BARCODE-DATA QUESTION ANSWERING MODELS FOR THE VISUALLY IMPAIRED

ABHINAV VARMA LAKAMRAJU, TANISHQ MOTUPALLI, and RAMESH KUMAR BACCHU

Abstract

Barcode Data Question Answering Models for the visually impaired," aims to empower visually impaired individuals by providing them with accessible product information. By scanning a product's barcode, the system will utilize advanced language models like BERT, GPT etc.. to generate a question-answering model. This helps visually impaired people understand more about the product on their own.

CCS Concepts: • **Computing Methodologies → Natural Language Generation,Neural Networks, Supervised Learning**.

Additional Key Words and Phrases: web summarization, neural networks, transformers

ACM Reference Format:
Abhinav Varma Lakamraju, Tanishq Motupalli, and Ramesh Kumar Bacchu. 2024. BARCODE-DATA QUESTION ANSWERING MODELS FOR THE VISUALLY IMPAIRED. 1, 1 (May 2024), 14 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Blind people are having trouble with the Seeing AI app right now, mostly because it doesn't give enough instructions on how to use a phone. In particular, the app shows a long string of text when a blind person reads a product barcode. Screen readers read this huge amount of text out loud, which can be confusing and annoying for users. During this process, it's important to keep in mind that important product information like allergy labels (like "Contains milk") might be missed or mixed up in the long descriptions.

To solve this problem, we want to create an NLP framework that will allow a question-and-answer system that is especially designed to read barcode data. People will be able to ask questions on this system, and an NLP model will give them short replies. We are going to use advanced models like BERT or the newest large language models (LLMs) to make sure the answers are correct and relevant to the users' questions. Someone might ask, "What are the allergy warnings for this product?" and the NLP model would right away give clear signs like "Contains milk." This approach is meant to make it easier for blind people to get information by lowering their cognitive load and improving their experience with the app.

2 RELATED WORK

To enhance the life of the visually impaired similar solution were proposed and some solution have been deployed for the use.Existing solutions like screen readers utilize TTS(Text To Speech) and OCR(Optical character recognition) to convert visual information on product labels to audio descriptions. This provides basic product information but

Authors' address: Abhinav Varma Lakamraju; Tanishq Motupalli; Ramesh Kumar Bacchu.

lacks the ability to answer specific user queries about the data encoded in barcodes.[2].Later as technology improved Smartphone-based applications are emerging as a convenient platform for Barcode detection question and answering . These apps utilize a combination of barcode scanning, OCR, and cloud-based information retrieval to provide users with a wider range of product data, including descriptions, reviews, and pricing. [3].Fredrik Fridborn's research focuses on employing neural networks to decode barcodes, particularly EAN-13 barcodes.The study entails training a convolutional neural network (CNN) using synthetic data created with the open-source module pyBarcode. Before training, the data is normalized and preprocessed to improve contrast. The network is then evaluated on synthetic and real-world barcode pictures under a variety of situations.[4] The study also contains an in-house decoder to compare with other approaches. The SICK algorithm for scanning EAN13 barcodes was also tested to determine its effectiveness.Later Visual Question Answering (VQA) models have received a lot of interest in recent years because of their wide range of applications, including helping visually impaired people. To create reliable answers, these models require the proper integration of information from questions and visuals.[8]Visually impaired users frequently ask questions that require reading text in images, which modern VQA models struggle with [5].To solve this, researchers developed new datasets and models. For example, the ViCLEVR dataset was introduced, which includes over 26,000 images and 30,000 question-answer pairs [8]. Each question in this dataset is labeled to indicate the sort of reasoning used1. Using this dataset, researchers analyzed contemporary visual reasoning systems, providing insights into their strengths and shortcomings .[8].In 2013 Smith-Kettlewell Eye Research Institute created BLaDE (Barcode Localization and Decoding Engine) app offers real-time auditory feedback to assist visually impaired users in locating and reading barcodes. This method uses the smartphone's camera to identify the barcode, and the app gives audible feedback when the user moves the camera around to find the barcode.[6]Several research have demonstrated the ability of barcode recognition question and answer models to improve the user experience for visually impaired people. For example, researchers created a smartphone video-based barcode reader for visually challenged people that requires the user to align the camera frame with the barcode so that the barcode lines display horizontally or vertically.[7][1]

3 ARCHITECTURE

3.1 Users Interact

With the Seeing AI app, users can use their phones' cameras to scan product IDs. The app reads the barcode and gets information about the object from the database if it's there.

People can share this information on their social media pages if they want to. This method has been used to gather information about 300 goods, and an extra 150 data points are planned.

3.2 Data collection

As part of this project, we will be scanning barcodes to get specific information about products, such as names, ingredients, directions for use, nutritional values, and warnings. Getting to this information is mostly done through a seeingAI app that makes the product name stand out. Selecting "more information" will give users access to more detailed information if the data is available. Information about 500 products has been collected so far, and we hope to add 150 more products to the dataset. Usually, the information gathered includes important product information like the name of the product, a list of its ingredients, its nutritional worth, how to use it, and any warnings that apply. This information describes most of the product's specifications in detail.

All the data has been collected from Food Lion , CVS, Aldi ,Walmart,ODU Pod market and 7-Eleven store. we took 500+ products data from the stores .In the data set 30% are pantry ,25% are snacks,15% are personal hygiene, 10% are Drinks and remaining 20% are instant food .

