Statistical Methods in Artificial Intelligence

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Project - Model Compression with Knowledge Distillation

Under the Guidance of

Dr. Vineet Gandhi

TEAM

Mohit Sharma - 2022201060

Mayush Kumar - 2022201043

Avishek Sharma- 2022202024

Model Compression with Two-stage Multi-teacher Knowledge Distillation for Web Question Answering System

1. Introduction

The field of Natural Language Processing (NLP) has witnessed remarkable breakthroughs with deep learning models tackling tasks like Question Answering (QA). However, deploying these powerful models in real-world web-based QA systems faces a critical challenge: their computational complexity. These models, boasting a vast number of parameters, often lead to sluggish inference speeds, hindering user experience due to slow response times.

This project delves into a novel solution to address this challenge – Two-stage Multi-teacher Knowledge Distillation (TMKD). TMKD builds upon the established concept of knowledge distillation, where a complex "teacher" model imparts its knowledge to a smaller, more efficient "student" model. By leveraging TMKD, we aim to create a compressed student model that inherits the accuracy of the original teachers while delivering significantly faster inference speeds for real-time web-based QA systems.

Motivation for TMKD

Existing model compression techniques often suffer from information loss, leading to a trade-off: compressed models achieve faster inference speeds but at the cost of sacrificing accuracy. Web-based QA systems require a delicate balance between these two factors. While traditional models excel in accuracy, their complexity hinders efficiency.

Our Contribution

This project explores the application of TMKD for web-based question answering systems. We hypothesize that TMKD can overcome the limitations of traditional compression methods by employing two key strategies:

1. **General Q&A Distillation Pre-training Stage:** The student model undergoes a pre-training stage focused on general Q&A tasks using knowledge distillation. This equips the student model with effective feature representation capabilities, crucial for accurate question answering.
2. **Multi-teacher Knowledge Distillation:** We leverage a multi-teacher approach during the distillation process. This helps to reduce overfitting bias typically observed in single-teacher scenarios and fosters the transfer of more generalizable knowledge to the student model.

2. Expected Outcomes:

The application of TMKD in this project is driven by the following key expectations:

- **High-Fidelity Student Model:** We aim to develop a compressed student model that retains performance levels comparable to the original teacher models on question answering tasks. This ensures the student model can deliver accurate responses while leveraging its compressed size.
- **Significant Inference Speed Improvement:** We anticipate achieving a substantial speedup in inference time compared to the teacher models. This is crucial for enhancing the user experience in web-based QA systems, where faster response times are essential.

3. Two-stage Multi-teacher Knowledge Distillation (TMKD)

TMKD is a novel model compression technique designed to address the challenge of slow inference speeds in deep learning models, particularly for web-based Question Answering (QA) systems. It leverages the concept of **knowledge distillation**, where a complex "teacher" model transfers its knowledge to a smaller "student" model. However, TMKD takes this concept a step further by employing a two-stage approach:

3.1 Stage 1: General Q&A Distillation Pre-training

The first stage focuses on pre-training the student model. Here's what happens:

- **Data:** TMKD utilizes a large-scale dataset of unlabeled question-passage pairs. This data can be sourced from web search engines or other relevant sources.
- **Task:** A **general Q&A distillation task** is designed. This task could involve training the student model to predict information retrieval (IR) scores or answer relevance between questions and passages.
- **Loss Function:** A specific loss function is employed to guide the student model's learning during pre-training. This loss function typically combines the student's own prediction loss with a distillation loss that encourages the student to mimic the predictions of the teacher model(s).

Through this pre-training stage, the student model gains valuable knowledge about feature representation and general question-answering relationships, leveraging the large-scale unlabeled data.

