HSLU Hochschule Luzern

Hochschule Luzern
Computer Science Department

DSPRO2: Optimizing Machine Learning Models for Commonsense Question Answering

Lecturers / Coaches
Dr. Umberto Michelucci
Dr. Seraina Glaus
Dr. Curdin Derungs
Fabian Widmer

Candidates
Aditi Sharma
Luke Kronenberg
Daniel Betschart

Year 2025

Contents

Abstract

Commonsense question answering (CQA) remains a significant frontier in artificial intelligence, requiring models to perform human-like reasoning on everyday knowledge. This project, DSPRO2, undertakes a comprehensive exploration and optimization of various machine learning architectures for the CommonsenseQA benchmark dataset. We systematically evaluated three distinct categories of models: (1) traditional NLP approaches using FastText embeddings with feedforward and recurrent neural network (RNN/LSTM) classifiers; (2) transformer-based models, specifically ALBERT ('albert-base-v2'), both by fine-tuning pre-trained weights and training from random initialization; and (3) prompting techniques with the Large Language Model (LLM) 'google/flan-t5-large', including zero-shot, few-shot, zero-shot Chain-of-Thought (CoT), and Retrieval Augmented Generation (RAG) with live web search. Our findings reveal a clear performance hierarchy: traditional models achieved accuracies near random chance (≈20%), underscoring their limitations. Fine-tuned ALBERT showed substantial improvement (54.38% accuracy), while ALBERT trained from scratch performed poorly, highlighting the critical impact of pre-training. The Flan-T5-large model demonstrated the highest efficacy, with zero-shot and few-shot prompting achieving impressive accuracies of 85.83% and 85.09% respectively. RAG also showed promise (78.00% on a subset), while CoT prompting resulted in a performance decrease (62.98%). This study emphasizes the pivotal role of large-scale pre-training and advanced architectures (Transformers, LLMs) in tackling commonsense reasoning, and provides insights into the varying effectiveness of different prompting strategies.

1 Introduction

This report details the project "DSPRO2," which focuses on the optimization of machine learning models for the task of commonsense question answering. The primary objective, as outlined in the project's scope, is to explore various architectural approaches and methodologies to enhance model performance on questions that require a significant degree of commonsense reasoning. Commonsense question answering (CQA) is a challenging subfield of Natural Language Processing (NLP) that aims to develop systems capable of answering questions that humans typically find trivial, yet require a foundational understanding of the world.

The project utilizes the CommonsenseQA dataset, a widely recognized benchmark available on Hugging Face. This dataset consists of multiple-choice questions, where each question is accompanied by five possible answer choices, and the model's task is to select the most plausible answer. Successfully tackling this dataset necessitates models that can go beyond simple pattern matching and engage in deeper reasoning processes.

Throughout this project, several distinct architectural paradigms were implemented and evaluated. These include:

1. Traditional NLP approaches leveraging pre-trained word embeddings (FastText) in conjunction with:
    - A 2-layer feedforward neural network classifier.
    - A Recurrent Neural Network (RNN) architecture (LSTM/GRU) followed by a classifier.
2. Transformer-based models, specifically focusing on the ALBERT (‘albert-base-v2‘) architecture. Experiments were conducted by:
    - Fine-tuning the pre-trained ALBERT model.
    - Training an ALBERT model from random initialization to assess the impact of pre-training.
3. Prompting techniques with Large Language Models (LLMs), using ‘google/flan-t5-large‘ to explore:
    - Zero-shot prompting.
    - Few-shot prompting.
    - Zero-shot Chain-of-Thought (CoT) prompting.
    - A Retrieval Augmented Generation (RAG) pipeline incorporating live web search via SerpAPI.

The central research question guiding this project can be formulated as: **How do different neural network architectures, ranging from traditional word embedding-based models to advanced transformer and large language models, compare in performance and efficiency for commonsense question answering, and what are the key factors influencing their optimization?**

This report will detail the dataset, preprocessing steps, methodologies employed for each architectural approach, experimental setup including hyperparameter tuning and model validation strategies, and a comprehensive analysis of the results obtained. All experiments and metrics were tracked using Weights & Biases (W&B) to ensure reproducibility and facilitate comparison. The findings aim to provide insights into effective strategies for optimizing models for commonsense reasoning tasks.

2 Related Works (Literature Review)

Commonsense question answering (CQA) represents a significant challenge in artificial intelligence, requiring models to possess a level of understanding akin to human intuition. This literature review explores key advancements and methodologies relevant to this project, spanning word embeddings, recurrent and transformer architectures, fine-tuning strategies, and prompting techniques for large language models.

2.1 Commonsense Question Answering and Datasets

The CommonsenseQA dataset (Talmor et al., 2019) is a prominent benchmark specifically designed to test commonsense reasoning. It presents questions that require knowledge about everyday situations, object properties, and social interactions, making it distinct from factoid-based QA datasets. Other similar datasets, such as PIQA (Physical Interaction QA) (Bisk et al., 2020) and Social IQA (Sap et al., 2019), also aim to evaluate different facets of commonsense knowledge. The primary challenge in CQA lies in equipping models with the implicit knowledge humans use effortlessly. Early approaches often struggled due to the lack of explicit commonsense knowledge bases integrated into neural models. Research by Liu et al. (2020) on "Commonsense Reasoning for Goal-Directed Dialogue" further highlighted the need for models to incorporate diverse forms of commonsense for practical applications.

2.2 Word Embeddings for NLP

Word embeddings revolutionized NLP by representing words as dense vectors in a continuous space, capturing semantic relationships. FastText (Bojanowski et al., 2017), used in this project's initial experiments, enhances traditional word vector models like Word2Vec (Mikolov et al., 2013a) by incorporating subword information. This allows FastText to generate embeddings for out-of-vocabulary words and often better capture morphological nuances. While effective for many tasks, averaging word embeddings for sentences or longer texts, as done in one of our approaches, can lead to a loss of sequential information and nuanced meaning, which is often critical for complex reasoning tasks (Mikolov et al., 2013b).

2.3 Recurrent Neural Networks (RNNs) in Question Answering

RNNs, particularly LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs (Cho et al., 2014), were a significant step forward in processing sequential data like text. Their ability to maintain a hidden state allows them to capture dependencies across tokens in a sequence. In question answering, RNNs have been used to encode both the question and the answer choices (or context passages) into fixed-size vectors, which are then used for classification or similarity computation (Wang & Jiang, 2016). For instance, Hermann et al. (2015) demonstrated the use of attentive LSTMs for reading comprehension. However, RNNs can struggle with very long-range dependencies and are computationally intensive to train compared to newer architectures.