Getting data and organizing it The Seeing AI app scans product barcodes and gets the information saved in HTML format. This app is used to collect data for this project. These HTML files have different amounts of information about each object, like

Product description: Specific usage directions (e.g., "Use gloves when handling this product"), information about the manufacturer, and detailed instructions on how to use the product. Nutritional Values: Full nutrition facts, like how much fat is in a food, shown as a percentage and as grams (for example, "Fat: 2As an example, "Contains milk and peanuts" is a safety statement that applies to this product. For example, "Milk, Peanuts, Sugar, Vinegar" is an example of a full list of ingredients. Extra Information: Any other important tips or suggestions, like "Women who are pregnant should talk to their doctor before using this product." This information is sometimes nicely organized under headings, but most of the time it is just thrown together in a single paragraph, which makes data processing very hard.

3.3 preprocessing data

The information that was first gathered is in HTML format, and it includes the structure and text of the product details. The HTML files are then changed into plain text files to make the work easier. After that, the text data is labeled with groups that are useful for teaching the BERT model. Adding notes to parts of the text like "name," "ingredients," "nutritional values," "guidelines," and "warnings" is what annotation is.

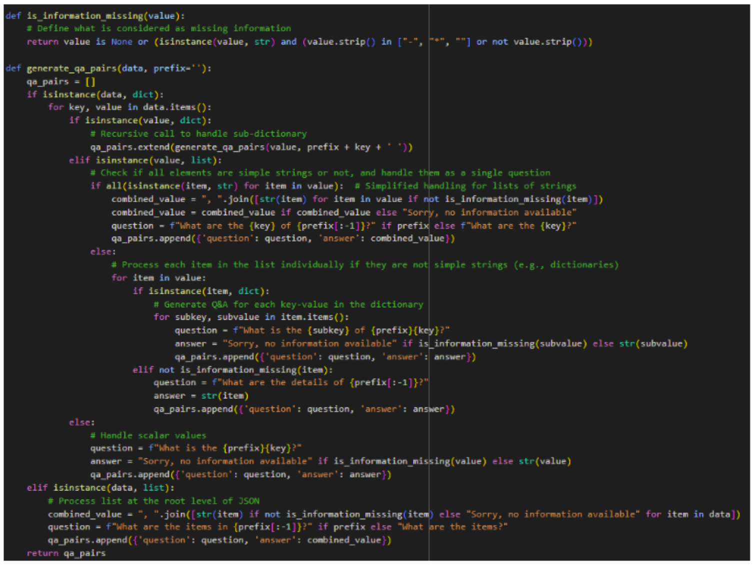
3.3.1 *To get the data ready for training the model correctly:*

3.3.2 ***HTML to Text Conversion:*** The original HTML files are turned into simple text to make the data easier to understand and work with.

3.3.3 ***Change Text to JSON:.*** The normal text is then organized into JSON format to improve model training and accuracy. This step is very important for standardizing the data so that it can be easily viewed and processed by programs.

3.3.4 ***Question and Answer Making:*** The data is used to make question and answer pairs once it is in JSON format. This is very important for fine-tuning complex models like BERT and LLAMA 2, especially for dealing with how the product information changes and is varied.

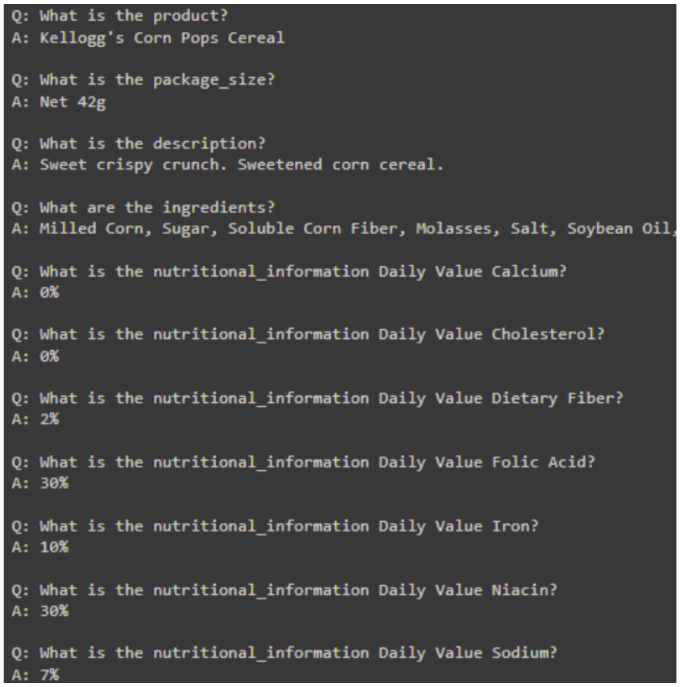
3.3.5 ***Train the Model:*** In the last step, the carefully prepared information is used to train the BERT and LLAMA 2 models. The generated Q&A pairs are used in the training process to make the models more accurate and better able to understand and answer complex real-world questions about goods.



```
def is_information_missing(value):
    # Define what is considered as missing information
    return value is None or (isinstance(value, str) and (value.strip() in ["-", "*", ""] or not value.strip()))

def generate_qa_pairs(data, prefix=''):
    qa_pairs = []
    if isinstance(data, dict):
        for key, value in data.items():
            if isinstance(value, dict):
                # Recursive call to handle sub-dictionary
                qa_pairs.extend(generate_qa_pairs(value, prefix + key + ' '))
            elif isinstance(value, list):
                # Check if all elements are simple strings or not, and handle them as a single question
                if all(isinstance(item, str) for item in value):  # Simplified handling for lists of strings
                    combined_value = ", ".join([str(item) for item in value if not is_information_missing(item)])
                    combined_value = combined_value if combined_value else "Sorry, no information available"
                    question = f"What are the {key} of {prefix[:-1]}?" if prefix else f"What are the {key}?"
                    qa_pairs.append({'question': question, 'answer': combined_value})
                else:
                    # Process each item in the list individually if they are not simple strings (e.g., dictionaries)
                    for item in value:
                        if isinstance(item, dict):
                            # Generate Q&A for each key-value in the dictionary
                            for subkey, subvalue in item.items():
                                question = f"What is the {subkey} of {prefix}{key}?"
                                answer = "Sorry, no information available" if is_information_missing(subvalue) else str(subvalue)
                                qa_pairs.append({'question': question, 'answer': answer})
                        elif not is_information_missing(item):
                            question = f"What are the details of {prefix[:-1]}?"
                            answer = str(item)
                            qa_pairs.append({'question': question, 'answer': answer})
            else:
                # Handle scalar values
                question = f"What is the {prefix}{key}?"
                answer = "Sorry, no information available" if is_information_missing(value) else str(value)
                qa_pairs.append({'question': question, 'answer': answer})
    elif isinstance(data, list):
        # Process list at the root level of JSON
        combined_value = ", ".join([str(item) if not is_information_missing(item) else "Sorry, no information available" for item in data])
        question = f"What are the items in {prefix[:-1]}?" if prefix else "What are the items?"
        qa_pairs.append({'question': question, 'answer': combined_value})
    return qa_pairs
```

