UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Knowledge Distillation in Large Language Models for Check-worthy Statement Detection

Diploma Thesis

Aristea Koutsomarkou

Supervisor: Eleni Tousidou

September 2025

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Knowledge Distillation in Large Language Models for Check-worthy Statement Detection

Diploma Thesis

Aristea Koutsomarkou

Supervisor: Eleni Tousidou

September 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Απόσταξη γνώσης σε Μεγάλα Γλωσσικά Μοντέλα για την Ανίχνευση Δηλώσεων που Χρήζουν Ελέγχου

Διπλωματική Εργασία

Αριστέα Κουτσομάρκου

Επιβλέπουσα: Ελένη Τουσίδου

Σεπτέμβριος 2025

Approved by the Examination Committee:

Supervisor **Eleni Tousidou**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Member **Michael Vassilakopoulos**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Athanasios Fevgas**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

I would like to express my deepest appreciation to my supervisor, Prof. Eleni Tousidou, whose guidance and advice have been fundamental to the completion of my thesis. I am thankful for her thoughtful suggestions and steady support, which guided me through the challenges of this work and contributed to the final thesis.

I would also like to express my sincere gratitude to the other members of my thesis committee, Prof. Michael Vassilakopoulos and Prof. Athanasios Fevgas, for their time and careful review of my thesis.

Finally, I would like to thank my family and friends for their support throughout my studies, their constant encouragement, and their understanding during challenging times. Their love and guidance have been invaluable and have kept me motivated throughout this journey.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Aristea Koutsomarkou

Diploma Thesis

Knowledge Distillation in Large Language Models for Check-worthy Statement Detection

Aristea Koutsomarkou

Abstract

In the current digital landscape, information travels at unprecedented speed, but its trustworthiness often remains uncertain. The proliferation of misinformation has made the identification of check-worthy statements an important task. A recent study evaluates numerous open-source large language models for this task. It focuses on political statements from the CT24 dataset provided by CheckThat Lab in the 2024 competition. Their study highlights that Llama2-7b, when combined with prompt engineering, is the most effective approach. This thesis builds on that work and reproduces those findings, extending the analysis to the CT22 dataset of COVID-19–related tweets to evaluate how well the method performs across different contexts and domains. Moreover, to overcome the computational limits layed by Llama2-7b due its requirements in GPU memory, we perform Knowledge Distillation techniques using a combined loss function to transfer knowledge coming from Llama2-7b (the teacher), to smaller but efficient models (the students). We experiment with both Kullback-Leibler (KL) divergence and Universal Logit Distillation (ULD) approaches within the combined loss framework. The results show that ULD outperforms KL divergence and maintains comparable performance to the teacher model while reducing computational requirements.

Keywords:

check-worthiness, LLMs, claim detection, NLP, Knowledge distillation, teacher model, student model, ULD, KL divergence

Διπλωματική Εργασία

Απόσταξη γνώσης σε Μεγάλα Γλωσσικά Μοντέλα
για την Ανίχνευση Δηλώσεων που Χρήζουν Ελέγχου

Αριστέα Κουτσομάρκου

Περίληψη

Στη σύγχρονη ψηφιακή εποχή, η διάδοση τοσο των πληροφοριών όσο και των ειδήσεων γίνεται ολοένα και ταχύτερη, ενώ η αξιοπιστία τους συχνά παραμένει αμφίβολη. Η ραγδαία εξάπλωση της παραπληροφόρησης καθιστά αναγκαία την ανάπτυξη εργαλείων για την ανίχνευση δηλώσεων που χρήζουν ελέγχου. Μια πρόσφατη μελέτη, εξερεύνησε και αξιολόγησε πληθώρα μεγάλων γλωσσικών μοντέλων για αυτόν το σκοπό. Η μελέτη επικεντρώθηκε σε πολιτικές δηλώσεις από το σύνολο δεδομένων CT24, που παρέχεται από το CheckThat Lab στον διαγωνισμό του 2024, όπου διαπίστωσε ότι το μοντέλο Llama2-7b, σε συνδυασμό με τεχνικές prompt engineering, αποτελεί την πιο αποτελεσματική προσέγγιση. Η παρούσα εργασία επεκτείνει την προηγούμενη μελέτη, αναπαράγοντας τα αποτελέσματα της και εφαρμόζοντας την ίδια μέθοδο στο σύνολο δεδομένων CT22, το οποίο περιλαμβάνει αναρτήσεις στο Twitter σχετικές με την COVID-19. Στόχος είναι η αξιολόγηση της απόδοσης της μεθόδου σε διαφορετικά θεματικά πεδία και είδη περιεχομένου. Προκειμένου να υπερβούμε τους περιορισμούς που θέτει το Llama2-7b λόγω των απαιτήσεών του σε σημαντικούς υπολογιστικούς πόρους και μνήμη GPU, εφαρμόσαμε τεχνικές Απόσταξης Γνώσης χρησιμοποιώντας μια συνάρτηση κόστους που συνδυάζει δύο όρους, προκειμένου να μεταφέρουμε γνώση από το ισχυρό μοντέλο Llama2-7b σε μικρότερα αλλά αποδοτικά μοντέλα. Πραγματοποιήθηκαν πειράματα χρησιμοποιώντας τόσο την απόκλιση Kullback–Leibler (KL) όσο και την Universal Logit Distillation (ULD) στο πλαίσιο της συνάρτησης κόστους. Τα αποτελέσματα αποδεικνύουν ότι η ULD υπερέχει σε σχέση με την KL και διατηρεί συγκρίσιμη απόδοση με το μεγαλύτερο και πιο ισχυρό μοντέλο, μειώνοντας σημαντικά τις υπολογιστικές απαιτήσεις.

Λέξεις-κλειδιά:

Αξιοεπαλήθευτο, Μεγάλα Γλωσσικά Μοντέλα, Ανίχνευση ισχυρισμών, Επεξεργασία Φυσικής Γλώσσας, Απόσταξη Γνώσης, Μοντέλο Διδάσκων, Μοντέλο Μαθητευόμενο, ULD, KL απόκλιση

Table of contents

List of figures

List of tables

Abbreviations

KL	Knowledge Distillation
ULD	Universal Logit Distillation
KL	Kullback-Leibler Divergence
LLMs	Large Language Models
SVM	Support Vector Machines
LSTM	Long Short-Term Memory
LoRA	Low-Rank Adaption
GPU	Graphics Processing Unit
CNN	Condensed Nearest Neighbor
LogReg	Logistic Regression
SMOTE	Synthetic Minority Over-sampling Technique
GRU	Gated Recurrent Unit
RNN	Recurrent Neural Network
CBOW	Continuous Bag of Words
PMI	Pointwise Mutual Information
TF-IDF	Term Frequency-Inverse Document Frequency
RF	Random Forest Classifier
NB	Naive Bayes Classifier
GloVe	Global Vectors for Word Representation
RAG	Retrieval-Augmented Generation

Chapter 1

Introduction

The widespread use of social media networks and internet platforms facilitates access to information. Despite the availability of vast corpora and easy access to information, authenticity and reliability remain a major concern, due to the quick spread of misinformation. This rapid diffusion of false information leads to the necessity for automated systems that can identify statements that require fact-checking. Consequently, check-worthy statement detection is a critical task that distinguishes verifiable claims from statements or opinions. As a core mechanism supporting fact-checking pipelines, it is important to prevent misinformation and disinformation while maintaining information quality.

The innovative applications of LLMs offer powerful capabilities to enhance information analysis. They support natural language understanding and generation and enable several tasks that include summarization, question answering, and automated fact-checking. Nevertheless, the computational requirements of these models bring significant limitations. Memory requirements, processing time, inference speed, and operational costs, among others, are all equally challenging, especially in demanding cases like when dealing with the content from social media or news feeds that require near-real-time processing.

Knowledge distillation provides an effective approach to address this issue. It transfers knowledge from large, computationally demanding teacher models to smaller, more efficient student models while preserving comparable performance levels.

1.1 Thesis Subject

This thesis explores knowledge distillation methods in LLMs for check-worthy statement detection. It compares large teacher models with efficient student architectures. Recent work,

such as CheckThat Lab (CTL) competitions [1], has applied large language models to check-worthiness detection and show impressive results but with heavy computational costs. The current thesis initially reproduces previous findings on check-worthiness, applied on a dataset containing political statements. Secondly, it extends the evaluation to a different dataset with tweets that are related to COVID-19. However, the primary contribution of this thesis is the investigation of knowledge distillation strategies for the check-worthiness detection task. In particular, we examine whether knowledge transfer from a large teacher model to smaller student models may preserve competitive performance and, at the same time, reduce memory and inference constraints. To this end, the thesis explores knowledge distillation strategies and focuses on a comparison between KL divergence and the recently introduced Universal Logit Distillation methodology which enables knowledge transfer across different model families.

1.2 Thesis Structure

The thesis is organized in seven chapters. In Chapter 2 we review related work on check-worthy detection and knowledge distillation tasks. Chapter 3 presents the theoretical background of what i being used. Chapter 4 describes the methodology implemented, including the workflow of LLMs for the check-worthy statement detection task and knowledge distillation methods. Chapter 5 presents details of the datasets and the pre-processing steps. In Chapter 6 we see the experimental evaluation, including setup, model performance analysis, and a comparison of knowledge distillation strategies. Finally, in Chapter 7, we conclude the thesis and discuss directions for future research. Supplementary material that includes some preliminary experiments and method exploration on the dataset of tweets related to COVID-19, is provided in the Appendices A, B.

[1] https://checkthat.gitlab.io/clef2025/

Chapter 2

Related Work

2.1 Check-worthiness Detection Techniques

Check-worthiness prediction is an important initial phase in fact-checking pipelines. The goal is to identify and prioritize the most important information that is likely to require further verification, rather than attempting to check everything. Several methods have been developed over the years to address this task, ranging from feature representation techniques with traditional machine learning models to approaches based on LLMs. Recent advances in deep learning and pre-trained language models have also significantly improved the ability to detect check-worthy content in various domains. Research in this area was largely catalyzed by the 2016 U.S. presidential election, with most initial studies utilizing American political discourse. Consequently, the field has been English-dominated, though some work has explored other languages, including Arabic and Turkish.

2.1.1 Machine learning Methods

One of the first approaches for detecting check-worthy claims is ClaimBuster, introduced by Hassan et al. [6]. It relies on the extraction of important features across various categories from the text. The extracted features reach a total of 6,615. These include sentiment, Term Frequency-Inverse Document Frequency (TF-IDF), weighted bag-of-words (BoW), word count, part-of-speech tags and entity types. The authors experimented with multiple combinations of these features alongside several supervised learning algorithms, including multinomial Naive Bayes (NB), Support Vector Machine (SVM), and Random Forest Classifier (RF), to identify the most effective approach for check-worthiness prediction. Among

these, the SVM classifier achieved the best performance.

Ullah [7] introduced a model to predict the check-worthiness of information, integrating both semantic and content-quality features. The method utilizes pre-trained Word2Vec embeddings to capture semantic meaning, along with selected attributes from the Information Nutritional Label framework [8], specifically factuality, emotion, dispute and technicality. The combination of these features is used to train a stochastic gradient descent classifier with a logistic regression function, achieving promising results for the task.

In their study Lespagnol et al. [9] examined several models, including logistic regression (LR), SVM and RFs. They incorporated diverse feature sets such as part-of-speech tags, named entities, word embeddings, syntactic dependency relationships, and "information nutritional" attributes that capture aspects like credibility, controversy, emotional content, factuality, and technical complexity. The best-performing model was logistic regression utilizing word embeddings and information nutritional features, achieving superior results in CTL 2018.

2.1.2 Deep Learning and Pretrained Language Model Methods

In addition to classical ML models, recent studies explore deep learning techniques, transformer-based models, pretrained language models, and other advanced methods. One notable approach is G2CW framework by Ria Jha et al. [10] for multi-class classification. They use pre-trained global vectors for word representation (GloVe) embeddings to obtain vector representations, which are then processed by a Gated Recurrent Unit (GRU) to capture sequential dependencies for sentence classification. The GRU outputs are passed through fully connected dense layers with softmax activation function to classify sentences into non-factual, unimportant factual, and check-worthy factual.

CheckThat! lab[1] is an annual competition since 2018 which focuses on misinformation detection and contains several tasks that support many different stages of a fact-checking process. Over time, the CheckThat! lab has expanded beyond core verification tasks to include auxiliary subtasks such as subjectivity, evidence retrieval, and detection of political bias, among others, that support the fact-checking workflow. One of the several models developed for CheckThat! lab is the proposed method [11] in the CLEF-2019[12] competition, which achieved the first place in the check-worthiness detection task for political content.

[1] https://checkthat.gitlab.io/editions/

They introduce the Neural Check-Worthiness Model (NCWM) that represents each word using a word2vec embedding and a one-hot encoding of its syntactic dependency tag. These are concatenated and fed into a long short-term memory (LSTM) network, with outputs aggregated via an attention-weighted sum. The final check-worthiness score is obtained through a sigmoid layer.

