



Roll No.- 2024201043

Report on the Language Model Built:

Drive link for train and test perplexity files-

-  Ulysses
-  Pride&Prejudice

(*The file links are available on README.md as well)

FFNN:

1. Introduction

A Feedforward Neural Network (FFNN) is a foundational architecture in neural network models, where information moves unidirectionally from input to output layers without any cycles or loops. In the context of language modeling, FFNNs predict the probability of a word based on its preceding words, capturing the statistical properties of language sequences.

2. Model Architecture

The FFNN language model processes a fixed-size context of preceding words to predict the next word in a sequence. The architecture comprises:

- **Input Layer:** Represents a context of 'n' previous words, each transformed into dense vector embeddings.

- **Hidden Layers:** One or more fully connected layers that process the concatenated embeddings, introducing non-linear transformations to capture complex patterns in the data.
- **Output Layer:** A softmax layer that outputs a probability distribution over the vocabulary, indicating the likelihood of each word being the next in the sequence.

The model's parameters are optimized to minimize the difference between the predicted and actual next words in the training data.

3. Training and Evaluation

The FFNN language model was trained on two distinct literary corpora: *Ulysses* by James Joyce and *Pride and Prejudice* by Jane Austen. The training process involved adjusting the model's weights to minimize perplexity—a metric that measures how well the model predicts a sample.

Perplexity Trends:

- **Training Set:** The model typically achieves lower perplexity on the training set, as it has directly learned from this data.
- **Test Set:** Perplexity is generally higher on the test set, reflecting the model's performance on unseen data and its ability to generalize.

A significant gap between training and test perplexity may indicate overfitting, where the model learns the training data too well, including its noise, and fails to generalize to new data.

Context Size Impact:

- **Smaller Context (e.g., 3 words):**
 - Provides less information for prediction, often leading to higher perplexity.
 - However, a smaller context can sometimes generalize better, reducing overfitting.
- **Larger Context (e.g., 5 words):**
 - Expected to improve predictions by incorporating more context.
 - However, in our experiments, **the perplexity for 5-gram was slightly higher than for 3-gram**, indicating that adding more context did not necessarily lead to better generalization. This could be due to overfitting or sparsity issues in training data.

This suggests that **increasing context size does not always guarantee better performance**, and optimal context length depends on the dataset and model capacity.

4. Results

The FFNN language model was evaluated on both the *Ulysses* and *Pride and Prejudice* datasets. The results demonstrated the model's capability to learn and predict word sequences, with performance metrics aligning with expectations based on the respective complexities and styles of the texts.

5. Conclusion

The FFNN language model effectively captures the statistical properties of language sequences, making it a valuable tool for tasks like next-word prediction and text generation. While it has limitations in modeling

long-range dependencies due to its fixed context window, it serves as a foundational approach in the field of neural language modeling.

Predictions with FFNN:

- **For Context-size=3 (Pride & Prejudice)**

```
python3 ffnn_ankita.py -f "/home/ankita/Downloads/Pride and  
Prejudice - Jane Austen.txt" 10
```

Input sentence: I have a

Word: <unk> | Probability: 0.3568

Word: good | Probability: 0.1284

Word: the | Probability: 0.0993

Word: a | Probability: 0.0725

Word: he | Probability: 0.0633

Word: most | Probability: 0.0239

Word: lucas | Probability: 0.0223

Word: many | Probability: 0.0177

Word: friend | Probability: 0.0168

Word: thing | Probability: 0.0155

- **For Context-size=5 (Pride & Prejudice)**

```
python3 ffnn_ankita.py -f "/home/ankita/Downloads/Pride and  
Prejudice - Jane Austen.txt" 10
```

Input sentence: I have a

Word: <unk> | Probability: 0.3546

Word: the | Probability: 0.1214

Word: he | Probability: 0.0879

Word: without | Probability: 0.0861

Word: a | Probability: 0.0708

Word: here | Probability: 0.0252

Word: happy | Probability: 0.0216

Word: lady | Probability: 0.0184

Word: something | Probability: 0.0165

Word: man | Probability: 0.0143

- **For Context-size=3 (Ulysses)**

```
python3 ffn_ankita.py -f "/home/ankita/Downloads/Ulysses -  
James Joyce.txt" 10
```

Input sentence: The man gave

Word: <unk> | Probability: 0.9827

Word: the | Probability: 0.0010

Word: not | Probability: 0.0009

Word: looked | Probability: 0.0006

Word: through | Probability: 0.0005

Word: matter | Probability: 0.0004

Word: at | Probability: 0.0003

Word: boy | Probability: 0.0003

Word: is | Probability: 0.0003

Word: towards | Probability: 0.0003

- **For Context-size=5 (Ulysses)**

```
python3 ffnn_ankita.py -f "/home/ankita/Downloads/Ulysses -  
James Joyce.txt" 10
```

Input sentence: I saw a

Word: <unk> | Probability: 0.8417

Word: not | Probability: 0.0464

Word: to | Probability: 0.0284

Word: the | Probability: 0.0229

Word: is | Probability: 0.0126

Word: through | Probability: 0.0087

Word: were | Probability: 0.0067

Word: be | Probability: 0.0052

Word: made | Probability: 0.0017

Word: white | Probability: 0.0017

RNN:

1. Introduction

A **Recurrent Neural Network (RNN)** is designed for sequential data, making it well-suited for language modeling. Unlike Feedforward Neural Networks (FFNNs), RNNs maintain a **hidden state** that captures information from previous time steps, allowing them to model dependencies across sequences. This makes RNNs more effective at capturing context compared to FFNNs.