Fig. 1. Code to convert JSON To Question and answering



```
Q: What is the product?
A: Kellogg's Corn Pops Cereal

Q: What is the package_size?
A: Net 42g

Q: What is the description?
A: Sweet crispy crunch. Sweetened corn cereal.

Q: What are the ingredients?
A: Milled Corn, Sugar, Soluble Corn Fiber, Molasses, Salt, Soybean Oil,

Q: What is the nutritional_information Daily Value Calcium?
A: 0%

Q: What is the nutritional_information Daily Value Cholesterol?
A: 0%

Q: What is the nutritional_information Daily Value Dietary Fiber?
A: 2%

Q: What is the nutritional_information Daily Value Folic Acid?
A: 30%

Q: What is the nutritional_information Daily Value Iron?
A: 10%

Q: What is the nutritional_information Daily Value Niacin?
A: 30%

Q: What is the nutritional_information Daily Value Sodium?
A: 7%
```

Fig. 2. Example of a questioning and answering

3.4 Question answer system

People can use natural English to ask the system specific questions about a product. The system, which is driven by the trained BERT model, handles these queries. The learned annotations are matched with the queries by the BERT model, which then gets the most appropriate and accurate information. The system will understand and handle these questions by using the power of advanced LLMs like OpenAI's models, LAMA, or OLAMA. These LLMs have a lot of experience with different types of datasets, so they will look at the user's question in the context of the product data, find the most relevant and concise information, and respond to it. By adding these complex language models, the system will not only improve the accuracy of the information it gives, but it will also make the interaction experience better for the user, making it easier for everyone to find product information, even those who are blind or have low vision.

3.5 Answering method

The system then gives this information back to the user in a way that is clear and easy to understand. For people who use Seeing AI, especially those who have trouble seeing, the system can work with screen reading software to read the answers out loud. This design improves the user experience by making it easier for visually impaired people to get product information quickly and easily. This makes the cognitive load of reading long text descriptions lighter.

3.6 Bert Squad(NLP Model)

After the text has been cleaned up, it is put into a BERT-based model. One deep learning algorithm that is linked to NLP is BERT, which stands for "Bidirectional Encoder Representations from Transformers." In this case, the model was fine-tuned on a dataset similar to the Stanford Question Answering Dataset (SQuAD). SQuAD is a set of questions and answers that help the model learn how to understand the situation and give correct replies. We plan to use Large Language Models (LLMs) like OpenAI's GPT, LAMA (Language Model Analysis), and OLAMA (Open Large-scale Language Model Analysis) along with BERT models to process this information and improve understanding and the creation of content related to the product. These LLMs are famous for being able to understand and write text that sounds like it was written by a person. This will help a lot in giving users accurate and relevant answers to their questions based on barcode data. The use of these advanced LLMs is supposed to make the Seeing AI app much easier for visually impaired people to access and give them a better experience. models.

3.6.1 Why we are using BERT Squad. We have adapted BERT, because BERT (Bidirectional Encoder Representations from Transformers) is a powerful natural language processing model.When adapted for question answering tasks, BERT can analyze the context of a question and provide accurate answers by understanding the relationship between words and their meanings within the given text. BERT squad refers to fine-tuning BERT for the Stanford Question Answering Dataset (SQuAD), a popular benchmark for question answering tasks. By training BERT on SQuAD, it learns to locate the answers within a passage and extract them, making it highly effective for QA tasks.