Li et al. [13] approached the challenge of detecting check-worthy content in political transcripts, for the CheckThat! 2024 competition, by fine-tuning eight open-source large language models. The authors used Low-Rank Adaptation (LoRA) for memory-efficient fine-tuning and they also implemented a comprehensive prompt engineering strategy to optimize model performance for political text analysis. A significant contribution of the research is the proposal of a two-step data pruning that refines the training set for check-worthiness detection by filtering for informative sentences and under-sampling from the majority class. The pipeline first selects informative sentences according to four criteria, check-worthy status, presence of named entities, usage of informative verbs, and sentence length. Approximately 44% of the original data was maintained after removing non-informative cases (10.5%) and applying Condensed Nearest Neighbor method to balance classes. Despite utilizing less data, the model achieved accuracy and F1 scores comparable to full-data training, saving fine-tuning time by more than a half. The Llama2-7b model, which has 7 billion parameters, achieved state-of-the-art performance on the check-worthiness estimation task, securing first place in the competition with an F1-score of 0.82. To ensure prediction reliability, the authors employed multiple inference runs with majority voting.

Kartal et al. [14] present a hybrid approach that combines fine-tuned mBERT predictions with various engineered features. The study investigates multiple strategies for expanding the volume of labeled data to enhance model training. These include augmenting the number of positive samples (IncPos) and implementing active learning (AL) techniques. Through extensive experimentation, they demonstrate that their approach outperforms all existing state-of-the-art models on the CLEF CheckThat! Lab (CTL) benchmark datasets from 2018 and 2019. The results highlight the effectiveness of combining transformer-based architectures with feature engineering for check-worthiness detection.

2.2 Knowledge Distillation

Knowledge Distillation (KD) is a type of model compression that was initially shown by Bucilua et al. [15] in 2006. Knowledge distillation was originally proposed by Hinton et al. [16], as a method for training a smaller, "student" model to mimic the behavior of a larger, pre-trained "teacher" model. Rather than training the student on hard labels, the student learns from the teacher model's soft labels, which are the probability distribution across classes. Following this, KD has been extensively explored across various domains, with researchers proposing several techniques to transfer knowledge from large, complex models to smaller, but efficient ones.

2.2.1 Knowledge Distillation in Large Language Models

While transformer-based models are becoming increasingly powerful, their computational requirements make knowledge distillation necessary for developing more efficient alternatives. Wang et al. [17] introduce the MMIDR framework which is designed for training LLMs to analyze multimodal misinformation and explain their reasoning. The approach uses response-based knowledge distillation where a teacher LLM generates rationales based on processed multimodal inputs. Visual inputs are initially converted to text through OCR and image captioning, then combined with retrieved supporting evidence to instruct a teacher LLM. The rationales are then used to fine-tune a smaller student LLM with LoRA, which freezes the base model and only updates the rank-decomposition parameters in Transformer layers.

Gu et al. [18] propose MiniLLM, a technique to enhance the efficiency of large language models through reduced resource intensity. In this approach, unlike most KD methods that use forward Kullback–Leibler (KL) divergence, which forces the student to imitate the teacher's complete output distribution, MiniLLM uses reverse KL divergence. Forward KL frequently leads to poor results in text generation tasks because smaller models cannot accurately capture the complex distribution of large teachers. In contrast, reverse KL divergence encourages students to focus on the high-probability modes of the teacher's distribution. This mode-seeking behavior helps the learner produce higher-quality and more faithful responses to the teacher, making the technique more appropriate for distilling large language models.

Zhang et al. [19] propose BD-LLM, a method that combines Bootstrapping and Distilling

LLMs for the task of toxic content detection. First the authors present a prompting method called Decision-Tree-of-Thought (DToT) which repeatedly selects fine-grained contexts to bootstrap high-confidence predictions and extract rationales from the LLM. DToT utilizes a confidence checker, context tree, and context selector to guarantee that the LLM's outputs are reliable and contextually appropriate. After collecting answers and rationales from LLMs through DToT, BD-LLM then distills the knowledge from the LLM into a smaller student model that is fine-tuned to reproduce answers and rationales, achieving high accuracy with a much smaller model size.

Chapter 3

Theory Background

3.1 Machine Learning Models

Machine learning uses a wide range of algorithms and architectures. Each one addresses specific computational challenges and data characteristics. This part discusses the key machine learning models that form the theoretical and practical foundation for the models that have been tested in the check-worthiness estimation task.

3.1.1 Traditional Machine Learning Models

There are many traditional machine learning models that have been used in a variety of tasks. Some of them are discussed below. The key models relevant to our work are discussed below.

Support Vector Machine (SVM)

A Support Vector Machine [20] is a supervised machine learning algorithm that can be used for both classification and regression tasks. The core idea of SVM is to find an optimal decision boundary, known as hyperplane, to separate data into different classes (Figure 3.1). The main objective of SVM is to maximize the margin, which is the distance between the hyperplane and the closest data points from each class. A larger margin results in a more robust and generalizable model.

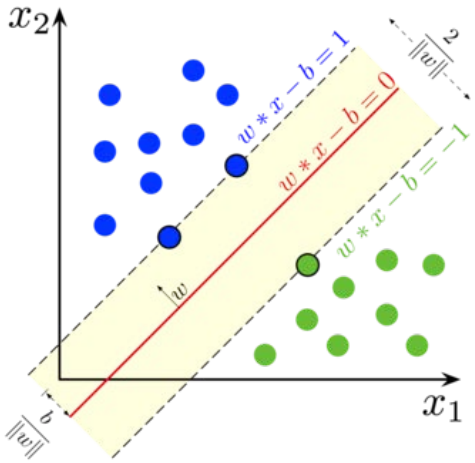

Figure 3.1: Linear SVM with Maximum-Margin Hyperplane and Support Vectors [1]

eXtreme Gradient Boosting (XGBoost)

XGBoost is a fast and efficient machine learning algorithm known for its high performance. It is an advanced gradient boosting method and a type of ensemble learning algorithm. It combines many weak learners, decision trees, to create a robust predictive model. The method builds decision trees sequentially with each new tree correcting the mistakes of the previous one (Figure 3.2). It also employs optimization techniques and regularization methods to prevent overfitting and enhance performance.

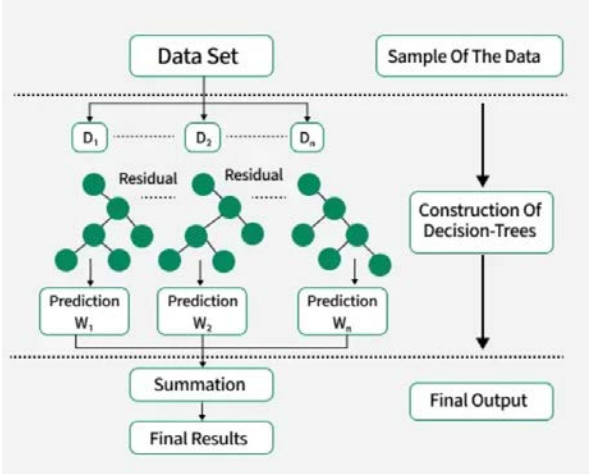

Figure 3.2: XGBoost workflow [2]

Logistic Regression (LR)

Logistic Regression is a supervised machine learning algorithm designed for classification problems, primarily used for binary classification tasks [21]. It analyzes how the input

features affect the probability of a categorical result. In contrast with linear regression, which predicts continuous values, logistic regression estimates the probability that a data point belongs to a particular class. The method utilizes a sigmoid function, which converts the raw output of the model to probability values ranging from 0 to 1.

Adaptive Boosting (AdaBoost)

AdaBoost is a boosting technique introduced by Freund and Schapire [22]. It is an ensemble classifier which combines several weak learners, typically using shallow trees, to form a powerful predictor. At first, this algorithm assigns to all training samples the same weight and then updates the weights in each round to emphasize the misclassified instances. The core idea is to iteratively resolve the mistakes of prior learners. The final classification decision is determined by combining the predictions of all weak learners via a weighted majority vote.

3.1.2 Neural-Based Models

There are also neural-based models that have revolutionized natural language processing and sequence modeling tasks and they achieve state-of-the-art performance across various domains.

Long Short-Term Memory (LSTM)

Long Short-Term Memory network [23] represents a particular type of Recurrent Neural Network (RNN) designed to address the vanishing gradient problem in traditional RNN architectures. LSTMs show excellent performance when learning long-term dependencies. The key innovation is in their memory cell mechanism. This cell can keep information over long sequences, thereby enables the network to remember relevant context from much earlier in the sequence and also process inputs. Three gating components enable this capability, an input gate that controls what new information is stored, a forget gate determines what existing information is deleted and an output gate which controls what information outputs from the memory cell.

3.2 Large Language Models (LLMs)

Large Language Models (LLMs) are AI advanced systems built using deep neural networks that have been trained on vast amounts of data to understand and generate natural language. They mark a significant breakthrough in natural language processing (NLP). These models have found applications across numerous fields, such as healthcare, education, finance, customer service, software development, and scientific research. They perform several tasks including document analysis, automated writing, translation, sentiment analysis, conversational question answering, text translation, classification, and generation.

Most LLMs are based on a transformer architecture which was proposed by Vaswani et al.[3] and utilizes attention mechanisms, particularly self-attention mechanisms, that allow parallel processing and capture long-range dependencies within sequences effectively. These types of mechanisms grant models the ability to assess the overall context of an input, thereby leading to thorough contextual understanding.

3.2.1 Transformer Architecture

The use of neural networks in NLP led to an important evolution, with RNN and LSTM networks enhancing the ability to model sequences. Yet, these architectures struggled with long sequences and complex dependencies. The introduction of the transformer architecture, as mentioned previously, signifies a revolutionary shift in how neural networks process sequential data. The transformer relies completely on attention mechanisms, which, as noted earlier, give the model the ability to consider the full context of a sequence and capture relevant dependencies and information more effectively.

As illustrated in Figure 3.3, the transformer has two primary components, a decoder and an encoder. The encoder transforms the input sequence into contextual representations, and the decoder produces the output sequence using these representations along with the tokens it has already generated.

Core Components

The core innovation of transformers is the self-attention mechanism, which enables each position in a sequence to attend to every other position in the input. More specifically, the use of multi-head attention serves as a key element within the transformer architecture. It runs

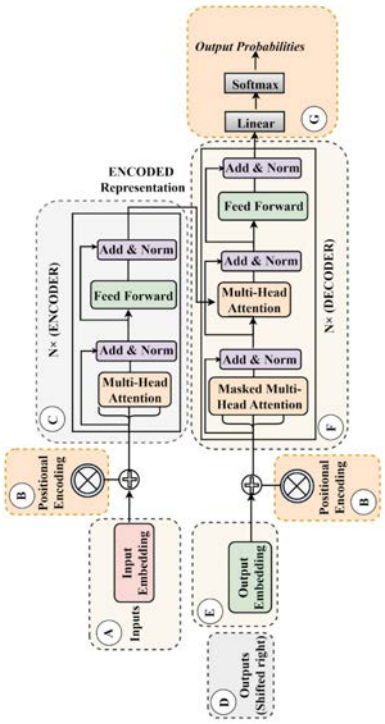

Figure 3.3: Transformer architecture [3]

many attention mechanisms in parallel and enables the model to consider various relational aspects concurrently. Since transformers process tokens in parallel, they rely on positional encodings that are added to the input embeddings in order to information about token order and preserve the sequence structure.

Each layer contains a feed-forward network applied independently to each position in the sequence. There are two linear transformations with a ReLU activation in between for additional processing capacity. Residual connections are present and surround each sub-layer, followed by layer normalization. This enhances training stability and deeper architecture can be constructed to prevent gradient vanishing problems that can appera in these model architectures.

This combination of these components enables transformers and makes them ideal to handle sequences in parallel while also capturing complex long-range dependencies, which makes them especially effective for language modeling.

Training Methodologies

To operate effectively across multiple tasks, LLMs utilize numerous training and inference strategies that offer flexibility across applications and domains.

Pre-training is the fundamental of current LLMs. The models acquire semantics, linguistic patterns and word knowledge from large corpus. They achieve this through approaches like next-token prediction or masked language modeling, which produce general-purpose representations that capture wide knowledge.

Fine-tuning adapts pre-trained models to particular tasks or domains using task-specific datasets. It enables specialization while keeping the general knowledge during pre-training. It uses less data and computational resources than training from scratch. Further details on this technique are provided in a subsequent section.

Few-shot and zero-shot learning highlight LLMs' ability to perform new tasks with minimal or no exposure to specific training data. In the context of few-shot learning, the model receives and is trained in a small number of examples within the prompt. In contrast, zero-shot learning performs tasks without any sample given, depending exclusively on natural language instructions and model's pre-trained knowledge.

In-context learning describes a model's capability to adapt to new tasks and provide responses based only on information in the input context. This method allows task execution without any parameter updates or extra training and shows the model's flexibility across several tasks.

LLM Architectures and Capabilities

LLMs have three primary transformer architectures, each one optimized for a different language processing task. There are encoder-only, decoder-only and encoder-decoder models. **Encoder-only** models, such as BERT and RoBERTa, process text bidirectionally and each token can attend to all others. This makes them effective for many tasks such as classification and sentiment analysis, but are not suitable for generation tasks. **Decoder-only** models, like GPT, adopt an autoregressive method and each token attends only to previous tokens. This implies that they are ideal for text generation and completion tasks. **Encoder-decoder models**, such as T5 and BART, integrate both architectures. They use bidirectional encoding for input and autoregressive decoding for output, therefore being well-suited for sequence-to-sequence tasks like translation and summarization.

Modern LLMs exhibit enhanced capabilities thanks to their architecture and scalability. Contextual understanding helps these models to capture semantic relationships, while also preserve coherence over long texts. Zero-shot and few-shot inference let them perform tasks

with a few or no examples, whereas prompt engineering has emerged and guides the model's behavior with carefully crafted input instructions.

