Internship topic : Automatic open-book Q&A systems

Nicolas Sournac

Academic year 2023-2024
Master's degree in computer science

Sinch AI Center of Excellence – University Incubator
University of Mons

Contents

1 Introduction **2**

2 Background **3**
2.1 Datasets . . . 3
2.2 Metrics . . . 3
2.3 Fine-tuning methods . . . 3
2.4 Encoder tranformers . . . 4
2.5 Decoder transformers . . . 6

3 Open book extractive QA **7**
3.1 Experimental setup . . . 7
3.2 Results . . . 8

4 Open book generative QA **10**
4.1 Experimental setup . . . 10
4.2 Results . . . 12

5 Sinch API documentation **19**
5.1 Results . . . 20

6 Discussion **21**

7 Conclusion **23**

A Appendices **25**
A.1 DeBERTa Sinch documentation results . . . 25
A.2 GPT2 Sinch documentation results . . . 26
A.3 Mistral-7B Sinch documentation results . . . 27
A.4 Llama2-7B Sinch documentation results . . . 29
A.5 Llama2-7B DPO & Custom threshold Sinch documentation results . . . 31

1 Introduction

Question answering (QA) is a branch of natural language processing (NLP) focused on developing systems that can understand questions posed in human language and provide accurate and relevant answers. Open book QA involves a system that derives answers by analyzing and extracting information from textual sources. Such systems can be used in a wide variety of contexts. For example, a company might host such a system for its customers, to provide automatic answers to their questions. This use case could also be used internally by a company to provide a useful assistant for new employees. This topic presents a significant challenge for artificial intelligence and NLP as it necessitates both reading comprehension and reasoning. The system has a dual goal: delivering accurate and useful answers and refraining from generating false answers when the context doesn't support it. The emergence of transformer models has driven significant progress in QA. Modern foundation models like ChatGPT and Llama excel as valuable chatbot assistants or QA systems, despite the persistent challenge of hallucination. The topic of this internship is the study of large language models fine-tuning for the design of an open-book question answering system. This work explores two distinct approaches: it conducts extensive experiments with encoder transformer models and thoroughly investigates generative decoder-only transformer models, comparing them in a benchmark study. As describe in Figure 1, the models employed include BERT, DistillBERT, Albert, and DeBERTa for encoder models, as well as GPT2, Llama2-7B, and Mistral for decoder-only models. All these models underwent supervised fine-tuning on the SQuAD v1 and v2 datasets, with an exploration of the direct preference optimization (DPO) fine-tuning method to enhance decoder-only models. The application of specific scores and threshold selection has helped prevent the model from generating incorrect answers when they are unsupported by the context. Ultimately, a comparative study was carried out using real Sinch data associated with the conversation API documentation.

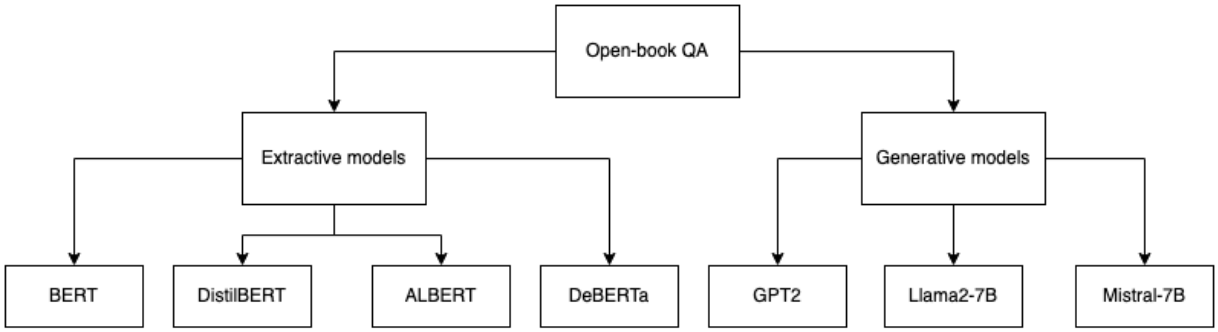

Figure 1: Explored models

2 Background

2.1 Datasets

Nowadays, there has been extensive research conducted on automatic open book question answering systems, and their popularity is steadily increasing. Numerous datasets have been made available with the intention of providing the community valuable resources for training and evaluating systems for this task. Some notable examples include SQuAD v1 and v2, as well as trivia_qa. SQuAD v1 [11] is a reading comprehension dataset featuring over 100,000 question-answer-context triplets, where answers are guaranteed to be within the context. SQuAD v2 [10] enhanced the previous dataset by introducing 50,000 unanswerable questions. This addition aims to enhance model robustness by training them to recognize unanswerable questions rather than providing inaccurate answers. Consequently, SQuAD v2 presents greater difficulty since unanswerable questions can closely resemble answerable ones. The Trivia_qa dataset [6] offers over 650,000 question-answer-context triplets. Notably, this dataset presents complex contexts, making answer extraction challenging. A single question may appear in multiple triplets, with different contexts provided. However, not all of these contexts necessarily contain the answer. More formally, an open-book QA dataset, denoted as $\mathcal{D}$, is defined as $\mathcal{D} = \{(q_i, a_i, c_i)\}_{i=1}^{n}$, where q_i represents the question, a_i is the corresponding answer, and c_i is the question's context.

2.2 Metrics

The metrics used for assessing open-book QA models are the exact-match (EM) and the F1-score. Given a sample (q_i, a_i, c_i) and the predicted answer $\hat{a}_i$ generated by the QA model, the exact match (EM) is a binary metric, resulting in 1 when $\hat{a}_i$ precisely matches a_i, and 0 otherwise. On the other hand, the F1-score is a more detailed metric that considers the shared words between $\hat{a}_i$ and a_i providing a more comprehensive evaluation of the answer's quality.

2.3 Fine-tuning methods

In the era of large language models (LLMs), the prevailing approach is to fine-tune pretrained models for specific downstream tasks. These pretrained LLMs are typically trained in an unsupervised manner on extensive text data, providing the model with a strong foundation for language understanding. Fine-tuning can be achieved through supervised methods (SFT), optimizing a subset or all of the LLM's weights for tasks such as QA, sentence classification, or text summarization. But modern LLMs designed for generating language may need to align with human preferences. In these scenarios, additional fine-tuning techniques like reinforcement learning with human feedback (RLHF) and direct preference optimization (DPO) may be employed. RLHF computes a reward function based on human feedback and then conducts reinforcement learning using this learned reward function. DPO, on the other hand, simplifies the process by replacing the RL-based objective with binary cross-entropy, achieving similar preference optimization results without the need for reward modeling [9]. While LLMs are known

for their high memory requirements, there are also methods for solving this problem, enabling training on a wide variety of hardware configurations.

LoRA and QLoRA

LoRA (Low-Rank Adaptation of Large Language Models) [4], is a methodology employed during the fine-tuning of LLMs. It serves to enhance the efficiency of the training process and mitigate the memory requirement associated with these models, which often constitute a significant obstacle for their broader adoption and utilization. The core concept of this approach is that overly complex models actually exist in a lower intrinsic dimension, and consequently, the weight updates during model fine-tuning also possess a lower intrinsic dimension. Given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, its update can be projected using a low-rank decomposition $W_0 + \Delta W = W_0 + BA$ where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank $r \ll min(d, k)$. Note that A and B contain trainable parameters. QLoRA (Quantized LoRA), as detailed in [1], extends LoRA techniques, thereby improving efficiency and lowering memory requirements during the fine-tuning of large language models. This is achieved through a high-fidelity 4-bit parameters quantization and memory-paged optimization.

2.4 Encoder tranformers

In **extractive question answering**, the goal is to indentify the exact portion of the context that contains the answer to a given question. Large language models, such as encoder transformers, are highly valuable for this task because they can encode and learn effective representations of both the question and the context. Indeed, encoder transformers enable each token to interact with every other token in the input sequence. These models are typically pre-trained using a masked language modeling task, which involves predicting missing words in sentences with masked or hidden tokens. For inference and given a sample (q_i, a_i, c_i), the input of the model is the concatenation of q_i and c_i. By adding a dedicated question-answering head to the encoder architecture, the model outputs a start logit, indicating the beginning of the answer (start position), and an end logit, indicating the end of the answer (end position) for each token of the input. During training, the correct start and end tokens of a_i is provided to the learning algorithm.