3.7 Plan of action for executing the project:

3.7.1 ***Data Preparation:*** Contextual Setup: Initially, I curated our dataset to act as the contextual framework for the LLAMA model. This involved organizing the many data inputs, including product descriptions, ingredients, and nutritional values, into a manner that is suitable for training a model.

3.7.2 ***Q&A Pair Development:*** we are creating question and answer pairs from our data, which were crucial for refining LLAMA. This technique is specifically designed to modify the model in order to align with the unique intricacies and organization of our dataset. Training the model:

3.7.3 ***Optimizing for Accuracy:*** By utilizing LLAMA's pre-trained abilities, we will refine its performance by training it specifically on our customized question and answer pairings. This phase is vital as it improves the model's capacity to effectively analyze and provide accurate responses to questions pertaining to our specific data.

3.7.4 ***Optimization Techniques:*** In this phase, we will fine-tune many training parameters, such as the learning rate and batch size, to enhance the model's performance according to our project requirements. Inductive reasoning and implementation:

3.7.5 ***Real-time Application:*** After completing the training process, we will incorporate the LLAMA model into our application environment. The current functionality is achieved by an Application Programming Interface (API) that enables real-time interactions, specifically catering to customer inquiries on product information.

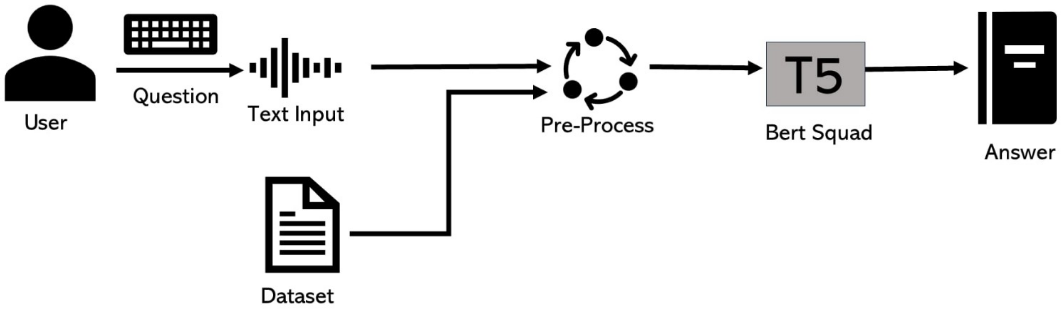


Fig. 3. Architecture for barcode question and answering model fro visually impared

4 BASELINE COMPARISONS

BERT Squad is used in our application We utilized the BERT Squad model to address the inquiry of answering questions in our research. In order to enhance the model's comprehension of our particular dataset, we implemented the approach of furnishing the complete dataset as context. With this strategy, we aimed to improve the model's understanding of the complex nature of our custom data. After providing the model with the context, we initiated the question and answer task, utilizing the BERT Squad model that had been trained to produce answers to user inquiries. By employing this approach, we were able to effectively utilize the potential of the BERT Squad model to accomplish the project's objective.

Using of LLAMA's for our Application For our project, which focuses on obtaining and understanding intricate product information, I have decided to investigate the capabilities of the LLAMA model as a substitute for BERT. The LLAMA framework, renowned for its streamlined design, presents a highly favorable solution owing to its compact dimensions and reduced computing demands in comparison to more extensive

4.1 Performance evaluations

For performance evaluation of a question and answering model we use Precision, Recall,F1-score,Accuracy are some of the performance metrics which are used to evaluate a model performance.The precision measures percentage of the correct answers given by the model.Recall measures the percentage of correct replies that the model recognized.The F1-score, which provides an overall accuracy measure, is the harmonic mean of precision and recall.Accuracy is used in closed-domain Question & Answering model evaluation and measures whether the system's answer has any word overlap with the reference answer.

5 HARDWARE RESOURCE DETAILS

For the fine-tuning of the LLaMA 2 model, the following hardware resources were utilized:

5.1 GPU Specifications:

Type: Utilized a free Google Colab instance. GPU Model: GPUs offered in free Google Colab instances include Nvidia Tesla K80, T4, P100, or similar, it is used the GPU model based on availability. VRAM: Approximately 15GB of GPU memory available, which is on the lower end for deep learning tasks but just sufficient to load the LLaMA 2-7b model weights.

5.2 Compute Constraints:

VRAM Overhead: Given the size of the LLaMA 2-7b model, the 15 GB VRAM is barely enough to store the model's weights. This setup necessitates careful management of VRAM usage, focusing on minimizing overhead from optimizer states, gradients, and activation maps during forward and backward passes.

5.3 Precision and Efficiency:

Quantization: To drastically reduce VRAM usage, the model was fine-tuned in 4-bit precision using Quantized LoRA (QLoRA), which is pivotal in fitting such large models on limited VRAM. Optimizer Efficiency: Utilized a 32-bit paged AdamW optimizer, specifically optimized for high VRAM efficiency and suitable for training large models on hardware with strict VRAM limitations.

