Interpreting parking signs with lightweight large language models (LLMs) (Custom Project)

Uche Ochuba
Department of Computer Science
Stanford University
Stanford, CA 94305
uochuba@cs.stanford.edu
Mentor: Anna Goldie
Sharing Project: CS210B (Software Project with Corporate Partners)

Abstract

This paper addresses the issue of trying to interpret parking signs given the text transcription of a sign and the current time of day and week. The approach of this paper is to use lightweight large language models, that is, models with 3B parameters or fewer, to solve this problem. A successful solution could automatically run on the hardware of one's car, to quickly, effectively, and confidentially (i.e., without cloud API calls) interpret parking signs and help users avoid violations and fines. My contribution is to curate a dataset tailored to this problem, as well as quantizing, finetuning, and writing applications to integrate the models into an application that can perform this interpretation task end-to-end. My main findings were that models with 3B parameters or fewer were successful at this task, with all achieving over 90% accuracy on the holdout set.

1 Introduction

There has been a variety of key research in large language models over recent years that has greatly enhanced their abilities to answer questions, generate code, translate, summarize, and perform other tasks. The most successful models seek to scale the transformer model and attention mechanisms. There have been large advances in accuracy as the sizes of models and datasets have increased. However, smaller language models still have a critical role to play, as models on consumer hardware and edge devices is critical. Some good examples in recent research are Falcon, Llama, Qwen, Mistral, and Mixtral. These models can still produce stunning results when they are finetuned for the specific use case. I hope to prove that it is possible to build a strongly finetuned chat model under 2B parameters, optimizer to run for speed on consumer hardware.

The main goal of this project is to demonstrate if lightweight large language models (2B parameters, such that they can perform inference in 1 or 2 seconds on the hardware of a 2024 laptop computer) can be used to interpret parking signs in real time for a driver. This project will be working with small transformers to test their accuracy on text examples where they receive a transcription of a parking sign, as well as the current time of day and day of the week, and then they are tasked with classifying the example based on whether or not one can park at the location at the current time. This is important as it could greatly help drivers to adjust to their environment, for example when driving in a new location. This could have drastic results in improving safety, and helping drivers avoid legal penalties. The project relates to the chosen paper as it is all about running LLMs on consumer hardware, such that the solution is portable, mobile, and unexposed to the security risks of cloud computing.

This problem is difficult because, as all drivers know, many parking signs can be highly confusing, especially when there are multiple present at one location. Therefore, large transformer models might be able to more easily interpret the parking signs to help users avoid costly mistakes and parking fines. The problem is fundamentally difficult because signs apply at different times of day and different days of the week, and the user needing to interpret the sign must cross-reference the current time with the street parking sign to determine whether or not parking is allowed. Indeed, the city of San Francisco alone distributes approximately 1 million parking citations per year [1]. This is a clear example of how current methods fail: humans are generally unreliable at interpreting

Computer Science 224N: Natural Language Processing with Deep Learning (Spring 2024) Stanford, USA.

Figure 1: Just one example of a confusing parking sign in San Francsico, California.

signs, and there are few cases of using software to automatically interpret signs. In the future, one might imagine a vehicle scanning the environment for potential violations, and alerting the user if they are about to leave the vehicle in a location that might expose them to fines.

Therefore, my approach to solving this problem involves using Large Language Models (LLMs) to interpret parking signs. In particular, this project aims to solve this problem using models that can be run on consumer hardware. The motivation behind this comes with understanding that current state-of-the-art LLMs require many billions of parameters and therefore specialized hardware and resource-intensive GPUs to run effectively. Thus, such "heavyweight" models are often operated over cloud applications, with the results sent back to the user over an internet connection. In the context of parking sign interpretation, this can be ethically and functionally problematic. This is because such information, if compromised, could reveal the location and/or other sensitive data from the user. So my use of lightweight models, which can be operated on the hardware of a car at present or in the near future, will help protect user privacy. Yet, this means approaching the challenge of interpreting complex signs and scenarios with less sophisticated models, which is the key challenge of this project.