2.4 Transformer Models for Question Answering

The advent of the Transformer architecture (Vaswani et al., 2017), with its self-attention mechanism, marked a paradigm shift in NLP. Models like BERT (Devlin et al., 2019) and its variants demonstrated state-of-the-art performance on numerous benchmarks, including QA. ALBERT (A Lite BERT for Self-supervised Learning of Language Representations) (Lan et al., 2020), used in this project, introduced parameter-reduction techniques to lower memory consumption and increase training speed compared to BERT, while often maintaining or improving performance. ALBERT, like BERT, is pre-trained on massive text corpora using self-supervised objectives (like Masked Language Modeling and Sentence Order Prediction), enabling it to learn rich contextual representations. For multiple-choice QA, models like 'AlbertForMultipleChoice' are specifically designed to process a question along with several candidate answers and predict the most plausible

one. Further advancements like RoBERTa (Liu et al., 2019) showed that robustly optimizing the pre-training process of BERT-like models can lead to significant performance gains.

2.5 Fine-tuning Strategies for Pre-trained Language Models

Fine-tuning pre-trained language models (PLMs) on downstream tasks is a dominant paradigm in modern NLP (Howard & Ruder, 2018). The process typically involves adding one or more task-specific layers on top of the PLM and then training the entire model (or a subset of its layers) on task-specific labeled data. Effective fine-tuning often requires careful hyperparameter selection (e.g., learning rate, batch size, number of epochs) to avoid issues like catastrophic forgetting or overfitting to the smaller downstream dataset. Strategies such as using a smaller learning rate for the PLM layers compared to the task-specific layers, and employing gradual unfreezing, have been explored to optimize this process (Sun et al., 2019). Raffel et al. (2020) in their T5 paper also explored a unified text-to-text framework where various NLP tasks, including QA, are cast as text generation problems, relying on effective fine-tuning.

2.6 Prompting Techniques for Large Language Models (LLMs)

Large Language Models (LLMs) like GPT-3 (Brown et al., 2020) and Flan-T5 (Chung et al., 2022) have shown remarkable capabilities in performing tasks with minimal or no task-specific training, through prompting.

- **Zero-shot prompting** involves directly querying the LLM with the task instance without providing any examples.
- **Few-shot prompting** provides the LLM with a few examples of the task (input-output pairs) in the prompt before the actual query, enabling in-context learning.
- **Chain-of-Thought (CoT) prompting** (Wei et al., 2022) encourages LLMs to generate intermediate reasoning steps before arriving at a final answer, which has been shown to improve performance on complex reasoning tasks. Kojima et al. (2022) further showed that even a simple prompt like "Let's think step by step" can elicit reasoning in zero-shot settings.
- **Retrieval Augmented Generation (RAG)** (Lewis et al., 2020) combines PLMs with non-parametric memory, typically a retriever that fetches relevant documents or text snippets from a large corpus (like the web in this project via SerpAPI) to provide context for the LLM's generation process. This helps ground LLMs in factual information and can reduce hallucination.

The effectiveness of these techniques often depends on the LLM's scale, the nature of the task, and the quality of the prompts and retrieved context.

2.7 Evaluation Metrics and Benchmarks for CQA

Accuracy is the primary metric for multiple-choice CQA tasks like CommonsenseQA. However, analyzing precision, recall, and F1-score per class (or macro-averaged) can provide deeper insights into a model's performance, especially in cases of class imbalance or specific weaknesses. Confusion matrices are also vital for understanding error patterns. Beyond CommonsenseQA, benchmarks like ARC (AI2 Reasoning Challenge) (Clark et al., 2018) and WinoGrande (Sakaguchi et al., 2020) further probe commonsense reasoning abilities. Ongoing research focuses on developing more robust evaluation methods that go beyond simple accuracy and test for true understanding and reasoning, addressing issues like dataset biases and model shortcuts (Rogers et al., 2023).

3 Data

3.1 Dataset Description

This project utilizes the CommonsenseQA dataset sourced from Hugging Face. The dataset is specifically designed for commonsense reasoning and comprises multiple-choice questions. Each instance in the dataset consists of:

- **id**: A unique identifier for the question.
- **question**: The question text which requires commonsense reasoning.
- **question_concept**: The main concept the question revolves around.
- **choices**: A dictionary containing:
  - **label**: A list of labels for the choices (typically 'A', 'B', 'C', 'D', 'E').
  - **text**: A list of five potential answer texts corresponding to the labels.
- **answerKey**: The label of the correct answer (e.g., 'A').

The objective for the model is to select the most plausible answer from the five given choices. For this project, the standard dataset splits were used:

- **Training set**: 'train[:-1000]' from the Hugging Face dataset.
- **Validation set**: 'train[-1000:]' from the Hugging Face dataset.
- **Test set**: 'validation' split from the Hugging Face dataset.

The 'id' and 'question_concept' columns were removed during preprocessing as they were not directly used for model training.

3.2 Data Preprocessing

Different preprocessing steps were applied depending on the architectural approach.

3.2.1 Preprocessing for FastText-based Models (Approach 1)

The preprocessing for models utilizing FastText embeddings involved several steps to prepare the textual data:

1. **Tokenization**: Questions and answer choices were tokenized into individual words using NLTK's 'word_tokenize()' function. This function handles punctuation, special characters, and word boundaries effectively for general-purpose tokenization. The rationale for choosing NLTK's tokenizer was its ease of use and general effectiveness for English text.
2. **Lemmatization**: Tokens were lemmatized using NLTK's 'WordNetLemmatizer()'. Lemmatization was chosen over stemming as it maps words to their dictionary forms (e.g., "better" to "good") using linguistic context, ensuring more accurate normalization without significant loss of meaning, which is crucial for commonsense understanding.
3. **Case Sensitivity**: Lowercasing was intentionally avoided. The rationale was that preserving case sensitivity retains semantic distinctions that can be critical for proper nouns, acronyms, and context-specific capitalization (e.g., "US" as a country vs. "us" as a pronoun), which might be relevant in commonsense scenarios.
4. **Stopword and Punctuation Removal**: Stopwords and punctuation were not removed. The justification is that these elements often provide critical syntactic, semantic, and tonal context. For instance, negation words are essential for sentiment and logical reasoning, and punctuation can define sentence boundaries or emphasis necessary for full comprehension.

5. **Handling Unknown Words**: For words not present in the pre-trained FastText vocabulary, a zero vector of the appropriate dimension (300 for 'fasttext-wiki-news-subwords-300') was used. This ensures that all tokens have a representation, even if out-of-vocabulary terms do not contribute learned semantic information from pre-training. The rationale is to maintain consistent input dimensionality.

After these steps, the question and each choice were represented by the average of their token embeddings. These average embeddings were then concatenated to form the input for the classifier models.

3.2.2 Preprocessing for Transformer-based Models (Approach 2 - ALBERT)

For the ALBERT model, preprocessing was handled by the 'AutoTokenizer' from the Hugging Face 'transformers' library, specifically 'albert-base-v2'. The rationale for using the model-specific tokenizer is to ensure consistency with its pre-training.

1. **Input Formatting**: For each question and its five answer choices, five distinct input sequences were created. Each sequence consisted of the question text concatenated with one of the answer choices. This formatting is standard for multiple-choice tasks with BERT-like models, allowing the model to process each question-choice pair in context.