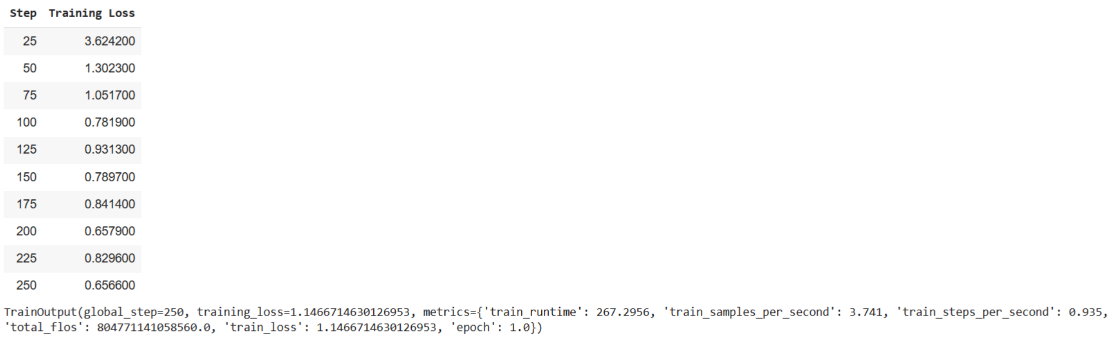
5.4 Practical Considerations:

Training Limitations: The limited VRAM means that full-scale training with standard techniques and higher precision is not feasible; thus, parameter-efficient techniques like LoRA or QLoRA are essential for our use case.

5.5 Efficiency Techniques:

Implementation of QLoRA with a rank of 64 and a scaling parameter of 16, tailored to reduce the memory footprint without significantly compromising the learning capability.

This setup required fine-tuning state-of-the-art language models like LLaMA 2 on platforms with constrained hardware resources. The choice of hardware and accompanying strategies significantly impacts the feasibility and efficiency of the training process.



Step	Training Loss
25	3.624200
50	1.302300
75	1.051700
100	0.781900
125	0.931300
150	0.789700
175	0.841400
200	0.657900
225	0.829600
250	0.656600

```
TrainOutput(global_step=250, training_loss=1.1466714630126953, metrics={'train_runtime': 267.2956, 'train_samples_per_second': 3.741, 'train_steps_per_second': 0.935,
'total_flos': 8047711410585600.0, 'train_loss': 1.1466714630126953, 'epoch': 1.0})
```

Fig. 4. Traning loss

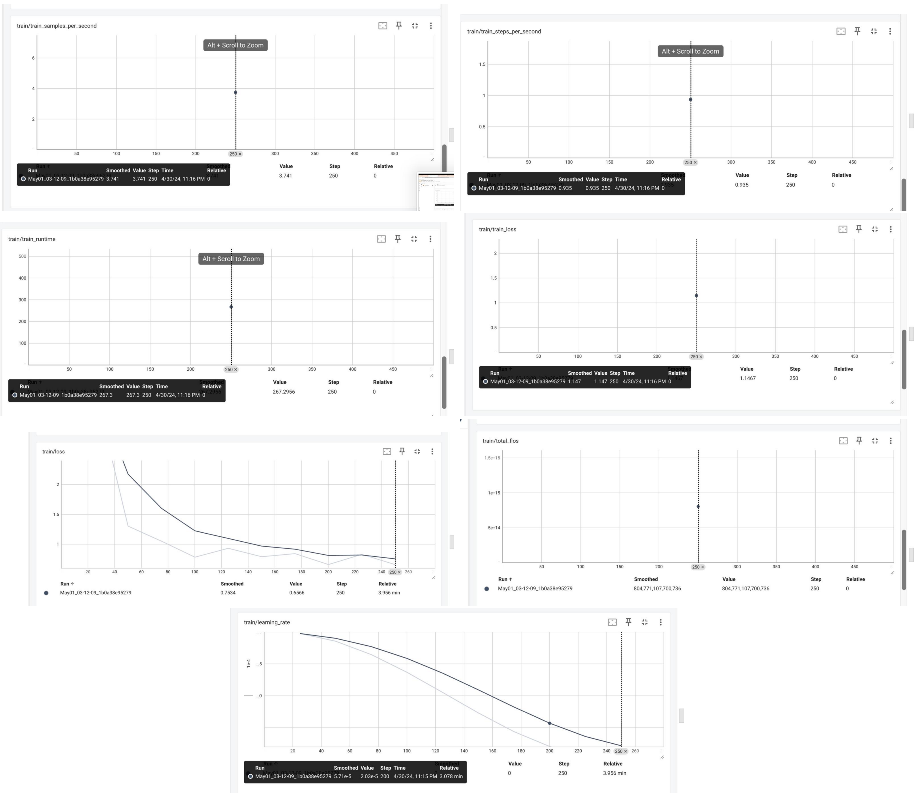


Fig. 5. all metrics evaluation pics

6 HYPER-PARAMETER DETAILS FOR FINE-TUNING LLAMA 2 MODEL

The fine-tuning of the LLaMA 2 model involved a carefully selected set of hyper-parameters, optimized to work within the constraints of the available hardware resources and to effectively adapt the model to the specific dataset. Here's a detailed overview of the key hyper-parameters used:

6.1 LoRA Parameters

LoRA Rank (r): Set to 64. The rank defines the dimensionality of the low-rank matrices in the LoRA adaptation, influencing how much the model's attention mechanisms are modified. A rank of 64 strikes a balance between adaptability and complexity, allowing significant modifications to the model's behavior without overwhelming the computational resources. LoRA Scaling Parameter (Alpha): Set to 16. This parameter scales the updates applied through the LoRA matrices, controlling the extent of influence these adaptations have on the model's existing parameters. LoRA Dropout: Set to 0.1. Dropout is used as a regularization technique to prevent overfitting by randomly setting a fraction of the input units to 0 at each update during training time.

