provided passage within a defined window. While our initial work demonstrated the effectiveness of this approach, several dimensions remained unexplored, including the influence of different window sizes on detection performance, the method's cross-lingual capabilities when applied to non-English content, and more comprehensive feature selection techniques examination that could answer the question which feature selection method is most suitable. Moreover, the computational efficiency of proposed methods were also not conducted.

2.2 Summary of the literature review

Current approaches to hallucination detection in LLMs offer diverse methodologies but each presents significant limitations. External response comparison methods like SelfCheckGPT assess factuality by measuring consistency across multiple outputs, but compromise efficiency by requiring repeated generations and struggle with tasks producing deterministic responses. Similarly, semantic entropy approaches provide statistically robust uncertainty measurements through bidirectional entailment clustering but face identical efficiency constraints from multiple sampling requirements. Knowledge-based methods like FLEEK leverage external verification against curated databases and web sources but operate reactively, requiring complete response generation before verification begins. Fine-tuning strategies can effectively reduce domain-specific hallucinations but risk catastrophic forgetting and demand substantial labeled data and computational resources. Internal state analysis via hidden states offers promising parameter-free detection but has been limited to binary classification scenarios with unclear layer selection guidelines. Attention map aggregation methods like Lookback Lens demonstrate strong cross-task transferability but suffer from input sensitivity, diminishing effectiveness with longer responses, and limited evaluation scope across datasets and model architectures. Our AggTruth method improves on these approaches through targeted attention analysis of context-response relationships, though questions remain regarding different window sizes, cross-lingual capabilities, and comprehensive feature selection techniques that could further enhance computational efficiency.

3. Basic Concepts

3.1 Tensor

A tensor can be represented as a (potentially multidimensional) array. Tensors are mathematical objects that can be used to describe physical properties, similar to scalars and vectors. A scalar, vector, and matrix are, respectively, tensors of order zero, one, and two. Due to this property, tensors are a generalization that represents an n-dimensional structure [26]. An example of a third-order tensor is a color image in RGB format, where its width and height are the first and second dimensions, and the third dimension corresponds to its color channels.

3.2 Classification

Classification is one of the three main tasks in machine learning and represents a supervised learning problem. The objective is to develop a model that, given input data x (features), can correctly assign objects to specific classes y [3]. An example of classification is a spam filter, which is trained on a set of emails labeled as spam or not spam and must learn to classify new messages [9]. During training, the model has access to both features and labels, allowing it to learn the relationships between them. A well-trained model should be able to accurately classify new input samples that represent objects from the classes encountered during training. When the model distinguishes between only two classes, the task is called binary classification; when the number of classes increases, it becomes multiclass classification [9].

A classifier outputs the probabilities that a given input sample belongs to each of the classes considered during training. In binary classification, a random classifier would achieve 50% prediction accuracy. Most classifiers are eager learners, acquiring their predictive abilities during training using a training dataset. Examples of such classifiers include neural networks, random forests, and logistic regression. In contrast, lazy algorithms do not learn from the training set but use it during inference; an example is the k-nearest neighbors algorithm [3]. Training a classification model involves optimizing a loss function, which quantifies the error made by the classifier on the training data. The most popular loss function is cross-entropy, defined as:

$$-\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3.1}$$

where n is the number of training examples, y_i is the true label of the i-th sample, and $\hat{y}_i$ is the predicted probability for that sample. Cross-entropy loss is used in neural networks and logistic regression. The value of the loss function indicates the error made by the classifier, and the goal of training is to minimize it. In neural networks, to obtain class probabilities, the output layer uses the sigmoid function for binary classification and the softmax function for multiclass classification. For multiclass problems, the softmax activation is defined as:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{3.2}$$

where x is the raw output from the neural network and k is the number of classes. The i-th output of the softmax function represents the predicted probability that the sample belongs to class j [3].

3.3 Neural Networks

A deep neural network is a set of machine learning methods known as deep learning. It uses interconnected nodes or neurons in a layered structure, either as a sequence or a directed graph, where each layer is a parameterized linear transformation of the input vector, followed by a nonlinear transformation [10]. This adaptive structure learns from its errors using a process called backpropagation [3]. In its second phase, the backpropagation algorithm works by moving from the output layer to the input layer, propagating the error gradient. When the algorithm computes the gradient of the loss function with respect to each parameter in the network, it uses these gradients to update each parameter. The update of the connection weights, which link neurons in subsequent layers, is performed using the chain rule [10].

3.4 Transformer

3.4.1 Tokenization

Tokenization represents a critical preprocessing step in natural language processing where text is segmented into smaller units called tokens. This process creates the bridge between human-readable text and the numerical representations that language models can process. Early tokenization methods relied primarily on simple word boundaries like spaces or punctuation, but these approaches proved inadequate for handling complex linguistic structures and frequently encountered unknown words. Modern large language models employ sophisticated subword tokenization techniques that decompose text in more flexible ways. Methods like Byte Pair Encoding (BPE) [28] and WordPiece [31] systematically divide text into meaningful subword units by identifying common patterns in a training corpus. BPE works by iteratively combining the most frequently occurring

adjacent character pairs, while WordPiece selects subword fragments that optimize language model probability.

3.4.2 Recurrent Neural Networks

Sequence modeling, in the context of machine learning, refers to a type of model that operates on a sequence of values in discrete time steps. In sequence modeling, the current output depends on the previous input, and the input length is not fixed. To solve such problems, recurrent neural networks have an "internal state" that is passed as part of the input data in the next iteration [9]. Since the output of a recurrent neuron at time step t is a function of all inputs from previous time steps, it can be said to have a form of memory. The part of the neural network that retains some state across different time steps is called a memory cell [9]. These networks are usually trained by "unrolling" them over several iterations, effectively creating a network consisting of a chain of modules with shared weights. Recurrent networks were originally used for tasks such as natural language translation or audio processing, but they have primarily found their application in time series modeling.

Unfortunately, recurrent neural networks are difficult to train because they suffer from exploding or vanishing gradients. As the algorithm moves to lower layers, the gradients often become smaller and smaller. As a result, the gradient update leaves the connection weights of the lower layers practically unchanged, and the training never converges to a good solution. The opposite phenomenon is when the gradients grow larger and larger, causing the layers to receive very large weight updates, which results in the learning algorithm diverging [9].

3.4.3 Hidden states

In models which process sequences, hidden states are internal vector representations that encode evolving contextual meaning as information traverses through the network architecture [10]. Each layer computes a new hidden state that captures both the current input and relevant contextual information from the sequence. In transformers, these hidden states are processed through self-attention and layer normalization, while in recurrent architectures (such as LSTMs and GRUs), they're updated through gated mechanisms that control information flow. As signals propagate through network layers, hidden states are progressively refined, integrating increasingly complex linguistic patterns. Early layers typically capture surface-level features, while deeper layers abstract higher-level concepts like entity relations or discourse structure. In the final layer, hidden states represent deeply contextualized understanding of the input, combining both local and global context.

3.4.4 Transformer architecture

The transformer architecture is a deep neural network model widely adopted in artificial intelligence, especially in natural language processing. It was introduced to address the challenges of modeling long sequences and to enable efficient parallel training.

Unlike recurrent neural networks, which process data sequentially, transformers operate on entire sequences simultaneously. This approach accelerates training and allows the model to capture dependencies between distant elements in the input.

The architecture consists of two main components: the encoder and the decoder. The encoder analyzes the input data and generates contextualized representations, while the decoder uses this information to produce the output, such as a translated sentence or summary. Both components are composed of multiple layers that refine the information as it passes through the network.

The transformer architecture has enabled the development and training of large-scale language models. Its ability to process sequences in parallel and handle long-range dependencies has established it as a standard in the field of artificial intelligence.

3.4.5 Positional Encoding

Unlike recurrent architectures that process tokens sequentially, transformer models handle all input tokens simultaneously. This parallel processing creates a fundamental challenge: the self-attention mechanism is inherently position-agnostic, treating input tokens as an unordered set. Without positional context, semantically distinct sentences like "cat chases ball" and "ball chases cat" would be represented identically, despite conveying opposite meanings.

To overcome this limitation, transformers incorporate positional encodings—specialized vectors added to token embeddings before they enter the attention layers. These encodings embed spatial information about each token's location within the sequence. The original Transformer paper [29] introduced fixed sinusoidal functions to generate these positional representations, while later implementations adopted learnable positional embeddings [30]. Fixed encodings provide mathematical regularities that generalize well to unseen sequence lengths, whereas learned encodings can better adapt to dataset-specific positional patterns. In both approaches, the addition of positional information enables the model to differentiate between tokens based on both their semantic content and sequential position.

For large language models (LLMs), effective positional encoding becomes increasingly crucial when processing extended contexts. As sequence lengths grow, proper positional representation helps maintain coherence across distant tokens, preserve the alignment between generated content and input prompts, and support complex reasoning chains. Inadequate handling of positional information can manifest as structural inconsistencies, logical contradictions, or hallucinated content, particularly when models operate near their maximum context window capacity.

To overcome the constraint of maximum context length in large language models, Rotary Position Embedding (RoPE) [21] scaling offers a promising technique for extending their processing capabilities beyond their initial training parameters. By modifying the rotary base parameter from its standard value of 10000 to a larger value, researchers have demonstrated significant improvements in models' ability to maintain coherent representations across longer sequences. This mathematical adjustment effectively "stretches" the positional encoding, allowing models to interpret token positions beyond their original training context without requiring complete retraining. Recent

advancements have shown that fine-tuning RoPE-based models with carefully calibrated base values can dramatically enhance extrapolation capabilities, enabling context lengths up to 1 million tokens with minimal additional training.

3.4.6 Dense embeddings

Dense embeddings are representations in the form of vectors of real numbers. Each vector represents one token, which can be a word or a sub-word. The vectors corresponding to the tokens that have similar linguistic features and appear in a similar context, which indicates similar meaning, are close in a vector space. The representations are learned by an embedding model during training on a corpus of text. In recent years, the most effective models are based on the Transformer Encoder architecture, such as BERT [5].

3.4.7 Multi-Head Attention

Transformer employs a multi-head attention mechanism, enabling the model to selectively attend to the most relevant tokens within a sequence. Multi-head attention is constructed from H individual scaled dot-product attention modules, whose outputs are concatenated and subsequently processed by a linear transformation. Each attention module receives as input a set of queries Q, keys K, and values V, all of which are generated by applying linear projections to the input. In the self-attention configuration, the same word embeddings, combined with positional encodings ($WE + PE$), are used as input for all three projections. The attention weights are computed by taking the dot product of the Q and K matrices, producing a similarity score between tokens. These scores are then scaled and normalized using the softmax function. The resulting attention weights are applied to the V matrix, yielding a filtered value matrix A, where features deemed more significant receive greater emphasis (see eq. 3.3). Multi-head attention consists of H such modules, each capable of capturing different aspects of linguistic information, resulting in diverse filtered value matrices.

$$\begin{aligned} Q = IW_q, K = IW_k, V = IW_v, \\ A = \text{softmax}(\frac{QK^T}{\sqrt{d}})V, \end{aligned} \tag{3.3}$$

where $I \in \mathcal{R}^{N \times d}$ represents input vectors, N is the length of the input sequence, d is is the dimension of hidden layers, $W_q, W_k, W_v \in \mathcal{R}^{d \times d}$ are the projection matrices.

3.5 Semantic Search

Semantic search refers to information retrieval techniques that go beyond simple keyword matching by understanding the contextual meaning of queries and documents. Modern semantic search systems leverage deep learning models, such as transformers,

to generate dense vector representations (embeddings) of both queries and documents. These embeddings capture semantic relationships, enabling the retrieval of relevant information even when there is little or no lexical overlap between the query and the documents. Semantic search has been widely adopted in various applications, including web search, question answering, and recommendation systems.

3.6 Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) is a framework that combines the strengths of large language models with external knowledge retrieval. In RAG, a retriever component first selects relevant documents from a large corpus based on the input query, and then a generator component, typically a language model, synthesizes a response using both the retrieved documents and the original query. This approach enhances the factual accuracy and grounding of generated responses, making it particularly effective for open-domain question answering and knowledge-intensive tasks [20]. RAG has demonstrated significant improvements over traditional generation-only models by incorporating up-to-date and domain-specific information from external sources.

3.7 Contextual Hallucinations

Contextual hallucinations in large language models (LLMs) refer to instances where the model generates content that is plausible within the given context but is not factually accurate or grounded in reality. These hallucinations are particularly concerning because they can mislead users by producing information that appears coherent and contextually appropriate, yet lacks a factual basis. The phenomenon of contextual hallucination highlights the challenges in ensuring the reliability of LLMs, especially in information retrieval and decision-making applications [13].

3.8 LLM as a Judge

The ability of large language models (LLMs) to mimic human reasoning and assess inputs according to established criteria has led to the emergence of the "LLM-as-a-Judge" paradigm. Research demonstrates that LLMs are scalable, adaptable, and cost-efficient, making them suitable for handling an increasing range of evaluative tasks that were once the domain of human evaluators [11]. These characteristics enable LLMs to be applied flexibly across diverse evaluation contexts and requirements. Consequently, the use of LLMs for evaluation purposes has expanded significantly in recent years. While LLMs were initially developed for tasks involving language understanding and generation, advances in training methods such as Reinforcement Learning from Human Feedback (RLHF) have improved their alignment with human judgment and reasoning.