2. **Tokenization**: The 'tokenizer' converted these input texts into token IDs, including special tokens required by ALBERT (e.g., '[CLS]', '[SEP]').

3. **Padding and Truncation**: All tokenized sequences were padded to a 'max_length' of 80 tokens (determined by analyzing the dataset's tokenized length distribution, where the maximum was 73) or truncated if they exceeded this length. Padding was applied to ensure uniform input tensor shapes for batch processing.

4. **Attention Masks**: Attention masks were generated alongside token IDs to indicate which tokens are actual content and which are padding, enabling the self-attention mechanism to focus appropriately.

The output of the tokenizer (input IDs and attention masks) for each of the five question-choice pairs, along with the correct label index (0-4), formed the input to the 'AlbertForMultipleChoice' model.

3.2.3 Preprocessing for LLM Prompting (Approach 3 - Flan-T5)

For the Flan-T5-large model, the raw text of the question and choices was formatted into specific prompt structures. The rationale was to leverage the instruction-following capabilities of Flan-T5.

1. **Prompt Formatting**: A helper function 'format_example' was used to structure the input clearly, mimicking a standard multiple-choice question format that the LLM can easily parse.

2. **Zero-Shot**: The direct prompt was used to assess the LLM's inherent commonsense ability without task-specific examples.

3. **Few-Shot**: Providing two examples was intended to give the model in-context learning cues about the task format and expected output style.

4. **Zero-Shot CoT**: Appending "Let's think step by step." was a direct application of CoT prompting to encourage explicit reasoning steps.

5. **RAG**: Providing web search snippets as context aimed to ground the LLM's responses in potentially relevant external information, testing if this improves reasoning.

The tokenizer for 'google/flan-t5-large' was used to convert these prompts into input IDs.

3.3 Dataset Usage

The dataset was consistently split into training, validation, and test sets as described in Section 3.1 for all model training and evaluation phases. This ensures comparability of results across different approaches. For the LLM prompting techniques (zero-shot, few-shot, CoT, RAG), evaluation was primarily performed on the test set, with few-shot examples drawn from the training set.

4 Methods

This section details the different architectural approaches and methodologies employed to address the commonsense question answering task. The rationale behind each choice is also discussed.

4.1 Approach 1: Word Embeddings with Conventional Classifiers

This approach was chosen as a baseline to evaluate the effectiveness of traditional NLP techniques using pre-trained word embeddings before moving to more complex architectures.

4.1.1 Word Embeddings

FastText embeddings ('fasttext-wiki-news-subwords-300') were selected due to their ability to handle out-of-vocabulary words through subword information, which is beneficial for datasets like CommonsenseQA that might contain diverse vocabulary. Averaging token embeddings provided a simple yet standard way to obtain fixed-size representations for variable-length texts (questions and choices).

4.1.2 Model 1.1: FastText + Feedforward Classifier

- **Input Representation**: Concatenating the average question embedding with each average choice embedding aimed to provide the classifier with a combined representation of the question-choice pair.
- **Architecture**: A 2-layer feedforward network was chosen for its simplicity and as a common baseline classifier. The ReLU activation was used for non-linearity. The rationale was to see if a straightforward non-linear mapping could discern commonsense relationships from the concatenated embeddings.
- **Training**: Standard CrossEntropyLoss and Adam optimizer were used, common choices for classification tasks.

4.1.3 Model 1.2: FastText + RNN (LSTM) + Classifier

- **Input Representation**: Using sequences of embeddings directly, rather than averaging, was intended to preserve sequential information within the question and choices.
- **Architecture**: An LSTM was chosen for its ability to capture long-range dependencies in sequences, which could be beneficial for understanding the context of questions and choices. The final hidden state was taken as a summary of the sequence. This was then fed to a similar 2-layer feedforward classifier. The rationale was that an LSTM might better model the contextual nuances than simple embedding averaging.
- **Training**: Similar to Model 1.1, with the addition of gradient clipping to stabilize RNN training.

4.2 Approach 2: Transformer-based Models (ALBERT)

Transformers were chosen due to their proven state-of-the-art performance on various NLP tasks, including question answering. ALBERT ('albert-base-v2') was selected as a lighter yet effective variant of BERT.

4.2.1 Model 2.1: Fine-tuning Pre-trained ALBERT

- **Model**: 'AlbertForMultipleChoice' is specifically designed for this type of task. Fine-tuning a pre-trained model leverages the vast knowledge learned during its initial training on large corpora. The rationale is that this pre-trained knowledge provides a strong foundation for understanding language and, potentially, commonsense.

- **Input Representation**: The standard input format for 'AlbertForMultipleChoice' (concatenated question-choice pairs) was used.
- **Training**: Fine-tuning involved updating the weights of the pre-trained model on the CommonsenseQA dataset. AdamW optimizer is standard for transformers.

4.2.2 Model 2.2: Training ALBERT from Random Initialization

- **Rationale**: This experiment was designed to isolate and quantify the contribution of pre-training. By training an ALBERT model with the same architecture but random initial weights, we can compare its performance directly against the fine-tuned pre-trained version.

4.3 Approach 3: Large Language Models (Flan-T5-large) - Prompting Techniques

Flan-T5-large was chosen as a representative instruction-tuned LLM, known for its strong performance across a wide range of tasks with prompting. The rationale for exploring prompting was to assess how well these large models can perform commonsense reasoning without task-specific fine-tuning.

4.3.1 Model 3.1: Zero-Shot Prompting

- **Rationale**: To establish a baseline for Flan-T5's out-of-the-box commonsense capability.

4.3.2 Model 3.2: Few-Shot Prompting

- **Rationale**: To assess if providing a small number of in-context examples helps the LLM better understand the task format and improve its reasoning.

4.3.3 Model 3.3: Zero-Shot Chain-of-Thought (CoT) Prompting

- **Rationale**: To investigate whether prompting the LLM to generate intermediate reasoning steps enhances its ability to arrive at the correct answer for commonsense questions, which often require implicit reasoning.

4.3.4 Model 3.4: Retrieval Augmented Generation (RAG) with Live Web Search

- **Rationale**: To explore if grounding the LLM with external information retrieved from live web searches can improve its performance, particularly for questions where commonsense might intersect with or benefit from factual knowledge.

4.4 Experiment Tracking and Evaluation

The use of Weights & Biases was motivated by the need for systematic tracking of numerous experiments and hyperparameters. Standard metrics like accuracy, precision, recall, F1-score, and confusion matrices were chosen as they are common for classification tasks and provide a comprehensive view of model performance.

5 Model Validation, Experiments, and Results

This section presents the experimental setup, training procedures, hyperparameter tuning efforts, and the performance results for each of the architectural approaches explored in this project. Model validation was primarily conducted by evaluating performance on a dedicated validation set during hyperparameter tuning and assessing generalization on a final held-out test set.

