Title:

Next-Word Prediction While Typing a Message

Author:

Zhihe Tu

Abstract:

This project investigates the task of next-word prediction while a user types a message. I compare two Transformer-based language models: GPT-2 and Unsloth Zephyr (4-bit quantized). I apply Top-k sampling to predict the three most likely next words for partially typed prompts. Experiments are conducted using normal English prompts and stress testing with noisy inputs (typos, random text, and code-like snippets). Results show that both models achieve comparable performance on normal English prompts, reaching 80% accuracy. Unsloth achieves higher prediction accuracy than GPT-2 (33% compared to 0) under stress conditions. Error analysis reveals systemic biases towards certain structured data and difficulties handling typos. Future improvements include fine-tuning on noisy datasets and incorporating preprocessing modules for error correction.

Introduction:

The ability to predict the next word while typing is crucial for improving communication speed and user experience in messaging applications, smart keyboards, and AI writing assistants. Accurate next-word prediction can help users formulate their messages faster, correct typos, and suggest relevant continuations. However, models often struggle with noisy or informal input, such as misspellings or code snippets, which occur frequently in real-world typing scenarios. This project aims to assess how well existing language models handle both clean and noisy prompts and investigate methods to improve prediction robustness.

Background / Related Work:

- Radford, A., et al. Language Models are Unsupervised Multitask Learners. OpenAI, 2019. Introduced GPT-2, showing language models can learn tasks without supervised fine-tuning.
- Brown, T., et al. Language Models are Few-Shot Learners. NeurIPS, 2020. Demonstrated that larger models like GPT-3 can perform few-shot learning with minimal examples.

- Dettmers, T., et al. LLM.int8(): 8-bit Matrix Multiplication for Transformers. NeurIPS, 2022. Proposed efficient low-bit quantization techniques to accelerate inference without sacrificing model quality.
- Xie, Q., et al. CoKe: Contextual Knowledge Selection for Pre-trained Language Models. ACL, 2020. Explored improving pre-trained models through better contextual knowledge selection during inference.
- Kneser, R., and Ney, H. Improved Backing-Off for N-gram Language Modeling. ICASSP, 1995.

  Proposed a smoothing technique that improves n-gram models by adjusting word probabilities based on the diversity of contexts in which words appear.

Differences: Our project directly evaluates next-word prediction in real typing scenarios, including both natural language and noisy input, rather than focusing on general multitask learning or improving traditional statistical models.

This project investigates next-word prediction for typed messages using two models:

- GPT-2 (from Hugging Face)
- Unsloth Zephyr SFT (4-bit quantized)

Both models predict the next likely word based on previous tokens. Specifically, given a sequence $x_1,x_2,\ldots,x_t$, the models output the probability distribution:

$P(x_{t+1}|x_1,\ldots,x_t)=\text{softmax}(Wh_t+b)$ where h_t is the hidden state at the last token.

We apply Top-k Sampling to select the top 3 predictions for each prompt:

top_k_probs, top_k_indices = torch.topk(probabilities, k=3, dim=-1).

Additionally, we introduce stress testing (random inputs, typos, and code-like input) to assess model robustness.

Experiment:

Datasets

- Normal Prompts: Simple English phrases like "The quick brown fox".
- Stress Testing Prompts: Noisy or malformed inputs.

Each prompt has a corresponding ground truth for evaluation.

Evaluation Metric:

Top-k accuracy: a prediction is considered correct if the ground-truth word appears in the top-3 outputs.

Accuracy is computed as:

Accuracy=Correct Predictions\Total Samples

Results:

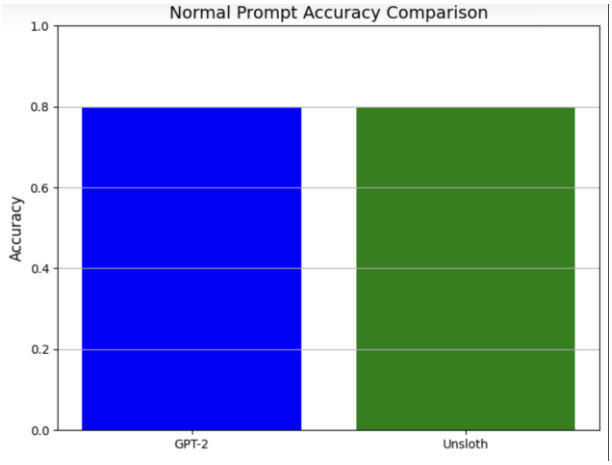
GPT-2 Normal Accuracy: 4/5 = 80.00%

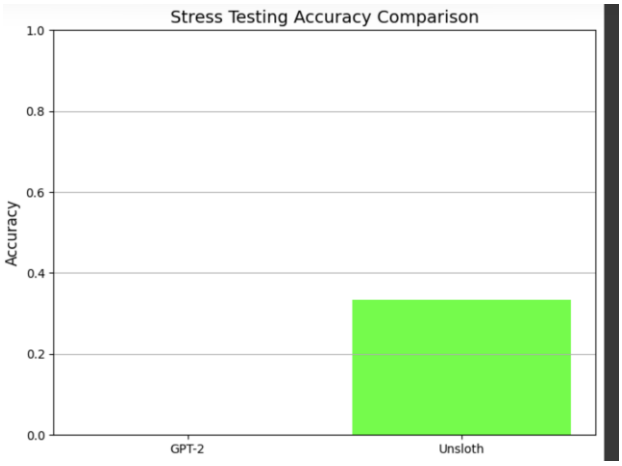
Unsloth Normal Accuracy: 4/5 = 80.00%

GPT-2 Stress Accuracy: 0/3 = 0.00%

Unsloth Stress Accuracy: 1/3 = 33.33%

Two bar charts visualize the normal vs. stress-test accuracies separately:





Both models seem to struggle significantly with unusual or challenging inputs (e.g.,'segdgegedd', 'Heo wrld'). However, Unsloth appears to perform slightly better, with 33.33% accuracy. It suggest a systemic limitation in handling rare or unnatural patterns.

The systems exhibit a bias toward certain structured data. For example, Unsloth has a clear advantage when dealing with code-like syntax ('def calculate_sum(a, b)'), as its highest probability prediction ('') aligns with the expectation for a valid continuation in this context.

Conclusion:

This project demonstrates that Unsloth Zephyr and GPT-2 achieve about the same performance on normal English text. Also, both models show vulnerability to stress testing, with Unsloth performing slightly better than GPT-2 on it. Future directions include:

- Fine-tuning models on noisy, typo-ridden datasets
- Incorporating error-correction preprocessing
- Expanding from next-word to next-phrase prediction

The results suggest that we can achieve similar results for typing scenarios with smaller models like Unsloth Zephyr. The stress-testing remains a challenging for deploying reliable typing assistance systems in real-world applications.

References:

- Radford, A., et al. *Language Models are Unsupervised Multitask Learners*. OpenAI, 2019.
- Brown, T., et al. *Language Models are Few-Shot Learners*. NeurIPS, 2020.
- Dettmers, T., et al. *LLM.int8(): 8-bit Matrix Multiplication for Transformers*. NeurIPS, 2022
- Xie, Q., et al. *CoKe: Contextual Knowledge Selection for Pre-trained Language Models*. ACL, 2020.
- Kneser, R., and Ney, H. *Improved Backing-Off for N-gram Language Modeling*. ICASSP, 1995.

```
===== Stress Testing Results =====

Stress Test Prompt: 'segdgegedd'
GPT-2 Predictions: [('.', 0.049510207027196884), ('ge', 0.04614612087607384), ('g', 0.03368943929672241)]
Unsloth Predictions: [('', 0.3876953125), ('ge', 0.1590576171875), ('g', 0.0182647705078125)]

Stress Test Prompt: 'Heo wrld'
GPT-2 Predictions: [('the', 0.037606097757816315), ('s', 0.031966812908649445), ('not', 0.026993615552783012)]
Unsloth Predictions: [('.', 0.07080078125), (',', 0.038482666015625), ('s', 0.0335693359375)]

Stress Test Prompt: 'def calculate_sum(a, b)'
GPT-2 Predictions: [('', 0.1964280605316162), ('{', 0.06657429039478302), ('return', 0.042080558836460114)]
Unsloth Predictions: [('', 0.9716796875), (':', 0.0172576904296875), ('#', 0.006862640380859375)]
```