This progress has facilitated the shift of LLMs from purely generative applications to evaluative functions. In essence, the LLM-as-a-Judge approach involves leveraging LLMs to assess objects, actions, or decisions based on specified rules, standards, or preferences.

4. Methods

4.1 Research problems

Due to the objective of this work, which is to develop a robust and efficient method for hallucination detection in Retrieval-Augmented Generation (RAG) systems, the following research problems have been identified:

1. **Robustness to model and task selection:** Is the approach effective regardless of the choice of LLM and the specific task?
2. **Impact of window size in aggregation:** How does the window size of the span affect hallucination detection performance? Is it possible to use smaller windows without loss of quality, thus detecting hallucinations in the smaller spans of tokens?
3. **Cross-lingual transfer:** Can the method effectively detect hallucinations when applied to models and tasks in languages other than the source language?
4. **Impact of feature selection:** Which attention head selection method is most suitable for reducing model complexity while maintaining detection accuracy? Furthermore, is it possible to train a hallucination detector using a reduced set of features—selected attention heads—and achieve better detection accuracy compared to using all available features without selection?
5. **Optimization of time and memory:** Which method is the most computationally efficient, and how can its time and memory consumption be minimized without compromising detection accuracy?

Solving the above research problems is essential to achieve the main goal of this work, which is to create a robust and efficient hallucination detection method suitable for practical use in various RAG scenarios.

4.2 Reference Methods

4.2.1 Hidden States

Recent research suggests that the hidden states of large language models (LLMs) can be leveraged as informative features for hallucination and factuality detection [1]. This approach, referred to as Statement Accuracy Prediction based on Language Model

Activations (SAPLMA), is grounded in the hypothesis that the hidden layers of an LLM encode signals about whether the model "believes" a generated statement to be true or false. However, it is not clear which layers should be taken into account. Although the final hidden layer is directly involved in the prediction of the next token, it may not always be optimal for assessing the factuality of the response.

Building on these insights, following [4], hidden state representations from the upper layers of the model were taken, specifically from layers L, $L-4$, and $L-8$, where L denotes the total number of layers in the architecture. For each selected layer, the hidden states are averaged across tokens within a specified window, producing a compact feature vector. These aggregated representations are then used as input features for downstream classification models tasked with distinguishing between factual and hallucinated statements. This strategy enables the classifier to exploit the rich, contextual information embedded in the model's internal activations, potentially improving the detection of hallucinations in generated text [23].

4.2.2 Lookback-Lens

Lookback-Lens is a method designed to detect contextual hallucinations in large language models by analyzing the distribution of attention within transformer architectures [4]. The core idea is to compute a *lookback ratio*, which quantifies the proportion of attention focused on the prompt tokens versus the attention placed on newly generated tokens during the generation process. In contrast to our approach presented in [23], the attention placed on the entire provided prompt is analyzed (including the system prompt and query), which naturally may lack robustness for changing input.

Formally, for a transformer with L layers and H attention heads, processing an input sequence $X = \{x_1, x_2, \ldots, x_N\}$ and generating tokens $Y = \{y_1, y_2, \ldots, y_{t-1}\}$, the lookback ratio for each head h in layer l at time step t is defined as:

$$A_t^{l,h}(\text{prompt}) = \frac{1}{N}\sum_{i=1}^{N}\alpha_{h,i}^{l}, \tag{4.1}$$

$$A_t^{l,h}(\text{new}) = \frac{1}{t-1}\sum_{j=N+1}^{N+t-1}\alpha_{h,j}^{l}, \tag{4.2}$$

$$\text{LR}_t^{l,h} = \frac{A_t^{l,h}(\text{prompt})}{A_t^{l,h}(\text{prompt}) + A_t^{l,h}(\text{new})}, \tag{4.3}$$

where $\alpha_{h,i}^{l}$ and $\alpha_{h,j}^{l}$ are the softmax-normalized attention weights assigned to prompt and newly generated tokens, respectively.

To detect hallucinations, the lookback ratios across all heads and layers are concatenated into a feature vector for each generation step. For a span of generated text $\{y_t, y_{t+1}, \ldots, y_{t+T-1}\}$, the corresponding lookback ratio vectors are averaged to form a single feature vector $\bar{v}$. A logistic regression classifier is then used to predict whether the span is factual or hallucinated based on $\bar{v}$ [4].

4.2.3 AggTruth

Each response generated by a transformer model results in a four-dimensional tensor with shape $L \times H \times N \times C$, where L is the number of layers, H is the number of attention heads, N is the number of generated tokens, and C is the number of context tokens. To enable effective training of a hallucination detection model, it is necessary to aggregate the attention scores assigned to the passage for each generated token, thereby reducing the dimensionality and size of the resulting data.

For every generated token, its attention allocation over the passage is summarized into a lower-dimensional representation.

Let $a_{l,h,t,i}$ denote the attention score for layer $l \in \{1, \ldots, L\}$, head $h \in \{1, \ldots, H\}$, generated token $t \in \{1, \ldots, N\}$, and context token $i \in \{1, \ldots, C\}$. The vector $\mathbf{a}_{l,h,t}$ collects all such scores for the context tokens at a given layer, head, and generated token.

Main to the AggTruth approach is the observation that the attention map produced during the generation of a new token reflects the influence of all preceding tokens, but not the token currently being generated. In other words, the attention scores are computed before the new token is actually selected, meaning that the choice of token—whether determined by greedy decoding or another sampling strategy—does not affect the attention map at that step. A token only begins to impact the attention distribution in subsequent generation steps, after it has been produced. Therefore, to accurately construct features that characterize token t, it is essential to utilize the attention map obtained during the generation of token t+1. Failing to do so results in each token being represented by features that actually correspond to its predecessor, rather than itself.

Apart from method which has been introduced by me, three other aggregation strategies are proposed in the paper [23]:

AggTruth CosSim

This method characterizes each attention head within a layer by the attention it allocates to passage tokens during the generation of a new token. To quantify the similarity of attention patterns among heads, the cosine similarity is computed between each head and all other heads in the same layer, and the results are averaged (see Equation 4.4).

$$\text{CosSim}_{l,h,t} = \frac{1}{H-1} \sum_{\substack{h'=1 \\ h' \neq h}}^{H} \frac{\mathbf{a}_{l,h,t} \cdot \mathbf{a}_{l,h',t}}{\|\mathbf{a}_{l,h,t}\| \|\mathbf{a}_{l,h',t}\|} \tag{4.4}$$

This yields a single value per head, reflecting how closely its attention distribution aligns with those of other heads in the layer for a given token. The underlying intuition is that, when the model generates unreliable or hallucinated content, the attention becomes similar to each other, indicating a lack of distinct focus.

Beyond quantifying the amount of attention, it is also important to understand how the model's attention is distributed across context tokens. In this work, attention scores are interpreted as pseudo-probabilities, enabling the use of metrics from probability

theory. To ensure the sum-to-one property without losing information about the original scale and distribution, for Aggtruth Entropy and JS-DIV the attention vector is augmented with an additional value equal to the difference between one and the sum of the attention scores, rather than simply normalizing [23].

AggTruth Entropy

Uncertainty in attention allocation can be indicative of hallucination. The entropy-based approach (Equation 4.5) measures whether the model's attention is concentrated on specific tokens or dispersed across the entire context.

$$\text{Entropy}_{l,h,t} = -\sum_{i=1}^{C} a_{l,h,t,i} \log_2 a_{l,h,t,i} \tag{4.5}$$

AggTruth JS-Div

Hallucinated outputs can be viewed as outliers, often arising when the model lacks sufficient knowledge for the task. The central hypothesis is that hallucinated and factual outputs can be distinguished by examining the consistency of attention distributions across layers. Treating attention scores as probability distributions, the average attention distribution for all heads in a layer is computed. The Jensen-Shannon divergence is then used to assess how much each head's distribution deviates from this average (Equation 4.6) [23].

$$\text{JS-Div}_{l,h,t} = \sqrt{\frac{1}{2}\sum_{i=1}^{C}\left(a_{l,h,t,i} \ln \frac{a_{l,h,t,i}}{m_{l,h,t,i}} + a_{l,\text{ref},t,i} \ln \frac{a_{l,\text{ref},t,i}}{m_{l,h,t,i}}\right)} \tag{4.6}$$

where:

$$a_{l,\text{ref},t,i} = \frac{1}{H}\sum_{h=1}^{H} a_{l,h,t,i}, \qquad m_{l,h,t,i} = \frac{a_{l,h,t,i} + a_{l,\text{ref},t,i}}{2}$$

4.3 Proposed Method

4.3.1 AggTruth Sum

AggTruth Sum is also one of the aggregation methods within the AggTruth method that constructs dedicated features by aggregating attention scores. Among other aggregation methods, it is the most straightforward, as it simply computes the sum of the attention weights assigned to the passage tokens during the generation of a given token (Equation 4.7) [23].

$$\text{Sum}_{l,h,t} = \sum_{i=1}^{C} a_{l,h,t,i} \tag{4.7}$$

This sum provides a straightforward measure of how much total attention the model allocates to the context at each generation step [23]. A higher sum could indicate that the model is strongly referencing the provided context when generating the next token, which may suggest better grounding and alignment with the source information. Conversely, a lower sum could imply that the model is relying more on previously generated tokens or its internal knowledge, potentially increasing the risk of hallucination. Among the other AggTruth aggregation methods, summing the attention scores stands out as the simplest, most interpretable, and computationally efficient approach. This properties are particularly valuable in production environments, where computational efficiency and interpretability are critical.

4.3.2 Feature (heads) selectors

Central tendency measure ratio ($\mathbf{Center_r}$)

Any central tendency statistic that can be computed for both hallucinated and non-hallucinated examples can be utilized in this approach. The core idea is to identify the r fraction of attention heads that show the most pronounced difference in a chosen central tendency measure between the two classes. Specifically, the top $r/2$ heads with the highest ratio and the bottom $r/2$ heads with the lowest ratio between the classes are retained. This method serves as a straightforward baseline for head selection, providing a simple yet interpretable way to focus on the most discriminative heads [23].

Let $M_h^{(0)}$ and $M_h^{(1)}$ denote the central tendency measure (e.g., median) for head h in the non-hallucinated and hallucinated classes, respectively. The ratio for each head is:

$$R_h = \frac{M_h^{(1)}}{M_h^{(0)}} \tag{4.8}$$

Select the top $\frac{r}{2}$ heads with the highest R_h and the bottom $\frac{r}{2}$ heads with the lowest R_h.

Above random feature performance ($\mathbf{Random_{n,k}}$)

In this method, a synthetic feature with random values is added to the dataset to establish a baseline for feature importance. Only those heads whose absolute importance, as measured by the magnitude of their logistic regression coefficients, exceeds that of the random feature are retained. This process is repeated n times, and only heads that are selected in at least k iterations are kept. This approach is inspired by the Boruta feature selection algorithm [18], but simplifies it by omitting the binomial test and using logistic regression instead of random forests. While the original Boruta algorithm is more computationally intensive due to the need for more iterations, our simplified version achieves comparable results with significantly reduced computation time. This method helps ensure that only features with importance above random noise are selected, improving model robustness [23].

Let $\beta_h^{(i)}$ be the absolute value of the logistic regression coefficient for head h in iteration i, and $\beta_{\text{rand}}^{(i)}$ for the random feature. Head h is selected in iteration i if:

$$|\beta_h^{(i)}| > |\beta_{\text{rand}}^{(i)}| \tag{4.9}$$

A head is retained if it is selected in at least k out of n iterations.

Above random feature performance - positive coefficients $\left(\mathbf{Random^{+}_{n,k}}\right)$

This variant follows the same procedure as the above method, but restricts selection to heads with positive logistic regression coefficients. By focusing only on features that contribute positively to the prediction, this approach can be particularly useful in identifying heads that strongly indicate hallucination [23].

Same as for $\mathbf{Random_{n,k}}$, but only heads with positive coefficients are considered:

$$\beta_h^{(i)} > \beta_{\text{rand}}^{(i)} \quad \text{and} \quad \beta_h^{(i)} > 0 \tag{4.10}$$

A head is retained if it meets the criterion in at least k out of n iterations.

Lasso based selection (Lasso)

Lasso-based selection leverages L1 regularization, which encourages sparsity by shrinking some model coefficients to exactly zero. Only features with non-zero coefficients after fitting the Lasso model are retained. This results in a compact and interpretable set of features, as irrelevant or redundant heads are automatically excluded. Lasso is especially effective when dealing with high-dimensional data, as it helps prevent overfitting and enhances model generalization [23].

Let β_h be the coefficient for head h after fitting the Lasso model:

$$\text{Select all } h \text{ such that } \beta_h \neq 0 \tag{4.11}$$

Spearman correlation with the target ($\mathbf{Spearman_r}$)

Among attention heads there is a high degree of multicollinearity, making traditional selection methods based on inter-feature correlation or variance inflation factor less effective. Instead, this method selects heads based on their Spearman correlation with the target variable. The top r fraction of heads with the highest statistically significant Spearman correlation coefficients are chosen. Additionally, an automatic mode ($r = \text{auto}$) is provided, where only features with a correlation greater than half of the maximum observed value are selected. This approach ensures that only the most relevant and statistically significant features are included in the model [23].

Let ρ_h be the Spearman correlation coefficient between head h and the binary target y:

$$\rho_h = \text{Spearman}(X_h, y) \tag{4.12}$$

Select the top r fraction of heads with the highest statistically significant ρ_h. For $r = \text{auto}$, select heads with $\rho_h > 0.5 \cdot \max_h |\rho_h|$.