5.1 Experiment Tracking

All experiments involving model training (Approaches 1 and 2) were rigorously tracked using Weights & Biases (W&B). This included logging hyperparameters, training/validation losses, accuracy, and other relevant metrics for each run. The W&B project page for these experiments can be found at: `https://wandb.ai/aditi-sharma-00073-hochschule-luzern/` (links for specific sub-projects are in the notebooks).

5.2 Approach 1: Word Embeddings with Conventional Classifiers

Experiments for this approach were conducted using pre-trained FastText embeddings.

5.2.1 Model 1.1: FastText + Feedforward Classifier

- **Training**: The model was trained for 3 epochs using Adam optimizer and CrossEntropyLoss.
- **Hyperparameter Tuning (Optuna)**: Validation accuracy on the validation set was the target metric for Optuna.
  - Search Space: Learning rate (1e-5 to 1e-2), batch size ([16, 32, 64]), hidden dimension (128 to 512), weight decay (1e-6 to 1e-3).
  - Number of Trials: 4.
  - Best Parameters Found: Learning rate $\approx 1.1 \times 10^{-4}$, batch size 64, hidden dimension 227, weight decay $\approx 7 \times 10^{-6}$.
  - Best Validation Accuracy during tuning: 45.3%.
- **Final Training with Best Parameters**: The model trained with best parameters on the combined train+validation set achieved a training accuracy of approximately 38.3% over 3 epochs.
- **Evaluation on Test Set**:
  - Accuracy: 20.56%
  - Precision (macro): 7.20%
  - Recall (macro): 19.70%
  - F1-score (macro): 7.13%
- **Critical Evaluation**: The very low test accuracy and F1-score, along with the heavily biased confusion matrix (Figure 1), indicate that this model failed to learn meaningful commonsense reasoning. The performance is close to random guessing for a 5-choice task (20%). The validation accuracy achieved during tuning (45.3%) was significantly higher than the test accuracy, suggesting potential overfitting or that the validation set was not entirely representative.

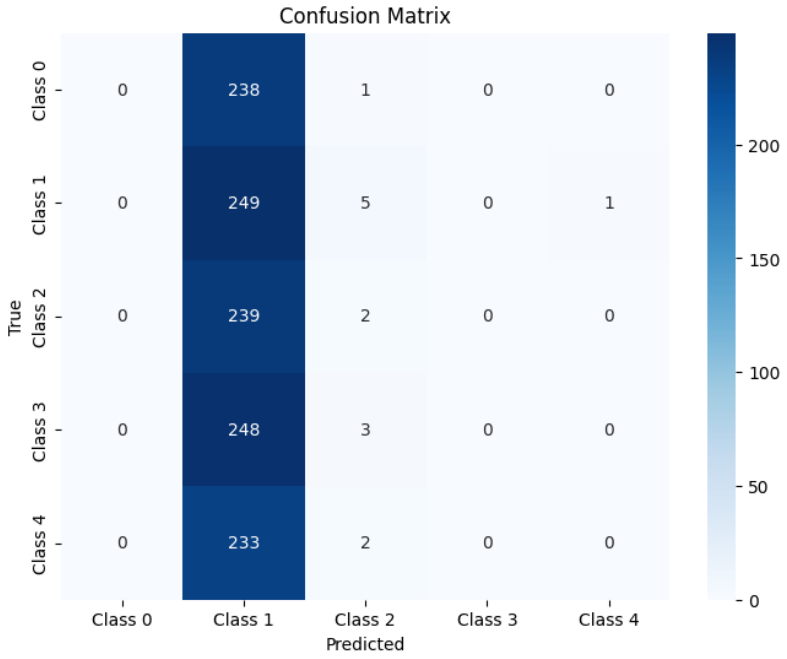

Figure 1: Confusion Matrix for FastText + Feedforward Classifier (Model 1.1) on Test Set.

5.2.2 Model 1.2: FastText + RNN (LSTM) + Classifier

- **Training**: Trained for 3 epochs with Adam optimizer, CrossEntropyLoss, and gradient clipping. Max sequence length for padding was 22 tokens.
- **Hyperparameter Tuning (Optuna)**: Validation accuracy on the validation set was the objective.
  - Search Space: Learning rate (1e-5 to 1e-2), batch size ([16, 32, 64]), RNN hidden dimension (128 to 512), RNN layers (1 to 3), dropout (0.1 to 0.5), weight decay (1e-6 to 1e-3).
  - Number of Trials: 5.
  - Best Parameters Found: Learning rate $\approx 8.3 \times 10^{-5}$, batch size 32, hidden dimension 470, weight decay $\approx 2.7 \times 10^{-4}$, 3 LSTM layers, dropout ≈ 0.46.
  - Best Validation Accuracy during tuning: 21.3%.
- **Final Training with Best Parameters**: The model trained on the full train+validation set with these parameters achieved a training accuracy of approximately 18.85% over 3 epochs.
- **Evaluation on Test Set**:
  - Accuracy: 19.08%
  - Precision (macro): 18.07%
  - Recall (macro): 19.37%
  - F1-score (macro): 9.91%
- **Critical Evaluation**: This model also performed very poorly, achieving an accuracy barely distinguishable from random guessing. The confusion matrix (Figure 2) showed a strong bias towards predicting class 'A'. The low validation accuracy during tuning (21.3%) already indicated that the RNN architecture with FastText embeddings was not effective for this task.

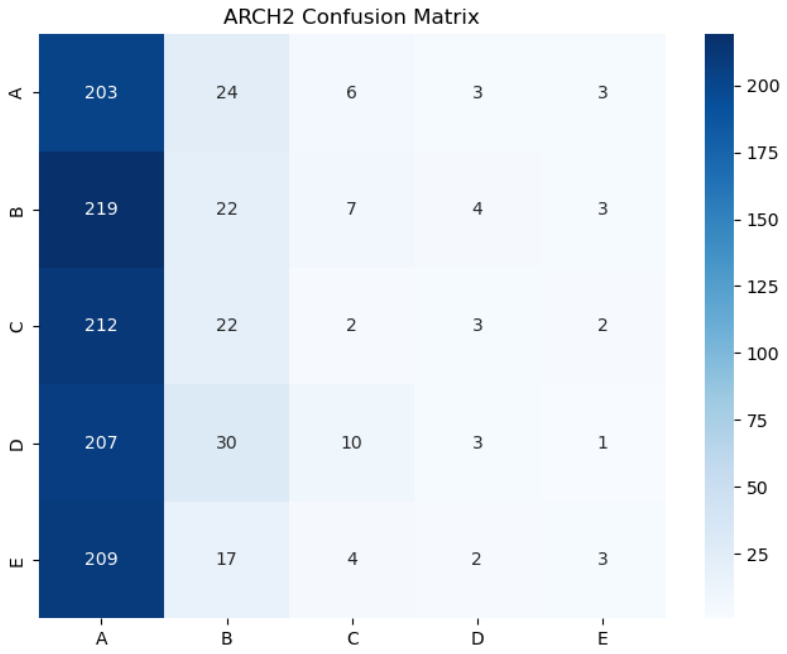

Figure 2: Confusion Matrix for FastText + RNN Classifier (Model 1.2) on Test Set.