The scalability of LLMs allows them to handle extensive inputs and adapt to various domains, making them versatile NLP tools. Beyond simple text processing, current LLMs thrive at reasoning, complex problem-solving and summarization. They are able to combine information from multiple sentences or documents, perform complex, step-by-step reasoning and generate clear, structured outputs for tasks such as summarization, question answering, and dialogue. These models' adaptability makes them able to be applied to specific domains, such as technical, scientific, or even legal texts. This suggests their versatility and effectiveness in real-world language processing.

3.3 Model Fine-Tuning

Fine-tuning is a method in ML that leverages existing pre-trained models and adapts them for specific tasks through additional training with new data. Instead of developing a model from scratch that requires extensive computational resources, fine-tune begins with a model that has already captured general patterns. The model's parameters are modified with the use task-specific data, which allows specialization for particular applications while the model keeps its general knowledge. This technique is less resource-intensive and usually performs better than starting from scratch.

3.3.1 Fine-tuning methods

There are several approaches for fine-tuning implementation. The differences in the approaches are about the computational cost, parameter scope and the type of task they have to perform. Some of the most commonly used fine-tuning approaches are described below.

Full fine-tuning is one of the methods, which updates all the parameters of the neural network. It follows a process comparable to pre-training but they differ because the full fine-tuning starts from an already pre-trained model and adapts it to a task-specific dataset and does not train it on a large, general corpus.

Parameter efficient fine-tuning (PEFT) is another fine-tuning method that can adapt large pre-trained models to several tasks. It does this but it a different way where it updates only a small subset of parameters and keeps the majority of the initial model unchanged.

There is no need for full retraining and this reduces the computational and memory demands. Techniques such as adapters and gradient checkpoints enhance efficiency.

Partial fine-tuning also reduces the demands beacuse it updates a subset of pre-trained parameters that are critical to downstream task. The rest of the parameters remain frozen. The standard approach is to fine-tune only the outer layers because the inner layers often encode general features,such as edges and textures in CNNs, that the model frequently leverages for related tasks. The more the new task aligns with the original, the less extensive the layer updates are needed.

As defined eralier fine-tuning adapts large pre-trained language models for specific tasks. This technique enhances their usability and matches them with user objectives. Full, partial, PEFT adjust all, some, or a limited number of parameters, respectively. Instruction tuning, reinforcement learning from human feedback (RLHF), and additive methods like adapters or LoRA further improve efficiency and task-specific performance while they keep the pre-trained knowledge.

3.4 Knowledge Distillation

Knowledge Distillation is a ML method, specifically a model compression technique, that seeks to transfer the knowledge from a larger and complex model (the teacher) to a smaller and efficient and lightweights model (the student). The student model learns from the original training data and the teacher's probabilistic outputs, which contain richer information than simple class labels. Through this method, smaller architectures can inherit the advanced capabilities of larger models, eliminating the need to train from the beginning and improving access to powerful modeling techniques [4][24]. The ultimate goal is to preserve as much of the teacher's predictive performance as possible while substantially reducing the computational and memory requirements of the model, thereby enabling efficient deployment. Teacher models are typically 2x to 10x larger than their student models. The compression ratio varies and depends on specific deployment context and the acceptable performance trade-offs.

As discussed in Section 2.2, the concept of transferring knowledge between models was first used in the model compression work by Buciluǎ et al. [15]. Hinton et al. [16] later popularized the modern framework of KD, which includes the use of probability distributions to

transfer richer information from the teacher to the student. These innovative studies established KD as a widely adopted method in deep learning, with applications that extend from simple compression [25].

3.4.1 Knowledge Distillation Methods

Knowledge distillation techniques are typically classified into three categories according to the type of knowledge that the teacher transfers to the student (Figure 3.4)[26][27] [28]. The knowledge might come from the output layer, from intermediate hidden representations, or from the structural relational patterns that exist between samples. These correspond to the three primary categories [29], response-based distillation [30] [31], feature-based distillation [32] and relation-based distillation [33], respectively.

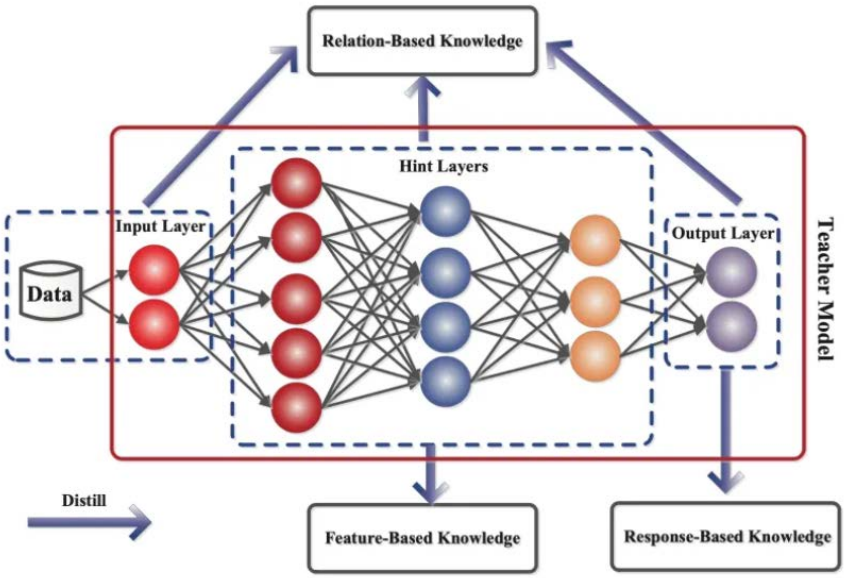

Figure 3.4: Different forms of Knowledge [4]

Beyond the knowledge type, distillation methods are also defined by their type of training, offline, online, and self-distillation, depending on whether the teacher's parameters remain fixed or are updated as the student is trained. Building on these foundational categories, numerous KD algorithms have been developed, each one adapted to specific architectural requirements and optimization objectives. Examples include adversarial distillation [34] [35], multi-teacher distillation, and cross-modal distillation, among others. Knowledge distillation has found widespread application across various domains, from image classification and object detection to language modeling and speech processing, especially where computational

constraints are limited.

Knowledge Types

1. **Response-based knowledge**

The most frequent type of knowledge distillation is the response-based knowledge, which is usually used in supervised learning. It involves transferring information from the teacher model's final output layer. In standard response-based KD approaches, the goal is to train the student model to match the output logits of the teacher model. A distillation loss is used to capture the difference between the logits of the student and the teacher model. By minimizing this loss during training, the student model gradually learns to replicate the teacher model's predictions.

In practice, response-based KD uses soft targets. Soft targets correspond to the probability distribution over all classes and they provide more nuanced information than a single ground-truth , which helps the student understand inter-class similarities. A softmax function is used to compute the soft targets along with a Temperature parameter (T). When temperature increases, the softmax outputs flatten. Consequently the probability distribution becomes smoother providing richer information that help the student learn more effectively.

The training process employs a combined loss that incorporates both distillation and standard cross-entropy loss to learn from both the teacher's predictions and the actual labeled data. This specific type of KD has several advantages. It is known for its simplicity and effectiveness since it requires only the teacher's predictions. Despite that response-based KD is the simplest approach to KD, it remains highly effective, especially when compressing large models into smaller ones in resource-constrained environments.

2. **Feature-based knowledge**

Feature-based knowledge emphasizes on the intermediate (hidden) layers of a neural network. In these hidden layers, the network processes information to discover and learn the underlying patterns and structures within the data. Feature-based distillation trains the student model to match these internal feature representations at corresponding layers. This approach transfers richer knowledge because it gives to the student

access to the teacher's reasoning process and observe the internal feature extraction. To achieve this, feature-based distillation utilizes a loss function that assesses the difference between the teacher's and student's feature activations.

3. **Relation-based knowledge**

Unlike the previous approaches, relation-based knowledge distillation concentrates on the dependencies between different layers or between feature maps. It provides a more comprehensive way to teach the student network to replicate the teacher's underlying reasoning. These relationships are often represented by similarity matrices, correlation coefficients, graphs or feature embeddings, which reflect the interactive patterns between network components rather than individual layer outputs.

Training type

1. **Offline distillation**

Offline distillation is the most widely used approach. In this type of distillation, the teacher has already been trained, and the student model learns under its supervision. The teacher model is trained independently, before being employed in the distillation process, and then provides supervision to train the smaller, student model and the weights of the teacher remain unchanged. The growth of deep learning has made many trained neural networks publicly available and offers flexibility in teacher model selection based on specific applications. Offline distillation has proven to be a reliable and practical technique for knowledge transfer.

2. **Online distillation**

Offline approaches generally depend on large, already trained teacher networks and assume access to these models. However, suitable pre-trained models are not always accessible for specific applications. Online distillation overcomes this challenge. It provides an alternative and allows for the training of teacher and student networks simultaneously during a single training phase. Parallel processing can significantly improve the computational efficiency of this training method.

3. **Self-distillation**

Self-distillation uses a single network architecture that serves dual roles as teacher and student model. This method enables knowledge flow within the same architecture from its deeper, more complex layers to capture patterns, to shallower layers learning basic features. Consequently, in these methods deeper network layers guide the learning of shallower layers, while this approach also allows knowledge from earlier training phases to inform later epochs. Self-distillation is a variant of online distillation and does not require any additional pre-trained models.

3.4.2 KD Algorithms and applications

Several distillation algorithms exist that they boost performance and tackle many challenges that may arise. These include adversarial, multi-teacher,which aggregates knowledge from multiple models [36] [37], cross-modal [38], graph-based [39] and attention-based [40], which preserve relational and contextual information, data-free and quantized, which reduce data and memory requirements.

There are many applications of knowledge distillation across various domains [41]. In computer vision, it helps to improve image classification, face recognition, and image segmentation [42]. In NLP [43], it improves tasks such as text classification, question answering, machine translation and document retrieval. Speech recognition systems also benefit from distillation [44]. These examples show KD's effectiveness and value across many tasks.

3.5 Text Representation and Vectorization

Contemporary machine learning algorithms for NLP require the transformation of textual data into numerical representations. These models are built to operate on quantitative data. This is a fundamental step that can be accomplished through various methods. Traditional techniques such as Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) rely on frequency-based representations, whereas more recent methods like Word2Vec and GloVe produce dense vector embeddings hat encode semantic meaning.

3.5.1 BoW

BoW is a commonly used approach to text vectorization, especially in NLP tasks. This method converts textual content, like documents' paragraphs or sentences, to numerical vec-

tors. Specifically it creates a vocabulary from all the unique words in the input, and then represents each piece of text as a vector based on this vocabulary. This technique considers text as an unordered set of words and ignores the grammar and word sequence, while it records only the frequency of each word in the text.

3.5.2 TF-IDF

TF-IDF is a text representation method which assesses the importance of a word to a given document in a larger set. This method uses two essential aspects to determine the literal significance of words. The TF term counts the number of times that a word is present in a document. It assigns higher weights to the terms that occur the most frequently. The IDF term measures how rare or common a word is throughout a collection of documents. It reduces the weights of terms that appear in several documents.

The final TF-IDF calculation creates a balance that emphasizes terms that appear frequently inside specific documents, but uncommon in the overall corpus. This is beneficial for information retrieval and document analysis tasks.

3.5.3 Word2Vec

Word2Vec, developed by Mikolov et al. [45] [46], produces vector representations of words that preserve semantic meaning, unlike traditional frequency-based methods. These vectors encode semantic information by analyzing the context in which words appear. Word2Vec develops these representations through training on large amounts of text data. The technique relies on the concept that words with related meanings should result in similar vector forms. Word2Vec is built on two architecture frameworks, Continuous Bag of Words (CBOW) and Skip-gram. Both methods consider individual words alongside a context window as they move through the text data. The main difference between these methods is their learning direction. CBOW predicts a target word from its surrounding context, while Skip-gram predicts the context words from a given target word. Both approaches work within a specific window that determines how many surrounding words to consider.

3.5.4 GloVe

GloVe is also a word embedding technique, introduced by Pennington et al. [47], that incorporates global statistical information with local context. It analyzes word co-occurrence patterns in huge corpora to capture semantic relationships and it produces dense vector representations. Unlike purely predictive models, GloVe uses matrix factorization of the co-occurrence matrix that is constructed. In the co-occurrence matrix each cell indicates how frequently a particular word pair is observed within the surrounding context. Once the co-occurrence matrix is created, GloVe adjusts the word vectors in order to make the dot product of any two vectors reflects the pointwise mutual information (PMI) [48] between the associated words. More precisely, PMI is computed from the probabilities of words that appear together in the corpus, and GloVe optimizes a weighted least squares objective that aligns the dot product of word vectors with the logarithm of these co-occurrence probabilities. This method allows GloVe embeddings to efficiently capture both syntactic and semantic meaning which is very important for several NLP tasks.

3.5.5 Sentence Embeddings (SentEmbeddings)

Sentence embeddings is an extension of word embeddings as they represent entire sentences as dense vectors in a high-dimensional space. This allows them to capture not only the meaning of individual words but also the semantic relationships expressed within a sentence as a whole. Sentence embeddings can be generated from pretrained models such as Sentence-BERT and they are particularly useful for NLP tasks like text classification, semantic similarity, and information retrieval.