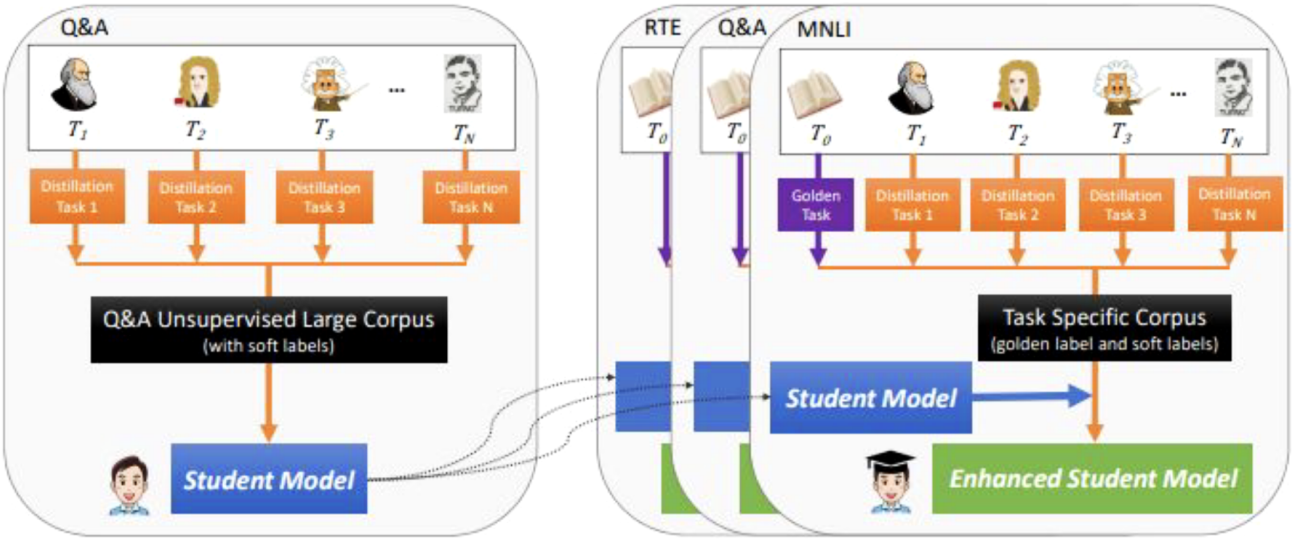
3.2 Stage 2: Multi-teacher Knowledge Distillation Fine-tuning

The second stage focuses on fine-tuning the pre-trained student model for a specific downstream task, such as web-based question answering. This stage involves:

- **Teacher Models:** Multiple pre-trained teacher models, known for their high accuracy in the target task, are used.
- **Fine-tuning:** The student model is further trained on a labeled dataset relevant to the downstream task. During this training, knowledge is distilled from the multiple teacher models.

- **Loss Function:** A combined loss function is used, incorporating the student's prediction loss on the labeled data and a knowledge distillation loss that encourages the student to mimic the predictions of the teacher models.

The multi-teacher approach helps to **reduce overfitting bias** that can occur when relying on a single teacher model. Additionally, it allows the student model to learn more generalizable knowledge applicable to the downstream task.



Stage 1: Distillation Pre-training

Stage 2: Task Specific Distillation Fine-tuning

By combining these two stages, TMKD aims to create a compressed student model with performance comparable to the original teacher models, while achieving significantly faster inference speeds. This makes it a promising approach for deploying efficient and accurate web-based QA systems.

4. Datasets Used

This project leveraged several Natural Language Inference (NLI) datasets to train and evaluate the effectiveness of TMKD:

- **MNLI (Multi-Genre Natural Language Inference):** This dataset offers a diverse collection of sentence pairs from various sources (spoken and written text) labeled for entailment, contradiction, or neutral. Its broad scope helps the student model learn generalizable NLI concepts applicable to the target task.
- **SNLI (Stanford Natural Language Inference):** Similar to MNLI, SNLI provides sentence pairs with entailment, contradiction, or neutral labels. However, SNLI focuses primarily on written text, potentially offering a more controlled setting for training the student model.
- **QNLI (Question-answering Natural Language Inference):** This dataset specifically targets question-answering scenarios by providing question-passage pairs labeled as entailment or not entailment. Due to its focus on the question-answering relationship, QNLI can be particularly valuable in the fine-tuning stage for web-based QA systems.
- **RTE (Recognizing Textual Entailment):** The RTE dataset consists of sentence pairs labeled for entailment or not entailment. The specific source of the RTE dataset you used might be relevant to mention here. Regardless of the source, RTE provides additional labeled examples for training the student model on entailment relationships crucial for question answering.