5.3 Approach 2: Transformer-based Models (ALBERT)

The ALBERT ('albert-base-v2') model was used with a maximum sequence length of 80. Model validation involved monitoring validation accuracy during training epochs to select the best performing checkpoint.

5.3.1 Model 2.1: Fine-tuning Pre-trained ALBERT

- **Hyperparameter Sweep**: A manual sweep was conducted. The best validation accuracy during the sweep (around 55.3% in epoch 3) was achieved with lr=2e-5, dropout=0.1, wd=0.0, batch_size=16.
- **Final Training (using best sweep config for 5 epochs)**: Validation accuracy peaked at 55.30% in epoch 3, then started to decrease, indicating overfitting.
- **Evaluation on Test Set (using model from epoch 3 of final training)**:
  - Accuracy: 54.38%
  - Precision (weighted): 54.61%
  - Recall (weighted): 54.38%
  - F1-score (weighted): 54.43%
- **Critical Evaluation**: Fine-tuned ALBERT showed a marked improvement over FastText models, demonstrating the power of pre-trained transformers. The accuracy is significantly above random guessing. However, the model started overfitting after 3 epochs, indicating that with the given dataset size, more prolonged training was not beneficial. The confusion matrix (Figure 3) was more balanced, but still showed room for improvement across all classes.

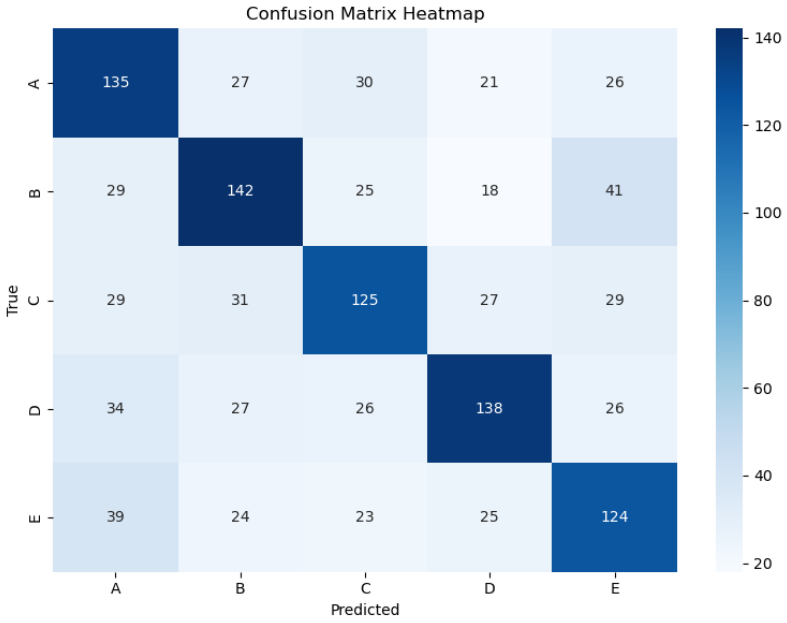

Figure 3: Confusion Matrix for Fine-tuned ALBERT (Model 2.1) on Test Set.

5.3.2 Model 2.2: Training ALBERT from Random Initialization

- **Training**: Trained for 5 epochs using the same hyperparameters as the best fine-tuned ALBERT. Validation accuracy remained low throughout training, peaking at 23.20%.
- **Evaluation on Test Set (using model from epoch 5)**:
  - Accuracy: 20.80%
  - Precision (weighted): 20.81%
  - Recall (weighted): 20.80%
  - F1-score (weighted): 20.80%
- **Critical Evaluation**: The performance was abysmal, similar to the FastText models. This experiment effectively validates the hypothesis that the strength of models like ALBERT heavily relies on their pre-training. Without it, the model could not learn the task from the given dataset. The confusion matrix (Figure 4) showed near-random performance.

5.4 Approach 3: Large Language Models (Flan-T5-large) - Prompting Techniques

Model validation for these approaches is direct evaluation on the test set, as no training is involved.

5.4.1 Model 3.1: Zero-Shot Prompting

- **Accuracy on Test Set**: 85.83%
- **Critical Evaluation**: Flan-T5-large exhibited remarkable zero-shot performance, significantly outperforming all trained models. This indicates strong inherent commonsense reasoning capabilities learned during its extensive pre-training and instruction tuning. The confusion matrix (Figure 5) was highly diagonal, showing excellent classification.

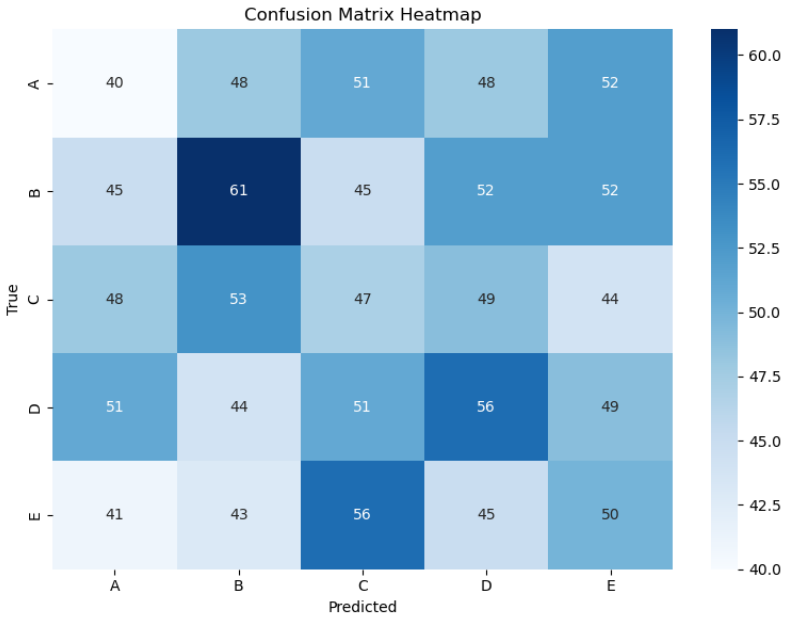

Figure 4: Confusion Matrix for ALBERT Trained from Scratch (Model 2.2) on Test Set.