3.6 Evaluation Metrics

Assessing a model's performance requires selection of appropriate metrics. They help us evaluate different models and identify best-performing model. In the context of knowledge distillation, metrics are used not only to evaluate the final accuracy of the student model but also to ensure that it effectively captures the behavior of the teacher model. In this study we utilize accuracy, precision, recall, and F1-score to provide a comprehensive evaluation.

The evaluation metrics utilize confusion matrix components, where TP (True Positives)

represent correctly predicted positive cases, TN (True Negatives) represent correctly predicted negative cases, FP (False Positives) represent incorrectly predicted positive cases, and FN (False Negatives) represent incorrectly predicted negative cases.

- **Accuracy**: is one of the most common metrics that is used in classification tasks. It serves as a comprehensive metric for the evaluation of the model's performance. Accuracy is defined as the number of correct predictions divided by the total number of predictions. However, it is not always the most reliable metric, especially with imbalanced datasets because a model can achieve a high value by simply predicting the majority class. It is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

- **Precision**: measures the ratio of true positive predictions to all positive predictions from the model. This measure becomes especially critical in situations where false positives have severe consequences. It is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

- **Recall**: evaluates the ability of a model to correctly detect true positive instances among all actual positives. This metric is particularly important when false negative (missed positive) errors are more costly than false positive errors, such as in disease detection where identifying all positive cases is critical. It is calculated as:

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

- **F1-Score**: combines precision and recall into a performance metric and represents their harmonic mean. It ensures that high performance in one metric cannot compensate for poor performance in the other because it penalizes extreme values. It is calculated as:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.4}$$

Chapter 4

Methodology

4.1 LLMs for Check-worthy Statement Detection

Open-source LLMs provide notable benefits across multiple dimensions, cost effectiveness, transparency, and ethical advantages. Based on these advantages, several major open-source LLMs were examined for the fact-checking worthiness classification task. The models evaluated include Llama-2-7b (7 billion parameters), GPT-2 (124 million parameters), Llama-3.2-1b (1.23 billion parameters), TinyLlama (1.1 billion parameters), and Microsoft Phi-2 (2.7 billion parameters). 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 (the 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. Initially, the binary classification task that is formulated as a text generation problem using structured prompt templates. Next, the models undergo parameter-efficient fine-tuning to adapt them for the task. 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.

4.1.1 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 standard format with three components: one instruction section that carries the system message and 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 that 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.

The prompt used in this project is based on the design introduced by Yufeng Li et al [13].

Prompt Check-worthy Statement Detection

Instruction:

Read the statement provided as below.

Your task is to evaluate whether the statement contains information or claims that are worthy to be verified through fact-checking. If the statement presents assertions, facts, or claims that would benefit from verification, respond with 'Yes'. If the statement is purely opinion-based, trivial, or doesn't contain any verifiable information or claims, respond with 'No'.

Input Sentence: `<input sentence>`

Response: `<Yes/No>`

4.1.2 Fine-tuning

The selected models went through supervised fine-tuning, which is a process where labeled training data is used to modify pre-trained causal language models to a specific task. A causal language model is an autoregressive model that predicts each token in a sequence based on all preceding tokens. For our task, although the output is just a single word (“Yes” or “No”), the model internally processes the input token by token, using the context of the entire sequence to generate the answer. As discussed earlier in Section 3.3, a fine-tuning approach leverages the model’s 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) [49] 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.

4.1.3 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.

4.2 Knowledge Distillation in LLMs for Check-worthy Statement Detection

While LLMs demonstrate exceptional performance, their computational complexity and resource demands limit their deployment in resource-constrained environments. KD addresses this problem through the transfer of knowledge from computationally expensive teacher models to lightweight student models. It enables efficient inference while preserving accuracy.

4.2.1 Knowledge Distillation Strategy

Knowledge distillation, as discussed in Section 3.4, is a model compression technique where 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 response-based knowledge distillation (Figure 4.1), which focuses on teacher's outputs, where the student model is "driven" to match the softer probability distributions (logits) of the teacher model, rather than based just on hard

ground truth labels. Our method employs an offline distillation strategy, indicating that the teacher model is fully pre-trained and fixed before the student model begins its training. For this work, 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. To guide the learning process, 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.

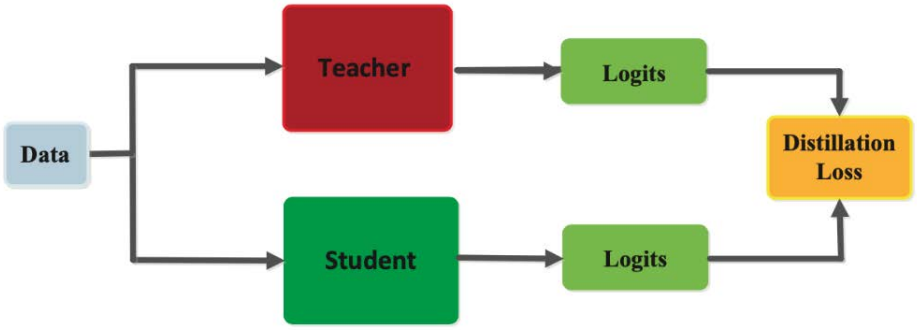

Figure 4.1: Response-based knowledge distillation [4]

This work follows a knowledge distillation process that explores the limitations coming from vocabulary alignment between teacher and student models. In this context, vocabulary alignment refers to the extent to which the teacher and student share the same tokens and token ids. The challenges arise when the teacher and student models differ in architecture. When the teacher and student use different tokenizers or vocabularies, their output probabilities cannot always be directly mapped, limiting the effectiveness of traditional knowledge distillation methods. Traditional methods typically assume substantial vocabulary overlap, restricting their application in cross-family model cases. The study investigates distillation performance in both same-family and different-family models, using two distinct loss functions for knowledge transfer.

This methodology allows to analyze how vocabulary alignment impacts distillation effectiveness. It also provides insights into how robust different loss formulations are across various model pairing scenarios. In particular, we compare two loss formulations approaches: (i) KL divergence, which in its standard form can only be used when teacher and student share the same vocabulary and (ii) the Universal Logit Distillation (ULD) loss (Figure 4.2) that is vocabulary independent and can be applied to different teacher/student pairs.

To enable the application of KL divergence in cross-family model pairs, we extend it

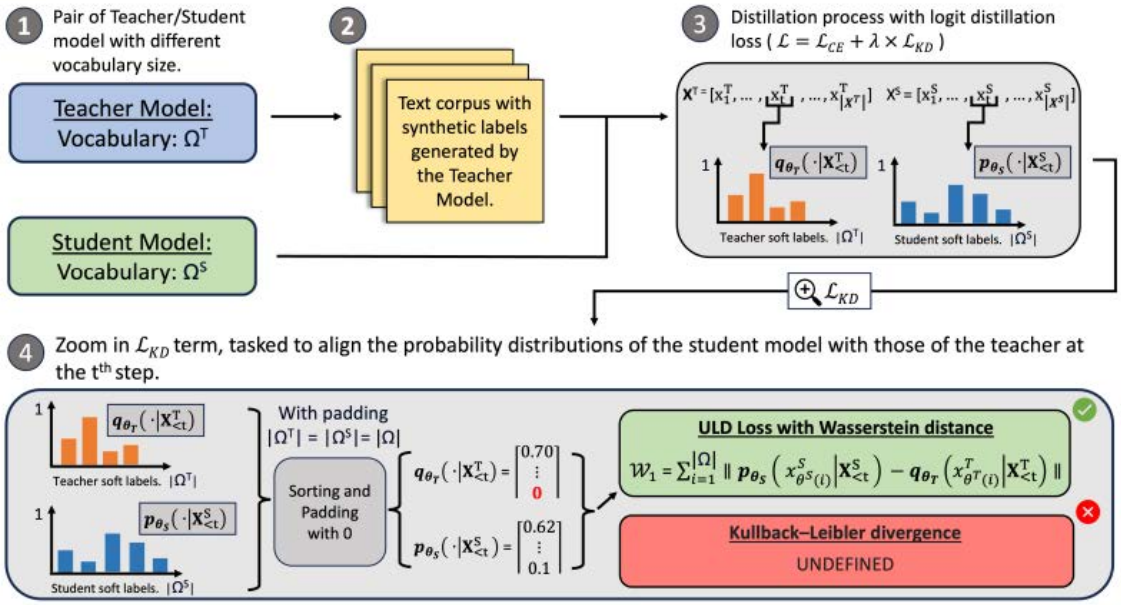

Figure 4.2: **Knowledge Distillation with ULD loss [5]**. In 4 KL divergence is undefined because the two distributions differ. To resolve this, we apply the ULD loss, which uses a closed-form Wasserstein distance.

with the creation of a vocabulary mapping between teacher and student based on token string overlap. Specifically, we identify tokens shared by both vocabularies and align their corresponding logits during training to enable direct comparison. Through this mapping the teacher's output logits are aligned to the student's vocabulary space, while unmatched tokens are masked out. The KL divergence loss is computed only for the aligned tokens, and distillation proceeds only if the overlap exceeds a minimum threshold. While this method extends KL-based distillation beyond same-family models, the alignment is inherently approximate and may suffer in quality when token overlap is limited. In contrast, ULD does not require token alignment, making it naturally suitable for cross-family models while being applicable to same-family models.

4.2.2 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 the model trained for the check-worthy detection task following the three-phase approach

described in Section 4.1. After full training, the model is stored. Having it stored, we load it to begin the training of the student of the Knowledge Distillation process. The teacher's objective is to generate "soft targets", i.e. 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 described in Section 4.1.2.

General Distillation Framework

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.

$$L_{total} = \alpha \cdot L_{CE} + (1 - \alpha) \cdot L_{KD} \tag{4.1}$$

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 0.3 - 0.5 being commonly used in the experiments.

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.

$$L_{CE} = -\sum y_{true} \cdot \log p_{student} \tag{4.2}$$

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. In this study, we explore and compare two different methods for this loss.

1. 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.

$$L_{KD} = KL(p_t||p_s) = \sum p_t \log \frac{p_t}{p_s} \tag{4.3}$$

$$p_t = softmax(\frac{z_t}{T}) \quad p_s = softmax(\frac{z_s}{T}) \tag{4.4}$$

where p_t and p_s 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.

2. **Universal Logit Distillation (ULD) Loss:** It is a recent innovative distillation approach proposed by Martins et al.

   [5] that uses optimal transport to allow knowledge transfer between different teacher and student models. In contrast to the classic KL divergence, ULD does not rely on

token-level alignment, it compares the overall shape of the predicted probability distributions, from teacher and student models, rather than the tokens themselves.

Specifically, it computes the average ℓ_1 distance between the sorted, temperature-scaled softmax distributions of the teacher and student models. By sorting the distributions in descending order of probability, the loss becomes independent of matching token identity, allowing for unaligned comparison between vocabularies. This enables knowledge to be transferred efficiently and flexibly from any generative teacher to any student model, regardless of architectural or tokenizer differences.

Although this method approximates some aspects of optimal transport, it does not compute the true Wasserstein distance. Instead, it uses a closed form solution for ULD loss, a simple and efficient proxy that aligns probabilities based on confidence rather than token identity. The loss is calculated as:

$$L_{KD} = \frac{1}{B}\sum_{i=1}^{B}\frac{1}{T_i}\sum_{t=1}^{T_i}|p_s^{(i,t)} - p_t^{(i,t)}| \tag{4.5}$$

where $p_s^{(i,t)}$ and $p_t^{(i,t)}$ denote the student's and teacher's temperature-scaled softmax distributions sorted in descending order. B is the batch size, and T_i number of answer tokens in sample i.

The temperature-scaled softmax distributions are defined as:

$$p_s^{(i,t)} = softmax(\frac{z_s^{(i,t)}}{T_s}) \qquad p_t^{(i,t)} = softmax(\frac{z_t^{(i,t)}}{T_t}) \tag{4.6}$$

where $z_s^{(i,t)} \in \mathbb{R}^V$ are student logits for token t in sample i and $z_t^{(i,t)} \in \mathbb{R}^V$ are teacher logits. V is the vocabulary size. T_s and T_t are the temperature values applied to the student and teacher logits that control the softness of the output distributions. Temperature acts as described previously, as the temperature increases, the distributions become softer, revealing more of the teacher's "dark knowledge" - informative cues that help guide the student during distillation.

Specifically, ULD sorts the logits of both instructor and student outputs in descending order before comparing the sorted probability distributions. Since the comparison is performed on sorted values it does not rely on exact token-to-token correspondence. This allows ULD to capture the similarity between the teacher and student's predicted

distributions even when their vocabularies differ or are not perfectly aligned. Therefore, ULD focuses on the relative importance of logits regardless of which token they belong to, making it robust to vocabulary mismatches that would may hurt token-level alignment methods like KL divergence.

Distillation Training Process

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, using either Kullback-Leibler divergence or Universal Logit Distillation, depending on the specific experimental setup. 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.

Chapter 5

Datasets & Preprocess

In this study we evaluate the KD method in LLMs for the task of check-worthy statement detection. To this end, we utilize two datasets provided by CheckThat! Lab. Specifically, the first dataset (CT24) [1] was released as part of the CheckThat! Lab at CLEF 2024, and the second (CT22) [2] originates from the 2022 edition of the same lab. Both datasets correspond to task 1, which focuses on estimating check-worthiness in statements. While the datasets include versions in multiple languages, we exclusively use the English version of the data in this study.

