Chapter 3

Methodology

3.1 LLMs for Check-worthy Statement Detection

Open-source LLMs provide notable benefits across multiple dimensions, cost-effectiveness, transparency, and ethical implications. Based on these advantages, several major open-source LLMs were examined for the fact-checking worthiness classification task, The models that we tested were Llama-2-7b, GPT-2, Llama-3.2-1b, TinyLlama, and Microsoft Phi-2. The objective is to determine whether the given statement contains claims or information that requires verification. The task is presented as a case of binary classification with two possible results Yes (the statement is check-worthy) and No (he statement is not check-worthy).

This study employs a comprehensive methodology for check-worthy statement detection using multiple open-source language models. The approach consists of three main phases, the binary classification task is formulated as a text generation problem using structured prompt templates, then the models undergo parameter-efficient fine-tuning to adapt them for the task and finally, a robust evaluation strategy employing multiple sampling and statistical analysis is used to assess model performance. This methodology is designed to provide reliable and statistically significant results while working within computational constraints.

3.1.1 Prompt Engineering and Task Formulation

The task of fact-checking worthiness classification in this project is formulated as text generation problem with a well-structured prompt template. This method converts the binary classification problem to a natural language generation framework and allows the models to leverage their pre-trained conversational abilities. The prompt's structure follows a stan-

dard format with three components: one instruction section that carries the system message refers to the description of the task, an input sentence section providing the statement for classification, and a response section which defines the expected result from the model.

The above template design supports several purposes. It defines clear task boundaries, it ensures consistent formatting across all training entries, and it also matches instruction-follow up patterns that are often employed in language model training. The structured format makes sure models can correctly analyze the input and comprehend the desired response format.

This setup allows models to use their natural language generation abilities to make decisions in a more flexible and context-aware way. By generating responses instead of choosing from fixed labels, they can better understand the nuances of each statement and determine whether it contains a verifiable claim. To keep outputs consistent and easy to interpret, the models are guided to respond with a simple ”Yes” for check-worthy statements, or ”No” for those that are opinion-based or trivial.

3.1.2 Fine-tuning

The selected models went through supervised fine-tuning, which is a process when labeled training data is used to modify pre-trained causal language models to a specific task. This approach leverages the models' existing language understanding capabilities while teaching them to distinguish between statements that require fact-checking verification and those that are purely opinion-based or trivial.

Considering the computational constraints and memory limitations for full fine-tuning of large language models, Low-Rank Adaptation (LoRA) was used, which is a parameter-efficient technique. LoRA tackles the difficulty of fine-tuning large models by decomposing weight updates into low-rank matrices, considerably reducing the number of trainable parameters while preserving model performance.This technique is particularly beneficial for academic research environments where computational resources, and more importantly GPU memory, are limited. Rather than updating all model parameters, LoRA freezes the original weights of the pre-trained model and introduces lightweight, trainable low-rank matrices within specific layers. These low-rank matrices serve as adaptation modules and they adjust the model's outputs for the new task while the original weights remain unchanged. By modifying only a small subset of parameters, LoRA significantly reduces computational and memory overhead. This approach enables fast and efficient fine-tuning, making it ideal for

research environments with limited hardware resources.

To further handle memory constraints, 4-bit quantization has been implemented, in the training process, in order to reduce the memory use and enable efficient training on limited hardware. It is a model compression technique that reduces the precision of model weights and activations from 16- or 32-bit floating-point representations to 4-bit ones. This type of quantization is employed during both training and inference in order to load the whole model into GPU memory utilizing significantly fewer resources.

3.1.3 Inference Strategy and Prediction Aggregation

In order to overcome the inherent randomness in language model generation, and to ensure more consistent, reproducible results and improved prediction reliability, a multiple sampling approach is implemented. Each test sample is processed five times independently, producing multiple predictions for the same input. This strategy serves two key purposes by providing insight into the model's confidence and uncertainty and it enables the construction of more robust final predictions. The final predictions are determined using majority vote on the five generated responses per sample. This ensemble method minimizes the impact of individual errors and leads to more stable classifications.

3.2 Knowledge Distillation in LLMs for Check-worthy Statement Detection

3.2.1 Knowledge Distillation Strategy

Knowledge distillation is a model compression technique in which a smaller, more efficient ”student” model is trained to mimic the behavior of a larger, previously trained ”teacher” model. Given the complexity of its neural structure, the teacher model requires the tuning of a substantial number of parameters to accurately learn and represent patterns present in the training dataset. The fundamental goal of Knowledge Distillation is to transfer the generalized knowledge embedded in the teacher's predictions to the student, allowing the learner to achieve similar performance with much fewer parameters and reduced computational overhead. This approach is well-suited for deployment in real-world scenarios where available computing resources are limited. In this study, this transfer is accomplished by ”driving”

the student model to match the softer probability distributions (logits) of the teacher model, rather than based just on hard ground truth labels. In this implementation, the teacher model has been fine-tuned on the identification of check-worthy statements task and represents the ”ground truth” knowledge that needs to be transferred. In our approach we utilize a combined loss function that balances standard cross-entropy loss with knowledge distillation loss, as relying solely on soft targets risks error propagation while hard labels alone would discard the teacher’s learned inter-class relationships.