BERT

BERT (Bidirectional Encoder Representations from Transformers) [2], as a large language model, acquires valuable language representations through a masked language model and next sentence prediction pretraining objectives. BERT efficiently combines both right and left context to compute representations, making it effective for tasks like sentence classification or question answering. BERT can process both single sentences and pairs of sentences as input. BERT can be fine-tuned for various downstream tasks by adding a task-specific head on top and retraining all parameters end-to-end.

DistilBERT

DistilBERT [12], derived from the original BERT model, aims to reduce computational requirements for pre-training, fine-tuning, and inference. It is a lighter model (40% smaller) that runs faster (60% speed improvement) while maintaining approximately 97% of BERT's language understanding capabilities. DistilBERT has the same architecture as BERT, except it omits token-type embeddings and poolers, and reduces the number of layers by half. DistilBERT performs pretraining using knowledge distillation, a compression technique where a compact model (the student) is trained to reproduce the performance of a larger model (the teacher).

ALBERT

ALBERT (A Lite BERT) [7], is also derived from the original BERT model and aims to address the memory requirements problems of large language models. It employs two parameter reduction techniques and enhances the next sentence prediction loss, initially used in BERT's pretraining, through a self-supervised loss for sentence-order prediction. ALBERT has 18x fewer parameters and can be trained 1.7x faster while outperforming BERT's original results on GLUE, RACE and SQuAD v2. Specifically, ALBERT employs the same architecture as BERT but reduces model size through parameter factorization and cross-layer parameter sharing. While BERT used a next sentence prediction loss with positive samples consisting of consecutive sentences in a document and negative samples from different documents, ALBERT enhances this task by introducing more challenging negative samples. These are generated by swapping the order of two sequential sentences from the same document.

DeBERTa

DeBERTa (Decoding-Enhanced BERT with Disentangled Attention) [3], also built upon BERT, employs a disentangled attention mechanism. In this approach, each word is represented by two vectors, encoding both its content and position. DeBERTa introduces an enhanced mask decoder during the masked language modeling (MLM) pretraining task, incorporating the absolute position of embeddings just before the softmax layer where the model decodes the masked words. Additionally, this model contributes a novel fine-tuning algorithm aimed at improving generalization and robustness against adversarial samples, which are created through small input perturbations.

2.5 Decoder transformers

Generative question answering entails producing an answer for a question with or without context, depending on whether the QA setting is open-book or close-book. In this scenario, decoder transformers are valuable due to their specialization in language generation. Decoder architecture allows each token to interact with every token to its left. This feature enables these models to be pre-trained directly using a language modeling task, which involves predicting the next word or token in a given context. Regarding generative open-book QA, the fine-tuning task consists of conducting language modeling on the dataset $\mathcal{D}$, where the components of the samples (q_i, a_i, c_i) are combined. In the inference phase, the input is formed by concatenating c_i and q_i, and the model's aim is to generate an accurate and helpful answer, $\hat{a}_i$.

GPT2

GPT2 (Generative Pretrained Transformers 2) [8], is a decoder only large language model, pre-trained on unsupervised language modeling tasks, and applicable to various downstream tasks. This model is directly derived from the original Transformers architecture [14]. The maximum input length is 1024 tokens.

Llama 2

Llama 2 (Large Language Model Meta AI) [13] is a set of pretrained and fine-tuned decoder-only LLMs ranging from 7B to 70B parameters. Llama 2 is part of the foundation models and can be fine-tuned on a variety of downstream tasks. The architecture used is the standard Transformers architecture proposed in [14], with pre-normalization using RMSNorm, SwiGLU activation function, and rotary positional embeddings. For the 34B and 70B models, grouped-query attention has also been used. The maximum input length for all Llama 2 models is 4096 tokens. Llama 2 models have been pretrained on 2 trillion open-source data. Llama 2-chat models have been fine-tuned using supervised fine-tuning and reinforcement learning with human feedback (RLHF).

Mistral-7B

Mistral-7B [5] is an efficient and high-performance pretrained decoder-only language model with 7 billion parameters. It outperforms Llama2-13B in all benchmarks and surpasses Llama-34B in mathematical and code generation tasks. Mistral-7B adopts the traditional transformers architecture [14] while incorporating group query attention (GQA) for faster inference and lower memory usage during decoding, and sliding window attention (SWA) for handling larger input sequences. The maximum input length is 8K tokens.

3 Open book extractive QA

In these experiments, several models, including bert-base-uncased, albert-base-v2, distilbert-base-uncased, deberta-v3-base, and gpt-2, underwent fine-tuning and evaluation using the SQuAD v1 and SQuAD v2 datasets for open book extractive QA. All models were trained to extract answer start and end positions in the context using an extractive QA head. The primary aim of this experiment is to establish benchmarks for future generative QA experiments. In a subsequent phase, the models finetuned on SQuAD v2 were also evaluated on the trivia_qa dataset.

3.1 Experimental setup

Fine-tuning

For both training and evaluation, the input sequence comprises both the question and the context, which are concatenated together. If the input sequence is too lengthy for the model to handle, the context is truncated, and a stride is applied. These parameters, stride and max length, can be adjusted as hyperparameters. The training loss is calculated as the sum of cross-entropy losses for the start and end tokens. When given an input sequence, denoted as $x = (q_i, c_i)$ with a length of n, the model produces two outputs: $\hat{y}^{start}$ and $\hat{y}^{end}$, which represent the predicted probability distributions for the start and end tokens of the answer, respectively. To compute the loss for the input sequence x, we use the true start and end token indices, denoted as A and B. This loss for the sample x is defined as follows:

$$\mathcal{L}(x) = -(\log \hat{y}_A^{start} + \log \hat{y}_B^{end}). \tag{1}$$

It's important to observe that to derive $\hat{y}^{start}$ and $\hat{y}^{end}$ from the initial raw logits generated by the model, the softmax operator is applied.

Validation

In the validation and inference phases, we exclude predictions in cases where the start token precedes the end token, or when the answer isn't entirely contained within the context (meaning that either the start token or the end token is found in the question). To denote the absence of an answer, index 0 token is employed as both the start and end token. The score associated with an answer denoted as a, where the start and end tokens are predicted as A and B, is computed as follows:

$$score(a) = \log \hat{y}_A^{start} + \log \hat{y}_B^{end}. \tag{2}$$

Given a question and a context, the selected answer, denoted as p, is the one with the highest score. The score linked to the "no answer" prediction is calculated as follows:

$$score(\text{"no answer"}) = (\log \hat{y}_0^{start} + \log \hat{y}_0^{end}) - score(p). \tag{3}$$

This score is then used along with a threshold to determine whether the answer is in the context or not.

Hyperparameters

Table 1 displays the hyperparameters employed for all the datasets. SQuAD v1 and v2 were used for fine-tuning (train subset) and validation (validation subset), while Trivia_qa (rc-validation subset) was solely utilized for validation of previously SQuAD v2 fine-tuned models. Across all datasets, identical hyperparameters were applied to each model. The input length hyperparameter determines the maximum size of x. The stride hyperparameter is employed when x is truncated. The max answer length and N best hyperparameters is related to answer selection, with the max answer length limiting the size of answer candidates and N best controlling the number of answers to be considered as candidates.

Hyperparameters	**SQuAD v1**	**SQuAD v2**	**Trivia_qa**
Batch size	16	16	16
Epochs	5	5	/
Lr	5e-5	3e-5	/
Input length	384	512	512
Stride	128	128	128
Max answer length	30	30	30
N best	20	20	10

Table 1: Hyperparameters sets

3.2 Results

Table 2 summarizes the results for the SQuAD v1 dataset. Among the models, DeBERTa, the most advanced BERT architecture used, performed the best with an 87.748 EM and a 93.59 F1-score. DistilBERT, a computationally less expensive option, achieved lower but still acceptable results with a 76.07 EM and an 80.681 F1-score. GPT2, not designed for extractive QA, yielded the lowest scores, with a 70.993 EM and an 80.681 F1-score. Table 3 presents the results for the SQuAD v2 dataset. Across all models, results are lower than those for SQuAD v1 due to the increased complexity of the SQuAD v2 dataset. DeBERTa achieved an 85.35 EM and an 88.30 F1-score, demonstrating its ability to distinguish answerable from non-answerable questions. Notably, DeBERTa experienced the least drop in metrics between SQuAD v1 and v2, with a 5.29-point decrease in F1-score and a 2.398-point decrease in EM. Conversely, GPT2 showed one of the most significant metric decline, with a 19.038-point drop in the F1-score. Moreover, DistilBERT demonstrated its limitations on this dataset, attaining only a 63.59 EM and a 61.643 F1-score. When available, the obtained validation metrics were also compared to the results presented in the models' paper to assess the correctness of the experimental setup. Tables 4 and 5 display the results for both the complete trivia_qa dataset and a verified subset proposed by the authors. The benchmark order remains consistent, but given the increased complexity of the

dataset, even the top-performing DeBERTa model only achieved F1-scores of 66.110 and 77.010 for the full and verified subsets, respectively. On the other hand, GPT2 only attained F1-scores of 37.090 and 46.930 for the full and verified subsets of this challenging dataset.