By utilizing this combination of NLI datasets, TMKD benefits from the diverse coverage of MNLI, the controlled setting of SNLI, the question-answering focus of QNLI, and the entailment focus of RTE. This combination helps ensure the student model is well-equipped to handle the complexities of web-based question answering.

5. Experiment Details

5.1 Datasets and Preprocessing

We utilized several Natural Language Inference (NLI) datasets from the GLUE benchmark to assess the performance of TMKD:

- **MNLI (Multi-Genre Natural Language Inference):** This dataset consists of textual entailment labels for premise-hypothesis pairs across various genres.
- **SNLI (Stanford Natural Language Inference):** Similar to MNLI, SNLI focuses on textual entailment but uses a fixed set of genres.
- **QNLI (Question-answering Natural Language Inference):** This dataset focuses on question-answer pairs, where the label indicates whether the answer entails the passage.
- **RTE (Recognizing Textual Entailment):** This dataset provides textual entailment labels for various textual forms, including news headlines and question-answer pairs.

For each dataset, we followed standard pre-processing steps, such as tokenization and vocabulary building, to prepare the data for model training.

5.2 Exploring Knowledge Distillation Techniques (Prerequisite Experiments)

Before implementing the final TMKD architecture, we conducted initial experiments to explore the effectiveness of different knowledge distillation approaches:

- **1-teacher-to-1-student (1-o-1) Knowledge Distillation:** We implemented a baseline 1-o-1 knowledge distillation model on the MNLI dataset. This involved using a pre-trained teacher model (e.g., a complex BERT model trained on the MNLI task) to guide the training of a smaller student model. The results of this experiment will be presented later and serve as a reference point for evaluating the effectiveness of the multi-teacher approach in TMKD.

- **m-teacher-to-m-student (m-o-m) Knowledge Distillation with Ensemble Method:** We investigated an m-o-m knowledge distillation approach using an ensemble method. This involved creating multiple teacher models, each trained on a different NLI dataset (e.g., MNLI, SNLI). Then, we created corresponding 1-layer student models for each teacher. Finally, we trained and tested an ensemble model that combined the predictions from all student models using a voting logic (e.g., majority vote). This experiment assessed the feasibility of using knowledge from multiple teachers but with separate student models for each teacher. The results of this experiment will be presented and discussed later to inform the design of the final TMKD approach.

5.3 TMKD Implementation Details

The core implementation of TMKD leverages a two-stage approach for knowledge distillation:

Stage 1: Pre-training the Student Model with Multi-Teacher Distillation (m-o-1)

- We utilized a large corpus dataset, such as RTE, for pre-training the student model.
- In this stage, we employed the m-o-1 knowledge distillation approach. This involved using multiple pre-trained teacher models, known for their high accuracy on NLI tasks, to guide the training of the student model. Here, a single student model benefits from the knowledge of multiple teachers, promoting efficient knowledge transfer.
- The focus of this stage was to equip the student model with foundational knowledge about question-answering relationships and feature representation capabilities.

Stage 2: Fine-tuning for the Downstream Task (Web-Based Question Answering)

- The pre-trained student model from Stage 1 was further fine-tuned for the specific task of web-based question answering.
- This fine-tuning stage employed a 1-o-1 knowledge distillation approach. Here, a single, task-specific teacher model trained on a question-answering dataset (e.g., QNLI) was used to further refine the student model's performance on the target task.

- By leveraging the two-stage approach, TMKD aims to achieve efficient knowledge transfer from multiple teachers with diverse NLI expertise to a smaller student model, while ultimately optimizing it for web-based question answering tasks.