Point Biserial correlation with the target ($\mathbf{PBI_r}$)

In addition to the feature selection methods proposed in our work [23], in this work the use of Point Biserial correlation for selecting attention heads is introduced. Point

Biserial correlation is a statistical measure that quantifies the relationship between a continuous variable (such as an attention head feature) and a binary target variable (e.g., hallucinated vs. non-hallucinated). Unlike Spearman correlation, which assesses monotonic relationships, Point Biserial correlation is specifically designed for scenarios where the target is binary, making it particularly well-suited for binary classification tasks [17]. By ranking heads according to their Point Biserial correlation with the target, features that are most discriminative for distinguishing between the two classes can be more effectively identified. As with the Spearman-based approach, the top r fraction of heads with the highest statistically significant Point Biserial correlation values are selected, and for an automatic mode ($r = \text{auto}$), only features with a correlation greater than half of the maximum observed value are included.

Let $r_{pb,h}$ be the Point Biserial correlation coefficient for head h:

$$r_{pb,h} = \frac{\overline{X}1, h - \overline{X}0, h}{s_h} \sqrt{\frac{n_1 n_0}{n^2}} \tag{4.13}$$

where $\overline{X}1, h$ and $\overline{X}0, h$ are the means of head h for the two classes, s_h is the standard deviation, n_1 and n_0 are the sample sizes for each class, and $n = n_1 + n_0$. Select the top r fraction of heads with the highest statistically significant $r_{pb,h}$. For $r = \text{auto}$, select heads with $r_{pb,h} > 0.5 \cdot \max_h |r_{pb,h}|$.

5. Experimental Setup

To construct a high-quality dataset for subsequent evaluation, a comprehensive workflow (Figure 5.1) has been developed that transforms each prompt into features ultimately utilized for training a hallucination detection classifier. The workflow extracts LLM responses with their associated attention maps and forwards these responses to GPT-4o for assessment. The procedure then isolates attention scores assigned to passage tokens and pairs each generated token with a binary label (1 indicating hallucinated content, 0 indicating factual content). These attention scores are subsequently consolidated using the AggTruth Sum aggregation methodology, and all generated tokens are segmented through a windowing approach, whereby they are converted into overlapping sequences of n tokens each, advancing one token at a time. Any sequence containing a hallucinated token receives a label of one, otherwise zero. Moreover, for each sequence, the features are averaged across all tokens, yielding a single feature that represents an individual attention head. Following the processing of all examples, the results are merged into a consolidated dataset. This data then passes through a selector, producing the final feature set that enables classifiers to identify potential hallucinations within LLM responses.

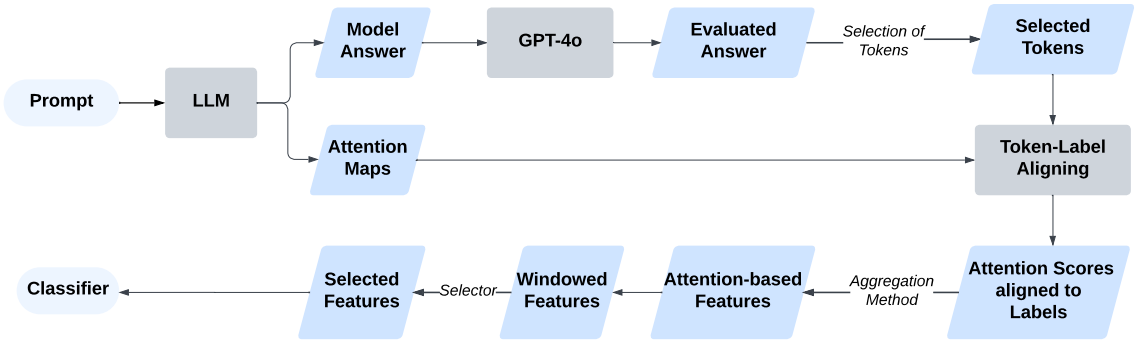

Figure 5.1: End-to-end pipeline of the process. It begins with a *Prompt* passed to an LLM and eventually results in *Selected Features* based on which the final *Classifier* detects potential hallucinations in the obtained answer. (source: [23])

5.1 Datasets

A comprehensive dataset of language model responses with token-level annotations was constructed (Table 5.1) to support the training of classifiers for hallucination

detection. The analysis was limited to a context length of 4,096 tokens, reflecting the constraints introduced by the Sliding Window Attention Mechanism [8], which restricts the extraction of attention scores across the entire passage once the window size is exceeded [23].

Evaluation was conducted on two fundamental natural language processing tasks: Question Answering (QA) and Summarization. For QA, the dataset included 1,535 examples from Natural Questions (NQ) [19] and 896 from HotPotQA [32]. The summarization task incorporated 1,000 examples each from CNN/Daily Mail (CNN/DM) [27] and XSum [24]. The choice of these datasets is conditioned by experiments conducted by [4] and then extended by our research group in [23].

To enable cross-language evaluation between English and Polish, the POQUAD_v2 and POLQA datasets were additionally included apart from the datasets used in [23]. Both of them are extractive question answering datasets, where POQUAD_v2 is a Polish adaptation of the SQuAD dataset, while POLQA is a curated dataset for Polish open-domain QA.

Dataset	Task Type	Number of Samples
Natural Questions (NQ)	QA	1,535
HotPotQA	QA	896
CNN/Daily Mail (CNN/DM)	SUM.	1,000
XSum	SUM.	1,000
POQUAD_v2	QA	1,500
POLQA	QA	710

Table 5.1: Evaluation datasets used during experiments. (source: own elaboration)

5.2 Tested LLMs

A diverse selection of large language models (LLMs) was incorporated into the experimental setup to ensure a comprehensive evaluation of the AggTruth Sum hallucination detection methodology in different architectural designs and training paradigms. These models were previously tested by our research group in [23], and to extend the analysis into cross-linguistic capabilities, the Polish large language language model Bielik-11B-v2.3-Instruct was added. Table 5.2 presents the configuration for each examined model.

5.3 Labels generation

Responses for all dataset entries were generated using greedy decoding, resulting in outputs that included both factual and hallucinated content. Ground truth labels at the token level were assigned using GPT-4o, following the "LLM-as-a-Judge" paradigm, in

LLM	Layers	Heads	Hidden Size
meta-llama/Llama-2-7b-chat-hf	32	32	4096
meta-llama/Llama-3.1-8B-Instruct	32	32	4096
unsloth/gemma-2-9b-it-bnb-4bit	42	16	3584
microsoft/Phi-3.5-mini-instruct	32	32	3072
speakleash/Bielik-11B-v2.3-Instruct	50	32	4096

Table 5.2: Configuration with key hyperparameters of examined LLMs. (source: own elaboration)

which advanced language models are employed to assess the outputs of other models. Utilizing GPT-4o as an automated judge provides several benefits, including scalable and cost-effective annotation of large datasets, thereby addressing the limitations of manual labeling such as time requirements and inter-annotator variability. Recent studies indicate that GPT-4o's judgments correspond with human assessments in 97% of cases, demonstrating high reliability. Additional validation involved a human expert annotating 75 samples, yielding a Cohen's Kappa of 0.7 between the expert and GPT-4o, which reflects substantial agreement. The factuality of all model responses across different datasets is presented in 5.3.

LLM	NQ	HotPotQA	CNN/DM	XSum	POLQA	POQUAD_v2
meta-llama/Llama-2-7b-chat-hf	54.9	62.0	74.4	63.2	44.6	49.9
meta-llama/Llama-3.1-8B-Instruct	75.8	56.1	86.8	77.0	65.8	83.5
unsloth/gemma-2-9b-it-bnb-4bit	84.4	85.3	92.6	75.5	80.4	88.6
microsoft/Phi-3.5-mini-instruct	61.0	74.2	75.7	66.6	44.6	57.8
speakleash/Bielik-11B-v2.3-Instruct	79.5	84.8	85.6	71.0	83.8	88.6

Table 5.3: Proportion of factual (non-hallucinated) content in LLM outputs across different datasets included in examinations. (source: own elaboration)

5.4 Evaluation Configurations

The framework implements evaluation in both same-task and cross-task configurations. The dataset is partitioned into training and test components, with the training component subjected to 5-fold cross-validation to determine model stability and validation effectiveness. The test component is further divided into subsets for same-task and cross-task evaluation to assess model generalization capabilities. A Logistic Regression model with class-weight balancing serves as the primary classifier (Table 5.4), selected for its computational efficiency, interpretability, and consistency with baseline methodologies. Alternative approaches, including boosted tree algorithms (such as LightGBM), were explored during preliminary experimentation but did not yield superior results. Time-series-specific architectures like LSTM networks were eliminated early in the test-

ing phase due to their suboptimal performance on hallucination detection tasks. Prior to classification, features undergo normalization using the min-max scaling strategy, transforming the data to a uniform range while preserving the original distribution characteristics. Different normalization techniques (such as standardization) were also rejected at the early stage of the experiments. To ensure a fair comparison with other baseline methods, hyperparameter tuning for logistic regression was not conducted. Regularization term and class weight balancing for effective learning of the minority class were implemented, as these modifications demonstrated more stable training during the early stages of evaluation [23].

Hyperparameter	**Value**
Maximum Iterations	10,000
Class Weight	balanced
Regularization Strength (C)	0.01

Table 5.4: Hallucination detection classifier hyperparameters for Logistic Regression used during all experiments. (source: own elaboration)

The evaluation structure follows two primary configurations: QA → SUM and SUM → QA as shown in Table 5.5. This dual-configuration approach enables to assess the model's ability to generalize across different question-answering and summarization tasks.

Source	**Target**	**Source**		**Target**	
		Train/Val	**Test**	**Test(1)**	**Test(2)**
QA	SUM.	NQ	HotPotQA	XSum	CNN/DM
SUM.	QA	CNN/DM	XSum	HotPotQA	NQ

Table 5.5: Description of datasets used at each stage of evaluation for each task setup. (source: [23]

To assess crosslingual capabilities, Polish and English (QA) datasets are used. This evaluation setting allows for testing whether attention patterns that signal hallucinations are consistent across linguistic boundaries, providing insights into the language-agnostic properties of tested methods. That crosslingual evaluation structure is presented in Table 5.6.

Source	Target	Source		Target	
		Train/Val	**Test**	**Test(1)**	**Test(2)**
EN	PL	NQ	-	polqa	POQUAD_v2
PL	EN	polqa	POQUAD_v2	NQ	HotPotQA

Table 5.6: Description of datasets used at each stage of evaluation for crosslingual setup. (source: own elaboration)

6. Experiments

The hallucination detection problem inherently involves imbalanced datasets where truthful responses significantly outnumber false or hallucinated content. In such scenarios, accuracy alone can be misleading as a model could achieve high accuracy by simply predicting the majority class. Area Under the Receiver Operating Characteristic curve (AUROC) provides a more robust evaluation metric as it measures a classifier's ability to discriminate between classes across different threshold settings, making it threshold-independent. AUROC represents the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative one, with values ranging from 0.5 (random guessing) to 1.0 (perfect classification).

To fairly compare different detection methods across various datasets, apart from AUC, a normalized performance metric called Gap is used [23]. For each model and source-target dataset pair, the Gap value is calculated using Equation 6.1:

$$\text{Gap}_m = \frac{1}{|S|} \sum_{s \in S} \frac{\max_{m' \in M} \text{AUC}^s_{m'} - \text{AUC}^s_m}{\max_{m' \in M} \text{AUC}^s_{m'}} \cdot 100\% \tag{6.1}$$

In this equation, m represents a specific detection method being evaluated within the set M of all methods under consideration. The set S encompasses all test datasets, while AUC^s_m denotes the AUROC score achieved by method m on test set s.

The Gap metric is averaged across test sets to provide insight into each method's consistency and reliability. This averaging approach is particularly important because direct comparison of raw AUROC values would be potentially misleading—different datasets contain varying proportions of positive samples and present different levels of difficulty, resulting in divergent ranges of achievable metric values. By normalizing performance relative to the best possible outcome for each dataset, the Gap measure enables meaningful cross-dataset comparisons of detection effectiveness.

6.1 Main setting

Table 6.1 presents the hallucination detection performance for Llama-2. Unlike hidden state-based approaches, attention-based methods achieved lower training AUCs, but resulted in a higher AUROC on out-of-domain target datasets. However, Lookback-Lens exhibits significant performance degradation in the SUM. → QA configuration for both original and retrained classifiers when compared to the results reported in [4]. Their observed Gap values exceed even those of hidden state-based approaches, with AUC scores hovering around values that suggest classifiers with little more than random prediction capability. It is observed that among all evaluated approaches, AggTruth

Method	Source	Target	Source			Target		Gap [%]
			Train	Val	Test	Test(1)	Test(2)	
Hidden States based								
24th Layer	QA	SUM.	0.954	0.945	0.717	0.625	0.629	8.752
	SUM.	QA	0.988	0.982	0.700	0.707	0.628	5.490
28th Layer	QA	SUM.	0.955	0.946	0.727	0.623	0.603	9.564
	SUM.	QA	0.988	0.981	0.691	0.704	0.611	6.860
32nd Layer	QA	SUM.	0.950	0.939	0.739	0.621	0.605	9.038
	SUM.	QA	0.986	0.978	0.678	0.660	0.573	11.239
Attention based								
Lookback Lens (paper)**	QA	SUM.	—	—	—	—	0.661	—
	SUM.	QA	—	—	—	—	0.660	—
Lookback Lens (original classifiers)	QA	SUM.	0.574	0.574	0.554	0.666	0.635	13.688
	SUM.	QA	0.634	0.634	**0.722**	0.506	0.506	19.407
Lookback Lens (retrained)	QA	SUM.	0.839	0.833	**0.752**	0.643	0.681	3.952
	SUM.	QA	0.898	0.882	0.700	0.521	0.523	18.931
AggTruth Sum	QA	SUM.	0.802	0.799	0.723	**0.670**	**0.710**	**2.612**
	SUM.	QA	0.895	0.886	0.707	**0.725**	**0.663**	**2.640**

Table 6.1: Comparison of Llama-2 hallucination detection (window size = 8) efficiency (AUROC) between different detection methods: hidden states based classifiers, attention based methods (Lookback Lens), and AggTruth Sum. The highest (lowest) values for each source-target configuration are in bold. (source: based on [23])

Sum consistently delivers the strongest performance, achieving the highest AUROC scores across all target tasks while maintaining robust results on source test settings, thereby yielding the lowest Gap values.