2 Related Work

A key paper to this project is *H2O-Danube-1.8B Technical Report* [2]. The key contribution of this paper is the introduction of new models, called H2O-Danube, and H2O-Danube2, each 1.8B parameters, trained on 1T and 3T tokens, respectively. They build on the Llama architecture and Mistral tokenizer by implementing several key, novel features, including sliding window attention, rotary positional embedding, grouped-query attention, and root mean square layer normalization.

Another important paper to this project is that of Google's Gemma model, which is 2B parameters and was trained 6T tokens [3]. This model had access to a larger and more varied amount of pre-training tokens than Danube, and has 200M more parameters, so one might expect more accurate performance on the classification task. Finally, I used a model from Zephyr's Stable Code technical report [4], which uses 3B parameters, and therefore I might expect to be more performant.

I chose these papers because they introduced new techniques that were demonstrated to be effective for lightweight LLMs, and produced the best model under and around 2B parameters. For my project, it is key that the model can be run on consumer hardware, as the goal for the project is to have model that can run on board a car, which has an amount of compute that is rather pedestrian. I have gained what I was hoping for, and am ready to begin applying the model to my task.

These papers deals with transformer architecture, which is clearly related to the topics of this class. The attention paradigm discussed was key in allowing the research world to more accurately model human language, allowing a quantum leap past RNNs and LSTMs. Indeed, language is also hard for computers to model. The best language models require trillions of parameters and even more tokens to achieve their staggering performance. This paper helps us solve the task of using fewer parameters to achieve passable results, with the ultimate goal of democratizing language models for everyone.

In this project, I am building upon the work in the mentioned papers because they do not apply the models to real-world tasks and bridge the gap to make the technology useful for everyone. This paper involved curating datasets from online and the San Francisco Municipal Transportation Agency (SFMTA), to test the models'

effectiveness. Additionally, my work takes the next step by setting up the models to accept input directly from a camera feed and transcribing them for input into the models.

3 Approach

The first step in creating this paper involved the curation of the dataset. The training and validation set examples included a 1376 unique examples. The examples are strings of text which contain transcriptions of parking signs, along with a question of whether or not the user can park at this location given the information. Each example has a corresponding label. This is stored in a comma-separated values format. Some training examples are shown as follows:

15 MINUTE PARKING 9 AM TO 5 PM MON-FRI. It is currently THU 2:45 PM Can I park here?,**1**
NO PARKING 2 AM TO 4 AM WEEKDAYS FOR STREET MAINTENANCE. It is currently TUE 3:15 AM Can I park here?,**0**
10 MINUTE PARKING. It is currently SUN 1:30 PM Can I park here?,**1**

All training examples were curated by me. In order to do this, I first began a correspondence with the SFMTA to get a dataset of nearly 2000 street parking signs within the city limits. I began to write my own examples, changing the times of day and labelling them with the correct response for the classification task. After completing many such examples, I also inputted these examples into text generation LLMs, prompting them to generate more and varied examples. I then checked the examples for quality and variety, ensuring that the labels were accurate and the examples were not repetitive or limited in scope and content. Thus, I was able to put together the 1376 examples. I also implemented algorithms to load the data into the various formats needed for the models to complete the classification task. The dataset split was 80:20 test:validation. Due to the limited amount of data, I trained once on the training set, and took my results by testing once on the held-out validation set. I also did quantization on all models to reduce their parameter precision and resources needed and increase their execution time. This is a common technique to reduce the resources needed for a model to run without compromising too much on performance. Based on the results of the paper, the effects of quantization were not to detrimental.

Additionally, I wrote scripts to measure summary statistics of the models' performance. This included accuracy $= \frac{TP+TN}{TP+TN+TP+TN}$, precision $= \frac{TP}{TP+FP}$, recall $= \frac{TP}{TP+FN}$, F1 $= \frac{2 \cdot precision \cdot recall}{precision+recall}$, AUROC (the area under the receiver operating characteristic curve), and AUPRC (the area under the precision-recall curve). The last two are particularly useful for handling class imbalance in the dataset. Where we have the following counts: TP for true positives, TN for true negatives, FP for false positives, FN for false negatives.