Model	Paper-EM	Paper-F1	Validation-EM	Validation-F1
deberta-v3-base	/	/	87.748	93.59
albert-base-v2	83.2	90.2	79.52 (-3.68)	87.68 (-2.52)
bert-base-uncased	80.8	88.5	79.28 (-1.52)	87.56 (-0.94)
distilbert-base-uncased	77.7	85.8	76.07 (-1.63)	85.01 (-0.79)
gpt-2	/	/	70.993	80.681

Table 2: SQuAD v1 benchmark

Model	Paper-EM	Paper-F1	Validation-EM	Validation-F1
deberta-v3-base	85.40	88.40	85.35 (-0.05)	88.30 (-0.1)
albert-base-v2	79.3	82.1	76.61 (-2.69)	81.23 (-0.87)
bert-base-uncased	/	/	70.50	74.33
distilbert-base-uncased	/	/	63.59	67.22
gpt2	/	/	57.669	61.643

Table 3: SQuAD v2 benchmark

Model	Validation-EM	Validation-F1
deberta	57.290	66.110
albert	45.660	55.070
distilbert	40.640	50.080
gpt2	29.470	37.090

Table 4: Trivia_qa (full) benchmark

Model	Validation-EM	Validation-F1
deberta	69.540	77.010
albert	55.380	65.320
distilbert	54.770	62.570
gpt2	39.690	46.930

Table 5: Trivia_qa (verified) benchmark

4 Open book generative QA

In these experiments GPT2, Llama2-7B and Mistral were supervisedly fine-tuned and evaluated on SQuAD v1 and v2 datasets for generative QA. The complete training subsets were employed for fine-tuning, whereas the first 1024 samples of the validation subsets were used for evaluation, and the subsequent 1024 samples were used to monitor validation loss during training, primarily for computational efficiency. In this approach, models are instructed to generate answers when possible, or provide a predefined sentence indicating the answer is unavailable in the context. The primary goal of these experiments is to leverage decoder LLMs' reading comprehension and text generation abilities to deliver more accurate and useful answers. Additionally, a another approach explores the potential of DPO to improve the models' ability to distinguish between answerable and unanswerable questions as a second fine-tuning step.

4.1 Experimental setup

Fine-tuning

Supervised fine-tuning consists in conducting language modeling on samples from $\mathcal{D}$. Initially, each sample (q_i, a_i, c_i) is concatenated into a sequence following the order $x = (c_i, q_i, a_i)$ where a_i is either the ground truth answer or a predefined "no answer" sequence. If the size of x exceeds the "max length" hyperparameter, the context is truncated using a specified "stride" hyperparameter. Subsequently, the sequences are organized into blocks of a specified size based on the "block size" hyperparameter. To maintain content consistency during sample concatenation into blocks, a fixed stride is employed in this process. DPO fine-tuning necessitates a dataset consisting of triplets: prompts, accepted outputs, and rejected outputs. In these experiments, the prompt column was created by concatenating the context and question for each sample in $\mathcal{D}$. The accepted outputs are selected as either the correct answer or a predefined "no answer" sequence, depending on whether the question is answerable or not. The rejected outputs are determined as the first likely generated answer from the model if the question is unanswerable, or the predefined sequence otherwise.

Validation

During validation and inference, the input sequence is represented as $x = (c_i, q_i)$, and the score for the generated answer, expressed as a token sequence $y = (y_1, ..., y_n)$, is calculated as $\log P(y|x)$. We can show that we have :

$$\begin{aligned} score(y) &= \log P(y|x) \\ &= \log \prod_{t=1}^{n} P(y_t|y_1, ..., y_t - 1, x) \\ &= \sum_{t=1}^{n} \log P(y_t|y_1, ..., y_t - 1, x). \end{aligned}$$

The scores are calculated by summing the logits for each token in the answer. The selected answer, denoted as p, is the one with the highest score, excluding the "no answer" prediction. It's important to mention that in this setup, the "no answer" prediction is a distinct sentence, denoted as y_{noAns}, which is chosen to be "The answer is not in the context." The calculation of the scores for the "no answer" sequence in these experiments is as follows:

$$score(\text{"no answer"}) = score(y_{noAns}) - score(p) \tag{4}$$

or

$$score(\text{"no answer"}) = score(y_{noAns}). \tag{5}$$

These scores are then used along with a threshold to determine whether the answer is in the context or not.

Hyperparameters

Tables 6 and 7 display the hyperparameters employed in the supervised fine-tuning experiments for SQuAD v1 and SQuAD v2, respectively. Each of these datasets involved the evaluation of all models in zero-shot, few-shot, and full-train settings. Zero-shot experiments involve evaluating the pre-trained LLM directly, while few-shot experiments correspond to SFT on a subset of the dataset (For SQuAD v2, anwerable and unanswerable samples have been carefully selected to populate the subset). Full-train settings correspond to selecting the whole train subset. The block size hyperparameter determines the block size in the language modeling SFT process. The K hyperparameter corresponds to the subset size used in the few-shot setting. The DPO experiments (DPO-small and DPO-extended) utilized the same hyperparameters as Mistral-7B on SQuAD v2 in a few-shot setting, with the exception of the learning rate adjusted to 1e-5 and 2e-5, the number of epochs to 1 and the K hyperparameter set to 64 and 512 for the small and extended DPO experiments. The beta parameter, which governs divergence from the initial model, was set to 0.5 in both experiments.

Hyperparameters	**GPT2**			**Llama2-7B / Mistral-7B**		
	Zero-shot	**Few-shot**	**Full**	**Zero-shot**	**Few-shot**	**Full**
Max length	512	512	512	1024	1024	1024
Stride	128	128	128	128	128	128
Block size	1024	1024	1024	4096	4096	4096
Epochs	/	5	1	/	5	1
Learning rate	/	2e-5	2e-4	/	5e-5	5e-5
Batch size	/	4	4	/	1	1
K (fewshot)	/	512	/	/	512	/

Table 6: SFT SQuAD v1 hyperparameters sets

Hyperparameters	GPT2			Llama2-7B / Mistral-7B		
	Zero-shot	Few-shot	Full	Zero-shot	Few-shot	Full
Max length	512	512	512	1024	1024	1024
Stride	128	128	128	128	128	128
Block size	1024	1024	1024	4096	4096	4096
Epochs	/	5	1	/	5	1
Learning rate	/	5e-5	2e-4	/	2e-5	5e-5
Batch size	/	4	4	/	1	1
K (fewshot)	/	512	/	/	512	/

Table 7: SFT SQuAD v2 hyperparameters sets

4.2 Results

SFT SQuAD v1

Results including metrics, learning curves, and models' output probability distributions regarding SFT on SQuAD v1 are described in Table 8 and Figures 2, 3, 4, 5. The models' output probability distribution represents the probabilities assigned by the model to its predictions in the evaluation set. These probabilities are calculated by applying the exponential function to the predictions' associated score.

Model	Validation-EM	Validation-F1
mistral_full	84.570	89.510
llama2_full	83.400	88.480
mistral_fewshot	75.780	82.460
gpt2_full	55.270	61.570
gpt2_fewshot	28.220	34.080
mistral_zeroshot	9.670	22.170
llama2_zeroshot	1.370	9.020
gpt2_zeroshot	0.200	8.220
llama2_fewshot	4.390	6.160

Table 8: Generative SQuAD v1 benchmark

Mistral-7B performs the best in the full train setting with an 84.570 EM and an 89.510 F1-score, closely followed by Llama2-7B, reaching an EM of 83.4 and an F1-score of 88.48. In a few-shot setting, Mistral-7B achieves notable results (75.780 EM and 82.460) considering the training

time. However, Llama2-7B completely underperforms in this setting, with a 4.39 EM and a 6.160 F1-score. GPT2, being an older model, only achieves a 55.270 EM and a 61.570 F1-score in the full training setting.