Method	Source	Target	Source			Target		Gap [%]
			Train	Val	Test	Test(1)	Test(2)	
AggTruth CosSim	QA	SUM.	0.858	0.853	**0.731**	0.587	0.713	6.209
	SUM.	QA	0.884	0.876	0.658	0.720	**0.687**	3.935
AggTruth Entropy	QA	SUM.	0.807	0.805	0.726	0.651	0.721	2.959
	SUM.	QA	0.894	0.885	0.699	**0.742**	0.674	**1.718**
AggTruth JS-DIV	QA	SUM.	0.767	0.765	0.727	0.639	**0.721**	3.478
	SUM.	QA	0.839	0.833	0.674	0.684	0.609	8.627
AggTruth Sum	QA	SUM.	0.802	0.799	0.723	**0.670**	0.710	**2.612**
	SUM.	QA	0.895	0.886	**0.707**	0.725	0.663	2.640

Table 6.2: Comparison of Llama-2 hallucination detection (window size = 8) efficiency (AUROC) between different AggTruth methods. The highest (lowest) values for each source-target configuration are in bold. (source: own elaboration)

When examining other AggTruth methods (Table 6.2) for Llama-2, AggTruth Sum exhibits stable performance with the lowest Gap for the QA → SUM. setting and the second lowest Gap for SUM. → QA, behind only AggTruth Entropy. Table 6.2 reveals higher Gap values for AggTruth CosSim and JS-DIV methods, with AggTruth JS-DIV on the SUM. → QA setting showing a Gap more than three times larger than that observed for AggTruth Sum.

Table 6.3 presents results for other examined LLMs. For each model, the hidden states approach is presented for the layer with the lowest observed Gap. All AggTruth methods are also shown. For Llama-3 and Phi-3.5, AggTruth JS-DIV achieved the lowest Gap values across both evaluation settings. For Bielik, the lowest Gap values in both settings were observed with AggTruth Sum. For Gemma-2, in the QA → SUM. setting, hidden states from the 34th layer performed best, while for SUM. → QA, AggTruth again demonstrated the lowest Gap among other methods. However, the hidden states-based approach for Gemma-2 produced a Gap more than three times higher than AggTruth, representing the second highest Gap among all methods and models apart from Lookback Lens. Beyond the Gemma-2 evaluation, AggTruth CosSim showed higher Gaps and lower AUCs, especially for Phi-3.5 QA → SUM. and Bielik SUM. → QA. AggTruth Sum obtained the second lowest Gaps for Llama-3 and the second lowest Gap for SUM. → QA when evaluating Phi-3.5, performing worse only in the QA → SUM. setting.

To fully assess robustness across all examined LLMs and evaluation settings, Table 6.4 presents the means and standard deviations of Gap metrics calculated across all models for each specific evaluation setting. For AggTruth Sum, the lowest mean Gap was observed among all methods in the SUM. → QA setting, measuring 20% lower than the second-best performer, AggTruth JS-DIV. In the QA → SUM setting, AggTruth Sum demonstrated a mean Gap only 0.36 percentage points higher than AggTruth JS-DIV, which achieved the lowest Gap value. Notably, AggTruth Sum exhibited lower standard deviations for both evaluation settings compared to JS-DIV. Among all AggTruth methods, AggTruth CosSim consistently demonstrated the highest Gap values on average. All hidden states approaches yielded higher Gap values for both evaluation settings than their AggTruth counterparts (however, with statistical significance observed only for SUM. → QA, where the mean Gap of the L-th layer hidden states exceeded that of AggTruth CosSim according to a one-sided Welch's t-test with p-value 0.05). Within the hidden states-based approaches, the L - 8 layer demonstrated the lowest mean Gap, with values progressively increasing as layer depth increased toward the final layer.

LLM	Method	Source	Target	Source			Target		Gap [%]
				Train	Val	Test	Test(1)	Test(2)	
llama-3.1-8B-Instruct	Hidden States (24th Layer)	QA	SUM.	0.932	0.925	**0.877**	0.606	0.623	8.957
		SUM.	QA	0.996	0.991	0.680	**0.860**	0.768	5.225
	AggTruth CosSim	QA	SUM.	0.799	0.797	0.849	0.641	0.687	5.336
		SUM.	QA	0.948	0.930	0.667	0.755	0.846	6.920
	AggTruth Entropy	QA	SUM.	0.862	0.858	0.861	0.638	0.632	7.551
		SUM.	QA	0.896	0.878	0.691	0.750	0.851	5.801
	AggTruth JS-DIV	QA	SUM.	0.854	0.851	0.845	**0.689**	**0.722**	**1.554**
		SUM.	QA	0.890	0.872	**0.719**	0.766	**0.853**	**3.807**
	AggTruth Sum	QA	SUM.	0.870	0.866	0.847	0.681	0.700	2.883
		SUM.	QA	0.915	0.897	0.710	0.749	0.850	4.975
phi-3.5-mini-instruct	Hidden States (24th Layer)	QA	SUM.	0.939	0.927	0.702	0.611	0.634	6.401
		SUM.	QA	0.967	0.953	0.670	0.523	**0.648**	11.161
	AggTruth CosSim	QA	SUM.	0.676	0.674	0.652	0.597	0.592	11.446
		SUM.	QA	0.812	0.799	0.647	**0.709**	0.623	5.069
	AggTruth Entropy	QA	SUM.	0.801	0.795	0.713	0.634	0.637	4.580
		SUM.	QA	0.828	0.819	0.701	0.678	0.631	3.513
	AggTruth JS-DIV	QA	SUM.	0.797	0.790	0.712	**0.674**	**0.647**	**2.172**
		SUM.	QA	0.849	0.835	**0.703**	0.702	0.636	**2.029**
	AggTruth Sum	QA	SUM.	0.754	0.750	**0.718**	0.611	0.619	6.427
		SUM.	QA	0.800	0.790	0.695	0.697	0.632	2.843
gemma-2-9b-it-bnb-4bit	Hidden States (34th Layer)	QA	SUM.	0.961	0.952	**0.812**	0.579	**0.644**	**4.383**
		SUM.	QA	0.999	0.996	0.582	0.604	0.610	15.199
	AggTruth CosSim	QA	SUM.	0.841	0.830	0.793	0.608	0.591	6.299
		SUM.	QA	0.704	0.689	0.576	**0.795**	0.648	5.518
	AggTruth Entropy	QA	SUM.	0.840	0.834	0.734	**0.651**	0.622	4.884
		SUM.	QA	0.749	0.741	0.624	0.718	0.617	7.968
	AggTruth JS-DIV	QA	SUM.	0.772	0.764	0.756	0.591	0.593	8.566
		SUM.	QA	0.738	0.732	0.626	0.777	0.625	5.037
	AggTruth Sum	QA	SUM.	0.841	0.833	0.753	0.619	0.613	6.301
		SUM.	QA	0.681	0.679	**0.627**	0.790	**0.626**	**4.410**
Bielik-11B-v2.3-Instruct	Hidden States (42th Layer)	QA	SUM.	0.983	0.976	**0.833**	0.659	0.659	5.622
		SUM.	QA	0.996	0.991	0.650	**0.811**	0.680	8.992
	AggTruth CosSim	QA	SUM.	0.837	0.830	0.784	**0.691**	0.663	5.817
		SUM.	QA	0.879	0.861	0.682	0.596	0.701	15.522
	AggTruth Entropy	QA	SUM.	0.850	0.844	0.818	0.690	0.675	3.939
		SUM.	QA	0.675	0.672	0.650	0.796	0.674	9.885
	AggTruth JS-DIV	QA	SUM.	0.848	0.842	0.805	0.682	**0.720**	2.803
		SUM.	QA	0.754	0.748	0.703	0.687	**0.791**	7.106
	AggTruth Sum	QA	SUM.	0.927	0.918	0.828	0.680	0.716	**2.134**
		SUM.	QA	0.800	0.789	**0.720**	0.714	0.766	**6.239**

Table 6.3: Comparison of hallucination detection (window size = 8) efficiency (AUROC) between hidden states features (baseline) and different AggTruth methods across various LLMs. The highest (lowest) values for each source-target configuration are in bold. (source: own elaboration)

Method	Source	Target	Gap Mean	Gap Std
Hidden States (L - 8 th Layer)	QA	SUM	6.98	2.04
	SUM	QA	9.10	5.00
Hidden States (L - 4 th Layer)	QA	SUM	7.20	1.68
	SUM	QA	11.90	7.90
Hidden States (L th Layer)	QA	SUM	7.53	2.10
	SUM	QA	14.98	9.54
AggTruth CosSim	QA	SUM	6.91	2.56
	SUM	QA	7.26	4.80
AggTruth Entropy	QA	SUM	4.67	1.88
	SUM	QA	5.64	3.51
AggTruth JS-DIV	QA	SUM	**3.60**	2.83
	SUM	QA	5.19	2.41
AggTruth Sum	QA	SUM	3.96	2.22
	SUM	QA	**4.08**	1.71

Table 6.4: Performance gap comparison of hallucination detection methods on cross-task evaluations using window size = 8. Lower values indicate better cross-task robustness. (source: own elaboration)

6.2 Influence of window size

The methodology of [4] and [23], follows analysis of a fixed window size of 8 tokens. However, this research also investigates the impact of varying window sizes on detection performance. Operating with smaller windows presents a more challenging task, as it requires assessing the factuality of responses based on fewer tokens. In the extreme case (window size of 1), classifiers must detect hallucinations based solely on aggregated attention weights for a single token. The critical research question is how various methods perform under changing window sizes. A truly robust method should demonstrate minimal performance variation across different window settings. While changing window size alters the distribution of hallucinated and non-hallucinated examples, making AUC values not directly comparable between different window size configurations (even for the same model and task), the Gap value remains independent of this ratio and requires only reference maximum values. Therefore, method robustness can be effectively measured by comparing Gap values across different window settings. The hypothesis examines whether methods can achieve similar, stable Gap values across various window sizes.

The results for the window size 1 setting are presented in Table 6.5, while Table 6.6 shows the results for window size 4. For window size 1 (Table 6.5), several patterns similar to the main setting can be observed. The AggTruth JS-DIV method again achieved the lowest Gap values for Phi-3.5 in both evaluation settings, as well as for Llama-3 in the QA $\rightarrow$ SUM configuration, with a Gap value close to zero, indicating the highest AUCs across all test sets. However, for the SUM. $\rightarrow$ QA setting, AggTruth Sum achieved the lowest Gap value among the examined methods. Similarly, for Bielik (in both evaluation settings), the lowest Gap values were observed for AggTruth Sum, which also achieved the highest AUCs on 5 out of 6 test sets. For the Llama-2 model, AggTruth Entropy obtained the lowest Gap value for the SUM. $\rightarrow$ QA setting, while for QA $\rightarrow$ SUM, the hidden states approach resulted in the lowest Gap. The highest overall Gap values were observed for the Gemma-2 model, consistent with the window size 8 setting.

For window size 4, similar patterns can be seen as in other configurations. The lowest Gap values (and half of the highest AUCs for test sets) for Bielik were observed for AggTruth Sum in both evaluation settings. For Llama-3 and Phi-3.5, the AggTruth JS-DIV aggregation method achieved the lowest Gap values. For Gemma-2, the spread of results was the highest, and no method demonstrated stable performance across both evaluation settings. However, considering performance across both evaluation settings, the lowest Gap value was observed for AggTruth Sum.

A natural hypothesis that arises is whether it is possible to effectively detect hallucinations using different window sizes. Due to the changing distribution of hallucinated and non-hallucinated examples when varying window sizes, the AUCs—even for the same evaluation setting and model—are not directly comparable. As the window size decreases, the number of hallucinated examples also decreases, resulting in a more imbalanced dataset. Examining Tables 6.3, 6.5, and 6.6, it can be seen that for no model or evaluation setting, at any window size, are the observed AUCs trivial. When reducing the window size to one, several slightly lower AUCs were observed for some test datasets, but this was not a consistent trend, and for some evaluation settings, these AUCs were even higher than those for larger window sizes.

Table 6.7 presents the mean and standard deviation of overall Gap values for window size 1. Similarly to the window size 8 setting, AggTruth Sum achieved the lowest mean Gap value for the SUM. → QA evaluation setting. For QA → SUM, the lowest mean Gap value was observed for hidden states from the L - 8th layer. The results for AggTruth JS-DIV are close to those of AggTruth Sum. Among the AggTruth methods, AggTruth CosSim again exhibited the highest Gap values. As in the main setting, for hidden states, increasing the layer number leads to higher Gap values for both evaluation settings.