The baseline for this assignment is the naive bayes classifier [5]. I wrote a script to execute the multinomial naive bayes classifier, with tools from the scikit-learn library. Naive bayes is a standard baseline classifier, which uses the following steps, and is based on Bayes Theorem of probability, which states that $P(A|B) = P(A) * P(B|A)/P(B)$, where A and B are events. In pre-processing, all punctuation and special characters are removed from the input string, and the string is tokenized along space-separated 1-grams, and the unique 1-grams of the corpus are used to constitute the vocabulary of the dataset. Implicitly, this gives every unique word a unique "identification number", from 0 to $|V| - 1$, where $|V|$ is the size of the vocabulary. Based on this, each input string can be represented as a vector $v \in \mathbb{R}^{|V|}$, where each entry is the count of the corresponding 1-gram in the text. During the training phase, we iterate over the training data and calculate the probability of each word given each class label, which in our case is either positive or negative for parking available or unavailable, respectively, as well as the prior probability of each class label. This means that, for each class c and each word w in the vocabulary, we calculate: $P(w|c) = \frac{count(w,c)+\alpha}{count(c)+\alpha|V|}$, where count(w, c) is the number of times word w appears in texts of class c, count(c) is the total count of words in texts of class c, and alpha a number used to smooth the probabilities. This is done so that 0 counts do not create 0 probabilities! We set this value equal to 1 for Laplace smoothing. The prior probability of each class is calculated as: $P(c) = \frac{count(c)}{N}$, where count(c) is the number of

strings in class c, and N is the total number of examples. When making predictions on the validation parking set, for a new string d represented as a vector v, we calculate the posterior probability of each class c using Bayes' rule: $P(c|d) \propto P(c) \prod_{i=1}^{|V|} P(w_i|c)^{v_i}$, where w_i is the i-th word in the vocabulary, and v_i is the count of that word in string d. The class with the highest posterior probability is chosen as the predicted label for the example.

The three primary models used for this project are Danube, Gemma, and Stable LM, which follow the transformer architecture [6]. This architecture brings the key concept of attention to the table, which allows the model to attend to different parts of the sentence at once, and uses an encoder-decoder architecture, as discussed in lecture. The model passes a embeddings of the words in the sentence into the transformer encoder, where positional encodings are added, and the multi-head attention algorithm is applied. At a high level, the attentional ouptut of query matrix Q, key matrix K, and value matrix V is calculated with: $Attention(Q, K, V) = \frac{QK^T}{\sqrt{d_k}}V$, where d_k is the dimension of K. (the projection of a vector v onto a vector y is $\frac{v \cdot y}{v \cdot y} y$. The last part of classification is done with a multilayer perception (MLP) layer, which consists of individual processing units, termed "neurons," arranged hierarchically. I add this on to the transformers using code from the HuggingFace Transformers library. Mathematically, this involves multiplying each vector v of the layer by a normalized version of softmax(qk^T), where q, k are vectors "adjacent" to v. Each neuron in a layer receives input from all neurons in the preceding layer through weighted connections. The layer applies a transformation to the input data using the GeLU function, $GeLU(x) = x\Phi(x)$, (where $\Phi(x)$ is the cumulative distribution function of the Gaussian distribution) with their corresponding weights and summing them up, allowing the network to progressively extract higher-level representations as it propagates through the layers.