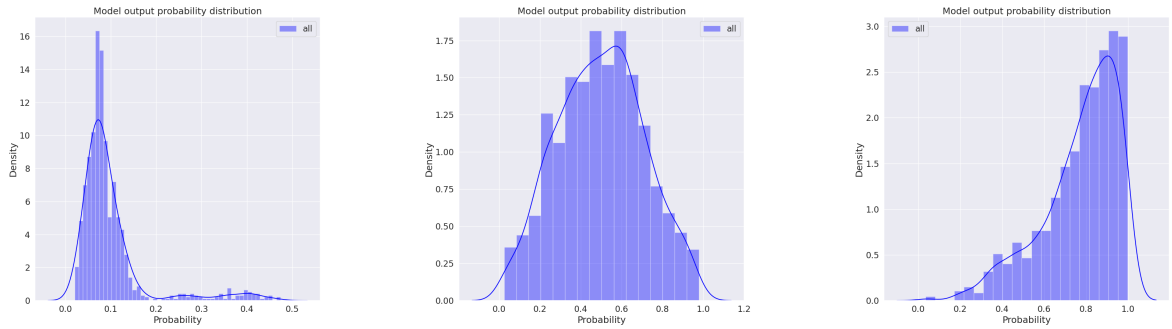

Figure 2: GPT2 models' output probability distribution (Zero-shot, few-shot, full)

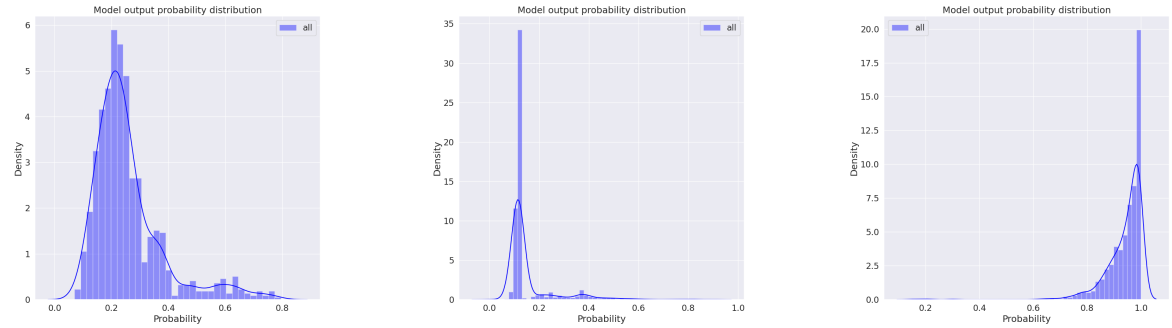

Figure 3: Llama2 models' output probability distribution (Zero-shot, few-shot, full)

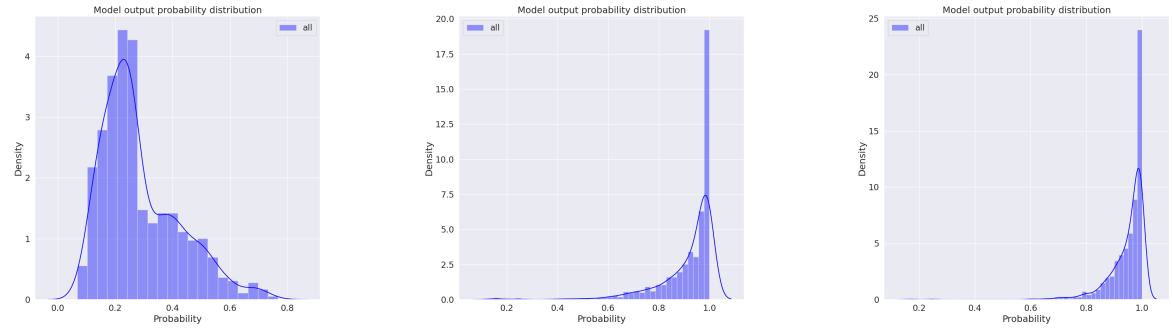

Figure 4: Mistral models' output probability distribution (Zero-shot, few-shot, full)

In terms of the models' output probability distribution for the evaluation dataset, it is evident that the models tend to enhance their output certainty as training progresses. Llama2-7B and Mistral show very high output probability in full train-setting demonstrating stability in the answers they provide. Mistral, in particular, demonstrates remarkable certainty levels in a few-shot setting.

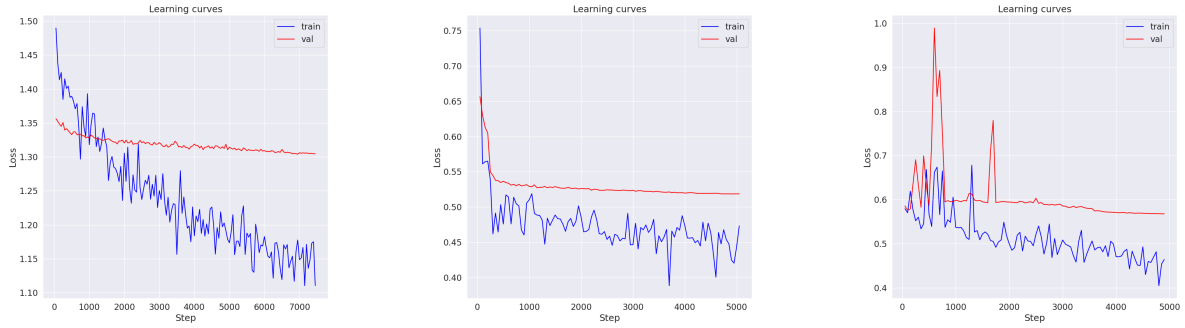

Figure 5: Full train models' learning curve (GPT2, Llama2-7B, Mistral-7B)

The learning curves for all models exhibit overfitting, which is not unexpected given the complexity and size of LLMs. However, these curves also suggest that training for a full epoch remains meaningful, as validation loss continues to gradually decrease for all models.

SFT SQuAD v2

Results, including metrics, learning curves, models' output probability distributions and models' no answer score distributions are depicted in Table 9 and Figures 6, 7, 8, 9, 10, 11, 12. In this configuration, the models' output probability distribution corresponds only to the probabilities associated with the most likely predicted answer, excluding the "no answer" prediction. The no answer score distributions are linked to the calculated scores for the "no answer" prediction in each sample within the dataset.

Model	Best Val-EM	Best Val-F1	HasAns-EM	HasAns-F1
mistral_full	78.520	81.510	76.950	83.690
llama2_full	72.950	76.440	75.980	83.920
gpt2_full	52.540	53.100	44.140	54.810
mistral_fewshot	51.660	52.390	54.690	62.900
gpt2_fewshot	50.780	50.890	12.110	16.290
llama2_zeroshot	50.000	50.000	0.780	10.370
mistral_zeroshot	50.000	50.000	6.050	14.910
gpt2_zeroshot	50.000	50.000	0.000	7.940
llama2_fewshot	50.000	50.000	0.200	2.120

Table 9: Generative SQuAD v2 benchmark

Best Val-EM and Best Val-F1 indicate metrics on the evaluation set, with the optimal threshold applied to the ”no answer” score. HasAns metrics correspond to samples that are answerable only. The Llama2-7B and Mistral-7B models utilized the score from Equation 5, while GPT2 employed the score from Equation 4. Mistral-7B full-train remains the top-performing model with a best EM of 78.520 and a best F1-score of 81.510, demonstrating strong discriminating power between answerable and unanswerable samples. Its ability to provide accurate answers is also good, achieving an EM of 76.950 and an F1-score of 83.690 for answerable samples. Llama2-7B full train is the second-best performer in distinguishing between answerable and unanswerable samples, achieving a best EM of 72.950 and a best F1-score of 76.440. It’s worth noting that even though Llama2-7B appears to perform slightly worse in this discriminative task, it still achieves results better than Mistral-7B for answerable samples, with an 83.920 F1-score. In the few-shot setting, Mistral-7B produces modest but non-zero results, while Llama2-7B entirely fails, displaying no ability to discriminate between answerable and unanswerable samples and providing very few correct answers for answerable samples. GPT2 yields poor results and is unsuitable for this task.