Table 6.8 shows the overall mean and standard deviation of Gap values across the examined LLMs for all methods. For the SUM. → QA setting, the best-performing method is AggTruth Sum, while for QA → SUM, AggTruth JS-DIV performs best, though with a value only 0.03 lower than AggTruth Sum (notably, the standard deviation for AggTruth Sum is two times lower). The best-performing hidden state approach, in terms of overall mean Gap value, was observed for the L - 8 layer.

For a clearer comparison between methods across the examined window sizes, Figures 6.1 and 6.2 present bar plots of the overall mean Gap for all AggTruth aggregation methods, as well as the best-performing hidden state layer, for the QA → SUM. and SUM. → QA evaluation settings, respectively. In Figure 6.1, it can be observed that the lowest mean Gap values were achieved by AggTruth JS-DIV, with slightly lower values than AggTruth Sum. AggTruth CosSim consistently exhibited higher Gap values than the other AggTruth methods, with performance comparable to the hidden states-based approach (except for window size 1). For the SUM → QA evaluation setting (Figure 6.2), the lowest Gap values were observed for AggTruth Sum, approximately 0.5 percentage points lower than AggTruth JS-DIV. Once again, the highest Gap values among all AggTruth methods were observed for AggTruth CosSim. For all examined window sizes, Gap values for hidden states-based approaches were higher than for any AggTruth method. Across both figures, high variances in the observed results are evident for all methods; however, AggTruth Sum demonstrates the smallest variance among them.

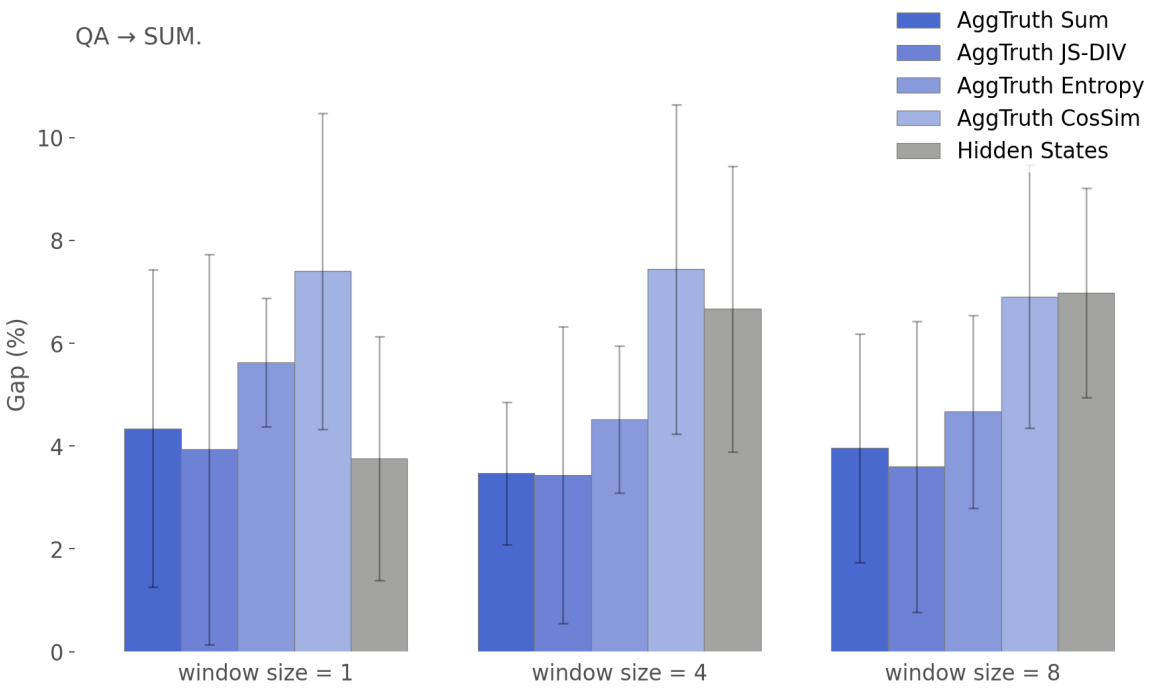

Figure 6.1: Performance gap comparison of hallucination detection methods on QA → SUM. task for different window sizes. Lower values indicate better cross-task robustness. (source: own elaboration)

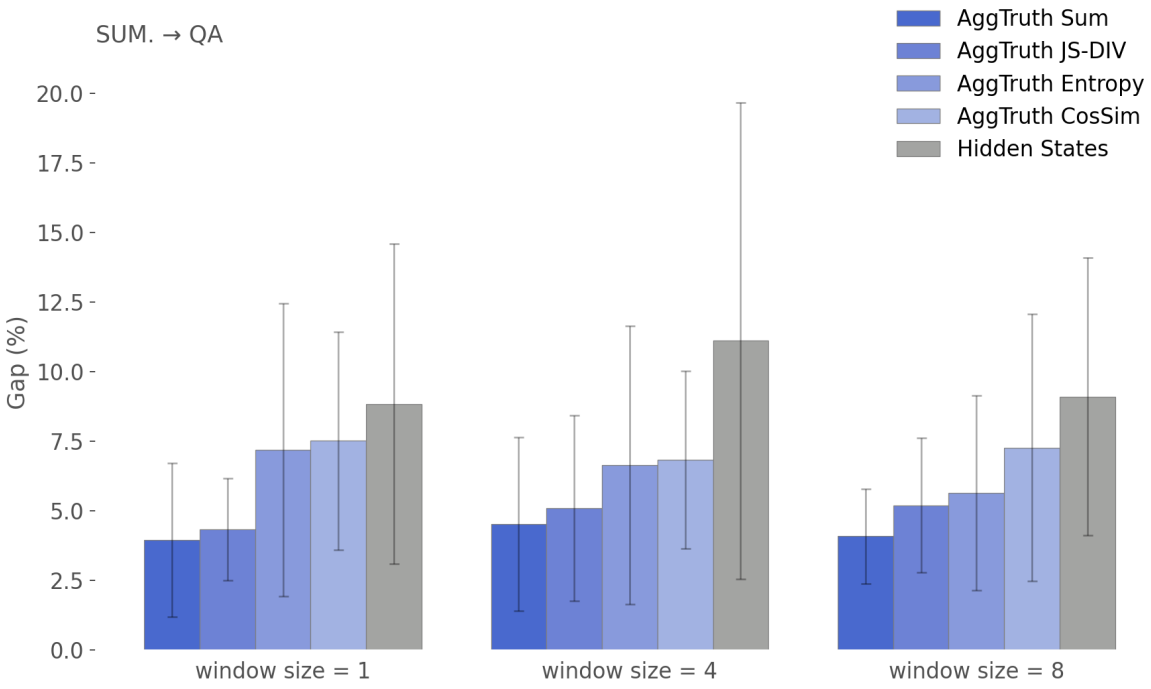

Figure 6.2: Performance gap comparison of hallucination detection methods on SUM → QA. task for different window sizes. Lower values indicate better cross-task robustness. (source: own elaboration)

LLM	Method	Source	Target	Source			Target		Gap [%]
				Train	Val	Test	Test(1)	Test(2)	
llama-2-7b-chat-hf	Hidden States (24th Layer)	QA	SUM.	0.906	0.882	**0.748**	**0.629**	0.652	**2.846**
		SUM.	QA	0.960	0.918	**0.682**	0.691	0.609	2.865
	AggTruth CosSim	QA	SUM.	0.816	0.808	0.726	0.598	0.698	3.326
		SUM.	QA	0.866	0.848	0.646	**0.694**	**0.661**	1.799
	AggTruth Entropy	QA	SUM.	0.771	0.769	0.697	0.610	0.661	5.694
		SUM.	QA	0.861	0.845	0.664	0.691	0.654	**1.434**
	AggTruth JS-DIV	QA	SUM.	0.810	0.804	0.706	0.591	**0.703**	4.308
		SUM.	QA	0.845	0.830	0.659	0.664	0.623	4.564
	AggTruth Sum	QA	SUM.	0.763	0.761	0.683	0.613	0.674	5.507
		SUM.	QA	0.859	0.842	0.678	0.675	0.641	2.203
llama-3.1-8B-Instruct	Hidden States (24th Layer)	QA	SUM.	0.884	0.868	0.858	0.625	0.660	6.069
		SUM.	QA	0.987	0.923	0.671	0.780	0.851	5.010
	AggTruth CosSim	QA	SUM.	0.846	0.836	0.860	0.618	0.650	6.804
		SUM.	QA	0.893	0.848	0.652	0.769	0.846	6.482
	AggTruth Entropy	QA	SUM.	0.827	0.822	0.861	0.628	0.674	5.146
		SUM.	QA	0.898	0.858	0.681	0.845	0.767	5.250
	AggTruth JS-DIV	QA	SUM.	0.831	0.824	**0.868**	0.665	**0.742**	**0.008**
		SUM.	QA	0.887	0.848	0.699	0.773	0.853	3.869
	AggTruth Sum	QA	SUM.	0.837	0.829	0.866	**0.665**	0.712	1.395
		SUM.	QA	0.901	0.861	**0.714**	**0.853**	0.772	**3.238**
phi-3.5-mini-instruct	Hidden States (24th Layer)	QA	SUM.	0.849	0.811	0.672	0.590	0.619	5.272
		SUM.	QA	0.915	0.838	0.638	0.584	0.510	13.124
	AggTruth CosSim	QA	SUM.	0.665	0.658	0.616	0.573	0.578	10.973
		SUM.	QA	0.704	0.690	0.598	0.652	0.572	8.292
	AggTruth Entropy	QA	SUM.	0.729	0.717	0.654	0.601	0.597	6.704
		SUM.	QA	0.770	0.752	0.680	0.583	0.573	7.747
	AggTruth JS-DIV	QA	SUM.	0.679	0.672	0.647	**0.642**	**0.648**	**2.325**
		SUM.	QA	0.761	0.742	0.688	**0.666**	0.607	**1.488**
	AggTruth Sum	QA	SUM.	0.775	0.758	**0.695**	0.595	0.599	4.934
		SUM.	QA	0.811	0.778	**0.703**	0.595	**0.609**	4.145
gemma-2-9b-it-bnb-4bit	Hidden States (34th Layer)	QA	SUM.	0.911	0.877	**0.837**	**0.595**	**0.638**	**0.843**
		SUM.	QA	0.995	0.946	0.612	0.611	0.597	14.356
	AggTruth CosSim	QA	SUM.	0.800	0.778	0.807	0.549	0.536	9.901
		SUM.	QA	0.789	0.748	0.590	0.727	0.638	8.343
	AggTruth Entropy	QA	SUM.	0.817	0.801	0.786	0.573	0.585	6.837
		SUM.	QA	0.735	0.704	0.590	0.620	0.583	15.659
	AggTruth JS-DIV	QA	SUM.	0.702	0.694	0.795	0.518	0.554	11.120
		SUM.	QA	0.734	0.706	0.608	**0.766**	0.628	6.295
	AggTruth Sum	QA	SUM.	0.818	0.802	0.791	0.565	0.556	8.618
		SUM.	QA	0.812	0.762	**0.674**	0.601	**0.662**	8.556
Bielik-11B-v2.3-Instruct	Hidden States (42th Layer)	QA	SUM.	0.941	0.906	**0.854**	0.628	0.637	4.663
		SUM.	QA	0.984	0.908	0.643	0.675	0.625	11.425
	AggTruth CosSim	QA	SUM.	0.822	0.800	0.817	0.643	0.624	6.026
		SUM.	QA	0.852	0.802	0.657	0.600	0.652	12.644
	AggTruth Entropy	QA	SUM.	0.862	0.837	0.848	0.668	0.621	3.782
		SUM.	QA	0.817	0.790	0.692	0.723	0.652	5.814
	AggTruth JS-DIV	QA	SUM.	0.842	0.818	0.825	0.692	0.653	1.924
		SUM.	QA	0.738	0.723	0.665	0.754	0.659	5.402
	AggTruth Sum	QA	SUM.	0.857	0.832	0.832	**0.700**	**0.653**	**1.260**
		SUM.	QA	0.816	0.786	**0.695**	**0.791**	**0.678**	**1.543**

Table 6.5: Comparison of hallucination detection (window size = 1) efficiency (AUROC) between hidden states features (baseline) and different AggTruth methods across different LLMs. The highest (lowest) values for each source-target configuration are in bold. (source: own elaboration)