The first dataset, from the CLEF 2024 , consists of text-based instances drawn from diverse sources including debate and speech political transcriptions. This dataset includes the train, dev, dev-test and test partitions, with a total of 24,163 sentences from political transcriptions. Both the dev-test and test sets are used for evaluation only. The distributions of sentence lengths (measured in words) are shown in Figure 5.1. Across all dataset divisions, the majority of sentences are relatively short, with most including less than twenty words. Most sentences are brief, with medians between 10 and 14 words and a significant amount under 10 words, indicating that the statements convey only minimal detail. Furthermore, CT24 dataset presents a significant class imbalance, with the majority of instances labeled as “No” (non-check-worthy) and only a small proportion labeled as “Yes” (check-worthy), as illustrated in Table 5.1.

Following the data pruning methodology from Li et al. [13] as discussed in Section 2.1.2, we use the pruned training data for the remainder of our experiments to improve efficiency

[1] https://gitlab.com/checkthat_lab/clef2024-checkthat-lab/-/tree/main/task1/data

[2] https://gitlab.com/checkthat_lab/clef2022-checkthat-lab/clef2022-checkthat-lab/-/tree/main/task1/data

Dataset	Partition	Check-worthy	Non-check-worthy	Total
CLEF 2024	Train	5,413	17,086	22,499
	Dev	238	794	1,032
	Dev-Test	108	210	318
	Test	88	253	341

Table 5.1: CT24 Dataset Statistics

by reducing computational costs, training time and concentrating on high-quality, informative examples. This also contributes to ensure consistency and focus in the model evaluation process.

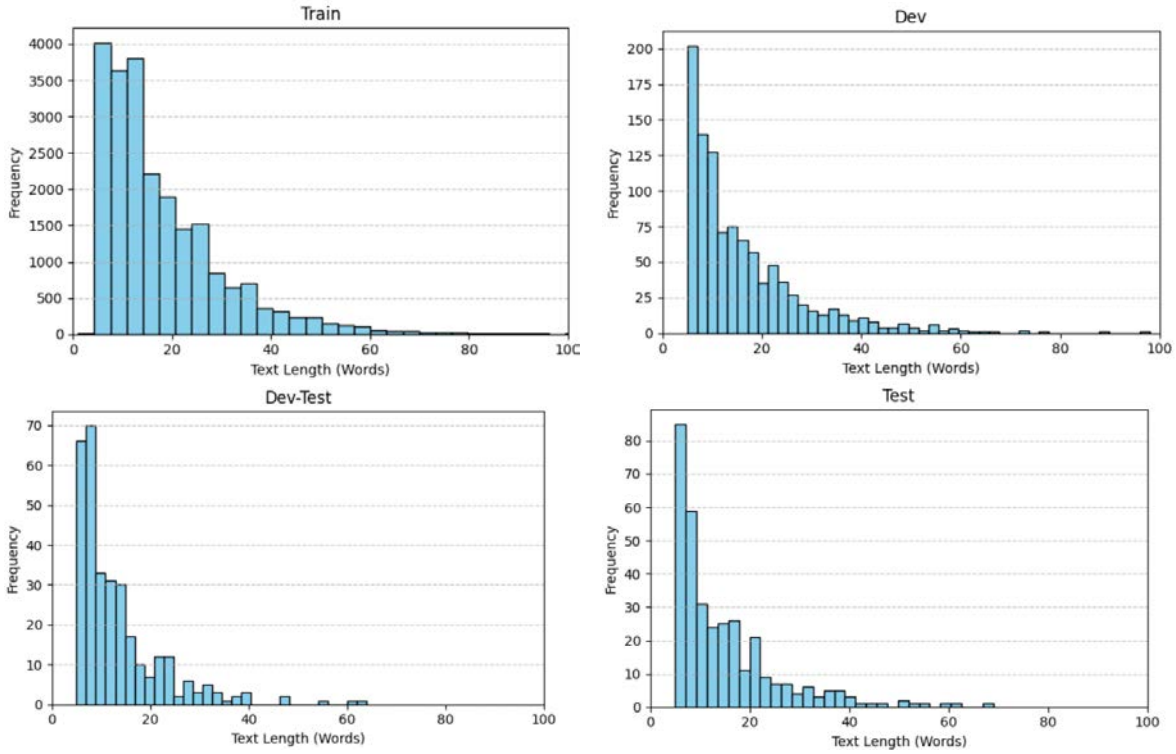

Figure 5.1: Text length Distribution of CT24

The second dataset, from the CLEF 2022, focuses exclusively on COVID-19-related tweets. CT22 is relatively smaller than the other dataset. The casual and conversational nature of tweets presents unique challenges for check-worthiness detection, as tweets often contain abbreviations, colloquialisms, and context-dependent references that differ significantly from formal political speeches or debate transcripts. As presented in Figure 5.2, the sentence lengths in this dataset are generally longer than those in the CT24 dataset, with a

higher proportion of tweets containing more than 20 words, approximately 30 to 40 words. This is expected, as tweets often include additional elements such as user mentions, urls, hashtags, and quoted content. These components contribute to increased token counts, even when the primary message is quite short. To handle the informal and noisy nature of tweet data, we tested models on both original and preprocessed text versions. The normalization process includes standard preprocessing steps such as lowercasing and the removal of user mentions, urls, and hashtags. These experiments allow us to evaluate the impact of preprocessing on model performance.

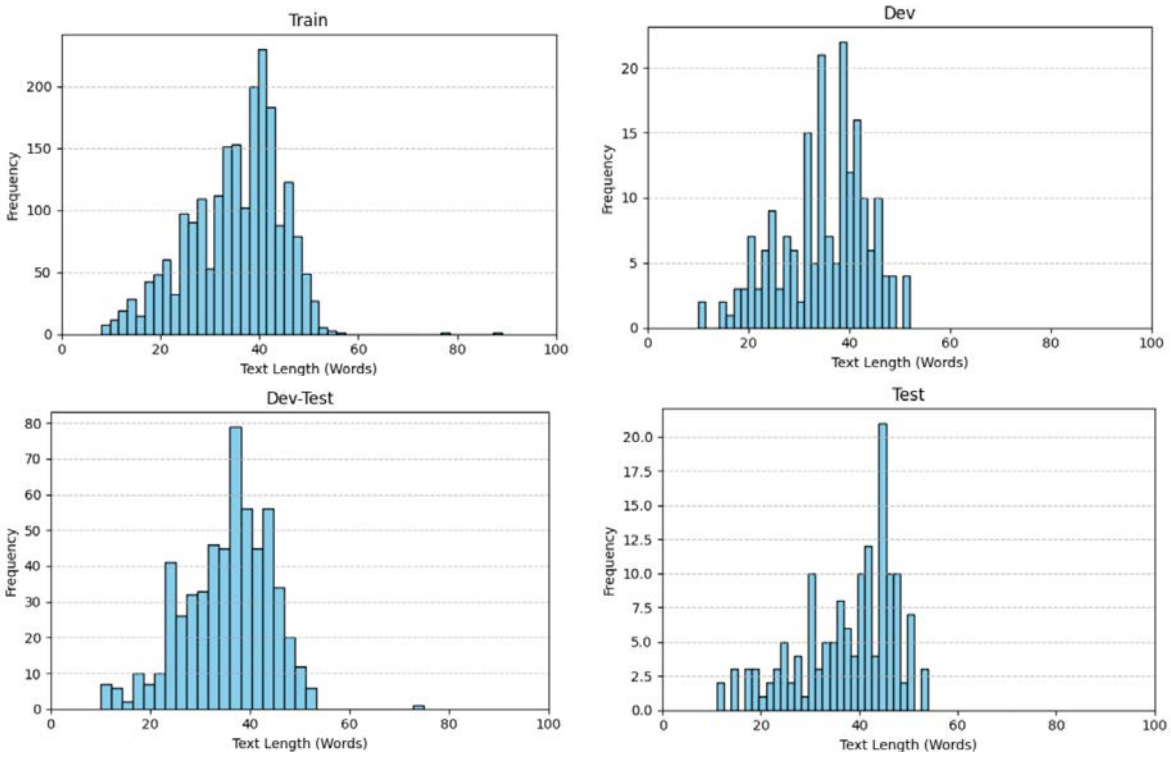

Figure 5.2: Text length Distribution of CT22

This dataset also exhibits an important class imbalance, as shown in Table 5.2. Due to the relatively small size of the train dataset (2,122 entries), the same pruning and class balancing strategy, as in the CT24 train dataset (22,499 entries), was not applied, as the size of the dataset did not result in high training time or computational demand. The implementation of such methods would have significantly reduced the number of available training examples, risked underfitting and limited the model's ability to generalize.

Dataset	Partition	Check-worthy	Non-check-worthy	Total
CLEF 2022	Train	447	1,675	2,122
	Dev	44	151	195
	Dev-Test	129	445	574
	Test	39	110	149

Table 5.2: CT22 Dataset Statistics

Initially, the same CNN-based balancing method used on the larger CT24 dataset was applied to this smaller dataset in preliminary experiments. However, this approach proved unsuitable for the current dataset's scale. The CNN led to a significant reduction in training size, retaining only a fraction of the original 2,122 samples, reducing them to just 999. It also results to significantly lower accuracy, likely due to underfitting by the limited number of retained samples. Given the size of this dataset, preserving as much training data as possible is a priority to maintain model capacity and generalization.

In an attempt to overcome the CNN method's limitations, we experiment with SMOTE for synthetic sample generation. However, this approach yielded no significant improvement over the original unbalanced dataset. Since the task focuses on detecting check-worthy statements, introducing synthetic data risks distorting the nuanced linguistic and contextual patterns essential for accurate detection. Therefore, preserving the integrity of the original data without artificial augmentation was deemed crucial to maintain model reliability and performance in this specific context. Our aim is to keep the authenticity and natural distribution of real-world samples, particularly in a sensitive domain like check-worthiness detection. Consequently, we avoided synthetic oversampling techniques such as SMOTE or text augmentation and keeping the dataset imbalanced. This is taken into account during evaluation and interpretation of results. This approach also allows for a more accurate comparison of method performance across different dataset types and distributions within the same task.

Chapter 6

Experimental Evaluation

The primary evaluation metric for task 1 (Check-worthiness Estimation) in CheckThat! 2024 is the F1 score on the positive class. However, to provide a more comprehensive assessment of model performance, we also report additional metrics including accuracy, precision, and recall averaged over the multiple runs. Since our goal also includes maintaining strong overall performance across all classes, we additionally consider accuracy as a secondary metric to assess how well the model performs on both positive and negative samples.

6.1 Experimental Setup

All experiments were conducted using Google Colab's cloud environment,using an Nvidia T4 GPU. The implementation was developed using PyTorch version 2.2.2 and executed in a Python environment configured with the accelerate library (version 0.29.1) to enable seamless GPU utilization. The software stack included essential packages from the Hugging Face , including `transformers` (v4.39.3), `datasets` (v2.18.0), `peft` (v0.10.0), and `bitsandbytes` (v0.42.0) for efficient model fine-tuning and inference. GPU acceleration was supported through CUDA 12, with packages such as `nvidia-cudnn-cu12`, `nvidia-cublas-cu12`, and related components.

6.1.1 Examined Models

To evaluate the performance and demonstrate the effectiveness of our proposed methodology, we first examine the baseline models provided by the CLEF CheckThat! Lab for task 1, as well as other methods that have been implemented for similar tasks such as check-worthy

or claim detection.

- The baselines provided by the CheckThat! Lab [1]for the task.[50]
  - **Random approach**: A random classifier that assigns class labels ("Yes" or "No") with equal probability to each test instance, establishing a naive performance baseline.
  - **N-gram**: A traditional machine learning approach using TF-IDF vectorization with unigram features (1-gram) and SVM classifier with linear kernel for text classification.
- **LSTM** [2] [51] [52]: A two-layer LSTM model initialized with pre-trained GloVe embeddings, followed by dropout and dense layers for binary classification.
- **Linear Experiments** [3] [53]: Multiple traditional machine learning models using various text representation strategies, which include:
  - **Embeddings**: Word2Vec (averaged per sentence)[45], TF-IDF, BoW, sentence embeddings (dense vectors from transformer models), and Pretrained GloVe[54].
  - **Models**: Logistic Regression, SVM, XGBoost and AdaBoost.
- **LLMs** [4]: Following the FactFinders methodology [55], we reproduce their LLM-based approach which demonstrated superior performance on this task.

  Tables 6.1 and 6.2 present the hyperparameters utilized to fine-tune the LLMs in the proposed method's experiments on the CT24 and CT22 datasets, respectively. The majority of hyper-parameters remain consistent across all tested LLMs, except for some cases, where the training batch size set to 1 because of memory constraints. To ensure reliable evaluation using established techniques, each fine-tuned model was run five times per input to account for generation variability, with majority voting determining the final predictions. The training process was repeated three times with averaged results for robust performance measurement.