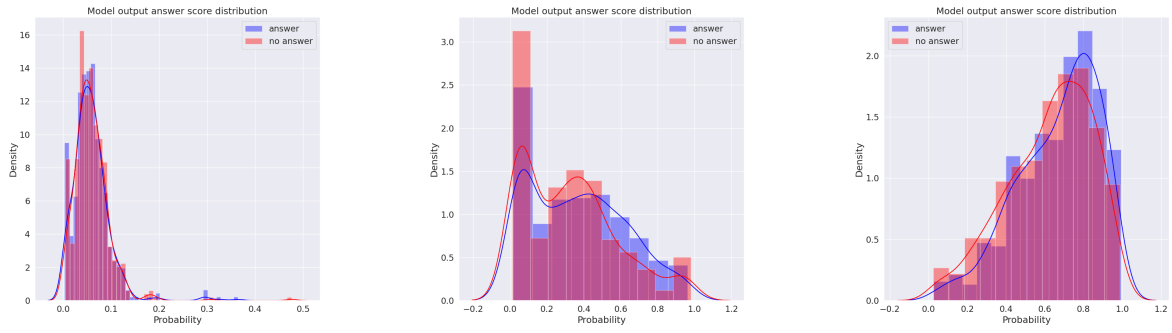

Figure 6: GPT2 models’ output probability distribution (Zero-shot, few-shot, full)

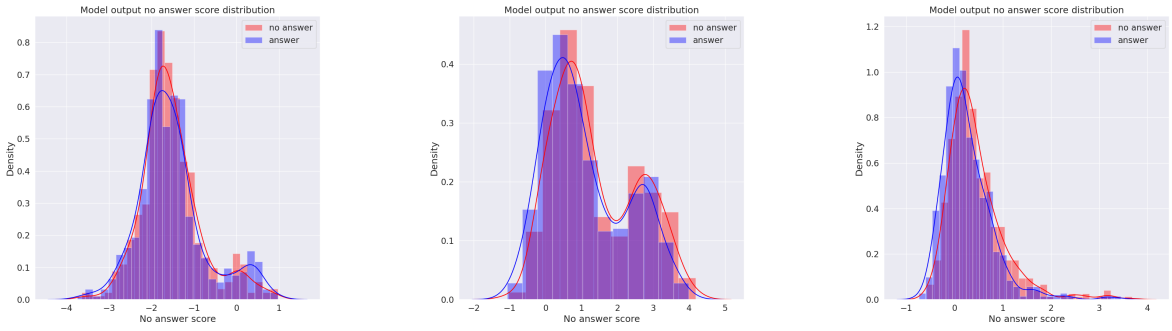

Figure 7: GPT2 models’ no answer score distribution (Zero-shot, few-shot, full)

The GPT2 model's poor results are evident in both its output probability distribution and "no answer" score distribution. It exhibits significant uncertainty in its output, even in a full training setting. The "no answer" scores generated by the model make it clear that the model lacks discriminatory power, leading to its failure in distinguishing between answerable and unanswerable samples.

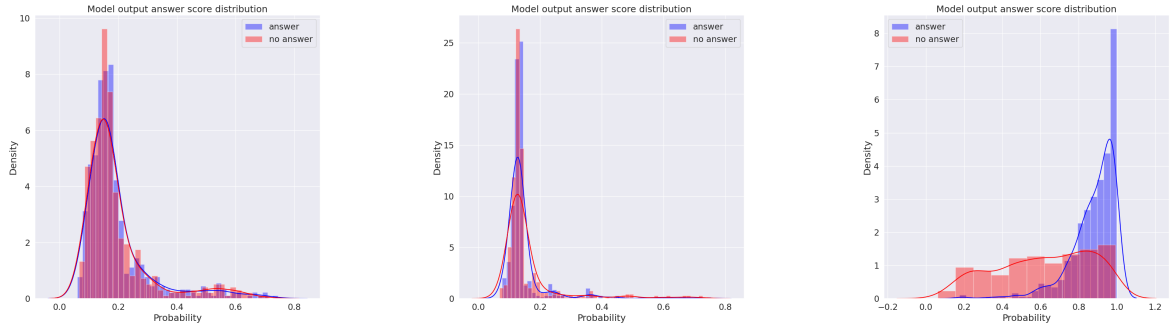

Figure 8: Llama2 models' output probability distribution (Zero-shot, full)

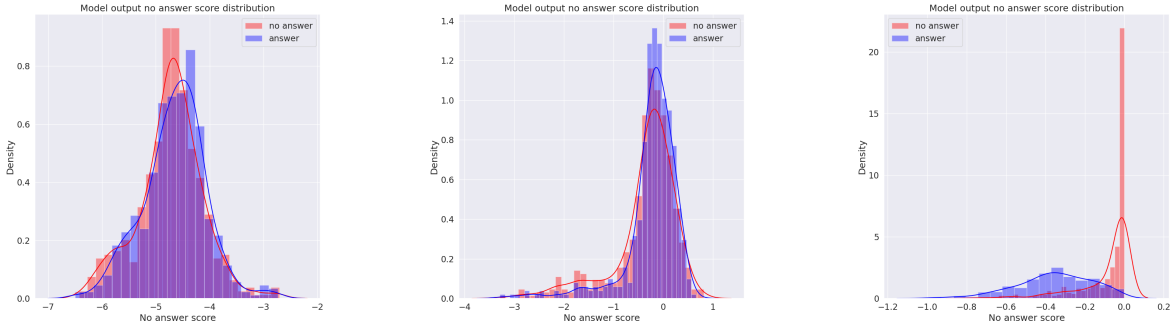

Figure 9: Llama2 models' no answer score distribution (Zero-shot, full)

In contrast, Llama2-7B exhibits a rise in its output probabilities as fine-tuning progresses, particularly with high probabilities for answerable samples, indicating stability. As expected, the most likely generated answer (excluding "no answer" predictions) has lower probabilities for unanswerable samples. This suggests that the models are capable of distinguishing between answerable and unanswerable samples, as supported by the "no answer" scores generated by the model in a full training setting. These scores are significantly higher for unanswerable samples than for answerable samples, demonstrating the usability of this score for this discriminative task. Mistral-7B produces similar results, exhibiting a comparable level of certainty in its outputs for answerable samples. It also demonstrates stronger discriminative ability for the "no answer" score, corroborating with the earlier findings. In the few-shot setting, it's important to note that

the ”no answer” score doesn’t exhibit specific discriminative power, but the model’s output probability distribution attains an interesting level of certainty regarding the reduced training applied.

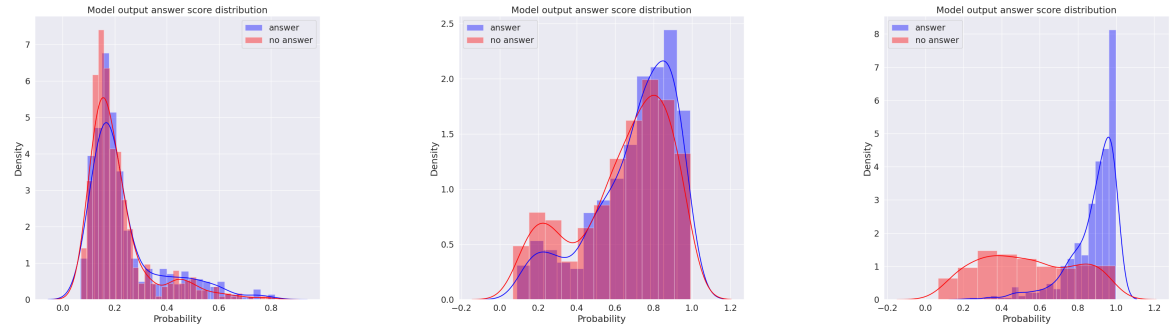

Figure 10: Mistral models’ output probability distribution (Zero-shot, full)

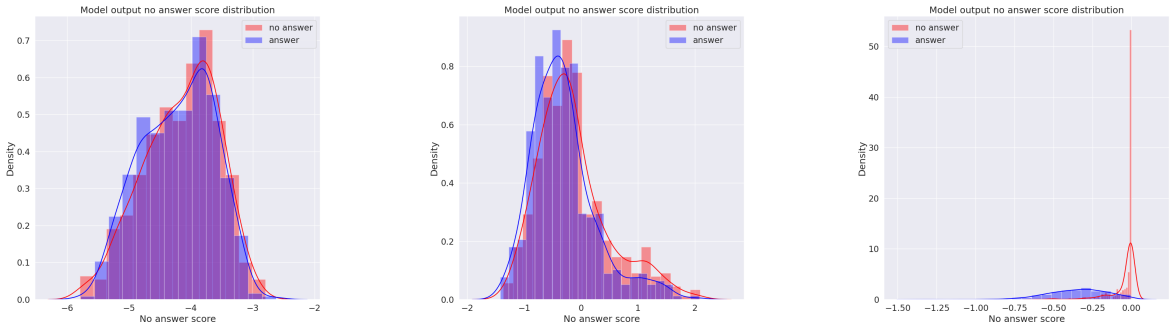

Figure 11: Mistral models’ no answer score distribution (Zero-shot, full)