LLM	Method	Source	Target	Source			Target		Gap [%]
				Train	Val	Test	Test(1)	Test(2)	
llama-2-7b-chat-hf	Hidden States (24th Layer)	QA	SUM.	0.942	0.929	**0.738**	0.632	0.645	5.875
		SUM.	QA	0.982	0.970	**0.697**	0.698	0.623	3.813
	AggTruth CosSim	QA	SUM.	0.845	0.840	0.732	0.599	**0.715**	4.586
		SUM.	QA	0.879	0.868	0.647	0.709	**0.682**	2.801
	AggTruth Entropy	QA	SUM.	0.800	0.798	0.723	**0.654**	0.703	**2.717**
		SUM.	QA	0.888	0.876	0.680	**0.718**	0.664	**1.704**
	AggTruth JS-DIV	QA	SUM.	0.768	0.766	0.717	0.645	0.712	3.030
		SUM.	QA	0.862	0.852	0.666	0.660	0.610	7.692
	AggTruth Sum	QA	SUM.	0.795	0.792	0.714	0.640	0.703	3.842
		SUM.	QA	0.886	0.875	0.690	0.704	0.654	2.330
llama-3.1-8B-Instruct	Hidden States (24th Layer)	QA	SUM.	0.910	0.902	0.841	0.612	0.630	9.347
		SUM.	QA	0.994	0.981	0.671	0.844	0.776	4.733
	AggTruth CosSim	QA	SUM.	0.790	0.788	0.827	0.629	0.675	7.131
		SUM.	QA	0.943	0.920	0.676	0.760	0.817	6.170
	AggTruth Entropy	QA	SUM.	0.850	0.845	**0.843**	0.646	0.657	6.501
		SUM.	QA	0.910	0.890	0.705	0.758	0.829	4.432
	AggTruth JS-DIV	QA	SUM.	0.844	0.840	0.840	**0.694**	**0.744**	**0.414**
		SUM.	QA	0.900	0.879	**0.722**	0.771	**0.831**	**3.039**
	AggTruth Sum	QA	SUM.	0.858	0.852	0.830	0.691	0.717	2.180
		SUM.	QA	0.928	0.908	0.715	**0.832**	0.759	3.832
phi-3.5-mini-instruct	Hidden States (24th Layer)	QA	SUM.	0.903	0.883	0.679	0.594	0.608	8.328
		SUM.	QA	0.956	0.929	0.637	0.601	0.523	13.889
	AggTruth CosSim	QA	SUM.	0.700	0.695	0.633	0.563	0.593	12.780
		SUM.	QA	0.743	0.733	0.628	**0.704**	0.602	5.242
	AggTruth Entropy	QA	SUM.	0.780	0.773	0.695	0.643	0.637	3.709
		SUM.	QA	0.849	0.832	**0.709**	0.635	0.616	3.904
	AggTruth JS-DIV	QA	SUM.	0.769	0.761	0.688	**0.669**	**0.652**	**1.999**
		SUM.	QA	0.816	0.801	0.708	0.692	0.624	**0.882**
	AggTruth Sum	QA	SUM.	0.784	0.776	**0.708**	0.626	0.605	5.597
		SUM.	QA	0.838	0.821	0.708	0.651	**0.624**	2.828
gemma-2-9b-it-bnb-4bit	Hidden States (34th Layer)	QA	SUM.	0.955	0.948	**0.814**	0.584	**0.630**	**3.121**
		SUM.	QA	0.999	0.998	0.608	0.539	0.572	21.998
	AggTruth CosSim	QA	SUM.	0.823	0.810	0.794	0.580	0.568	7.360
		SUM.	QA	0.765	0.743	0.576	0.771	0.633	11.014
	AggTruth Entropy	QA	SUM.	0.855	0.844	0.752	0.605	0.617	5.098
		SUM.	QA	0.752	0.737	0.606	0.686	0.607	14.310
	AggTruth JS-DIV	QA	SUM.	0.796	0.787	0.768	0.593	0.560	8.122
		SUM.	QA	0.721	0.711	**0.615**	**0.777**	0.629	**9.023**
	AggTruth Sum	QA	SUM.	0.751	0.742	**0.780**	**0.614**	0.620	3.390
		SUM.	QA	0.760	0.743	0.609	0.626	**0.760**	10.001
Bielik-11B-v2.3-Instruct	Hidden States (42th Layer)	QA	SUM.	0.972	0.960	**0.849**	0.647	0.646	6.372
		SUM.	QA	0.995	0.985	0.651	0.743	0.650	9.724
	AggTruth CosSim	QA	SUM.	0.812	0.803	0.811	0.660	0.686	5.354
		SUM.	QA	0.856	0.834	0.683	0.662	**0.708**	8.879
	AggTruth Entropy	QA	SUM.	0.847	0.838	0.829	0.693	0.655	4.553
		SUM.	QA	0.802	0.788	0.712	0.664	0.679	8.777
	AggTruth JS-DIV	QA	SUM.	0.877	0.865	0.824	**0.702**	0.669	3.630
		SUM.	QA	0.792	0.779	0.692	**0.787**	0.678	4.801
	AggTruth Sum	QA	SUM.	0.841	0.833	0.837	0.679	**0.710**	**2.338**
		SUM.	QA	0.854	0.838	**0.725**	0.762	0.694	**3.560**

Table 6.6: Comparison of hallucination detection (window size = 4) efficiency (AUROC) between hidden states features (baseline) and different AggTruth methods across different LLMs. The highest (lowest) values for each source-target configuration are in bold. (source: own elaboration)

Method	Source	Target	Gap Mean	Gap Std
Hidden States (L - 8 th Layer)	QA	SUM	**3.76**	2.38
	SUM	QA	8.84	5.75
Hidden States (L - 4 th Layer)	QA	SUM	5.04	2.12
	SUM	QA	12.38	8.85
Hidden States (L th Layer)	QA	SUM	5.19	2.51
	SUM	QA	13.84	8.84
AggTruth CosSim	QA	SUM	7.41	3.08
	SUM	QA	7.51	3.92
AggTruth Entropy	QA	SUM	5.63	1.25
	SUM	QA	7.18	5.26
AggTruth JS-DIV	QA	SUM	3.94	4.30
	SUM	QA	4.32	1.83
AggTruth Sum	QA	SUM	4.34	3.09
	SUM	QA	**3.94**	2.77

Table 6.7: Performance gap comparison of hallucination detection methods on cross-task evaluations using window size = 1. Lower values indicate better cross-task robustness. (source: own elaboration)

Method	Source	Target	Gap Mean	Gap Std
Hidden States (L - 8 th Layer)	QA	SUM	6.67	2.78
	SUM	QA	11.11	8.57
Hidden States (L - 4 th Layer)	QA	SUM	7.96	2.61
	SUM	QA	13.55	10.86
Hidden States (L th Layer)	QA	SUM	7.50	2.60
	SUM	QA	15.59	11.59
AggTruth CosSim	QA	SUM	7.44	3.21
	SUM	QA	6.82	3.20
AggTruth Entropy	QA	SUM	4.52	1.43
	SUM	QA	6.63	5.00
AggTruth JS-DIV	QA	SUM	3.44	2.89
	SUM	QA	5.09	3.33
AggTruth Sum	QA	SUM	3.47	1.38
	SUM	QA	4.51	3.13

Table 6.8: Performance gap comparison of hallucination detection methods on cross-task evaluations using window size = 4. Lower values indicate better cross-task robustness. (source: own elaboration)

6.3 Cross-lingual evaluation

The methodology for cross-lingual evaluation extends beyond monolingual settings to assess hallucination detection capabilities across language boundaries. This research investigates two distinct cross-lingual configurations: English to Polish (EN → PL) and Polish to English (PL → EN). Operating across languages presents a significant challenge, as methods must generalize beyond the semantic and syntactic structures of a single language. The key research question is how various detection methods perform when applied to content generated in a different language than the one they were trained on. A truly robust cross-lingual method should demonstrate consistent performance regardless of the source and target language pairing. While the underlying LLM's proficiency may vary between languages, potentially affecting the nature and frequency of hallucinations, the Gap metric enables meaningful comparisons by measuring relative performance. Therefore, method robustness in cross-lingual settings can be effectively assessed by comparing Gap values between EN → PL and PL → EN configurations. The hypothesis examines whether methods can maintain stable performance when transferred across language boundaries, with minimal degradation compared to their monolingual counterparts.

Table 6.9 presents the results for cross-lingual settings for the examined LLMs. With the exception of two experiments, for AggTruth Sum, JS-DIV, and Entropy, the overall Gap value for the EN → PL evaluation setting is lower than for PL → EN. This is particularly evident for AggTruth Entropy in the EN → PL results, which—apart from those for Llama-3—are the lowest and close to the minimum achievable value. For Bielik, no different phenomenon was observed compared to other LLMs, despite it being a Polish model continually pretrained on massive Polish datasets. For the Llama-2 model, all differences between Gap values among the examined methods are not statistically significant. As in the main setting, AggTruth Sum achieved lower Gap values and higher AUCs for Gemma-2 and Bielik in the EN → PL setting compared to other methods.

Table 6.10 shows the overall mean and standard deviation of Gap values for the examined methods. For the EN → PL evaluation setting, the lowest mean Gaps were observed for AggTruth Entropy and AggTruth Sum, with values at least twice as low as those for the other methods. For the PL → EN setting, the lowest Gap (with the lowest standard deviation) was observed for the hidden states-based approach from the $L - 8$ th layer. As with different window sizes, an increase in Gap value was observed when increasing the number of layers. For the hidden states-based approach, lower Gap values were observed for the PL → EN setting, but for AggTruth methods (apart from CosSim), the opposite phenomenon was observed.

LLM	Method	Source	Target	Source			Target		Gap [%]
				Train	Val	Test	Test(1)	Test(2)	
llama-2-7b-chat-hf	Hidden States (24th Layer)	EN	PL	0.954	0.945	-	0.742	0.699	1.376
		PL	EN	0.964	0.955	0.727	**0.766**	0.709	1.794
	AggTruth CosSim	EN	PL	0.823	0.819	-	0.744	0.690	1.875
		PL	EN	0.870	0.863	0.722	0.733	0.712	3.309
	AggTruth Entropy	EN	PL	0.864	0.860	-	**0.747**	0.711	**0.200**
		PL	EN	0.865	0.860	**0.750**	0.744	0.724	**1.048**
	AggTruth JS-DIV	EN	PL	0.831	0.827	-	0.721	0.707	2.248
		PL	EN	0.834	0.829	0.744	0.748	**0.725**	1.119
	AggTruth Sum	EN	PL	0.830	0.827	-	0.738	**0.714**	0.644
		PL	EN	0.866	0.860	0.748	0.742	0.720	1.399
llama-3.1-8B-Instruct	Hidden States (24th Layer)	EN	PL	0.932	0.925	-	**0.784**	0.721	**1.511**
		PL	EN	0.898	0.886	**0.740**	**0.863**	0.786	**2.823**
	AggTruth CosSim	EN	PL	0.799	0.797	-	0.718	0.733	4.994
		PL	EN	0.750	0.746	0.691	0.738	0.844	7.617
	AggTruth Entropy	EN	PL	0.863	0.859	-	0.739	**0.744**	2.855
		PL	EN	0.802	0.798	0.717	0.758	**0.857**	5.138
	AggTruth JS-DIV	EN	PL	0.808	0.805	-	0.745	0.728	3.560
		PL	EN	0.763	0.760	0.712	0.756	0.843	5.991
	AggTruth Sum	EN	PL	0.865	0.860	-	0.750	0.740	2.403
		PL	EN	0.800	0.796	0.723	0.768	0.851	4.722
phi-3.5-mini-instruct	Hidden States (24th Layer)	EN	PL	0.939	0.927	-	0.591	0.559	6.754
		PL	EN	0.928	0.915	**0.675**	**0.764**	0.629	**0.024**
	AggTruth CosSim	EN	PL	0.676	0.674	-	0.527	0.512	15.751
		PL	EN	0.821	0.813	0.603	0.643	0.594	10.718
	AggTruth Entropy	EN	PL	0.721	0.717	-	**0.627**	**0.603**	**0.264**
		PL	EN	0.828	0.820	0.663	0.702	0.605	4.561
	AggTruth JS-DIV	EN	PL	0.697	0.695	-	0.579	0.598	4.475
		PL	EN	0.805	0.798	0.658	0.696	**0.613**	4.638
	AggTruth Sum	EN	PL	0.706	0.704	-	0.605	0.589	3.128
		PL	EN	0.812	0.806	0.662	0.660	0.602	6.643
gemma-2-9b-it-bnb-4bit	Hidden States (34th Layer)	EN	PL	0.961	0.952	-	0.677	0.689	4.556
		PL	EN	0.936	0.914	**0.696**	0.768	0.657	3.031
	AggTruth CosSim	EN	PL	0.840	0.830	-	0.686	0.669	5.305
		PL	EN	0.703	0.696	0.651	0.762	0.649	5.857
	AggTruth Entropy	EN	PL	0.842	0.834	-	0.693	**0.728**	**0.761**
		PL	EN	0.825	0.812	0.665	0.784	**0.667**	3.388
	AggTruth JS-DIV	EN	PL	0.838	0.830	-	0.685	0.704	2.962
		PL	EN	0.772	0.758	0.674	**0.793**	0.666	**2.666**
	AggTruth Sum	EN	PL	0.841	0.833	-	0.703	0.720	0.576
		PL	EN	0.827	0.811	0.670	**0.783**	0.649	4.071
Bielik-11B-v2.3-Instruct	Hidden States (42th Layer)	EN	PL	0.983	0.976	-	0.698	0.670	8.899
		PL	EN	0.991	0.984	0.660	0.834	0.700	7.596
	AggTruth CosSim	EN	PL	0.879	0.870	-	0.712	0.667	8.170
		PL	EN	0.964	0.954	0.657	0.805	0.721	8.076
	AggTruth Entropy	EN	PL	0.871	0.863	-	0.785	**0.715**	**0.178**
		PL	EN	0.946	0.936	0.690	0.831	0.751	**4.261**
	AggTruth JS-DIV	EN	PL	0.870	0.862	-	0.763	0.706	2.282
		PL	EN	0.875	0.866	0.688	0.717	0.812	6.329
	AggTruth Sum	EN	PL	0.868	0.861	-	**0.788**	0.711	0.303
		PL	EN	0.927	0.916	**0.695**	0.736	**0.824**	4.793

Table 6.9: Comparison of hallucination detection efficiency (AUROC) across different LLMs for cross-lingual (Polish-English) settings, showing performance of hidden states versus various AggTruth methods. The highest (lowest) values for each source-target configuration are in bold. (source: own elaboration)

Method	Source	Target	Gap Mean	Gap Std
Hidden States (L - 8 th Layer)	PL	EN	**1.92**	1.37
	EN	PL	3.55	2.59
Hidden States (L - 4 th Layer)	PL	EN	3.45	1.83
	EN	PL	6.96	3.63
Hidden States (L th Layer)	PL	EN	5.26	2.74
	EN	PL	7.06	2.67
AggTruth CosSim	PL	EN	7.12	2.75
	EN	PL	7.22	5.27
AggTruth Entropy	PL	EN	3.68	1.60
	EN	PL	**0.85**	1.15
AggTruth JS-DIV	PL	EN	4.15	2.22
	EN	PL	3.11	0.94
AggTruth Sum	PL	EN	4.33	1.90
	EN	PL	1.41	1.27

Table 6.10: Performance gap comparison of hallucination detection methods on cross-lingual (Polish-English) evaluations. Lower values indicate better cross-lingual robustness. (source: own elaboration)

6.4 Impact of feature selection

Prior work by [4] demonstrated performance improvements when increasing the number of attention heads used by their classifier. However, their investigation was constrained to small subsets of heads (10, 50, and 100) for Llama-2, representing merely 1%, 5%, and 10% of all available heads. Additionally, their evaluation examined predefined spans rather than windows, which are not accessible during the decoding process [23].