[1]https://gitlab.com/checkthat_lab/clef2024-checkthat-lab/-/tree/main/task1?ref_type=heads
[2]https://github.com/jrmtnez/NLP-IR-UNED-at-CheckThat-2020/blob/master/T1En/nlpir01/task1.py
[3]https://github.com/SamiNenno/Claim-Detection/blob/main/Experiments/LinearExperiments.py
[4]https://github.com/isyufeng/FactFinders/blob/main/src/train_evaluate.py

Parameter	Value
Epochs	3
Training batch size	2
Gradient accumulation steps	2
Optimizer	Paged AdamW 32bit
Learning rate	2e-4
Weight decay	0.001
Maximum gradient norm	0.3
Warmup ratio	0.03
Temperature	0.03
LoRA Alpha	16
LoRA dropout	0.1
LoRA rank	64

Table 6.1: Hyper-parameters used for Fine-tuning on CT24

Parameter	Value
Epochs	3
Training batch size	2
Gradient accumulation steps	2
Optimizer	Paged AdamW 32bit
Learning rate	2e-4
Weight decay	0.001
Maximum gradient norm	0.3
Warmup ratio	0.03
Temperature	0.03
LoRA Alpha	8
LoRA dropout	0.1
LoRA rank	16

Table 6.2: Hyper-parameters used for Fine-tuning on CT22

- **Knowledge Distillation in LLMs** [5] [5]: Building on the LLM proposed method, we investigate knowledge distillation methods, examining both same-architecture and cross-architecture teacher-student configurations for check-worthy statement detection.

To maintain fair experimental conditions, all baseline models were either implemented directly from their original descriptions or adapted from public available source code to work with our datasets. While certain approaches were originally developed for similar tasks such as claim detection, we modified them appropriately to adapt for the check-worthiness detection task and evaluated their performance under our experimental conditions. Baselines were executed 10 times, whereas the proposed models were run 3 times due to resource constraints, following the common practice in prior work [13].

6.1.2 Preliminary Experiments and Method Exploration

The baseline models were initially examined and tested on both the CT22 and CT24 datasets. Subsequently, LLMs were explored for the task of identifying check-worthy statements. The results of these experiments are presented in the Section 6.2. As seen by the results, the LLMs outperformed all other evaluated approaches across both datasets. While achieving strong performance on CT24, poorer results were observed on CT22 dataset. The performance difference can be due to several challenges inherented by the CT22 dataset. The dataset consists primarily of content from Twitter, which creates unique challenges for check-worthiness detection due to informal language, acronyms, and context-dependent references. Furthermore, the dataset has a notable class imbalance, with significantly fewer check-worthy instances, which is likely contributing to the overall lower performance.

To address this challenge, we first looked beyond causal language models to explore transformer models specifically designed for sequence classification (AutoModelForSequenceClassification), as these architectures are inherently better suited to classification tasks than causal models. Several models, including meta-llama/Prompt-Guard-86M [6], Sami92/XLM-R-Large-ClaimDetection [7] and rashmikamath01/distillbert-fine-tuned-claimbuster3C [8] were evaluated. However the performance remained similar and comparable to that of the causal LLMs.

[5] https://github.com/Nicolas-BZRD/llm-recipes/blob/main/models/distillation_model.py
[6] https://huggingface.co/meta-llama/Prompt-Guard-86M
[7] https://huggingface.co/Sami92/XLM-R-Large-ClaimDetection
[8] https://huggingface.co/rashmikamath01/distillbert-fine-tuned-claimbuster3C

Different techniques to address class imbalance were also investigated. One such approach involves the replacement of the default cross-entropy loss in the SFTrainer with Focal loss, which is a method that reduces the influence of well-classified samples and emphasizes harder towards the misclassified instances. This shift in focus helps the model learn from the underrepresented class more effectively during fine-tuning. The focal loss implementation used parameters $\alpha = 0.25$ and $\gamma = 2.0$, following standard practices for addressing class imbalance in binary classification tasks. A second, more comprehensive strategy was subsequently implemented. This approach retained the focal loss mechanism while incorporating additional balancing techniques: automated class weight computation, which scales weights based on class frequency, and adjustment of decision criteria to better identify check-worthy content. This involved optimizing the confidence threshold,the minimum probability required, for the model to categorize information as check-worthy. Despite these modifications, class imbalance strategies did not result in significant performance improvements.

Given these results, simpler and more focused approaches were investigated in order to achieve better performance for the CT22 dataset. Specifically, BERT-based models were explored, which achieved better results than previous approaches. Appendix B contains results of the BERT-based models that yielded the best results among those tested.

In addition, as discussed in Chapter 5, models were evaluated on both the original (raw) and normalized tweet data. However, the raw data consistently yielded better performance. Therefore, all subsequent experiments for the knowledge distillation were conducted using the raw data. Results for models trained on normalized data, including baseline experiments, are also provided in Appendix A.

While BERT-based models have shown improvements on CT22 dataset, our BERT-based experiments focused on this particularly challenging dataset due to its known difficulty for large-scale generative models. BERT as an encoder-only model, is optimized for understanding rather than generation. For generative tasks, we focused on decoder-only LLMs, which have recently demonstrated state-of-the-art performance across a variety of tasks. Since LLMs, particularly autoregressive transformer models like Llama2-7b and GPT, already achieve strong performance on CT24 dataset, we decided not to test BERT on that dataset.

The core research interest lies in investigating knowledge distillation strategies, as it means to retain the capabilities of large decoder-only LLMs while reducing computational

requirements. Given that the LLMs examined, such as Llama, GPT, and others, demonstrated strong overall performance, we focused on exploring how their knowledge could be effectively transferred to smaller, more efficient models. Specifically, Llama2-7b was selected to serve as the teacher, in this study, due to its strong overall performance and reliability across the specific task. To evaluate the efficacy of this knowledge transfer, distillation was applied from Llama2-7b to smaller models, TinyLlama and Microsoft's Phi-2. This technique aims to test whether these lightweight student models could demonstrate improved performance or even approach the teacher's level of performance, while improving computational efficiency.

6.2 Baseline Models Performance

This section presents a comprehensive evaluation of baseline models, which establish performance benchmarks using traditional machine learning approaches, for check-worthy statement detection across the CT24 and CT22 datasets.

Tables 6.3, 6.4, 6.5, and 6.6 present the evaluation results of the baseline models for the task of check-worthy statement detection across the CT24 and CT22 datasets, including all evaluation partitions (Test and Dev-Test), respectively. For the CT24 dataset, the experimental results show that SVM combined with SentEmbeddings has the best overall performance, with F1 scores of 0.7326 (Dev-Test) and 0.5224 (Test), and accuracy values of 0.8554 and 0.8123, respectively. While LSTM has a higher F1 score on the Test set (0.5443), SVM with SentEmbeddings appears to be stronger because it demonstrates more consistent performance across both partitions and significantly outperforms LSTM on accuracy, which serves as our secondary evaluation metric. The results suggest that sentence-level embeddings outperform typical word-level representations.

Partition	Representation	Model	Accuracy	Precision	F1	Recall
Dev-Test		Random	0.5377	0.3818	0.4615	0.5833
		N-gram	0.8082	0.8406	0.6534	0.5370
		LSTM	0.8283	0.8602	0.6976	0.5917
	SentEmbeddings	LogReg	0.8491	0.9286	0.7303	**0.6019**
		SVM	**0.8554**	0.9327	**0.7326**	0.5833
		XGBoost	0.7956	0.8909	0.6012	0.4537
		AdaBoost	0.8050	0.8833	0.6310	0.4907
	BoW	LogReg	0.8019	0.8689	0.6272	0.4910
		SVM	0.7704	**0.9487**	0.5034	0.3426
		XGBoost	0.7610	0.8636	0.5000	0.3519
	GloVe	LogReg	0.7327	0.7674	0.4371	0.3056
		SVM	0.7327	0.8571	0.3529	0.2222
		XGBoost	0.7453	0.8140	0.4636	0.3241
		AdaBoost	0.7422	0.8250	0.4460	0.3056
	Word2Vec	LogReg	0.7830	0.8305	0.5868	0.4537
		SVM	0.7987	0.8926	0.6098	0.4630
		XGBoost	0.7862	0.8704	0.5803	0.4352
		AdaBoost	0.7987	0.9074	0.6049	0.4537

Table 6.3: Experimental results of examined models on CT24 Dev-test set

Partition	Representation	Model	Accuracy	Precision	F1	Recall
Test		Random	0.4575	0.2291	0.3071	**0.4660**
		N-gram	0.8094	0.7447	0.5185	0.3977
		LSTM	0.8047	0.6942	0.5443	0.4568
	SentEmbeddings	LogReg	0.7830	0.6522	0.4478	0.3409
		SVM	**0.8123**	0.7609	**0.5224**	0.3977
		XGBoost	0.7859	0.6471	0.4748	0.375
		AdaBoost	0.7801	0.6226	0.4681	0.375
	BoW	LogReg	0.8123	0.7727	0.5151	0.3836
		SVM	0.7947	**0.875**	0.375	0.2386
		XGBoost	0.7771	0.6875	0.3667	0.25
	GloVe	LogReg	0.7501	0.5366	0.3411	0.25
		SVM	0.7742	0.72	0.3186	0.2046
		XGBoost	0.7713	0.6389	0.3710	0.2614
		AdaBoost	0.7713	0.625	0.3906	0.2841
	Word2Vec	LogReg	0.7771	0.6364	0.4242	0.3182
		XGBoost	0.8006	0.7273	0.4849	0.3636
		SVM	0.8035	0.8	0.4553	0.3182
		AdaBoost	0.8065	0.775	0.4844	0.3523

Table 6.4: Experimental results of examined models on CT24 test set

For the CT22 dataset, LSTM achieves the best performance, with F1 scores of 0.498 (Dev-Test) and 0.492 (Test), and accuracy scores of 0.775 and 0.714 respectively. While BoW with LogReg has slightly higher accuracy on the Dev-Test set (0.798), LSTM is considered as more reliable bacause it has the highest F1 score, which serves as our primary evaluation metric, while maintaining competitive accuracy performance.

Partition	Representation	Model	Accuracy	Precision	F1	Recall
Dev-Test		Random	0.658	0.226	0.221	0.217
		N-gram	**0.805**	0.673	0.371	0.256
		LSTM	0.775	0.509	**0.498**	**0.521**
	SentEmbeddings	LogReg	0.793	0.609	0.320	0.217
		SVM	0.780	0.646	0.350	0.240
		XGBoost	0.756	0.438	0.358	0.302
		AdaBoost	0.760	0.446	0.349	0.287
	BoW	LogReg	0.798	0.578	0.453	0.370
		XGBoost	0.793	0.561	0.436	0.357
	GloVe	LogReg	0.803	0.682	0.347	0.233
		XGBoost	0.744	0.4	0.329	0.279
		AdaBoost	0.784	0.544	0.333	0.240
	Word2Vec	LogReg	0.803	**0.7**	0.331	0.217
		SVM	0.794	0.690	0.253	0.155
		XGBoost	0.775	0.5	0.416	0.357
		AdaBoost	0.793	0.562	0.431	0.349

Table 6.5: Experimental results of examined models on CT22 Dev-test set

Partition	Representation	Model	Accuracy	Precision	F1	Recall
Test		Random	0.604	0.237	0.234	0.231
		N-gram	**0.772**	**0.692**	0.346	0.2307
		LSTM	0.714	0.398	**0.492**	**0.544**
	SentEmbeddings	LogReg	0.732	0.474	0.310	0.231
		SVM	0.745	0.556	0.208	0.128
		XGBoost	0.698	0.417	0.4	0.385
		AdaBoost	0.718	0.460	0.447	0.436
	BoW	LogReg	0.725	0.475	0.481	0.487
		XGBoost	0.711	0.444	0.427	0.410
	GloVe	LogReg	0.732	0.467	0.260	0.180
		XGBoost	0.711	0.429	0.358	0.308
		AdaBoost	0.758	0.579	0.379	0.282
	Word2Vec	LogReg	0.738	0.5	0.339	0.256
		SVM	0.741	0.536	0.203	0.129
		XGBoost	0.712	0.454	0.482	0.513
		AdaBoost	0.732	0.486	0.460	0.436

Table 6.6: Experimental results of examined models on CT22 test set

6.3 LLMs Performance

In this section the evaluation of the LLMs' performance using pre-trained transformer architectures, is analyzed.

The evaluation of LLMs on both the CT24 and CT22 datasets reveals that Llama-2-7b consistently outperforms all the other examined models, including the baselines and the other LLMs, in the task of detecting check-worthy statements. It achieves the highest F1 scores across both datasets, with 0.8708 (Dev-Test) and 0.8249 (Test) on CT24, and 0.5693 (Dev-Test) and 0.6003 (Test) on CT22. These results clearly surpass those of the competing models, as shown below, in Tables 6.7 and 6.8. Although microsoft/phi-2 achieved relatively high accuracy on the CT22 dataset, its F1 scores were consistently lower, indicating weaker per-

formance when evaluated under the primary metric. In contrast, Llama-2-7b demonstrated strong and consistent performance across all evaluation settings. Because of its robust and consistent performance, Llama-2-7b was selected as the teacher model for the distillation experiments that follow.