3.2.2 Model Architecture and Preparation

Teacher Model

We utilize a pre-trained large language model of substantial capacity for the teacher architecture. In the distillation process, the teacher remains strictly in inference mode. The teacher is derived from the above method explained in section 3.1 after its full training. It is stored and we load it for the Knowledge distillation process for the training of the student. The teacher’s objective is to generate ”soft targets” - probability distributions of all possible output tokens for a given input. These softened logits provide a more comprehensive signal than hard labels, conveying the teacher’s confidence and the relationships between different output classes. The teacher model’s parameters remain constant during the training of the student model.

Student Model

The student model is a comparatively smaller large language model, more computationally efficient model that serves as the knowledge recipient. To further optimize its memory footprint and accelerate training, the student model undergoes 4-bit quantization and also Low-Rank Adaptation is applied as we mention in 3.1.2.

Distillation Training Process

The student model’s training aims to incorporate teacher knowledge as well as learning from ground-truth labels. This is accomplished with a hybrid loss function that has two main components, the Standard Cross-Entropy Loss and Knowledge Distillation Loss.

$$L_{total} = \alpha \cdot L_{CE} + (1 - \alpha) \cdot L_{KD} \cdot T^2 \tag{3.1}$$

Alpha - Weighting Parameter (α) controls the balance between learning from ground truth (α closer to 1) versus learning from teacher (α closer to 0). Typical values range from 0.1 to 0.9, with α = 0.7 being commonly used. Temperature Squared compensates for the gradient magnitude reduction caused by temperature scaling. When temperature $T > 1$, the gradients from the distillation loss become smaller, so multiplying by T^2 ensures the distillation loss contributes meaningfully to the overall gradient updates.

The Cross-Entropy loss (L_{CE}) ensures that the student learns from the ground truth labels. It is the conventional loss between the student model's predicted logits and the true, one-hot encoded ground-truth labels. It assures the student learns to correctly classify the inputs based on the provided dataset.

$$L_{CE} = -\sum y_{true} \cdot \log p_{student} \tag{3.2}$$

y_{true} represents the one-hot encoded ground truth labels and $p_{student}$ is the student model's probability distribution over classes.

The Knowledge Distillation Loss (L_{KD}) quantifies the difference between the student's and teacher's softened output distributions. It captures the rich knowledge embedded in the teacher's soft predictions and it is computed as the Kullback-Leibler (KL) divergence between the student's log-softmax probability and the teacher's softmax probabilities. It is also known as relative entropy. The KL divergence measures how much the student's probability distribution differs from the teacher's distribution. This loss component enables the transfer of nuanced knowledge such as understanding which classes are more similar to each other, learning when to be confident versus uncertain in predictions and capturing the teacher's learned feature representations and decision-making patterns.

$$L_{KD} = KL(p_{teacher}||p_{student}) = \sum p_{teacher} \log \frac{p_{teacher}}{p_{student}} \tag{3.3}$$

$$p_{teacher} = softmax(logits_{teacher}/T) \quad p_{student} = softmax(logits_{student}/T) \tag{3.4}$$

$p_{teacher}$ and $p_{staudent}$ represent the teacher's and the student's softened probability distributions respectively. T is the temperature parameter that controls the softness of the distributions. Here the temperature parameter acts as a "knowledge controller" that determines how much of the model's internal reasoning becomes accessible. At T=1, the model produces

its natural, confident predictions where one class typically dominates (e.g., 0.9 vs 0.1), effectively hiding the model's understanding of class relationships. As temperature increases beyond 1, the distribution becomes progressively flatter, revealing increasingly subtle distinctions the model makes between classes - essentially showing the model's "thought process" about why certain alternatives were considered but ultimately rejected. In contrast, temperatures below 1 make distributions sharper and more decisive, approaching binary decisions that contain even less information than the model's natural predictions.

The knowledge distillation process operates through a systematic training procedure where both teacher and student models process the same input data simultaneously. During each training step, the input text is first fed to the teacher model, which generates soft probability distributions using temperature-scaled softmax operations. Concurrently, the same input is processed by the student model to produce its own predictions. The hybrid loss function then computes the combined objective by evaluating both the student's alignment with ground truth labels (cross-entropy component) and its similarity to the teacher's soft targets (KL divergence component). Through backpropagation, gradients flow only through the student model's parameters, gradually adjusting its weights to minimize the combined loss. This iterative process enables the student to progressively learn both task-specific accuracy and the nuanced decision-making patterns encoded in the teacher's predictions, ultimately achieving comparable performance with significantly reduced computational requirements.