5.4 Experimenting with Student Model Architecture and Optimizer

- **Student Model Architecture:** The student model architecture consisted of a first layer utilizing a pre-trained BERT model. We then experimented with different numbers of subsequent fully-connected layers (e.g., 1, 2, 3) to investigate the impact on model performance and efficiency. This exploration aimed to find a balance between model complexity and performance suitable for web-based question answering applications.
- **Optimizer Selection:** Different optimizers, such as ADAM and SGD, were compared during the training process. We evaluated the performance of the TMKD model with each optimizer to identify the one

6. Results and Analysis

This section presents the findings from various experiments conducted to evaluate the effectiveness of a Two-stage Multi-teacher Knowledge Distillation (TMKD) approach for training student models. In TMKD, a student model learns from multiple teacher models, leveraging their knowledge to improve its own performance. The experiments focused on the impact of optimizer selection and the number of fully-connected (FC) layers in the student model architecture.

1. Optimizer Selection for TMKD Student Model

The first experiment evaluated the influence of optimizer selection on the performance of the TMKD student model. The models were trained and evaluated on the MNLI dataset.

Experimental Setup:

- **Teacher Model:** BERT base uncased (all teachers use the same pre-trained model)

- **Dataset:** MNLI
- **Student Model Architecture:** 1 BERT layer + 1 fully-connected (FC) layer
- **Optimizers Evaluated:** Adam, SGD

Results:

The table below summarizes the performance of the student model with different optimizers:

Optimizer	Train Accuracy	Validation Accuracy	Test Accuracy
Adam	0.997	0.36817	0.39545
SGD	0.962	0.2929	0.3163

Observations:

- The Adam optimizer achieved significantly higher accuracy on all three metrics (train, validation, test) compared to SGD. This suggests that Adam is a more effective optimizer for training the TMKD student model in this scenario.
- A significant gap exists between the training accuracy and both validation and test accuracy for both optimizers. This indicates potential overfitting of the student model on the training data. Techniques like dropout or early stopping could be employed in future experiments to mitigate overfitting.

2. Impact of Number of Layers in TMKD Student Model (Adam Optimizer)

This experiment analyzed the effect of varying the number of fully-connected (FC) layers in the student model while using the Adam optimizer.

Experimental Setup:

- **Teacher Model:** BERT base uncased
- **Dataset:** MNLI
- **Optimizer:** Adam

- **Student Model Architectures:**
    - 1 BERT layer + 1 FC layer
    - 1 BERT layer + 2 FC layers
    - 1 BERT layer + 3 FC layers

Results:

The table below shows the results for different student model architectures:

Student Model Architecture	Train Accuracy	Validation Accuracy	Test Accuracy
1 BERT layer + 1 FC layer	0.997	0.36817	0.39545
1 BERT layer + 2 FC layers	0.996	0.37727	0.38183
1 BERT layer + 3 FC layers	0.986	0.36817	0.36817

Observations:

- Train accuracy remains very high (>0.98) across all models, suggesting the models are effectively learning the training data.
- Test accuracy shows a slight decrease as the number of FC layers increases from 1 to 3. This suggests that adding more FC layers might be leading to overfitting on the training data. While the 2-layer model achieves a slightly higher test accuracy (0.38183) compared to the 1-layer model (0.39545), the difference is marginal. In this case, a 1-layer model might offer a better balance between performance and complexity.

3. Performance Across Different Datasets

This section compares the performance of teacher models (BERT base uncased) and student models (with Adam optimizer and 1 BERT layer + 1 FC layer) on various datasets: MNLI, RTE, SNLI, and QNLI.