Partition	Model	Accuracy	Precision	F1	Recall
Test	Llama-2-7b	**0.9091**	0.8202	**0.8249**	**0.8295**
	SVM with SE	0.8123	0.7609	0.5224	0.3977
	GPT-2	0.8622	0.7412	0.7283	0.7159
	Llama-3.2-1b	0.8536	0.6857	0.746	0.8182
	TinyLlama	0.8788	0.7979	0.7576	0.7348
	microsoft/phi2	0.8827	**0.8637**	0.7402	0.6477
Dev-Test	Llama-2-7b	**0.9151**	**0.9010**	**0.8708**	**0.8426**
	SVM with SE	0.8554	0.9327	0.7326	0.5833
	GPT-2	0.8742	0.8333	0.8095	0.7870
	Llama-3.2-1b	0.8891	0.8649	0.8603	0.8389
	TinyLlama	0.9059	0.8956	0.8469	0.8179
	microsoft/phi2	0.8920	0.8831	0.8130	0.6914

Table 6.7: Performance of LLMs on CT24 Test and Dev-Test

Partition	Model	Accuracy	Precision	F1	Recall
Test	Llama-2-7b	**0.8153**	**0.5571**	**0.6003**	**0.6393**
	LSTM	0.714	0.398	0.492	0.544
	GPT-2	0.6465	0.2589	0.2176	0.1880
	Llama-3.2-1b	0.6309	0.3462	0.3956	0.4615
	TinyLlama	0.6309	0.2647	0.2466	0.2308
	microsoft/phi2	06935	0.4383	0.5090	0.6068
Dev-Test	Llama-2-7b	0.7944	0.5652	**0.5693**	**0.5735**
	LSTM	0.775	0.509	0.498	0.521
	GPT-2	0.7718	0.4720	0.1997	0.1266
	Llama-3.2-1b	0.7753	0.5	0.4267	0.3721
	TinyLlama	0.7619	0.4112	0.2053	0.1370
	microsoft/phi2	**0.8142**	**0.6382**	0.4921	0.4005

Table 6.8: Performance of LLMs on CT22 Test and Dev-Test

6.4 Knowledge Distillation Performance

This section presents a comprehensive evaluation of knowledge distillation experiments, which transfer knowledge from larger teacher models to smaller student models.

Tables 6.9 and 6.10 show the performance of the knowledge distillation method across teacher–student model pairs, trained both within the same model family and across different families, using a combined loss function that includes cross-entropy along with either KL divergence or ULD loss. KL divergence is commonly used when teacher and student share the tokenizer. In contrast, ULD loss uses the Wasserstein distance to align logit distributions across models with different token vocabularies because it works by comparing the overall distribution shapes rather than token-by-token probabilities as KL divergence does. The KL Divergence Loss in this design relies on an overlap between the teacher and student vocabularies to operate effectively. As a result, when applied to teacher-student pairs from distinct model families with limited vocabulary overlap, the distillation results may be less reliable.

After evaluation of different hyperparameter configurations, for all distillation experiments, we use temperature scaling with $T = 3.0$ and alpha parameter to $\alpha = 0.3$ for both

KL divergence and ULD losses. While higher temperatures are often used to extract more soft target information from the teacher, in our case, increasing T beyond 3 led to a drop in accuracy. Therefore, $T = 3$ was chosen as it provided consistently strong results across experiments.

Partition	Model	Accuracy	Precision	F1	Recall
Test	Llama-2-7b (Teacher)	0.9091	0.8202	0.8249	0.8295
	TinyLlama	0.8788	0.7979	0.7576	0.7348
	TinyLlama-KD (KL)	**0.8856**	**0.7816**	**0.7771**	**0.7727**
	TinyLlama-KD (ULD)	**0.8798**	**0.8**	**0.7592**	0.6705
	microsoft/phi2	0.8827	0.8637	0.7402	0.6477
	microsoft/phi2-KD (KL)	**0.8974**	**0.8046**	**0.8**	**0.7955**
	microsoft/phi2-KD (ULD)	**0.8856**	**0.8657**	**0.7484**	**0.6951**
Dev-Test	Llama-2-7b (Teacher)	0.9151	0.9010	0.8708	0.8426
	TinyLlama	0.9059	0.8956	0.8469	0.8179
	TinyLlama-KD (KL)	**0.8973**	**0.8215**	**0.8574**	**0.9046**
	TinyLlama-KD (ULD)	**0.9088**	**0.9438**	**0.8586**	0.787
	microsoft/phi2	0.8920	0.8831	0.8130	0.6914
	microsoft/phi2-KD (KL)	**0.9308**	**0.9479**	**0.8922**	**0.8426**
	microsoft/phi2-KD (ULD)	**0.8973**	**0.9631**	**0.8275**	**0.7253**

Table 6.9: Performance of proposed method on CT24 Test and Dev-Test

The results in Table 6.9 of the CT24 dataset demonstrate that knowledge distillation consistently improves the performance of the student models across both the Test and Dev-Test partitions. Both KL and ULD loss functions lead to improvements in F1 scores compared to the simple TinyLlama and microsoft/phi2 models, indicating successful transfer of knowledge from the teacher.

In the CT24 dataset, the KL-based distillation outperforms ULD in the TinyLlama model across the test partition. This is expected, as the teacher (Llama 2) and student belong to the same model family and thus share a similar vocabulary space, allowing KL divergence to operate effectively. However, in the Dev-Test partition, ULD achieves better results than KL

in most metrics, except for recall.

In the case of the microsoft/phi2 models, while the KL variant achieves higher scores in some metrics and especially in the Dev-Test partition, the performance is less reliable as discussed earlier. Instead, ULD which is designed for teacher-student pairs from different model families is considered more appropriate and stable choice in this design. ULD achieves better results in every metric compared to the simple phi2 model across both partitions. Notably, in some cases, ULD even enables the student to outperform the teacher in certain metrics, further supporting its effectiveness.

Partition	Model	Accuracy	Precision	F1	Recall
Test	Llama-2-7b (Teacher)	0.8153	0.5571	0.6003	0.6393
	TinyLlama	0.6309	0.2647	0.2466	0.2308
	TinyLlama-KD (KL)	**0.6913**	**0.4054**	**0.3947**	**0.3846**
	TinyLlama-KD (ULD)	**0.6577**	**0.4091**	**0.5143**	**0.6923**
	microsoft/phi2	06935	0.4383	0.5090	0.6068
	microsoft/phi2-KD (KL)	0.651	0.4177	**0.6078**	**0.8462**
	microsoft/phi2-KD (ULD)	**0.7383**	**0.5**	**0.5618**	**0.6113**
Dev-Test	Llama-2-7b (Teacher)	0.7944	0.5652	0.5693	0.5735
	TinyLlama	0.7619	0.4112	0.2053	0.1370
	TinyLlama-KD (KL)	**0.7927**	**0.5893**	**0.3568**	**0.2558**
	TinyLlama-KD (ULD)	**0.7753**	**0.5**	**0.5474**	**0.6047**
	microsoft/phi2	0.8142	0.6382	0.4921	0.4005
	microsoft/phi2-KD (KL)	0.8086	0.5584	**0.5593**	**0.6667**
	microsoft/phi2-KD (ULD)	**0.8206**	**0.6444**	**0.5297**	**0.4496**

Table 6.10: Performance of proposed method on CT22 Test and Dev-Test

In the CT22 dataset (Table 6.10), knowledge distillation results in significant improvements in both TinyLlama and microsoft/phi2 models across most metrics, particularly in terms of the F1 score, which is our primary evaluation metric. Despite the overall lower performance of all models compared to CT24 dataset, which is most likely related to the greater

complexity and different properties of the CT22 data, both KL and ULD loss functions lead to F1 score improvements over the baseline student models.

In contrast to the CT24 distillation for TinyLlama, in CT22 dataset the ULD clearly outperforms the KL, especially in F1 and recall. In the case of microsoft/phi2, although the KL variant achieves slightly higher F1 and recall scores on certain partitions compared to the ULD, it also causes a drop in accuracy and precision, highlighting instability in cross-family distillation. In contrast, ULD offers a more balanced and stable performance across all metrics, making it a more suitable and robust choice for heterogeneous teacher-student configurations like microsoft/phi2 and Llama-2.

Chapter 7

Conclusion & Future Work

In this thesis we explored Knowledge Distillation methods in LLMs for the task of check-worthiness detection. We built on earlier results that showed Llama2-7b model performs well. In order to handle the model's heavy computational demands, we focused on transferring the capabilities of this high-performing teacher model to smaller, more efficient student architectures. The experimental evaluation indicated that our distillation framework successfully handles the issues that arise with big language models. This work defined a combined loss function which incorporates cross-entropy and KL divergence or ULD for the knowledge transfer process. The results showed that ULD, which allows cross-model knowledge transfer without shared tokenizers, overall outperforms KL and transfers the teacher model's abilities in more efficient student architectures. This provides flexibility in model selection. The study tested the robustness of the methods on both a political transcription dataset and a COVID-19 tweet dataset, where the distilled architectures showed reliable detection capabilities across many text structures and domains. These findings prove that knowledge distillation helps the deployment of advanced check-worthiness detection systems in situations with limited computational resources and encourages broader implementation of automated fact-checking approaches.

Future research in Knowledge Distillation could focus on developing more sophisticated distillation losses that capture different aspects of the teacher model's knowledge. Furthermore, bias prevention approaches during knowledge distillation also require exploration, as the knowledge transfer process transfers any inherent biases present in the teacher's decision-making. The integration of knowledge distillation with Retrieval-Augmented Generation (RAG) systems can offer compact models that can leverage real-time information retrieval

to enhance their decision-making capabilities. This integration will help to address the limitation of static knowledge distillation, where student models are constrained to knowledge captured during training. RAG-enhanced distillation models could use up-to-date information sources during inference, allowing for real-time fact-checking against evolving knowledge bases. Moreover, RAG systems could retrieve semantically similar previously verified claims or content from training datasets, allowing the model to leverage historical verification patterns or examine comparable instances and their associated labels to inform current decisions, thereby improving consistency and accuracy via similarity-based reasoning.

One powerful extension to this work could be a multi-teacher ensemble distillation, where a student model can learn from a "committee" of multiple teachers. This approach will use domain-specialized teacher models to train a unified student through dynamic knowledge aggregation mechanisms. This addresses the limitation of single-teacher distillation by incorporating diverse expertise, potentially improving generalization across varied claim types.

Finally, a further direction is network-aware distillation, which incorporates social media propagation patterns into the knowledge transfer process. In this approach, check-worthiness assessment could benefit from analyzing not only textual content but also how information spreads and reproduces across social networks. A distilled model could learn to integrate propagation velocity, source diversity, and network dissemination patterns when determining whether claims warrant verification. This may enhance check-worthiness detection through social context-aware decision-making.

Bibliography

[1] Mohammed Abu-Jamous and Ashraf Yunis Maghari. UAV Detection Model Using Histogram of Oriented Gradients and SVM. Master of information technology, Islamic University of Gaza, December 2019. DOI:10.13140/RG.2.2.26164.99207.

[2] GeeksforGeeks. Implementation of XGBoost (eXtreme Gradient Boosting). https://www.geeksforgeeks.org/machine-learning/implementation-of-xgboost-extreme-gradient-boosting/, 2025. Accessed: 2025-09-08.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. arXiv 2023, arXiv: 1706.03762.

[4] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6):1789–1819, March 2021. http://dx.doi.org/10.1007/s11263-021-01453-z.

[5] Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. Towards Cross-Tokenizer Distillation: the Universal Logit Distillation Loss for LLMs. arXiv 2025, arXiv: 2402.12030.

[6] Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. Toward Automated Fact-Checking: Detecting Check-worthy Factual Claims by ClaimBuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1803–1812, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Md Zia Ullah. An ML Model for Predicting Information Check-Worthiness using a Variety of Features (short paper). In Josiane Mothe and Thi Bich Ngoc Hoang, editors,

Proceedings of the Workshop on Machine Learning for Trend and Weak Signal Detection in Social Networks and Social Media, TWSDetection 2020, Toulouse, France, February 27-28, 2020, volume 2606 of *CEUR Workshop Proceedings*, pages 56–61. CEUR-WS.org, 2020.

[8] Norbert Fuhr, Anastasia Giachanou, Gregory Grefenstette, Iryna Gurevych, Andreas Hanselowski, Kalervo Jarvelin, Rosie Jones, YiquN Liu, Josiane Mothe, Wolfgang Nejdl, Isabella Peters, and Benno Stein. An Information Nutritional Label for Online Documents. *SIGIR Forum*, 51(3):46–66, February 2018.

[9] Cédric Lespagnol, Josiane Mothe, and Md Zia Ullah. Information Nutritional Label and Word Embedding to Estimate Information Check-Worthiness. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 941–944, New York, NY, USA, 2019. Association for Computing Machinery.

[10] Ria Jha, Ena Motwani, Nivedita Singhal, and Rishabh Kaushal. Towards automated check-worthy sentence detection using Gated Recurrent Unit. *Neural Comput. Appl.*, 35(15):11337–11357, February 2023.

[11] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, and Christina Lioma. Neural weakly supervised fact check-worthiness detection with contrastive sampling-based ranking loss. volume 2380. ceur workshop proceedings, 2019. 20th Working Notes of CLEF Conference and Labs of the Evaluation Forum, CLEF 2019 ; Conference date: 09-09-2019 Through 12-09-2019.

[12] Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Giovanni Da San Martino, and Pepa Atanasova. Overview of the CLEF-2019 CheckThat!: Automatic Identification and Verification of Claims, 2021.

[13] Yufeng Li, Rrubaa Panchendrarajan, and Arkaitz Zubiaga. FactFinders at Checkthat! 2024: Refining Check-worthy Statement Detection with LLMs through data Pruning. arXiv 2024, arXiv: 2406.18297.

[14] Yavuz Selim Kartal and Mucahid Kutlu. Re-Think Before You Share: A Comprehensive Study on Prioritizing Check-Worthy Claims. *IEEE Transactions on Computational Social Systems*, 10(1):362–375, 2023.

[15] Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a Neural Network. arXiv 2015, arXiv: 1503.02531.