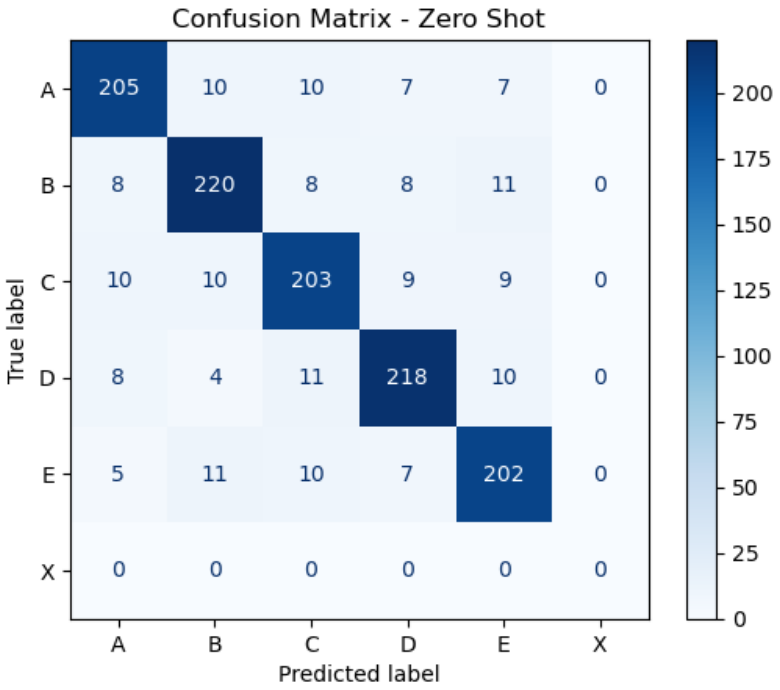

Figure 5: Confusion Matrix for Flan-T5-large Zero-Shot (Model 3.1) on Test Set.

5.4.2 Model 3.2: Few-Shot Prompting (2-shot)

- **Accuracy on Test Set**: 85.09%
- **Critical Evaluation**: Few-shot prompting performed comparably to zero-shot, with a very slight decrease. This suggests that for Flan-T5 on CommonsenseQA, minimal in-context examples are sufficient, and adding more might not always lead to gains, potentially due to prompt length or the choice of examples. The confusion matrix (Figure 6) remained strong.

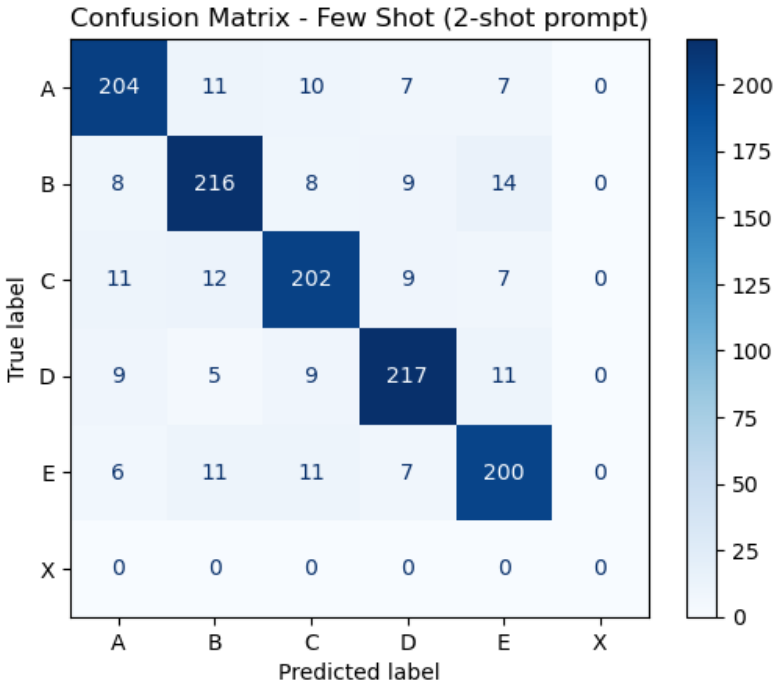

Figure 6: Confusion Matrix for Flan-T5-large Few-Shot (Model 3.2) on Test Set.

5.4.3 Model 3.3: Zero-Shot Chain-of-Thought (CoT) Prompting

- **Accuracy on Test Set**: 62.98%
- **Critical Evaluation**: CoT prompting led to a substantial drop in accuracy compared to direct zero/few-shot methods. While CoT is designed to elicit reasoning, it might have made the task more complex for the model in this multiple-choice setting, or the generated rationales were not always leading to the correct final answer label. The confusion matrix (Figure 7) showed more distributed errors.

5.4.4 Model 3.4: RAG with Live Web Search (SerpAPI)

- **Accuracy on 50 Test Samples**: 78.00%
- **Critical Evaluation**: On the subset, RAG improved over Zero-Shot CoT, suggesting that external context aids Flan-T5. However, it did not reach the levels of direct zero/few-shot prompting, possibly due to the nature of the retrieved snippets or the simplicity of the RAG integration. A full test set evaluation would be needed for a conclusive comparison.

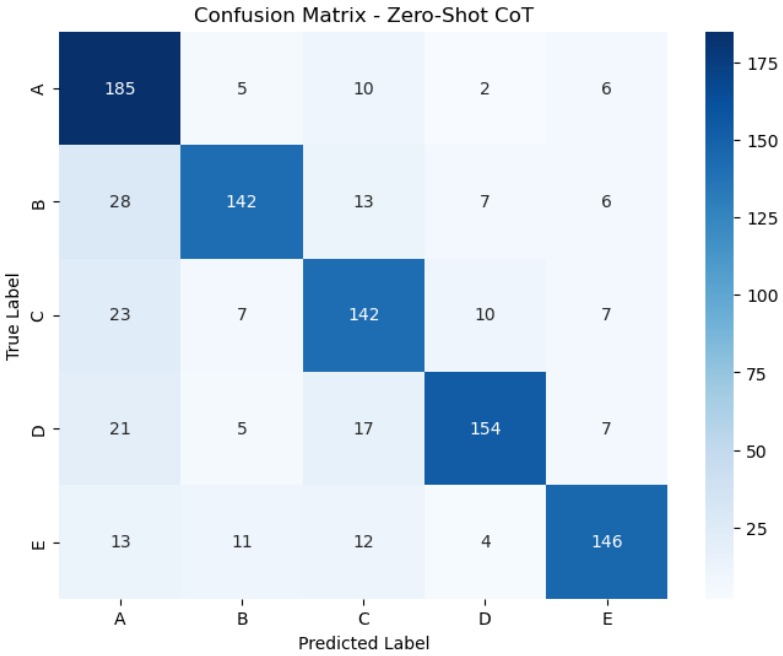

Figure 7: Confusion Matrix for Flan-T5-large Zero-Shot CoT (Model 3.3) on Test Set.

6 Machine Learning Operations (MLOps)

Effective MLOps practices were employed throughout this project to ensure reproducibility, systematic experimentation, and efficient model management.