From here, I wrote scripts to finetune each of the models. I wrote code to execute parameter-efficient finetuning (PEFT), which uses the Adam optimizer to finetune the linear and embedding layers of each transformer on the 1100 training examples [7]. I wrote scripts that use the Low-Rank Adaptation of Large Language Models (LoRA) configuration, based on a finetuning paper [8]. The optimization of this objective is done with the following equation, which is discussed in depth in the eponymous paper [8] $\max_\Theta \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right)$. LoRA performs updates to only a small number of parameters, meaning that it is more computationally and time efficient. This has the benefit of allowing finetuning on less sophisticated hardware. This is done by using rank decompositions of the parameters, to calculate low rank updates to the parameters. LoRA introduces two small matrices, Δ_w and Δ_b, for each layer of the pre-trained model, using the original weights W and biases b of the layer, as follows: $W' = W + U * \Delta_w * V^T$, $b' = b + \Delta_b$, where U and V are trainable matrices. The dimensions of U, Δ_w, and V are chosen such that the rank of the update ($U * \Delta_w * V^T$) is smaller than the rank of the original weight matrix W [8].

Finally, work performed also includes an iOS application, which can be pointed at parking signs, and automatically prompts the trained model with the time of day, such that it will make a prediction for the sign in the field of view of the camera, and output the interpretation result class to the user interface. The model runs in under 2 seconds on the device, and can be demonstrated at the poster session!

4 Experiments

The dataset used for this project was made up of 1376 unique examples of text transcriptions of parking signs, along with a corresponding question ("Can I park here?) and label. Each example is a string of text containing the parking sign information, followed by a question asking whether one can park at that location given the provided information. Labels are binary, with 0 indicating that parking is not allowed and 1 indicating that parking is allowed. Each row represents a single example. A variety of parking sign scenarios are present in the dataset, including time restrictions, day restrictions, and other parking regulations. The dataset was curated by myself, who initially obtained a dataset of nearly 2000 street parking signs within the city limits from the San Francisco Municipal Transportation Agency (SFMTA). I then created all examples by writing them with inspiration from the dataset and other parking signs found online. They

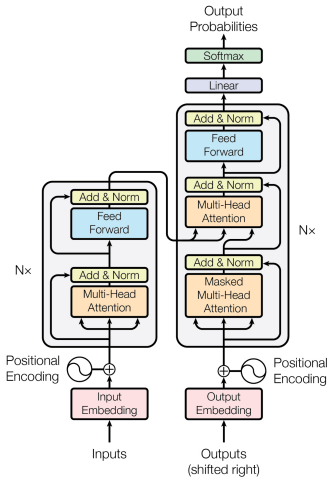

Figure 2: Original transformer model architecture [6].

Table 1: Model Performance Comparison

Model	Acc (Val)
Naive Bayes Baseline	72.5
Danube	90.9
Gemma	**95.3**
Stable LM	93.8

were then labeled with the correct response for the classification task. Furthermore, I used text generation language models to generate more varied examples, which were then checked for quality and accuracy before being included in the dataset. The dataset was split into training and validation sets, with an 80:20 ratio, respectively.

Models were fine-tuned with NVIDIA T4 GPUs for 10 epochs, taking about 4 hours to train on average. I chose 10 epochs as I noticed that eval accuracy had well converged by this point. In preliminary investigations, I discovered that a batch size of 16 led to the best evaluation accuracy and smallest evaluation loss, while not running out of memory on the machines. Additional training hyper-parameters included the following. Warmup steps = 1, batch size = 16, gradient accumulation = 1, optimizer = AdamW, and learning rate = 0.001. For LoRA, I used rank = 8, alpha = 32, and dropout = 0.01.

The evaluation statistics used are accuracy $= \frac{TP+TN}{TP+TN+TP+TN}$, precision $= \frac{TP}{TP+FP}$, recall $= \frac{TP}{TP+FN}$, F1 $= \frac{2 \cdot precision \cdot recall}{precision+recall}$, AUROC (the area under the receiver operating characteristic curve), and AUPRC (the area under the precision-recall curve). The last two are particularly useful for handling class imbalance in the dataset. Where we have the following counts: TP for true positives, TN for true negatives, FP for false positives, FN for false negatives.