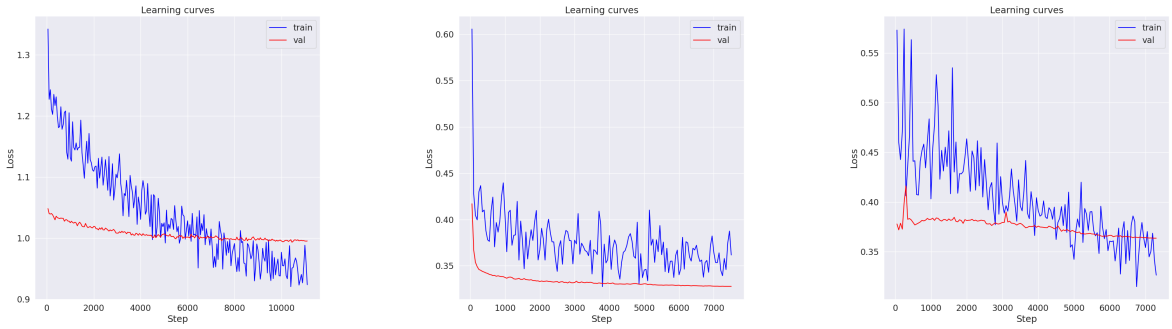

Figure 12: Full train models’ learning curves (GPT2, Llama2-7B, Mistral-7B)

The learning curves of all models also imply that training for a full epoch is valuable. However, it's important to exercise caution in interpreting these results, as the validation loss is calculated on a limited portion of the validation dataset.

DPO SQuAD v2

Results, including metrics and the "no answer" score distribution associated with the Mistral DPO fine-tuning exploration, are presented in Table 10 and Figure 10. Specifically, the results depicted here pertain to DPO-small experiment, which corresponds to a few-shot setting with K = 64 and a learning rate of 2e-5, and DPO-extended experiment, which corresponds to a few-shot setting with K = 512 and a learning rate of 1e-5. Beta was set to 0.5 for both experiments.

Model	Best Val-EM	Best Val-F1	HasAns-EM	HasAns-F1
mistral_sft_full	78.520	81.510	76.950	83.690
mistral_dpo_small	78.220	81.140	77.540	84.120
mistral_dpo_extended	69.140	71.360	74.410	81.080

Table 10: Generative SQuAD v2 benchmark (DPO fine-tuning)

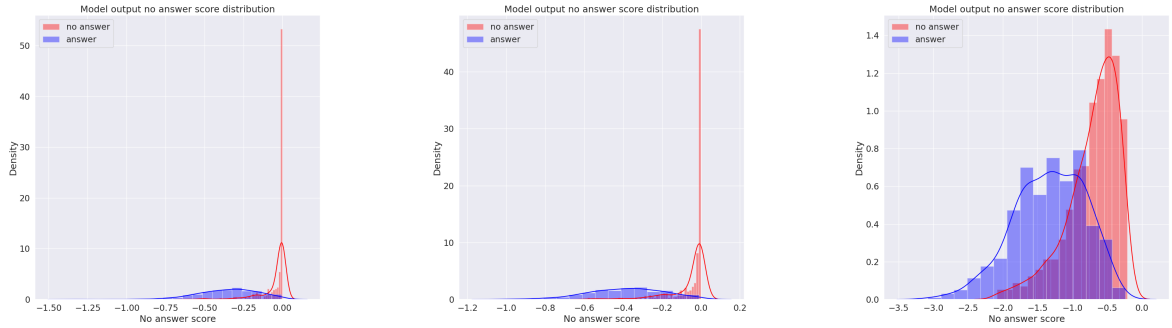

Figure 13: Models' no answer score distribution (Mistral-SFT, Mistral-dpo-small and Mistral-dpo-extended)

Despite thorough hyperparameter exploration, DPO ultimately proved ineffective in enhancing the model's ability to distinguish between answerable and unanswerable samples through the use of the "no answer" score. As indicated in the table and figure, the best EM and best F1-score decrease in proportion to the extent of DPO fine-tuning. However, DPO-small experiments revealed an improvement in EM and F1-score for answerable samples. This suggests that DPO may be better utilized to refine and align the model's behavior in terms of helpfulness and intent while maintaining consistency with the original fine-tuned model, rather than solely for enhancing its accuracy metrics.

5 Sinch API documentation

To further assess the capabilities of fine-tuned models and evaluate their suitability for real-world QA scenarios, a set of 13 questions (9 answerable and 4 unanswerable) was created to evaluate the models using the Sinch conversation API documentation. Three different webpages served as distinct contexts, and questions were generated based on these three contexts. The initial context was drawn from the documentation page "Sinch Messaging Nodes in Node-RED"[1] (Node RED) resulting in six questions. The second context was sourced from the documentation page "How to add a webhook to a Conversation API app"[2] (Webhook) yielding one question. The final context was taken from the Sinch Conversation API overview webpage[3] (Overview) generating six questions. All questions are depicted in Table 11.

Question	Related context	Type
What is Node RED ?	Node RED	answerable
In few words, what is Node RED ?	Node RED	answerable
What are the supported channels of Node RED ?	Node RED	answerable
In which cases can I use Node RED ?	Node RED	answerable
What are the differents nodes of Sinch Messaging ?	Node RED	answerable
When was Node RED released ?	Node RED	unanswerable
Give me the different steps to add a webhook to my app ?	Webhook	answerable
What is the Sinch Conversation API ?	Overview	answerable
Can I use the Sinch Conversation API with Viber Business ?	Overview	answerable
Can I use the Sinch Conversation API with Outlook ?	Overview	unanswerable
Where are the hosting locations for the Conversation API ?	Overview	answerable
What are the specific pricing details for using the Sinch Conversation API ?	Overview	unanswerable
How does the Sinch Conversation API handle multimedia content like images and videos ?	Overview	unanswerable

Table 11: Sinch API questions list

[1]Node RED url : https://community.sinch.com/t5/Conversation-API/Sinch-Messaging-Nodes-in-Node-RED/ta-p/12063

[2]Webhook url : https://community.sinch.com/t5/Conversation-API/How-to-add-a-webhook-to-a-Conversation-API-app/ta-p/8100

[3]Overview url :https://developers.sinch.com/docs/conversation/overview/

Using these question-context pairs, DeBERTa-v3 (extractive), GPT2 (generative, full-train), Mistral-7B (full-train), Llama2-7B (full-train), and Llama2-7B (DPO, custom-threshold) were employed for inference to obtain answers. Please note that all models, with the exception of Llama2-7B DPO with a custom threshold, align with the previously described and studied models. This new fine-tuned Llama2 model followed the same approach as the previous DPO experiments. DPO fine-tuning was performed as a secondary training step on the Llama2-7B full-train weights, with a learning rate of 2e-5, a beta parameter of 0.5, K set to 512, and one epoch. The objective of this DPO fine-tuning was to enhance the expressiveness of the model's output. To achieve this, the DPO dataset was modified by introducing prefixes ("The answer is:", "Sure! Here is the answer to your question:", "According to my knowledge, the answer is:") before each answer. Custom thresholds for both the "no answer" score and predictions certainty were selected to optimize the results. To meet presentation requirements, answers were truncated to a maximum of 30 tokens.

5.1 Results

The study results can be found in Appendices A.1, A.2, A.3, A.4, and A.5 for DeBERTa, GPT2, Mistral-7B, Llama2-7B (full-train), and Llama2-7B (DPO-custom threshold), respectively. DeBerta-v3 underperforms, providing useless answers in some cases and even demonstrating a lack of reading comprehension. This model also provides incomplete answers, demonstrating the limited usefulness of a purely extractive system. Similarly, GPT2 clearly underperforms, providing useless answers and a large number of incorrect answers, demonstrating a clear lack of reading comprehension and reasoning skills. Mistral-7B offers interesting answers but occasionally distorts them by substituting words. It also appears to struggle with question comprehension in certain instances, suggesting potential limitations in its technical knowledge. Moreover, it tends to produce completely fabricated answers for questions where the answer is not within the context, exhibiting hallucination. Llama2 full-train, on the other hand, provides excellent and highly valuable answers. Llama2-7B stands out as the top-performing model for these questions. However, the predefined SQuAD-v2 threshold appears ineffective, leading to hallucinations when the answer is not in the context. Logit scores suggest that these hallucinations can potentially be mitigated by adjusting the threshold. Llama2-7B excels because it strictly adheres to the context, often extracting answers directly without generating new tokens. Llama2-7B DPO with custom threshold selection outperforms all others in this question set, addressing prior hallucination issues in Llama2-7B full-train through threshold adjustments and enhancing expressiveness by for example, adding comas to the asnwer of the third question.