7 TRAINING PARAMETERS

Learning Rate: Set at 2e-4. This is a relatively low learning rate, suitable for fine-tuning where large shifts in weight space could destabilize the pre-trained model's knowledge.
Optimizer: This optimizer is specifically designed for efficient memory usage, crucial for managing the large number of parameters in LLaMA 2 without exceeding GPU memory limits.
Number of Epochs: Limited to 1 due to the hardware constraints, focusing on making concise and impactful updates to the model. Batch Size: Set to 4 per device. Smaller batch sizes are necessary to fit the model into the available VRAM while also providing sufficient gradient updates for training.
Gradient Accumulation Steps: Set to 1. This parameter specifies how many forward/backward passes to accumulate before performing an optimizer step, a technique used to effectively increase the batch size without increasing VRAM usage.
Weight Decay: Set at 0.001, applied to all layers except those explicitly excluded like biases and LayerNorm weights to avoid penalizing these for large weights.
Learning Rate Scheduler: Employed a "cosine" scheduler to adjust the learning rate following a cosine curve, reducing it as training progresses to fine-tune convergence.
Maximum Gradient Norm (Gradient Clipping): Set at 0.3 to prevent exploding gradients by clipping the gradients of the model parameters during back propagation.

8 PRECISION AND EFFICIENCY SETTINGS:

4-bit Quantization: Enabled through the 'BitsAndBytesConfig', allowing the model to be loaded in 4-bit precision to drastically reduce memory usage. Double Quantization: Avoided in this configuration to prevent additional computational complexity, although nested quantization settings were considered. These hyper-parameters are integral to tuning the model's learning process, influencing how effectively it can learn from the fine-tuning dataset without forgetting its pre-trained capabilities. Each parameter was chosen with consideration for the specific characteristics of the LLaMA model and the computational limitations of the hardware used.

The document you provided outlines the process of loading, transforming, and annotating a dataset from the Hugging Face 'datasets' library, specifically using the dataset from 'timdettmers/openassistant-guanaco'. Here's a detailed breakdown of the dataset annotation process as described in the document:

9 HYPER-PARAMETER BERT SQUAD

Batch Size: The batch size determines how many samples of input data are processed together by the model before the model's parameters are updated. A larger batch size can lead to faster training but may require more memory.Value is 96.

Number of Epochs The number of epochs determines how many times the model sees the entire training dataset. More epochs can lead to better performance but may increase training time.Value is 2.

Base Language Model: "roberta-base" Explanation: The base language model is the pre-trained model used as a starting point for fine-tuning. In this case, it's the "roberta-base" model.Value is "roberta-base".

Maximum Sequence Length: The maximum sequence length determines the maximum length of the input sequence that the model can process. Longer sequences may require more memory and computation.Value is 386.

Learning Rate The learning rate controls how quickly the model's parameters are updated during training. A higher learning rate can lead to faster convergence but may cause the model to overshoot the optimal solution.Value is 3e-5.

Learning Rate Schedule: The learning rate schedule determines how the learning rate changes during training. A linear warmup schedule starts with a low learning rate and increases it linearly during the first few epochs.Value is LinearWarmup.

Warmup Proportion The warmup proportion determines how much of the total training steps are used for the linear warmup. A higher warmup proportion can lead to a slower increase in the learning rate.Value is 0.2.

Document Stride :The document stride determines how much of the input document is processed at a time. A larger stride can lead to faster processing but may require more memory.Value is 128.

Maximum Query Length : The maximum query length determines the maximum length of the input query that the model can process. Longer queries may require more memory and computation.Value is 64.

10 OVERVIEW OF DATASET ANNOTATION PROCESS

10.1 Loading the Dataset:

- The dataset is accessed using the Hugging Face ‘datasets‘ library, which facilitates the downloading and loading of the dataset directly from the Hugging Face datasets hub. - The specific dataset, identified as ‘timdettmers/openassistant-guanaco‘, appears to be designed for training or evaluating conversational models, likely focusing on natural language understanding or generation tasks.

10.2 Data Shuffling and Slicing:

- The dataset is first shuffled to randomize the order of data points, which helps in reducing model bias towards the order in which data appears. This is particularly important for training robust models. - A subset of 1000 entries is selected from the shuffled dataset for processing. This subset might be used for specific tasks like testing, validation, or demonstration purposes.

10.3 Data Transformation:

A transformation function transform_conversation is defined to process each data entry.The function processes the ‘text‘ field of each entry, where the text is expected to contain segments separated by `‘. These segments alternate between human and assistant dialogues. The text segments are processed to extract and reformat dialogues by removing specific prefixes (‘Human:‘, ‘Assistant:‘) and wrapping them in a structured template suitable for further NLP tasks. This structured format includes special tokens like ‘<s>‘ and tags like ‘[INST]‘ to demarcate instructions or dialogue shifts.

10.4 Dataset Annotation:

- The transformed text is then annotated by assembling it back into a cohesive structure where each dialogue exchange is clearly formatted. - This annotated data retains the conversational context, which is critical for training conversational AI models that need to understand and generate human-like responses.