Table 2: Model Performance Metrics

Model	Precision	Recall	F1	AUROC	AUPRC
Danube	90.3	91.5	90.7	91.6	88.4
Gemma	**95.7**	**94.4**	**94.9**	**99.3**	**99.3**
Stable LM	93.0	94.1	93.5	98.1	97.8

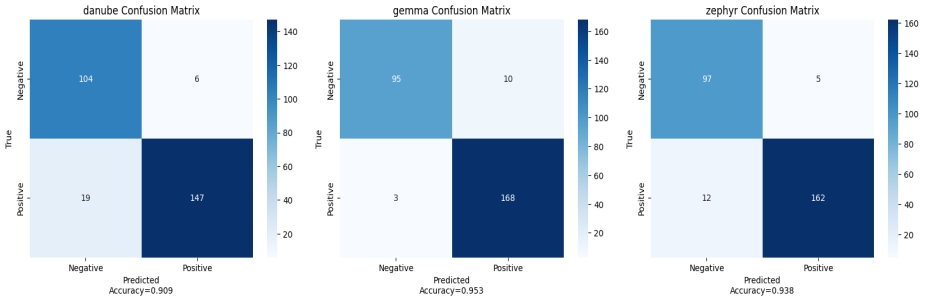

Figure 3: Left to right: confusion matrices for danube, gemma, and zephyr models.

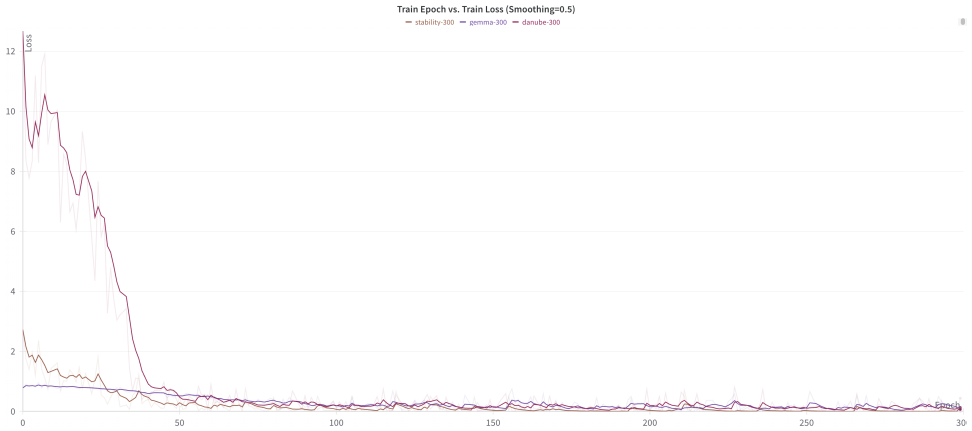

Figure 4: Statistics over training eopchs for various models.

My quantitative results showed that the finetuned Gemma model performed the best across all statistics. The accuracy of these models was surprisingly good for lightweight models, since they were all able to achieve above 90% accuracy. This is very promising and shows that any the models could be deployed in production with relatively performant results. So, overall these results are much better than I expected, surpassing the baseline with flying colors. Furthermore, all of the models take less than 2 seconds to run on the hardware of a 2023 M3 MacBook Pro, showing that their deployment is also realistic in terms of computational resources. I think that this is the case as the transformer architecture is highly capable of extracting information across strings, due to the features of attention as discussed in lecture. It could also suggest that there is the need for a larger and even more diverse dataset, such that the models could be tested in a greater variety of scenarios before deployment.