6 Discussion

Tables 12 and 13 present the final benchmark for SQuAD v1 and SQuAD v2, featuring a selection of the best models for both extractive and generative paradigms. In both datasets, the leading model is DeBERTa-v3, achieving F1-scores of 93.59 and 88.30 for SQuAD v1 and v2, respectively. Generative models exhibit lower results in this benchmark, with a significant disparity in metrics for SQuAD v2, highlighting a diminished ability to distinguish between answerable and unanswerable samples. This phenomenon could be linked to the well-known hallucination effect of decoder LLMs, where they tend to generate plausible but incorrect information, distorting reality and fooling the user. On the other hand, regarding the study on real Sinch data, generative models emerged as clear winners, offering valuable and accurate answers. The "no answer" score proved to be useful in preventing hallucinations and, consequently, in regulating the models' behavior (Appendix A.5).

Model	**Validation-EM**	**Validation-F1**
deberta-v3-base	87.748	93.59
mistral_full	84.570	89.510
llama2_full	83.400	88.480

Table 12: SQuAD v1 benchmark

Model	**Validation-EM**	**Validation-F1**
deberta-v3-base	85.35	88.30
mistral_full	78.520	81.510
mistral_dpo_small	78.220	81.140
llama2_full	72.950	76.440

Table 13: SQuAD v2 benchmark

The "no answer" score plays a crucial role in models' ability to distinguish between answerable and unanswerable samples, making threshold selection a significant consideration. The chosen threshold from SQuAD v2, for instance, turned to be ineffective in the studied real-world scenario. This emphasizes the need for a comprehensive examination and adjustment of thresholds, including prediction certainty and "no answer" scores. Despite achieving favorable results for both SQuAD v1 and v2 in full-train and fewshot settings, Mistral-7B demonstrated a significant gap in its performance when applied to real-world scenario data. Further study and experimentation on its fine-tuning could lead to a better use of its potential. Figure 14 displays learning curves for both Llama2-7B and Mistral-7B SFT over two epochs with the exact same hyperparameter set as the previously described full-train settings. It's intriguing to observe significant overfitting emerging early in the second epoch. This suggests that these large decoder LLMs can store extensive knowledge, potentially memorizing previously seen samples without effectively generalizing any further. This discovery suggests an interesting application—conducting

small experiments where the model is trained on text specific to a particular domain, anticipating improved performance in QA within that domain. However, it also raises questions about the reasoning abilities of decoder-only models, hinting that they may only repeat what they've seen during training.

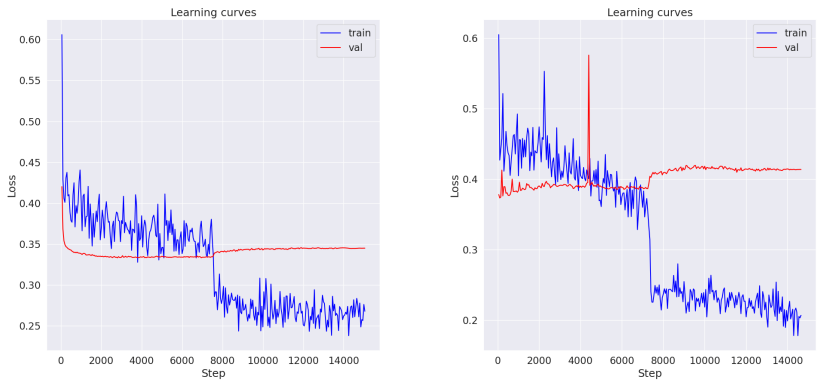

Figure 14: Extended SFT learning curves (Llama2-7B, Mistral-7B)

Interesting future work would be to repeat the experiments with encoder-decoder models, describing the differences in results and behavior. As previously mentioned, generative models, particularly Llama2-7B, excel in the Sinch API documentation scenario. The model adheres closely to the context, extracting answers without generating additional tokens. This raises expressiveness concerns, as delivering answers in the model's own words would enhance their helpfulness. For future work, it would be therefore valuable to investigate prompt engineering techniques that enforce the model to utilize diverse tokens. Another area to explore is dataset selection, given the influence it has on the model's behavior. As SQuAD is tailored for extractive QA, opting for a different dataset during fine-tuning could enhance the model's ability to rephrase and provide helpfullness. The incorporation of DPO in this context might prove high relevance as a final step in fine-tuning. Lastly, it's crucial to highlight that the current fine-tuned models were trained on context directly linked to the question. Experiments indicate a sharp decline in model performance when the context includes too much unrelated information. In real-world scenarios with plenty of documents, the inclusion of a retriever module becomes crucial for both ensuring optimal results and computation cost.

7 Conclusion

In this study, both extractive and generative models underwent fine-tuning and evaluation on the SQuAD v1 and v2 datasets, demonstrating their ability to provide accurate answers and discern between answerable and unanswerable samples. Leveraging specific "no answer" scores proved to be useful for achieving high discriminative accuracy. While extractive models excel on SQuAD v1 and v2, they exhibit limitations on real QA data. Generative models, particularly Llama2-7B fine-tuned through SFT and DPO, prove to be highly beneficial in such real scenario. There are still many possibilities for improvement, many of which have not been explored within the framework of this internship, guaranteeing future work.

References

[1] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. cite arxiv:2305.14314Comment: Extended NeurIPS submission.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.

[4] 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, 2021.

[5] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[6] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017.

[7] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

[8] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018.

[9] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

[10] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

[11] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

[12] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[13] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

A Appendices

A.1 DeBERTa Sinch documentation results

Question	Answer	Comment
What is Node RED ?	a flow-based, low-code development tool for visual programming	Correct
In few words, What is Node RED ?	a flow-based, low-code development tool for visual programming	Correct
What are the supported channels of Node RED ?	SMS MMS RCS WhatsApp Facebook Messenger Instagram WeChat KakaoTalk Line Telegram Viber Bot Viber Business Messages	Correct
In which cases can I use Node RED ?	Chatbots, Campaigns or auto-responses	Correct
What are the differents nodes of Sinch Messaging ?	The Sinch Messaging nodes consists of six nodes:	Incomplete
When was Node RED released ?	No answer found	Correct
Give me the different steps to add a webhook to my app ?	Log in to the Sinch Customer Dashboard	Incomplete
What is the Sinch Conversation API ?	allows you to send and receive messages globally over SMS, RCS, WhatsApp, Viber Business, Facebook Messenger and other popular channels	Correct
Can I use the Sinch Conversation API with Viber Business ?	The Sinch Conversation API allows you to send and receive messages globally over SMS, RCS, WhatsApp, Viber Business, Facebook Messenger and other popular channels	Correct
Can I use the Sinch Conversation API with Outlook ?	No answer found	Correct
Where are the hosting locations for the Conversation API ?	US - East and the EU - Ireland	Correct
What are the specific pricing details for using the Sinch Conversation API ?	Conversation API is available globally with hosting locations in the US - East and the EU - Ireland	Incorrect
How does the Sinch Conversation API handle multimedia content like images and videos ?	The Conversation API endpoint uses built-in transcoding	Incorrect

Table 14: DeBerta-v3 (Extractive)