Our study expanded this examination to address a fundamental question - what proportion of attention heads is optimal for achieving peak performance across various LLMs? The analysis revealed peak results typically occurred when leveraging at least half of the available attention heads across the evaluated LLMs. Notably, for specific scenarios, just $10 - 20\%$ of heads proved adequate without substantial performance loss (see Table 6.11). Some configurations with reduced head counts actually surpassed full-head models in AUROC metrics. Since these findings stemmed from specific test configurations rather than comprehensive hyperparameter optimization, additional performance gains through dataset-specific tuning remain a promising avenue for further investigation [23].

However, the topic of optimal feature selection methods was not explored in depth, as only some recommendations were provided without a thorough examination. For this reason, a comprehensive evaluation of feature selection methods was conducted in this work to determine which approach is the optimal choice. A robust feature selection method should perform well across all LLMs and tasks, demonstrate better performance than the no feature selection setting, and select as few features as possible while maintaining detection capability. As the analysis of window size and cross-lingual settings revealed that the lowest Gaps were achieved by AggTruth Sum and AggTruth JS-DIV, all applied feature selection methods were evaluated for these approaches in both monolingual and cross-lingual settings (using window size 8). Tables 6.12 and 6.13 present the means and standard deviations of the overall Gap across all LLMs for both evaluation settings for AggTruth Sum and AggTruth JS-DIV, respectively. The tables also show the mean percentage of heads selected by each feature selection method. For both aggregation methods, applying feature selection can result in a lower overall Gap compared to not applying it at all. However, not all feature selection methods had the same positive impact on overall performance. Selecting features with higher coefficients than random features yielded results for the Gap metric similar to not selecting features, and in some cases, even higher Gap values were observed for this version of positive coefficient selection. The two feature selection methods that consistently achieved the lowest Gap values for both evaluation settings and both examined aggregation methods were Spearman and PBI. For both aggregation methods and evaluation settings, the mean Gap values for Spearman and PBI were lower than those for not applying feature selection. The percentage of selected features was also similar, ranging from approximately 45% to 70%.

Tables 6.12 and 6.13 also present the means and standard deviations of the overall Gap across all LLMs for cross-lingual evaluation for AggTruth Sum and AggTruth JS-

DIV, respectively. The spread of results between selection methods is not as pronounced as in the monolingual setting. Moreover, the differences between these methods and not applying feature selection are negligible. The percentage of heads selected using Spearman or PBI is slightly higher than in the monolingual setting, ranging from approximately 55% to 70%.

LLM	AggTruth	Source	Target	Selector	Heads [%]	Test	Test(1)	Test(2)	Gap [%]
llama-2-7b-chat-hf	Sum	QA	SUM.	—	100.0	0.727	0.582	0.697	6.836
				$\text{Random}^{+}_{3,3}$	48.4	0.723	0.670	0.710	**2.048**
				$\text{Center}_{0.2}$	**19.9**	0.713	0.638	0.739	2.798
		SUM.	QA	—	100.0	0.705	0.722	0.660	2.159
				$\text{Spearman}_{1.0}$	96.2	0.706	0.724	0.660	**1.975**
				$\text{Random}^{+}_{3,3}$	**45.6**	0.699	0.692	0.639	4.776
llama-3.1-8B-Instruct	JS-Div	QA	SUM.	—	100.0	0.844	0.687	0.720	1.740
				$\text{Random}_{3,2}$	96.8	0.845	0.689	0.722	**1.554**
				Lasso	**19.2**	0.856	0.685	0.708	1.973
		SUM.	QA	—	100.0	0.714	0.765	0.854	3.847
				$\text{Random}_{3,1}$	86.7	0.715	0.766	0.853	**3.786**
				$\text{Random}^{+}_{3,3}$	**43.2**	0.702	0.770	0.847	4.512
phi-3.5-mini-instruct	JS-Div	QA	SUM.	—	100.0	0.715	0.651	0.633	3.905
				$\text{Spearman}_{1.0}$	76.6	0.712	0.674	0.647	**2.172**
				$\text{Center}_{0.2}$	**19.9**	0.659	0.660	0.663	4.406
		SUM.	QA	—	100.0	0.703	0.702	0.636	**2.029**
				$\text{Random}_{3,1}$	95.2	0.702	0.703	0.635	2.038
				$\text{Spearman}_{\text{auto}}$	**31.3**	0.696	0.713	0.633	2.045
gemma-2-9b-it-bnb-4bit	Sum	QA	SUM.	—	100.0	0.729	0.637	0.590	7.512
				$\text{Center}_{0.5}$	49.9	0.742	0.625	0.609	**6.606**
				$\text{Spearman}_{0.2}$	**19.9**	0.759	0.583	0.634	6.819
		SUM.	QA	—	100.0	0.662	0.697	0.572	8.642
				$\text{Spearman}_{0.5}$	49.9	0.659	0.748	0.593	**5.563**
				$\text{Spearman}_{0.1}$	**10.0**	0.617	0.749	0.625	5.901

Table 6.11: Influence of the percentage of selected heads on results with different selectors. Comparisons are made for each LLM, aggregation method, and task between no feature selection (marked with hyphens), best-obtained selection method w.r.t. Gap value, and method resulted in the lowest number of heads. (source: [23])

Feature Selection	Source	Target	Gap Mean	Gap Std	Mean [%] Heads
No Feature Selection	QA	SUM	5.79	2.59	100.00
	SUM	QA	6.56	4.56	100.00
Random	QA	SUM	5.59	2.47	95.55
	SUM	QA	6.55	4.31	93.64
Random^{+}	QA	SUM	6.17	4.47	47.24
	SUM	QA	11.74	9.42	44.34
Spearman$_r$	QA	SUM	**5.00**	1.75	64.81
	SUM	QA	**4.46**	2.01	45.89
Non Zero Feature	QA	SUM	5.55	2.34	46.51
	SUM	QA	6.75	4.35	78.62
Center$_r$	QA	SUM	5.00	1.74	29.91
	SUM	QA	9.56	7.50	41.93
PBI$_r$	QA	SUM	5.02	2.20	55.49
	SUM	QA	4.51	1.97	49.73

Table 6.12: Performance comparison of feature selection methods for hallucination detection across tasks for Aggtruth Sum. Gap Mean represents the percentage difference from best performance (lower values indicate better cross-task robustness), with Gap Std showing the variability across test datasets. Mean [%] Heads represents the percentage of attention heads used. (source: own elaboration)

Feature Selection	Source	Target	Gap Mean	Gap Std	Mean [%] Heads
No Feature Selection	QA	SUM	4.92	3.32	100.00
	SUM	QA	6.86	4.46	100.00
Random	QA	SUM	4.85	3.38	96.62
	SUM	QA	6.77	4.41	92.94
Random^{+}	QA	SUM	4.86	3.82	47.27
	SUM	QA	8.64	5.04	45.32
Spearman$_r$	QA	SUM	**4.19**	2.78	73.44
	SUM	QA	**5.44**	2.36	41.47
Non Zero Feature	QA	SUM	4.86	3.23	38.78
	SUM	QA	7.12	4.34	76.67
Center$_r$	QA	SUM	4.29	2.40	37.94
	SUM	QA	7.50	3.77	49.95
PBI$_r$	QA	SUM	4.65	3.12	73.84
	SUM	QA	5.47	2.71	55.44

Table 6.13: Performance comparison of feature selection methods for hallucination detection across tasks for Aggtruth JS-DIV. Gap Mean represents the percentage difference from best performance (lower values indicate better cross-task robustness), with Gap Std showing the variability across test datasets. Mean [%] Heads represents the percentage of attention heads used. (source: own elaboration)

Feature Selection	Source	Target	Gap Mean	Gap Std	Mean [%] Heads
No Feature Selection	EN	PL	5.15	3.62	100.00
	PL	EN	4.71	2.15	100.00
Random	EN	PL	5.23	3.42	95.54
	PL	EN	**4.69**	1.96	91.13
Random^{+}	EN	PL	**4.26**	0.92	47.27
	PL	EN	6.18	1.07	44.28
Spearman$_r$	EN	PL	4.44	1.66	61.69
	PL	EN	5.23	1.99	67.18
Non Zero Feature	EN	PL	5.44	2.79	38.78
	PL	EN	5.15	1.87	15.88
Center$_r$	EN	PL	5.39	1.88	26.60
	PL	EN	6.33	1.79	26.60
PBI$_r$	EN	PL	4.73	2.17	53.70
	PL	EN	5.24	1.98	65.29

Table 6.14: Performance comparison of feature selection methods for hallucination detection in cross-lingual tasks for AggTruth Sum. Gap Mean represents the percentage difference from best performance (lower values indicate better cross-lingual robustness), with Gap Std showing the variability across test datasets. Mean [%] Heads represents the percentage of attention heads used. (source: own elaboration)

Feature Selection	Source	Target	Gap Mean	Gap Std	Mean [%] Heads
No Feature Selection	EN	PL	4.81	4.51	100.00
	PL	EN	4.52	1.93	100.00
Random	EN	PL	4.63	4.49	95.43
	PL	EN	**4.52**	1.99	90.49
Random^{+}	EN	PL	4.42	4.51	47.24
	PL	EN	6.79	2.30	42.74
Spearman$_r$	EN	PL	2.88	1.00	56.70
	PL	EN	4.62	2.21	71.05
Non Zero Feature	EN	PL	4.54	4.15	46.51
	PL	EN	4.92	1.76	18.02
Center$_r$	EN	PL	**2.54**	1.31	41.95
	PL	EN	5.96	2.35	41.93
PBI$_r$	EN	PL	3.84	3.39	67.89
	PL	EN	4.54	2.07	64.04

Table 6.15: Performance comparison of feature selection methods for hallucination detection in cross-lingual tasks for AggTruth JS-DIV. Gap Mean represents the percentage difference from best performance (lower values indicate better cross-lingual robustness), with Gap Std showing the variability across test datasets. Mean [%] Heads represents the percentage of attention heads used. (source: own elaboration)

6.5 Computational efficiency

An efficient detection method, in addition to being robust and easily adaptable to new cases, should also be computationally efficient. Since the aggregation method prepares features for the hallucination detector, which operates in real time as tokens are generated, low computation time is a crucial requirement. Even if a classifier effectively identifies hallucinated tokens, if the method is too slow, it cannot be used in production settings. Figure 6.3 presents the average computation time (in seconds) required to create classifier-ready features from raw attention maps for all AggTruth methods. As expected, hidden state-based approaches do not require any additional computation time to aggregate features (apart from averaging across the window), so they are not shown in the figure. The Lookback Lens implementation is highly inefficient, with feature preparation sometimes taking several minutes. As a result, only AggTruth methods are presented in the figure. To ensure a fair comparison, 240 examples were included in the measurement for each aggregation method. The lowest average computation time was observed for AggTruth Sum, at approximately 0.17 seconds. The time required to compute one example for AggTruth JS-DIV is about 20 times greater than for AggTruth Sum. The observed average computation time for AggTruth Sum is twice as fast as AggTruth CosSim and about 2.5 times faster than AggTruth Entropy. All differences in average computation time between pairs of methods are statistically significant (regardless of threshold, using the Welch's t-test, one-sided).

The features passed to the classifier also impact memory usage. Since all features (for both hidden state and attention based approaches) are continuous and can be efficiently represented as 32-bit floats, the main differentiating factor is their number. Figure 6.4 shows the memory consumption of features (in kilobytes per example) for the examined methods using the Llama-2 model. The figure also presents the percentage reduction in memory compared to the most memory-intensive approach (hidden states). Using AggTruth Sum with feature selection, which achieved the lowest Gap, resulted in approximately 85% memory savings per example compared to the hidden states approach. With the memory-optimized version (using about 20% of features), the memory required to store features was about 20 times lower than for hidden states.