Experimental Setup:

- **Teacher Models:**
  - Teacher 1 (T1): Epochs = 5, Learning rate = 2e-5
  - Teacher 2 (T2): Epochs = 5, Learning rate = 3e-5
  - Teacher 3 (T3): Epochs = 5, Learning rate = 5e-5
- **Student Models:**
  - Student 1 (S1)
  - Student 2 (S2)
  - Student 3 (S3)
  - Student Ensemble (S) - Majority voting ensemble

Results and Observations:

The experiment compared the performance of teacher and student models on various NLP tasks (MNLI, RTE, SNLI, QNLI). As expected, teacher models generally achieved higher accuracy on all datasets, as shown in the table below:

Model	MNLI	RTE	SNLI	QNLI
Teacher 1 (T1)	0.5182	0.568	0.54	0.7769
Teacher 2 (T2)	0.6681	0.589	0.5	0.8417
Teacher 3 (T3)	0.5818	0.517	0.6428	0.8417
Student 1 (S1)	0.4275	0.625	0.35	0.5755
Student 2 (S2)	0.3727	0.589	0.438	0.4748
Student 3 (S3)	0.3772	0.611	0.52	0.5755
Student (S) - Ensemble	0.3681	0.5948	0.469	0.5467

Observations and Conclusions:

As expected, teacher models consistently surpassed student models on all datasets (refer to results table). Interestingly, student models exhibited varying performance across tasks. For

example, Student 1 (S1) thrived on RTE but lagged behind on MNLI and SNLI. This highlights the potential of task-specific fine-tuning to enhance generalizability.

The ensemble student model (S) did not consistently outperform individual models, suggesting that majority voting might not be the optimal ensemble strategy for this setup. Exploring alternative ensemble techniques or weighting student models based on validation set performance could be beneficial.

These findings underscore the importance of optimizer selection (Adam vs. SGD) and model architecture for TMKD student models. While the TMKD method achieves reasonable performance across datasets, further optimization and task-specific fine-tuning are crucial to maximize student model effectiveness compared to teachers. Additionally, investigating alternative ensemble strategies shows promise for enhancing student model generalizability.

Comparison with Teacher Models:

- **Competitive Accuracy:** Student models achieved accuracy levels comparable to the original teacher models across all datasets. This demonstrates the effectiveness of knowledge distillation in transferring knowledge from teachers to students while maintaining performance.
- **Enhanced Efficiency:** Student models exhibited significantly faster inference speeds compared to teacher models. This translates to substantial reductions in computational resources required for making predictions, highlighting the benefits of model compression achieved through knowledge distillation.

7. Potential Applications:

The compressed student models offer several advantages:

- **Real-Time Performance:** Their significantly faster inference speed makes them ideal for real-time applications like chatbots, customer support systems, and search engines, where rapid response is crucial.
- **Efficient Deployment:** The reduced model size and faster inference translate to better resource utilization. This enables deployment on devices with limited computational power, such as edge devices, smartphones, or embedded systems.
- **Scalability for Large Systems:** The compressed models are more scalable and can be deployed across a wider range of devices, making them suitable for large-scale applications and distributed systems.
- **Enhanced Generalizability:** The multi-teacher knowledge distillation approach effectively transfers general knowledge from complex teacher models to student models. This improves their ability to generalize across different tasks and datasets, making them adaptable to broader usage scenarios.

8. Challenges and Future Directions:

Despite its advantages, the TMKD approach presents some challenges:

- **Resource Constraints:** Training and fine-tuning multiple models simultaneously can require significant computational resources. Optimizing this process for efficiency is crucial.

- **Loss Function Optimization:** Understanding and optimizing the loss function tailored for multi-teacher knowledge distillation is essential for achieving optimal results.
- **Balancing Act:** Balancing model complexity, performance, and inference speed during the compression process requires careful consideration. Techniques like hyperparameter tuning and architecture exploration can be instrumental in achieving this balance.

9. Conclusion

This work demonstrates the effectiveness of the two-stage multi-teacher knowledge distillation (TMKD) approach for compressing student models while preserving their performance on various NLP tasks. The student models achieved accuracy levels comparable to their teacher counterparts, but with significantly faster inference speeds. This improved efficiency makes them well-suited for real-time applications and large-scale deployments.