[17] Longzheng Wang, Xiaohan Xu, Lei Zhang, Jiarui Lu, Yongxiu Xu, Hongbo Xu, Minghao Tang, and Chuang Zhang. MMIDR: Teaching Large Language Model to Interpret Multimodal Misinformation via Knowledge Distillation. arXiv 2024, arXiv: 2403.14171.

[18] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge Distillation of Large Language Models. arXiv 2024, arXiv: 2306.08543.

[19] Jiang Zhang, Qiong Wu, Yiming Xu, Cheng Cao, Zheng Du, and Konstantinos Psounis. Efficient Toxic Content Detection by Bootstrapping and Distilling Large Language Models. arXiv 2023, arXiv: 2312.08303.

[20] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Mach. Learn.*, 20(3):273–297, September 1995. https://doi.org/10.1023/A:1022627411411.

[21] D. R. Cox. The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[22] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, page 148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[23] Sepp Hochreiter and JΓOrgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. https://doi.org/10.1162/neco.1997.9.8.1735.

[24] Lin Wang and Kuk-Jin Yoon. Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):3048–3068, 2022.

[25] Fahad Sarfraz, Elahe Arani, and Bahram Zonooz. Knowledge Distillation Beyond Model Compression. arXiv 2020, arXiv: 2007.01922.

[26] Yugank .Aman. Knowledge Distillation for LLMs: Techniques and Applications. https://medium.com/@yugank.aman/knowledge-distillation-for-llms-techniques-and-applications-e23a17093adf, February 2025. Accessed: 2025-09-08.

[27] Haziqa Sajid. Knowledge Distillation: Transferring Knowledge from Large, Computationally Expensive LLMs to Smaller ones Without Sacrificing Validity. https://zilliz.com/learn/knowledge-distillation-from-large-language-models-deep-dive, 2024. Accessed: 2025-09-08.

[28] Dave Bergmann. What is Knowledge Distillation? https://www.ibm.com/think/topics/knowledge-distillation, April 2024. Accessed: 2025-09-08.

[29] Chuanguang Yang, Xinqiang Yu, Zhulin An, and Yongjun Xu. Categories of Response-Based, Feature-Based, and Relation-Based Knowledge Distillation. arXiv 2023, arXiv: 2306.10687.

[30] Liangchen Song, Xuan Gong, Helong Zhou, Jiajie Chen, Qian Zhang, David Doermann, and Junsong Yuan. Exploring the Knowledge Transferred by Response-Based Teacher-Student Distillation. In *Proceedings of the 31st ACM International Conference on Multimedia*, MM '23, page 2704–2713, New York, NY, USA, 2023. Association for Computing Machinery.

[31] Ying Jin, Jiaqi Wang, and Dahua Lin. Multi-Level Logit Distillation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24276–24285, 2023.

[32] Shuo Wang, Chihang Wang, Jia Gao, Zhen Qi, Hongye Zheng, and Xiaoxuan Liao. Feature Alignment-Based Knowledge Distillation for Efficient Compression of Large Language Models. arXiv 2024, arXiv: 2412.19449.

[33] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational Knowledge Distillation. arXiv 2019, arXiv: 1904.05068.

[34] Ilyas Oulkadda and Julien Perez. AKD : Adversarial Knowledge Distillation For Large Language Models Alignment on Coding tasks. arXiv 2025, arXiv: 2505.06267.

[35] Xin Lian, Zhiqiu Huang, and Chao Wang. AKD: Using Adversarial Knowledge Distillation to Achieve Black-box Attacks. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2023.

[36] Konrad Zuchniak. Multi-teacher knowledge distillation as an effective method for compressing ensembles of neural networks. arXiv 2023, arXiv: 2302.07215.

[37] Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. Multilingual Neural Machine Translation with Knowledge Distillation. arXiv 2019, arXiv: 1902.10461.

[38] Dino Ienco and Cassio Fraga Dantas. DisCoM-KD: Cross-Modal Knowledge Distillation via Disentanglement Representation and Adversarial Learning. arXiv 2024, arXiv: 2408.07080.

[39] Jing Liu, Tongya Zheng, Guanzheng Zhang, and Qinfen Hao. Graph-based Knowledge Distillation: A survey and experimental evaluation. arXiv 2023, arXiv: 2302.14643.

[40] Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. ALP-KD: Attention-Based Layer Projection for Knowledge Distillation. arXiv 2020, arXiv: 2012.14022.

[41] Abdolmaged Alkhulaifi, Fahad Alsahli, and Irfan Ahmad. Knowledge distillation in deep learning and its applications. *PeerJ Computer Science*, 7:e474, April 2021.

[42] Gousia Habib, Tausifa jan Saleem, Sheikh Musa Kaleem, Tufail Rouf, and Brejesh Lall. A Comprehensive Review of Knowledge Distillation in Computer Vision. arXiv 2024, arXiv: 2404.00936.

[43] Amir M. Mansourian, Rozhan Ahmadi, Masoud Ghafouri, Amir Mohammad Babaei, Elaheh Badali Golezani, Zeynab Yasamani Ghamchi, Vida Ramezanian, Alireza Taherian, Kimia Dinashi, Amirali Miri, and Shohreh Kasaei. A Comprehensive Survey on Knowledge Distillation. arXiv 2025, arXiv: 2503.12067.

[44] Yan Gao, Titouan Parcollet, and Nicholas Lane. Distilling Knowledge from Ensembles of Acoustic Models for Joint CTC-Attention End-to-End Speech Recognition. arXiv 2021, arXiv: 2005.09310.

[45] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. arXiv 2013, arXiv: 1301.3781.

[46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. arXiv 2013, arXiv: 1310.4546.

[47] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[48] Kenneth Ward Church and Patrick Hanks. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[49] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. arXiv 2021, arXiv: 2106.09685.

[50] Alberto Barrón-Cedeño, Firoj Alam, Tanmoy Chakraborty, Tamer Elsayed, Preslav Nakov, Piotr Przybyła, Julia Maria Struß, Fatima Haouari, Maram Hasanain, Federico Ruggeri, Xingyi Song, and Reem Suwaileh. The CLEF-2024 CheckThat! lab: Check-worthiness, Subjectivity, Persuasion, Roles, Authorities, andAdversarial Robustness. In Nazli Goharian, Nicola Tonellotto, Yulan He, Aldo Lipani, Graham McDonald, Craig Macdonald, and Iadh Ounis, editors, *Advances in Information Retrieval*, pages 449–458, Cham, 2024. Springer Nature Switzerland.

[51] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, and Christina Lioma. Neural weakly supervised fact check-worthiness detection with contrastive sampling-based ranking loss. volume 2380. ceur workshop proceedings, 2019. 20th Working Notes of CLEF Conference and Labs of the Evaluation Forum, CLEF 2019 ; Conference date: 09-09-2019 Through 12-09-2019.

[52] Juan Martinez-Rico, Lourdes Araujo, and Juan Martinez-Romo. NLP&IR@UNED at CheckThat! 2020: A Preliminary Approach for Check-Worthiness and Claim Retrieval

Tasks using Neural Networks and Graphs. In *Working Notes of CLEF 2020 – Conference and Labs of the Evaluation Forum*, volume 2696. CEUR-WS.org, September 2020.

[53] Sami Nenno. Propositional claim detection: a task and dataset for the classification of claims to truth. *J. Comput. Soc. Sci.*, 7(2):1727–1752, 2024. https://doi.org/10.1007/s42001-024-00289-0.

[54] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[55] Yufeng Li, Rrubaa Panchendrarajan, and Arkaitz Zubiaga. FactFinders at CheckThat! 2024: Refining Check-worthy Statement Detection with LLMs through Data Pruning. arXiv 2024, arXiv: 2406.18297.

[56] Petar Ivanov, Ivan Koychev, Momchil Hardalov, and Preslav Nakov. Detecting Check-Worthy Claims in Political Debates, Speeches, and Interviews Using Audio Data. arXiv 2024, arXiv: 2306.05535.

[57] Xinyi Zhou and Reza Zafarani. A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities. *ACM Comput. Surv.*, 53(5), September 2020.

[58] Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. A Survey on Automated Fact-Checking. *Transactions of the Association for Computational Linguistics*, 10:178–206, 2022.

[59] Pepa Atanasova, Alberto Barron-Cedeno, Tamer Elsayed, Reem Suwaileh, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. Overview of the CLEF-2018 CheckThat! Lab on Automatic Identification and Verification of Political Claims. Task 1: Check-Worthiness. arXiv 2018, arXiv: 1808.05542.

[60] Sanjai Balajee Kannan Giridharan, Sanjjit Sounderrajan, B. Bharathi, and Nilu R. Salim. SSN-NLP at CheckThat! 2024: Assessing the Check-Worthiness of Tweets and Debate Excerpts Using Traditional Machine Learning and Transformer Models. In *Working*

Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum, pages 483–488. CEUR-WS.org, 2024.

APPENDICES

Appendix A

Models performance on normalized CT22 set

Partition	Model	Accuracy	Precision	F1	Recall
Test	Llama-2-7b	0.8054	0.6667	0.5797	0.5128
	Llama-3.2-1b	0.5928	0.3537	0.4621	0.6667
	GPT-2	0.6395	0.2819	0.1594	0.1111
	TinyLlama	0.6398	0.3692	0.4404	0.5470
	microsoft/phi2	0.6846	0.4310	0.5155	0.6410
Dev-Test	Llama-2-7b	0.8066	0.7045	0.3584	0.2403
	Llama-3.2-1b	0.7631	0.4791	0.5416	0.6227
	GPT-2	0.7300	0.1579	0.1107	0.1003
	TinyLlama	0.7497	0.4273	0.3745	0.3333
	microsoft/phi2	0.7927	0.5463	0.4979	0.4574

Table A.1: Performance of LLMs on normalized CT22 Test and Dev-Test

Partition	Representation	Model	Accuracy	Precision	F1	Recall
Dev-Test		Random	0.657	0.226	0.221	0.217
		N-gram	0.801	0.660	0.352	0.240
		LSTM	0.786	0.546	0.422	0.364
	SentEmbeddings	LogReg	0.803	0.643	0.389	0.279
		SVM	0.810	0.679	0.411	0.295
		XGBoost	0.742	0.422	0.408	0.395
		AdaBoost	0.807	0.602	0.488	0.411
	BoW	LogReg	0.788	0.543	0.419	0.341
		XGBoost	0.789	0.549	0.427	0.349
		AdaBoost	0.779	0.545	0.159	0.093
	GloVe	LogReg	0.807	0.688	0.373	0.256
		XGBoost	0.772	0.489	0.396	0.333
		AdaBoost	0.768	0.471	0.332	0.256
	Word2Vec	LogReg	0.806	0.737	0.335	0.217
		SVM	0.798	0.686	0.293	0.186
		XGBoost	0.791	0.562	0.406	0.318
		AdaBoost	0.767	0.469	0.362	0.294

Table A.2: Experimental results of examined models on normalized CT22 Dev-test set

Partition	Representation	Model	Accuracy	Precision	F1	Recall
Test		Random	0.604	0.237	0.24	0.2301
		N-gram	0.758	0.58	0.357	0.256
		LSTM	0.680	0.405	0.413	0.446
	SentEmbeddings	LogReg	0.752	0.545	0.393	0.307
		SVM	0.711	0.409	0.295	0.237
		XGBoost	0.718	0.459	0.447	0.436
		AdaBoost	0.758	0.543	0.512	0.487
	BoW	LogReg	0.732	0.487	0.474	0.462
		XGBoost	0.725	0.475	0.481	0.487
	GloVe	LogReg	0.725	0.45	0.305	0.231
		XGBoost	0.617	0.286	0.296	0.308
		AdaBoost	0.725	0.45	0.305	0.231
	Word2Vec	LogReg	0.725	0.375	0.128	0.077
		SVM	0.732	0.445	0.167	0.103
		XGBoost	0.638	0.333	0.357	0.385
		AdaBoost	0.705	0.436	0.436	0.437

Table A.3: Experimental results of examined models on normalized CT22 Test set

Appendix B

Performance of BERT-based Models on CT22

Partition	Model	Base Architecture	Accuracy	Precision	F1	Recall
Test	Nithiwat/mdeberta-v3-base_claim-detection	mDeBERTa-v3-base	0.6230	0.4013	0.5634	0.9201
	xlm-roberta-base	XLM-RoBERTa-base	0.6811	0.4310	0.5471	0.7403
	roberta-base	RoBERTa	0.6810	0.4301	0.5471	0.7420
	bert-base-uncased	BERT-base	0.5110	0.3404	0.4911	0.8465
	WhispAI/ClaimBuster-DeBRTaV2	DeBERTa-v2	0.6702	0.4432	0.5810	0.8702
	eskimo7/distilbert-tweets	DistilBERT	0.5347	0.4132	0.5401	0.7718
Dev-Test	Nithiwat/mdeberta-v3-base_claim-detection	mDeBERTa-v3-base	0.8420	0.6330	0.6604	0.6821
	xlm-roberta-base	XLM-RoBERTa-base	0.7721	0.4905	0.5910	0.7420
	roberta-base	RoBERTa	0.7704	0.4890	0.5910	0.7430
	bert-base-uncased	BERT-base	0.7503	0.4712	0.6005	0.8210
	WhispAI/ClaimBuster-DeBRTaV2	DeBERTa-v2	0.8104	0.5602	0.6150	0.6817
	eskimo7/distilbert-tweets	DistilBERT	0.7713	0.5021	0.6231	0.8221

Table B.1: Performance BERT-Based Models on CT22 Dataset