The code:

```
!pip install -U bitsandbytes unsloth
!pip install transformers matplotlib
!pip install huggingface_hub[hf_xet]
import torch
import torch.nn.functional as F
import matplotlib.pyplot as plt
from unsloth import FastLanguageModel, PatchDPOTrainer
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Load GPT-2 Model and Tokenizer
def load_gpt2():
    model_name = "gpt2"
    tokenizer = GPT2Tokenizer.from_pretrained(model_name)
    model = GPT2LMHeadModel.from_pretrained(model_name)
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
    model.to(device)
    return model, tokenizer, device

# Load Unsloth Model and Tokenizer
def load_unsloth():
    PatchDPOTrainer()
    max_seq_length = 1024
    dtype = None
    load_in_4bit = True
    model, tokenizer = FastLanguageModel.from_pretrained(
        model_name="unsloth/zephyr-sft-bnb-4bit",
        max_seq_length=max_seq_length,
        dtype=dtype,
        load_in_4bit=load_in_4bit,
    )
    device = model.device
    return model, tokenizer, device

# Prediction Function
def predict_next_words(prompt, model, tokenizer, device, top_k=3):
    if not prompt.strip():
        return [("No input", 0.0)]
    inputs = tokenizer(prompt, return_tensors="pt").to(device)
    with torch.no_grad():
        outputs = model(**inputs)
        next_token_logits = outputs.logits[:, -1, :]
```

```
    probabilities = F.softmax(next_token_logits, dim=-1)
    top_k_probs, top_k_indices = torch.topk(probabilities, k=top_k, dim=-1)
    predictions = []
    for i in range(top_k):
        token_id = top_k_indices[0, i].item()
        token = tokenizer.decode([token_id]).strip()
        prob = top_k_probs[0, i].item()
        predictions.append((token, prob))
    return predictions

# Plotting Normal and Stress Testing Accuracies Separately
def plot_accuracy(normal_accuracies, stress_accuracies):
    plt.figure(figsize=(8, 6))
    models = ['GPT-2', 'Unsloth']
    plt.bar(models, normal_accuracies, color=['blue', 'green'])
    plt.ylim(0, 1)
    plt.ylabel('Accuracy', fontsize=12)
    plt.title('Normal Prompt Accuracy Comparison', fontsize=14)
    plt.grid(axis='y')
    plt.show()

    plt.figure(figsize=(8, 6))
    plt.bar(models, stress_accuracies, color=['cyan', 'lime'])
    plt.ylim(0, 1)
    plt.ylabel('Accuracy', fontsize=12)
    plt.title('Stress Testing Accuracy Comparison', fontsize=14)
```

```
    plt.grid(axis='y')
    plt.show()


# Main Testing Function
def test_models_on_prompts(gpt2_model, gpt2_tokenizer, gpt2_device,
                           unsloth_model, unsloth_tokenizer, unsloth_device,
                           prompts, ground_truths, top_k=3):
    gpt2_correct = 0
    unsloth_correct = 0


    for prompt, truth in zip(prompts, ground_truths):
        gpt2_preds = predict_next_words(prompt, gpt2_model, gpt2_tokenizer, gpt2_device,
top_k)
        unsloth_preds = predict_next_words(prompt, unsloth_model, unsloth_tokenizer,
unsloth_device, top_k)


        gpt2_predicted_words = [word.lower() for word, _ in gpt2_preds]
        unsloth_predicted_words = [word.lower() for word, _ in unsloth_preds]


        if truth.lower() in gpt2_predicted_words:
            gpt2_correct += 1
        if truth.lower() in unsloth_predicted_words:
            unsloth_correct += 1


    total = len(prompts)
    gpt2_accuracy_normal = gpt2_correct / total
    unsloth_accuracy_normal = unsloth_correct / total
```

```
# Stress Testing
stress_prompts = ["segdgegedd","Heo wrld","def calculate_sum(a, b)"]
stress_ground_truths = ["", "world", ":"]
gpt2_stress_correct = 0
unsloth_stress_correct = 0

print("\n===== Stress Testing Results =====")
for prompt, truth in zip(stress_prompts, stress_ground_truths):
    print(f"\nStress Test Prompt: '{prompt}'")
    gpt2_preds = predict_next_words(prompt, gpt2_model, gpt2_tokenizer, gpt2_device,
top_k)
    unsloth_preds = predict_next_words(prompt, unsloth_model, unsloth_tokenizer,
unsloth_device, top_k)
    print("GPT-2 Predictions:", gpt2_preds)
    print("Unsloth Predictions:", unsloth_preds)

    gpt2_predicted_words = [word.lower() for word, _ in gpt2_preds]
    unsloth_predicted_words = [word.lower() for word, _ in unsloth_preds]

    if truth and truth.lower() in gpt2_predicted_words:
        gpt2_stress_correct += 1
    if truth and truth.lower() in unsloth_predicted_words:
        unsloth_stress_correct += 1

total_stress = len(stress_prompts)
gpt2_accuracy_stress = gpt2_stress_correct / total_stress
```

```
    unsloth_accuracy_stress = unsloth_stress_correct / total_stress


    print("\n===== Accuracy Results =====")
    print(f"GPT-2 Normal Accuracy: {gpt2_correct}/{total} = {gpt2_accuracy_normal:.2%}")
    print(f"Unsloth Normal Accuracy: {unsloth_correct}/{total} = {unsloth_accuracy_normal:.2%}")
    print(f"GPT-2 Stress Accuracy: {gpt2_stress_correct}/{total_stress} = {gpt2_accuracy_stress:.2%}")
    print(f"Unsloth Stress Accuracy: {unsloth_stress_correct}/{total_stress} = {unsloth_accuracy_stress:.2%}")


    plot_accuracy([gpt2_accuracy_normal, unsloth_accuracy_normal], [gpt2_accuracy_stress, unsloth_accuracy_stress])


# Main Script
if __name__ == "__main__":
    # Load models
    gpt2_model, gpt2_tokenizer, gpt2_device = load_gpt2()
    unsloth_model, unsloth_tokenizer, unsloth_device = load_unsloth()


    # Define prompts and their ground truths
    prompts = [
        "The quick brown fox",
        "How are",
        "What are the cause of global",
        "Climate change continues to impact",
        "Do you want a cup of"
```

```
]

ground_truths = [
    "jumped",
    "you",
    "warming",
    "the",
    "tea"
]

# Run the tests
test_models_on_prompts(
    gpt2_model, gpt2_tokenizer, gpt2_device,
    unsloth_model, unsloth_tokenizer, unsloth_device,
    prompts,
    ground_truths
)
```