- **Experiment Tracking**: Extensive use of Weights & Biases (W&B) for logging hyperparameters, metrics (loss, accuracy, F1, precision, recall), and model configurations across all training runs for Approaches 1 and 2. This facilitated systematic comparison and reproducibility. W&B project links were provided in the respective Jupyter notebooks.
- **Hyperparameter Optimization**: Optuna was utilized for systematic hyperparameter tuning for the FastText-based models (Approach 1), with results logged to W&B. For ALBERT models (Approach 2), a manual sweep was conducted and tracked using W&B.
- **Model Saving and Management**:
  - For Optuna-tuned models, the best model state dictionary based on validation accuracy was determined, and a final model was trained using these best parameters.
  - For ALBERT models, the notebook 'NLP-2-AditiSharma.ipynb' includes logic to save the best model checkpoint (e.g., 'best_albert_random_model.pt') based on validation accuracy during epochal training. This allowed for reloading the best performing model for later evaluation.
- **Code and Environment Management**:
  - The project was developed primarily using Jupyter Notebooks, allowing for iterative experimentation and documentation of steps.
  - Dependency management was handled via 'pip install' commands within the notebooks, ensuring that necessary libraries such as 'datasets', 'wandb', 'transformers', 'nltk', 'gensim', 'torch', 'sklearn', 'optuna', 'accelerate', 'requests', and 'openai' were available in the execution environment.
- **Data Handling and Versioning**: The CommonsenseQA dataset was sourced directly from Hugging Face using their 'datasets' library. This library handles dataset downloading, caching, and provides mechanisms for dataset versioning by referring to specific dataset versions or commit hashes on the Hugging Face Hub. This ensures that the experiments used a consistent and well-defined version of the dataset, reducing variability due to data changes.
- **Reproducibility**: To enhance the reproducibility of experimental results, random seeds were set for 'random', 'numpy', and 'torch' libraries, particularly in the LLM experimentation phase (Approach 3) as detailed in 'NLP-2-AditiSharma.ipynb'.
- **Computational Resources**: Model training and experimentation, especially for the more computationally intensive transformer models, were conducted leveraging the GPUHub provided by the Hochschule Luzern (HSLU). This infrastructure provided access to necessary GPU resources for efficient training.
- **Deployment Readiness**: While actual deployment of a model into a production environment was outside the scope of this academic project, the MLOps practices employed contribute to deployment readiness. Systematic experiment tracking, version control of code (assumed via standard development practices), environment management through 'pip', and model checkpointing are foundational steps. These practices ensure that a selected model (e.g., the fine-tuned ALBERT or a prompted Flan-T5 setup) could be packaged and deployed more reliably, for instance, as an API endpoint, should a practical application be developed. The clear documentation within notebooks also aids in understanding the model's creation and facilitating its transition to a potentially operational state.

These MLOps practices, while adapted for a research-oriented student project, aimed to incorporate principles of good machine learning engineering.

7 Discussion

The project explored several architectural approaches for commonsense question answering, revealing significant differences in performance and highlighting key aspects of model optimization for this task.

Comparison of Approaches:

1. **FastText-based Models (Approach 1)**: Both the feedforward classifier (Model 1.1) and the RNN-based classifier (Model 1.2) using FastText embeddings performed poorly, with test accuracies around 20%. Their confusion matrices showed strong biases towards predicting specific classes, indicating a failure to capture the complex semantic relationships required for commonsense reasoning. The RNN model, despite its sequential processing capabilities, did not offer a significant improvement over the simpler feedforward network. The initial rationale for these models was to establish a baseline with traditional techniques, but their performance underscores the insufficiency of simple embedding averaging or basic RNNs for complex reasoning.

2. **Transformer-based Models (Approach 2 - ALBERT)**:

   - **Fine-tuned Pre-trained ALBERT (Model 2.1)**: This approach yielded a substantial improvement, with a test accuracy of 54.38%. The rationale for choosing a transformer like ALBERT was its proven ability to capture contextual information through self-attention. The fine-tuning process leveraged this pre-trained knowledge effectively. The model demonstrated a much better ability to distinguish between classes. Overfitting observed after 3 epochs suggests that for this dataset size, early stopping or further regularization might be beneficial.
   - **ALBERT Trained from Scratch (Model 2.2)**: This model performed exceptionally poorly (20.80% accuracy), similar to the FastText models. This starkly illustrates the critical importance of pre-training. The rationale for this experiment—to quantify pre-training's impact—was clearly validated. Without this foundational knowledge, ALBERT could not learn meaningful representations.

3. **Large Language Models (Approach 3 - Flan-T5-large)**:

   - **Zero-Shot and Few-Shot Prompting (Models 3.1, 3.2)**: Flan-T5-large's strong performance (85.83% zero-shot, 85.09% few-shot) confirmed the rationale that large, instruction-tuned LLMs possess significant inherent commonsense capabilities. The minor difference between zero-shot and few-shot suggests the model might already be well-aligned for this task format.
   - **Zero-Shot CoT (Model 3.3)**: The accuracy drop to 62.98% with CoT was unexpected. The rationale for CoT is to improve reasoning. However, for this multiple-choice task, forcing a reasoning chain might have led the model to generate plausible but incorrect rationales or become sidetracked, ultimately selecting a suboptimal answer.
   - **RAG with Live Web Search (Model 3.4)**: The 78.00% accuracy on a subset (vs. CoT's 62.98%) supports the rationale that external, factual context can aid even commonsense tasks, possibly by resolving ambiguities or providing specific grounding knowledge that might not be perfectly encoded even in a large LLM.

Key Insights and Critical Evaluation:

- **Impact of Pre-training and Model Scale**: The project overwhelmingly demonstrates that model performance on CQA is heavily tied to the scale of pre-training and the sophistication of the architecture. LLMs and fine-tuned transformers far surpassed other methods.
- **Effectiveness of Transfer Learning**: Fine-tuning pre-trained ALBERT was significantly more effective than training from scratch, confirming the power of transfer learning.

- **Prompting Nuances**: The success of prompting LLMs is highly dependent on the strategy. While direct prompting was very effective, CoT was less so for this specific setup, indicating that prompting methods are not universally optimal and require careful selection and tuning for the task at hand.
- **Limitations of Early Models**: The early models (FastText + NN/RNN) were clearly inadequate. Their failure highlights the complexity of commonsense reasoning, which requires more than simple semantic similarity or basic sequential modeling. Their poor validation scores during tuning were early indicators of these limitations.
- **Model Validation and Overfitting**: For trainable models like ALBERT, validation set performance was crucial for identifying the optimal training duration (e.g., 3 epochs for fine-tuned ALBERT) to prevent overfitting, which was evident in later epochs.

The progression from simple models to LLMs provided a clear learning curve, emphasizing the advancements in NLP and the current capabilities of large-scale models for complex reasoning.