5 Analysis

Overall, we see that the transformer models are able to capture a wide variety of examples and accurately predict whether or not parking is allowed at a specific time. Looking at the individual examples, the models were all capable of looking at the same parking restriction and flip their prediction of parking ability based on solely a time change. This is helpful in achieving a high accuracy on the validation set. This is likely attributable to the relatively large amount of parameters of the models, as well as the features of attention. In the end, I also wrote scripts to time the execution time of the different models, and the differences were negligible, showing that all of them could perform in under 2 seconds. This means that this task could be performed with smaller models, rather than needing extremely large language models to perform this task, which was one of the goals of the project. Overall, we see that the Stable LM and Danube models had the tendancy to overpredict the negative class, leading to more false negatives than positives. This could be for many reasons, including a bias of the models towards being more conservative in allowing parking to avoid potential fines or violations. The Gemma model was able to achieve the best balance, with very few errors in either class. On the contrary, it produced more false positives than any of the other models, which could be problematic in the real world. This positivity bias could be because of class imbalance in teh dataset. One might imagine prioritizing avoiding

false positives rather than false negatives (i.e. precision over recall) for this use case, since it is probably less problematic to tell a user that they cannot park at a location when they can, than telling them that they can park when they can't. However, for the majority of common parking sign scenarios, all models performed relatively well.

The following are qualitative examples of the outputs of the models. Here is one validation example that all models misclassified:

NO PARKING BETWEEN 6:30 AM - 8:30 AM AND 2:30 PM - 4:30 PM SCHOOL DAYS. It is currently SUN 3:55 PM and it is not a school day Can I park here? **1**.

This is a rather complicated example in the set, as it needs the model to understand that it is not a school day, and that the sign for no parking only applies during school days. This could plausibly be done by understanding that Sunday is not typically a school day, although it seem unlikely that the models could extract this information from a training set of only 1100 examples. Although, it is possible that they have learned this concept during the pre-training phase. It is more likely that they will be able to extract information from the prompt given, which clearly states that it is not a school day. Yet, all 3 models still failed to classify this example in the positive class. This shows that the models have clear limitations, and are perhaps less able to understand additional semantic information, or distinguish between the days of the week. The solution to this would be to increase the size of the models, or to obtain a larger and more extensive dataset.

Next is an example that was classified as a false negative by all models except Gemma.

NO PARKING 1 AM TO 5 AM DAILY. It is currently SUN 2:00 PM Can I park here? **1**

It is likely that this was caused by the fact that the failing models were less able to distinguish between AM and PM times. Clearly, if it is 2:00 AM, one cannot park at this location. But if it is 2:00 PM in the afternoon, than this is a positive example, which is the correct class. This is likely the misinterpretation that is happening with the models. They are misunderstanding and saying that the current time falls within the early morning time range given, when it is actually a time that is 12 hours later. It could be useful to use a 24-hour clock in the prompt to see how this affects the accuracy of the models.

Overall, this qualitative analysis suggests that the Gemma model is the best at this classification task, likely due to the larger number of tokens that it was pre-trained on. Note that it does not have the largest number of parameters, and still was the most successful at this task. This suggests that extensive pretraining is likely a strong predictor of which model will be most successful at this kind of task, especially since the model architectures are relatively similar. However, the Stable LM and Danube models performed better at keeping the number of false positives low, which could be determined to be a highly useful feature, since the primary goal of a system such as this might be to avoid parking fines, even if it is abundantly cautious and sometimes tells the user that they cannot park in a location where they actually can.

6 Conclusion

My project demonstrated the capabilities of modern lightweight large language models (LLMs) under 2B parameters to perform complex reasoning on real-world data. By training Danube, Gemma, and Stable LM models on a dataset of parking sign transcriptions and interpretations, relatively accurate performance above 90% was achieved in predicting whether parking was allowed in a given scenario. Gemma model achieved the best results with 95.3% accuracy, showing the benefits of more extensive pre-training. However, I saw that even compact LLMs can handle real-life language tasks when properly finetuned, as the other models also performed quite well. A key achievement was creating an iOS app demo deploying the LLM on-device for real-time parking sign interpretation.