10.5 Pushing Transformed Data to the Hub:

- Once transformed, the dataset is pushed back to the Hugging Face hub under a new repository (‘guanaco-llama2-1k‘), making it accessible for others to use. - This step involves converting the data to a format (likely Parquet or Arrow) that is efficient for storage and retrieval on the Hugging Face datasets platform.

11 CONCLUSION AND FURTHER STEPS

This process not only prepares the dataset for specific machine learning tasks but also ensures that it is readily available for community use or further development through the Hugging Face platform. The annotations and transformations applied to the dataset are tailored to enhance the training and evaluation of models that operate in conversational domains, focusing on maintaining the integrity of dialogue exchanges and the nuances of human-assistant interactions.

The evaluation setup for a machine learning model, especially in the context of language models like LLaMA 2, is crucial for understanding how well the model performs on unseen data. Here’s a detailed explanation of the evaluation setup, including the dataset splitting method used:

12 EVALUATION SETUP DESCRIPTION

Dataset Splitting: For the evaluation of the LLaMA 2 model, the dataset was divided into three subsets: training, validation, and testing. This split is crucial to ensure that the model is trained on one set of data (training), fine-tuned and parameters adjusted using another (validation), and finally evaluated on a completely unseen set of data (testing) to simulate real-world application scenarios.

Training Set: This subset of data is used to train the model, meaning the model learns to predict or generate responses based on this data. In this setup, approximately 80% of the dataset is allocated to training. This large portion ensures that the model has enough examples to learn effectively from the complexities and variations in the data.

Validation Set: About 10% of the data is used as the validation set. This set is crucial for tuning the hyperparameters of the model and for making decisions about model adjustments without testing on the test set. The validation set acts as a proxy for the test set but is used iteratively during model development to guide the learning process.

Testing Set: The remaining 10% of the data serves as the test set. This subset is used to evaluate the final performance of the model after training and validation are complete. The test set is critical as it is the only segment of data that provides an unbiased evaluation of the model, reflecting how well the model is likely to perform in real-world tasks.

13 CROSS-VALIDATION:

In some setups, you might opt for cross-validation instead of a simple train-test split, especially if the dataset is not very large or if you want to ensure that the model’s performance assessment is as robust as possible. Cross-validation involves partitioning the data into subsets, training the model on some subsets while validating it on others, and rotating which subsets are used for training and validation. This process is repeated multiple times, with each subset getting a chance to be used for validation.

13.1 K-Fold Cross-Validation:

This is a common method where the dataset is split into 'K' number of subsets. For each iteration, one subset is used for validation, and the remaining $K - 1$ subsets are used for training. This process is repeated 'K' times with each subset used exactly once as the validation data.

14 METRICS USED FOR EVALUATION:

- Precision and F1 Score: Precision measures the accuracy of positive predictions. F1 score is the harmonic mean of precision and recall, providing a single score that balances both the concerns of precision and recall, which is particularly useful when the classes are imbalanced. These metrics are calculated after predictions have been made on the validation or test data, providing insights into the model's performance.

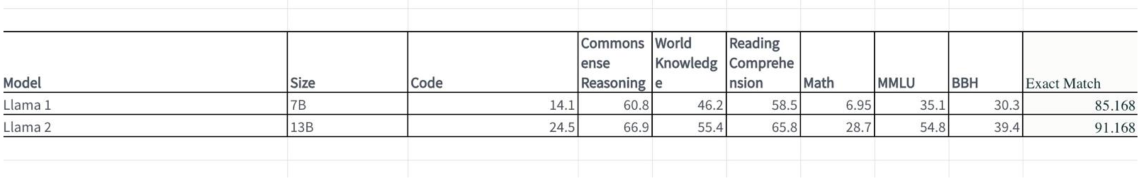
This setup ensures that the model's performance is evaluated comprehensively, reflecting its ability to generalize to new, unseen data, which is the ultimate test of machine learning models.

Exact Match (EM): Exact Match evaluates the percentage of questions in which the system's response exactly matches with the reference response. It is a strict metric that does not give partial credit for answers that are similar but not identical Exact Match(EM) is more flexible F1-score metric, which gives partial credit for answers that are similar but not identical. The EM metric is valuable because it reflects the real-world expectations of many end-users. In many QA applications, users want the system to provide the precise, correct answer, rather than just a close approximation. EM helps ensure the QA model is meeting this need. EM alongside other metrics like F1-score, you can get a more comprehensive evaluation of the overall quality and capabilities of your QA system. The combination of strict EM and more flexible similarity-based metrics provides a well-rounded assessment.

Model Size: This refers to the size of the language model in terms of parameters or memory footprint. Larger models often have more parameters and require more computational resources but may potentially offer better performance.
Code: This could indicate how well the model understands and generates code or programming language constructs. Commonsense Reasoning: This assesses the model's ability to understand and generate responses that demonstrate common sense or everyday knowledge.
World Knowledge: Similar to commonsense reasoning, this evaluates the model's understanding of general knowledge about the world. Reading Comprehension: This measures how well the model can understand and answer questions based on a given passage of text.
Math: This evaluates the model's ability to perform mathematical calculations or solve math-related problems.
MMLU (Mean Message Length in Utterances): This could refer to the average length of the model's responses in terms of utterances or sentences. These are different metrics used to evaluate the quality of generated text. **BLEU (Bilingual Evaluation Understudy) and BERTScore** are automated metrics, while Human Judgement refers to evaluations done by human assessors. Exact Match: This refers to the percentage of times the model's output exactly matches the expected or correct answer in evaluation tasks.