8 Conclusions and Future Work

8.1 Summary of Key Findings

This project, DSPRO2, systematically evaluated various machine learning architectures for the CommonsenseQA task. The key findings are:

1. **Traditional NLP models** (FastText with Feedforward/RNN classifiers) exhibited poor performance, with accuracies around 20%, and struggled with biased predictions. These models lacked the capacity for deep commonsense reasoning.
2. **Transformer-based models (ALBERT)** showed a significant improvement when fine-tuned from pre-trained weights, achieving a test accuracy of 54.38%. In contrast, ALBERT trained from scratch performed as poorly as the traditional models, underscoring the critical role of pre-training.
3. **Large Language Models (Flan-T5-large)** using prompting techniques achieved the highest performance. Zero-shot prompting yielded an impressive 85.83% accuracy, and few-shot prompting performed similarly at 85.09%. Zero-shot Chain-of-Thought (CoT) prompting, surprisingly, resulted in a lower accuracy of 62.98%.
4. **Retrieval Augmented Generation (RAG)** with Flan-T5, evaluated on a subset of data, achieved 78.00% accuracy, demonstrating the potential of incorporating external knowledge.

The results clearly indicate that pre-trained knowledge and architectural sophistication (Transformers, LLMs) are paramount for achieving strong performance on commonsense reasoning tasks.

8.2 Contributions of the Project

This project contributes by:

- Providing a comparative analysis of different classes of models (embedding-based, transformers, LLMs) on the CommonsenseQA benchmark.
- Quantifying the significant impact of pre-training by comparing fine-tuned ALBERT with an ALBERT model trained from scratch.
- Exploring various prompting strategies for LLMs and highlighting their differential effectiveness (e.g., zero-shot vs. CoT vs. RAG).
- Systematically tracking experiments using W&B and employing hyperparameter optimization techniques (Optuna, sweeps).

8.3 Limitations

The project faced certain limitations:

- **Computational Resources**: Training larger models or more extensive hyperparameter searches for transformers and LLMs can be computationally intensive. The RAG evaluation was limited to a small subset due to API call constraints or processing time.
- **Scope of Architectures**: While several architectures were explored, many other advanced models and techniques exist that were not covered.
- **Depth of CoT/RAG Exploration**: The CoT and RAG explorations were preliminary. More sophisticated prompt engineering or retrieval strategies could yield different results.
- **Interpretability**: While performance was the primary focus, deeper analysis into the reasoning processes or failure modes of the models, especially LLMs, was limited.

8.4 Future Work

Building on the findings and limitations of this project, several avenues for future work can be identified:

- **Advanced Transformer Architectures**: Experiment with more recent and larger transformer models (e.g., RoBERTa, DeBERTa, newer T5 variants, or GPT-family models if accessible).
- **Sophisticated Fine-tuning**: Explore advanced fine-tuning techniques for transformers, such as adapter-based tuning (LoRA), or multi-task learning if applicable.
- **Enhanced Prompt Engineering for LLMs**: Conduct a more thorough investigation into prompt engineering for Flan-T5 and other LLMs, including more complex CoT structures and dynamic few-shot example selection.
- **Scalable RAG Implementation**: Develop a more scalable RAG pipeline with optimized retrieval mechanisms and evaluate it on the full test set.
- **Model Ensembling**: Investigate ensembling techniques, combining predictions from different strong models (e.g., fine-tuned ALBERT and Flan-T5 with different prompting).
- **Error Analysis and Interpretability**: Conduct a detailed error analysis to understand common failure modes and explore model interpretability techniques to gain insights into their reasoning processes.
- **Knowledge Integration**: Explore methods for more explicitly integrating commonsense knowledge graphs or structured knowledge bases into the models.

8.5 Closing Statement

The DSPRO2 project successfully navigated a range of NLP model architectures, demonstrating a clear hierarchy in performance for commonsense question answering, with large pre-trained language models using appropriate prompting strategies achieving state-of-the-art results for this setup. The journey from simpler embedding-based models to complex transformers and LLMs provides valuable lessons in the ongoing quest for artificial general intelligence, particularly in the domain of commonsense reasoning.

References

- Bisk, Y., Zellers, R., Lebras, R., Gao, J., & Choi, Y. (2020). PIQA: Reasoning about Physical Commonsense in Natural Language. In *Proceedings of the AAAI Conference on Artificial Intelligence.* (`https://arxiv.org/abs/1911.11641`)

- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics, 5*, 135-146. (`https://arxiv.org/abs/1607.04606`)

- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020).* (`https://arxiv.org/abs/2005.14165`)

- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* (`https://arxiv.org/abs/1406.1078`)

- Chung, H. W., Narang, S., Fedus, W., et al. (2022). Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv:2210.11416.* (`https://arxiv.org/abs/2210.11416`)

- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457.* (`https://arxiv.org/abs/1803.05457`)

- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).* (`https://arxiv.org/abs/1810.04805`)

- Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. In *Advances in neural information processing systems 28 (NIPS 2015).* (`https://arxiv.org/abs/1506.03340`)

- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780.

- Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL).* (`https://arxiv.org/abs/1801.06146`)

- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022).* (`https://arxiv.org/abs/2205.11916`)

- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations (ICLR).* (`https://arxiv.org/abs/1909.11942`)

- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020).* (`https://arxiv.org/abs/2005.11401`)

- Liu, J., Chen, Y., Liu, K., & Zhao, J. (2020). Commonsense Reasoning for Goal-Directed Dialogue. In *Proceedings of the AAAI Conference on Artificial Intelligence.*

- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692.* (`https://arxiv.org/abs/1907.11692`)
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781.*
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013).*
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research, 21*(140), 1-67. (`https://arxiv.org/abs/1910.10683`)
- Rogers, A., Kovaleva, O., & Rumshisky, A. (2023). A Primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics.* (Note: This is a general reference, a specific 2023 paper on evaluation might be more targeted if one exists with this exact title.)
- Sakaguchi, K., Le Bras, R., Bhagavatula, C., & Choi, Y. (2020). WinoGrande: An Adversarial Winograd Schema Challenge at Scale. In *Proceedings of the AAAI Conference on Artificial Intelligence.* (`https://arxiv.org/abs/1907.10641`)
- Sap, M., Rashkin, H., Chen, D., Lebras, R., & Choi, Y. (2019). Social IQA: Commonsense Reasoning about Social Interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).* (`https://arxiv.org/abs/1904.09728`)
- Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to Fine-Tune BERT for Text Classification? In *Chinese Computational Linguistics (CCL 2019).* (`https://arxiv.org/abs/1905.05583`)
- Talmor, A., Herzig, J., Lourie, N., & Berant, J. (2019). CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).* (Accessed from `https://huggingface.co/datasets/tau/commonsense_qa`)
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017).* (`https://arxiv.org/abs/1706.03762`)
- Wang, B., & Jiang, T. (2016). A Compare-Aggregate Model for Matching Text Sequences. *arXiv preprint arXiv:1611.01747.*
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022).* (`https://arxiv.org/abs/2201.11903`)
- Hugging Face Transformers Library: `https://huggingface.co/transformers`
- NLTK Project: `https://www.nltk.org/`
- PyTorch: `https://pytorch.org/`
- Weights & Biases: `https://wandb.ai/`
- Optuna: `https://optuna.org/`
- SerpAPI: `https://serpapi.com/`