The primary limitation I had during this project was the relatively dataset size, and I am limited in my ability to write new and varied examples. With more hours and resources, I could potentially get more and higher quality examples. I could also look more into customizing the architecture of the transformers for this specific task. I did quantization to make the models run faster, but I could also change the model architecture more. Expanding this dataset and exploring larger compact architectures in the 5-10B range could further boost capabilities. Overall, this work validated the potential of LLMs to tackle impactful real-world reasoning tasks efficiently on consumer hardware, which could be an exciting field of research going forward. The original goal of this project including executing causal language modeling, attempting to predict not only whether or not one could park, but also for how long in the positive cases. But, this was scaled back as, after extensive effort, the lightweight models were not able to produce reliable and coherent results. This could be a further avenue for research in the future.

I would like to thank my esteemed project advisor, Anna Goldie.

References

[1] A. Rezal, "Maps show which San Francisco blocks are hot spots for parking tickets," *San Francisco Chronicle*. Accessed: Jun. 06, 2024. [Online]. Available: https://www.sfchronicle.com/sf/article/parking-tickets-18409424.php

[2] P. Singer *et al.*, "H2O-Danube-1.8B Technical Report." arXiv, Apr. 15, 2024. doi: 10.48550/arXiv.2401.16818.

[3] Gemma Team *et al.*, "Gemma: Open Models Based on Gemini Research and Technology." arXiv, Apr. 16, 2024. doi: 10.48550/arXiv.2403.08295.

[4] N. Pinnaparaju *et al.*, "Stable Code Technical Report." Accessed: Jun. 06, 2024. [Online]. Available: https://arxiv.org/html/2404.01226v1

[5] V. B. Vikramkumar and Trilochan, "Bayes and Naive Bayes Classifier." arXiv, Apr. 03, 2014. doi: 10.48550/arXiv.1404.0933.

[6] A. Vaswani *et al.*, "Attention Is All You Need." arXiv, Aug. 01, 2023. doi: 10.48550/arXiv.1706.03762.

[7] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, and F. L. Wang, "Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment." arXiv, Dec. 19, 2023. doi: 10.48550/arXiv.2312.12148.

[8] E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. doi: 10.48550/arXiv.2106.09685.

[9] Q. Lhoest *et al.*, "Datasets: A Community Library for Natural Language Processing," arXiv, Sep. 06, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2109.02846

[10] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," arXiv, Dec. 03, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1912.01703

[11] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," arXiv, Jan. 04, 2012. [Online]. Available: https://doi.org/10.48550/arXiv.1201.0490

[12] The pandas development team, "pandas-dev/pandas: Pandas (v2.1.4) [Computer software]," Zenodo, Oct. 14, 2023. [Online]. Available: https://doi.org/10.5281/zenodo.10304236

[13] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261-272, Mar. 2020. [Online]. Available: https://doi.org/10.1038/s41592-019-0686-2

[14] M. L. Waskom, "seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, Apr. 2021. [Online]. Available: https://doi.org/10.21105/joss.03021

[15] T. Wolf *et al.*, "HuggingFace's Transformers: State-of-the-art Natural Language Processing," arXiv, Oct. 08, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1910.03771

Ethical Considerations

In thinking of ethics, the purpose of this project is to encourage running models locally rather than in the cloud, in order to protect the data of users. This will be key in the scenario of mobility, because the location of the user is sensitive information that is important to protect from bad actors. One ethical concern is the data privacy of users. How I am practically addressing this is by not scraping the data of someone driving around, as this could lead to re-identification attacks. By data privacy concerns, I specifically mean that a bad actor could use this information to track down the place of work or home of a user, which could be used for targeted harassment or doxxing. Also, this could be used to expose people for going to places where they might not want others to know that they have been. Thus, this is a serious ethical concern. To combat this, I am also ensuring that we are not doing cloud computing with this data in the finished product, so that beneficiaries of this research will not be exposed to security risks involving the leaking of their past locations.

A second ethical concern is the reliability of the models that I am using. If the models are inaccurate, it could result in users parking in the wrong location, which would lead to fines. Action that I will take to avoid this happening is to make the models as accurate and robustly tested as possible, and also adding disclaimers that the models are not perfect and that the user should use their own discretion and use the product at their own risk. This makes sure that people are well informed of the risks of using technology.