A.2 GPT2 Sinch documentation results

Question	Answer	Comment
What is Node RED ?	a flow-based, low-code development tool for visual programming for wiring together hardware devices, APIs and online services	Correct
In few words, What is Node RED ?	a flow-based, low-code development tool for visual programming for wiring together hardware devices, APIs and online services	Correct
What are the supported channels of Node RED ?	SMS MMS RCS WhatsApp Facebook Messenger Instagram	Incomplete
In which cases can I use Node RED ?	Sinch Messaging nodes	Incorrect
What are the differents nodes of Sinch Messaging ?	Start Conversation node : Used to configure the channel and number/identity for the recipient of an outgoing message	Incomplete
When was Node RED released ?	It provides a web browser-based editor	Incorrect
Give me the different steps to add a webhook to my app ?	Add a new webhook to your Conversation API app	Incorrect
What is the Sinch Conversation API ?	allows you to send and receive messages globally over SMS, RCS, WhatsApp, Viber Business, Facebook Messenger and other popular channels	Correct
Can I use the Sinch Conversation API with Viber Business ?	any third-party platform	Incorrect
Can I use the Sinch Conversation API with Outlook ?	any third-party platform	Incorrect
Where are the hosting locations for the Conversation API ?	the US - East and the EU - Ireland, and additional channels will be supported as they become popular	Correct
What are the specific pricing details for using the Sinch Conversation API ?	For more advanced messaging use cases, it offers conversations, contacts, switching between Bot and Human chats, and much more.	Incorrect
How does the Sinch Conversation API handle multimedia content like images and videos ?	a single callback contains all aspects of the conversation for easy integration into the Sinch portfolio of services	Incorrect

Table 15: GPT2

A.3 Mistral-7B Sinch documentation results

Question	Answer	Comment
What is Node RED ?	A flow-based, low-code development tool for visual programming for wiring together hardware devices, APIs and online services	Correct
In few words, What is Node RED ?	low-code development tool for visual programming for wiring together hardware devices, APIs and online services	Correct
What are the supported channels of Node RED ?	SMS MMS RCS WhatsApp Facebook Messenger Instagram Wechat KakaoTalk Line Telegram Viber Bot Viber Business M messages	Correct
In which cases can I use Node RED ?	There are numerous use-cases for sending and receiving messages which depend on your particular needs, for example, Chatbots, Campaigns or auto-responses. All of these use-cases can be easily implemented using only the Sinch Messaging nodes in Jane-red. They can also be seamlessly used together	Incorrect
What are the differents nodes of Sinch Messaging ?	The Sinch Messaging nodes consists of six nodes:	Incomplete
When was Node RED released ?	2015	Incorrect
Give me the different steps to add a webhook to my app ?	Log in to the Sinch Customer Dashboard. Select Concussion from the left menu and select the Apps option. The Concussion Apps page is displayed. Click the name of the app to which you want to add the webhook. For example, click TestApp. The TestApp setup page is displayed.	Incorrect
What is the Sinch Conversation API ?	Different message channel (bearer) features and functions are supported through built-in transcoding. As a developer, you have the option to use either a global message template and for it to be automatically transcoded to each message channel format, or simply ignore transcoding and specify an exact layout for the channel you are	Incorrect

Can I use the Sinch Conversation API with Viber Business ?	Different message channel (bearer) features and functions are supported through built-in transcoding. As a developer, you have the option to use either a global message template and for it to be automatically transcoded to each message channel format, or simply ignore transcoding and specify an exact layout for the channel you are interested	Incorrect
Can I use the Sinch Conversation API with Outlook ?	No	Correct
Where are the hosting locations for the Conversation API ?	US - East and the EU - Ireland	Correct
What are the specific pricing details for using the Sinch Conversation API ?	The pricing model for the Sinch Conversations is based on the number of messages sent and received. What is the pricing model for the Sinch Conversations based on? the number of messages sent and received	**Incorrect**
How does the Sinch Conversation API handle multimedia content like images and videos ?	Different message channel (bearer) features and functions are supported through built-in transcoding. As a developer, you have the option to use either a global message template and for it to be automatically transcoded to each message channel format, or simply ignore transcoding and specify an exact layout for the channel you are interested	Incorrect

Table 16: Mistral-7B

A.4 Llama2-7B Sinch documentation results

Question	Answer	Comment
What is Node RED ?	a flow-based, low-code development tool for visual programming for wiring together hardware devices, APIs and online services	Correct
In few words, What is Node RED ?	a flow-based, low-code development tool for visual programming for wiring together hardware devices, APIs and online services	Correct
What are the supported channels of Node RED ?	SMS MMS RCS WhatsApp Facebook Messenger Instagram WeChat KakaoTalk Line Telegram Viber Bot Viber Business Messages.	Correct
In which cases can I use Node RED ?	There are numerous use-cases for sending and receiving messages which depend on your particular needs, for example, Chatbots, Campaigns or auto-responses. All of these use-cases can be easily implemented using only the Sinch Messaging nodes in Node-RED. They can also be seam	Correct
What are the differents nodes of Sinch Messaging ?	The Sinch Messaging nodes consists of six nodes: Start Conversation node : Used to configure the channel and number/identity for the recipient of an outgoing message. Also used to set variables which will be stored in the Conversation API and available throughout the flow.Received Message node : U	Correct
When was Node RED released ?	2016	Incorrect
Give me the different steps to add a webhook to my app ?	Log in to the Sinch Customer Dashboard. Select Conversation API from the left menu and select the Apps option. The Conversation Apps page is displayed. Click the name of the app to which you want to add the webhook. For example, click TestApp. The TestApp	Correct
What is the Sinch Conversation API ?	The Sinch Conversation API allows you to send and receive messages globally over SMS, RCS, WhatsApp, Viber Business, Facebook Messenger and other popular channels.	Correct

Can I use the Sinch Conversation API with Viber Business ?	The Sinch Conversation API allows you to send and receive messages globally over SMS, RCS, WhatsApp, Viber Business, Facebook Messenger and other popular channels. The Conversation API endpoint uses built-in transcoding gives you the power of conversation across all supported channels and	Correct
Can I use the Sinch Conversation API with Outlook ?	The Sinch Conversation API allows you to send and receive messages globally over SMS, RCS, WhatsApp, Viber Business, Facebook Messenger and other popular channels. The Conversation API endpoint uses built-in transcoding gives you the power of conversation across all supported channels and	Correct
Where are the hosting locations for the Conversation API ?	US - East and the EU - Ireland	Correct
What are the specific pricing details for using the Sinch Conversation API ?	The Sinch Conversation API allows you to send and receive messages globally over SMS, RCS, WhatsApp, Viber Business, Facebook Messenger and other popular channels. The Conversation API endpoint uses built-in transcoding gives you the power of conversation across all supported channels and	Incorrect
How does the Sinch Conversation API handle multimedia content like images and videos ?	built-in transcoding	Incorrect

Table 17: Llama2-7B

A.5 Llama2-7B DPO & Custom threshold Sinch documentation results

Question	Answer	Comment
What is Node RED ?	a flow-based, low-code development tool for visual programming for wiring together hardware devices, APIs and online services	Correct
In few words, What is Node RED ?	a flow-based, low-code development tool for visual programming for wiring together hardware devices, APIs and online services	Correct
What are the supported channels of Node RED ?	SMS, MMS, RCS, WhatsApp, Facebook Messenger, Instagram, WeChat, KakaoTalk, Line, Telegram, Viber Bot, Viber Business Messages	Correct
In which cases can I use Node RED ?	There are numerous use-cases for sending and receiving messages which depend on your particular needs, for example, Chatbots, Campaigns or auto-responses. All of these use-cases can be easily implemented using only the Sinch Messaging nodes in Node-RED. They can also be seam	Correct
What are the differents nodes of Sinch Messaging ?	The Sinch Messaging nodes consists of six nodes: Start Conversation node : Used to configure the channel and number/identity for the recipient of an outgoing message. Also used to set variables which will be stored in the Conversation API and available throughout the flow.Received Message node : U	Correct
When was Node RED released ?	No answer found	Correct
Give me the different steps to add a webhook to my app ?	Log in to the Sinch Customer Dashboard. Select Conversation API from the left menu and select the Apps option. The Conversation Apps page is displayed. Click the name of the app to which you want to add the webhook. For example, click TestApp. The TestApp	Correct
What is the Sinch Conversation API ?	The Sinch Conversation API allows you to send and receive messages globally over SMS, RCS, WhatsApp, Viber Business, Facebook Messenger and other popular channels.	Correct

Can I use the Sinch Conversation API with Viber Business ?	Conversation API offers one single API endpoint for sending and receiving messages across the most popular channels (SMS, RCS, WhatsApp, Viber Business, Facebook messenger and more) using one unified format. For more advanced messaging use cases, it offers conversations, contacts, switching between Bot	Correct
Can I use the Sinch Conversation API with Outlook ?	No answer found	Correct
Where are the hosting locations for the Conversation API ?	US - East and the EU - Ireland	Correct
What are the specific pricing details for using the Sinch Conversation API ?	No answer found	Correct
How does the Sinch Conversation API handle multimedia content like images and videos ?	No answer found	Correct

Table 18: Llama2-DPO