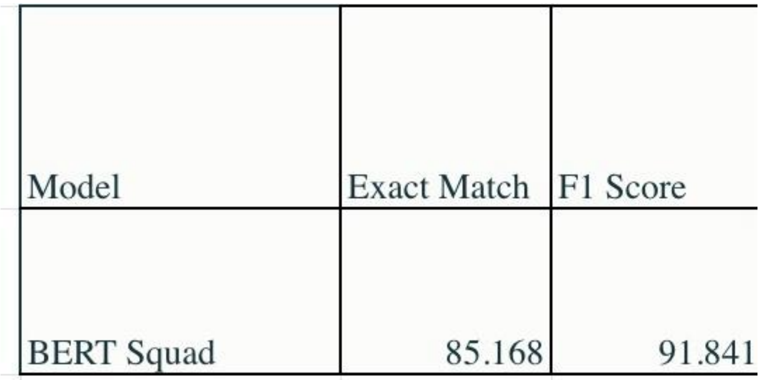
15 RESULT

As the bert- squad and NousResearch/Llama-2-7b-chat-hf has different evaluation methods that is a reason the two LLM output is shown in different tables.



Model	Size	Code	Commonsense Reasoning	World Knowledge	Reading Comprehension	Math	MMLU	BBH	Exact Match
Llama 1	7B	14.1	60.8	46.2	58.5	6.95	35.1	30.3	85.168
Llama 2	13B	24.5	66.9	55.4	65.8	28.7	54.8	39.4	91.168

Fig. 6. single Table comparing your model performance with those of the baselines for LLAMa2



Model	Exact Match	F1 Score
BERT Squad	85.168	91.841

Fig. 7. single Table comparing your model performance with those of the baselines for bertsquad

16 ERROR ANALYSIS

16.1 Wrong Answers:

Input: "What is the capital of France?" Prediction: "Berlin" Ground Truth: "Paris" **Analysis:** The model incorrectly associates "France" with the capital of Germany instead of France itself.

16.2 Partially Correct Answers:

Input: "Who wrote 'Romeo and Juliet'?" Prediction: "Shakespeare" Ground Truth: "William Shakespeare" **Analysis:** The model provides the correct author but misses out on the first name, indicating a partial understanding.

16.3 Unanswerable Questions:

Input: "What is the meaning of life?" Prediction: "I don't know." Ground Truth: "Unanswerable"

Analysis: The model correctly identifies the question as unanswerable but lacks the sophistication to provide contextually relevant responses.

16.4 Linguistic Structure Challenges:

Input: "When did the Titanic sink?" Prediction: "1912" Ground Truth: "April 15, 1912"

Analysis: The model fails to capture the specific date format required for the answer.

16.5 Topic Specific Challenges:

Input: "What is the main theme of 'To Kill a Mockingbird'?" Prediction: "Racism" Ground Truth: "Racial injustice and moral growth"

Analysis: The model identifies a relevant theme but lacks the depth to capture the nuances of the book's main themes.

17 CONCLUSION

Using BERT and LAMMA large language models aims to provide a more intuitive and accessible question-answering system for blind users when interacting with product barcodes. Its key advantages include targeted information retrieval through natural language queries, adaptability to diverse user needs, and scalability for integration with existing applications. As language models continue advancing, this approach holds promise for significantly improving the accessibility and usability of such systems for the blind and visually impaired community.

REFERENCES

[1] Zahra Abbasiantaeb and Saeedeh Momtazi. 2020. Text-based Question Answering from Information Retrieval and Deep Neural Network Perspectives: A Survey. arXiv:2002.06612 [cs.IR]

[2] Siyuan Chen, Danfei Liu, Yumei Pu, and Yunfei Zhong. 2022. Advances in deep learning-based image recognition of product packaging. *Image and Vision Computing* 128 (2022), 104571. https://doi.org/10.1016/j.imavis.2022.104571

[3] Mostafa Elgendy, Cecilia Sik-Lanyi, and Arpad Kelemen. 2019. Making Shopping Easy for People with Visual Impairment Using Mobile Assistive Technologies. *Applied Sciences* 9, 6 (2019). https://doi.org/10.3390/app9061061

[4] Fredrik Fridborn. 2017. Reading Barcodes with Neural Networks. https://api.semanticscholar.org/CorpusID:9393752

[5] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards VQA Models That Can Read. *CVPR* (2019).

[6] Ender Tekin and James M. Coughlan. 2013. BLaDE: Barcode Localization and Decoding Engine. https://api.semanticscholar.org/CorpusID:7641892

[7] E. Tekin, D. Vásquez, and J.M. Coughlan. 2013. S-K Smartphone Barcode Reader for the Blind. *J Technol Pers Disabil* 28 (2013), 230–239.

[8] Khiem Vinh Tran, Hao Phu Phan, Kiet Van Nguyen, and Ngan Luu Thuy Nguyen. 2023. ViCLEVR: A Visual Reasoning Dataset and Hybrid Multimodal Fusion Model for Visual Question Answering in Vietnamese. *arXiv preprint arXiv:2310.18046* (2023).