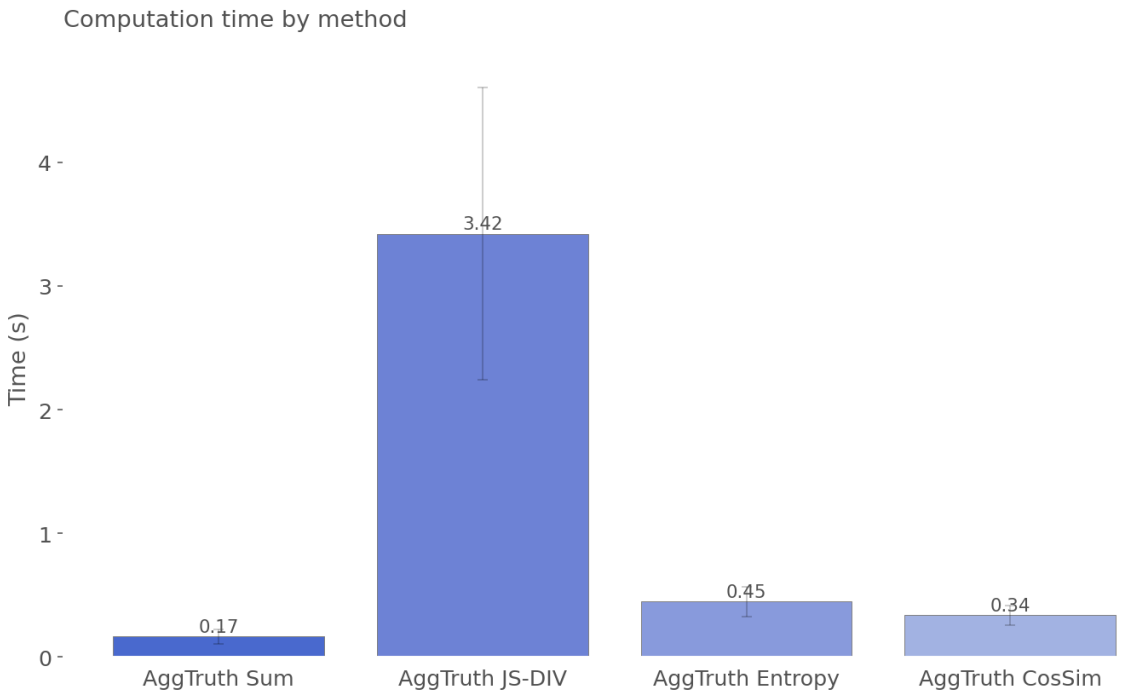

Figure 6.3: The average computation time (in seconds) required to create classifier-ready features from raw attention maps for all AggTruth methods. (source: own elaboration)

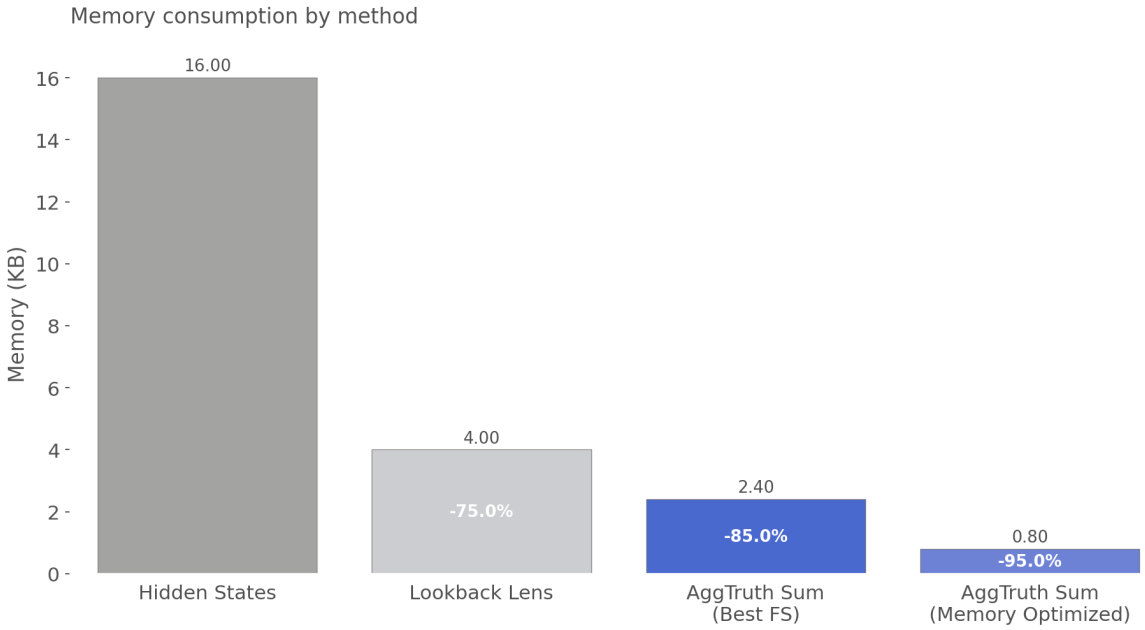

Figure 6.4: The memory consumption of features (in kilobytes per example) for the hidden state and attention based methods using the Llama-2 model. (source: own elaboration)

6.6 Main outcomes

Several experiments were conducted to compare different hallucination detection methods based on the internal states of large language models. To comprehensively evaluate these methods, their behavior was examined under varying window sizes, cross-lingual evaluation, and computational efficiency, including time and memory consumption. Additionally, various head selection methods for AggTruth aggregation were assessed. The experiments demonstrated that hallucination detection using a windowed approach is feasible even for window sizes of 4 and 1. Considering the overall mean Gap across all examined LLMs and window sizes, in the QA → SUM. evaluation setting, AggTruth JS-DIV and AggTruth Sum performed the best. Similarly, in the SUM. → QA setting, AggTruth Sum, with a slightly lower overall mean Gap than AggTruth JS-DIV, achieved the best results. AggTruth-based approaches (excluding CosSim) consistently outperformed hidden state-based approaches in most cases, indicating that attention-based methods contain information that can better differentiate between hallucinated and non-hallucinated examples for monolingual settings transferring between tasks (QA, SUM.) compared to hidden states. The experiments suggest that using hidden states from layer L - 8 may result in better detection performance than using L - 4 or the last layer, also showing that detection capabilities consistently decrease as the number of chosen layer increases. Moreover, results indicate that the hidden states approach works better when training on the QA task and transferring to SUM. than vice versa.

The lowest variances for AggTruth Sum in the conducted experiments, along with a stable, low Gap between window sizes, indicate that AggTruth Sum is the most robust approach for monolingual settings, being less influenced by the choice of task, dataset, model, or window size. The best hallucination detection performance can be achieved by selecting the appropriate aggregation method for each LLM. AggTruth Sum appears to be most aligned with Bielik and Gemma-2 models, while for Llama-3 and Phi-3.5, AggTruth JS-DIV outperformed other methods. For Llama-2, either AggTruth Entropy or AggTruth Sum seems appropriate. The results suggest that changing the window size does not alter this relationship.

Based on cross-lingual evaluation, further conclusions can be drawn. AggTruth methods (excluding CosSim) consistently performed better when trained on English and transferred to Polish than in the opposite scenario. The exact opposite was observed for hidden states. These results may indicate that attention patterns can serve as meaningful features for hallucination detection when the detector is trained on English and then used for Polish (a minority language). However, training on minority languages like Polish can yield better results when using hidden states. Using hidden states from the L - 8 layer results in a more capable and stable hallucination detector than using higher layers. Among AggTruth methods, for cross-lingual settings, AggTruth Entropy is the most effective, especially for the EN → PL configuration. Per-LLM capabilities were not maintained, as AggTruth Entropy appears to be the best option for cross-lingual settings across all models. The high performance degradation for AggTruth JS-DIV suggests it is not well-suited for cross-lingual tasks.

Applying feature selection methods results in higher hallucination detection performance compared to not applying them. A comprehensive examination indicates that

the Spearman feature selection method is the most stable across tasks, models, datasets, and aggregation methods. The proposed PBI method can serve as a second-best choice, sometimes outperforming Spearman in both mono- and cross-lingual settings. Results observed for selecting features with importance higher than random and positive coefficients indicate that this method is not suitable and performs worse than not applying feature selection at all. For cross-language tasks, results suggest that more heads are needed to detect hallucinations than in monolingual settings. Moreover, the benefit of applying feature selection is also reduced.

AggTruth aggregations are more computationally efficient in terms of memory required to store features compared to the Lookback lens and hidden state approaches, while AggTruth Sum aggregation is also the fastest to compute among AggTruth methods. As these hallucination detectors are intended for use during token generation in the decoding process of large language models, these characteristics make the AggTruth method more suitable and production-ready. Despite its strong hallucination detection capabilities (especially for Llama-3 and Phi-3.5), AggTruth JS-DIV is not acceptable for live token generation due to its 20 times higher compute time compared to AggTruth Sum. The experiments showed that AggTruth Sum can serve as an effective attention-based hallucination detection method, as it is the most robust approach, adapting to task, dataset, model, and window size, while also being optimized for compute time and memory usage.

7. Conclusions

This thesis introduced a novel method for hallucination detection in Retrieval-Augmented Generation systems based on attention map aggregation techniques within large language models. The proposed AggTruth approach effectively identifies contextual hallucinations while maintaining low computational requirements, demonstrating superior memory efficiency compared to Lookback Lens and hidden state approaches, with AggTruth Sum offering the fastest computation time among all AggTruth methods. Comprehensive experiments comparing various hallucination detection methods leveraging internal states of large language models showed AggTruth's robustness across different window sizes, evaluation settings, models, and datasets. This work also examined and compared feature selection methods which were shown to improve hallucination detection effectiveness, with the Spearman method proving most stable across tasks, models, and datasets, while the proposed PBI method served as a strong alternative.

AggTruth Sum also has its natural limitations, such as the access to context tokens' attention scores for generated tokens, which are not available when using flash-attention. Additionally, some attention patterns, such as sparse or window attention, can limit the use of AggTruth.

Several promising directions for future research emerge from this work. Expanding evaluation to include a broader range of languages beyond English and Polish would enhance understanding of multi-lingual hallucination detection capabilities. It is believed that analyzing specific parts of attention score vectors could open up new opportunities for detecting hallucinations, not only in RAG settings but also in other tasks.

7. Bibliography

[1] A. Azaria and T. Mitchell. The internal state of an llm knows when it's lying. *arXiv preprint arXiv:2304.13734*, 2023.

[2] F. F. Bayat, K. Qian, B. Han, Y. Sang, A. Belyi, S. Khorshidi, F. Wu, I. F. Ilyas, and Y. Li. Fleek: Factual error detection and correction with evidence retrieved from external knowledge. *arXiv preprint arXiv:2310.17119*, 2023.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[4] Y.-S. Chuang, L. Qiu, C.-Y. Hsieh, R. Krishna, Y. Kim, and J. Glass. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. *arXiv preprint arXiv:2407.07071*, 2024.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[6] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

[7] Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig. Does fine-tuning llms on new knowledge encourage hallucinations? *arXiv preprint arXiv:2405.05904*, 2024.

[8] T. Gemma and et al. Gemma 2: Improving open language models at a practical size, 2024.

[9] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Sebastopol, CA, 3 edition, 2022.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016.

[11] J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu, et al. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.

[12] L. Huang and et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. on Inform. Sys.*, 2024.

[13] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

[14] Z. Ji and et al. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, Mar. 2023.

[15] Z. Ke, H. Lin, Y. Shao, H. Xu, L. Shu, and B. Liu. Continual training of language models for few-shot learning. *arXiv preprint arXiv:2210.05549*, 2022.

[16] J. Kocoń, I. Cichecki, O. Kaszyca, M. Kochanek, D. Szydło, J. Baran, J. Bielaniewicz, M. Gruza, A. Janz, K. Kanclerz, et al. Chatgpt: Jack of all trades, master of none. *Information Fusion*, 99:101861, 2023.

[17] D. Kornbrot. *Point Biserial Correlation*, pages 1620–1621. Wiley, Oct. 2005.

[18] M. B. Kursa and W. R. Rudnicki. Feature selection with the boruta package. *Journal of statistical software*, 36:1–13, 2010.

[19] T. Kwiatkowski and et al. Natural questions: A benchmark for question answering research. *Transactions of the ACL*, 7:452–466, 2019.

[20] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[21] X. Liu, H. Yan, S. Zhang, C. An, X. Qiu, and D. Lin. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*, 2023.

[22] P. Manakul, A. Liusie, and M. J. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*, 2023.

[23] P. Matys, J. Eliasz, K. Kiełczyński, M. Langner, T. Ferdinan, J. Kocoń, and P. Kazienko. Aggtruth: Contextual hallucination detection using aggregated attention scores in llms. In *International conference on computational science*, page (accepted for publication). Springer, 2025.

[24] S. Narayan, S. B. Cohen, and M. Lapata. Don't give me the details, just the summary. *Topic-Aware Convolutional Neural Networks for Extreme Summarization. ArXiv abs/1808.08745*, 2018.

[25] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, M. Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[26] B. Porat. Constant tensors and constant linear transformations. In *A Gentle Introduction to Tensors*, pages 14–15. Technion, Haifa, Israel, 2nd edition, 2014. First published in 2010.

[27] A. See and et al. Get to the point: Summarization with pointer-generator networks. In *ACL 2017*, pages 1073–1083. ACL, 2017.

[28] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[30] Y.-A. Wang and Y.-N. Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. *arXiv preprint arXiv:2010.04903*, 2020.

[31] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[32] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[33] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.