In this report, we analyze the performance of an RNN-based language model trained on two literary datasets:

- *Ulysses* by James Joyce
- *Pride and Prejudice* by Jane Austen

Our evaluation focuses on **perplexity**, a key metric in language modeling that measures how well the model predicts the next word in a sequence.

2. Model Architecture

The RNN language model consists of:

- **Embedding Layer (nn.Embedding)** – Converts words into dense vector representations.
- **Recurrent Layer (nn.RNN)** – Processes sequential input and maintains a hidden state to store past information.
- **Fully Connected Layer (nn.Linear)** – Outputs a probability distribution over the vocabulary for the next word prediction.
- **Dropout (nn.Dropout)** – Helps prevent overfitting by randomly deactivating neurons during training.

The model learns to **predict the next word** given a sequence of preceding words.

3. Training and Evaluation

The model was trained using **cross-entropy loss** and optimized with **AdamW**. The datasets were split into training and test sets to evaluate generalization.

Perplexity Trends

- **Training Set** – Perplexity is lower since the model learns directly from this data.
- **Test Set** – Perplexity is higher as the model encounters unseen word sequences.

A small gap between training and test perplexity suggests good generalization, while a large gap may indicate **overfitting**.

4. Context Size Impact

- **Smaller Context (e.g., 3 words):**
 - Provides limited information for predictions.
 - **In the Pride and Prejudice dataset, perplexity was slightly lower for context size 3 than for context size 5.** This suggests that a shorter context was sufficient to model the relatively simpler sentence structures in Jane Austen's writing.
- **Larger Context (e.g., 5 words):**
 - Offers more context, potentially improving predictions.
 - **In the Ulysses dataset, perplexity was lower for context size 5 than for context size 3.** This indicates that James

Joyce's complex and fragmented writing style benefits from a longer context to better capture dependencies.

5. Results

The RNN model was evaluated on both datasets, and the results showed:

- **Better performance with context size 5** than with context size 3.
- **Higher perplexity in Ulysses** compared to Pride and Prejudice, likely due to the more complex structure and vocabulary of *Ulysses*.
- A reasonable balance between training and test perplexity, indicating good generalization.

6. Conclusion

The RNN-based language model effectively learns sequential patterns, outperforming FFNNs in handling longer dependencies. The results confirm that:

- **Larger context sizes (5-grams) improve performance** by capturing more information.
- **Smaller context sizes (3-grams) lead to higher perplexity**, likely due to insufficient context.

While RNNs improve language modeling over FFNNs, they still struggle with **long-range dependencies** due to vanishing gradients. Advanced models like **LSTMs** or **Transformers** could further enhance performance.

Predictions with RNN:

- **Pride & Prejudice Corpus-**

```
python3 ffnn_ankita.py -r "/home/ankita/Downloads/Pride and  
Prejudice - Jane Austen.txt" 10
```

Input sentence: I gave a

Word: <unk> | Probability: 0.0696

Word: regiment | Probability: 0.0610

Word: under | Probability: 0.0425

Word: the | Probability: 0.0311

Word: family | Probability: 0.0282

Word: letter | Probability: 0.0189

Word: have | Probability: 0.0187

Word: then | Probability: 0.0166

Word: happy | Probability: 0.0145

Word: our | Probability: 0.0134

- **Ulysses Corpus-**

```
python3 ffnn_ankita.py -r "/home/ankita/Downloads/Ulysses -  
James Joyce.txt" 10
```

Input sentence: He gave me

Word: to | Probability: 0.0706

Word: the | Probability: 0.0663

Word: not | Probability: 0.0546

Word: is | Probability: 0.0417

Word: <unk> | Probability: 0.0412

Word: so | Probability: 0.0336

Word: without | Probability: 0.0319

Word: be | Probability: 0.0258

Word: our | Probability: 0.0256

Word: all | Probability: 0.0254

LSTM:

1. Introduction

Long Short-Term Memory (LSTM) networks are a type of **Recurrent Neural Network (RNN)** designed to address the vanishing gradient problem in standard RNNs. LSTMs introduce **gates (input, forget, and output gates)** to regulate the flow of information, allowing them to capture **long-range dependencies** more effectively.

In this report, we analyze the performance of an **LSTM-based language model** trained on two literary datasets:

- *Ulysses* by James Joyce
- *Pride and Prejudice* by Jane Austen

Our evaluation focuses on **perplexity**, a metric that indicates how well the model predicts the next word in a sequence.

2. Model Architecture

The LSTM language model consists of:

- **Embedding Layer (nn.Embedding)** – Converts words into dense vector representations.
- **LSTM Layer (nn.LSTM)** – Processes input sequences while maintaining a memory state, helping capture long-range dependencies.
- **Fully Connected Layer (nn.Linear)** – Outputs a probability distribution over the vocabulary for next-word prediction.
- **Dropout (nn.Dropout)** – Prevents overfitting by randomly deactivating neurons during training.

Unlike standard RNNs, **LSTMs effectively retain relevant information across longer sequences**, making them well-suited for language modeling.

3. Training and Evaluation

The model was trained using **cross-entropy loss** and optimized with **AdamW**. The datasets were split into training and test sets to evaluate generalization.

Perplexity Trends

- **Training Set** – The model achieves lower perplexity as it learns from the data.
- **Test Set** – Perplexity is generally higher, reflecting how well the model generalizes to unseen text.

A small difference between training and test perplexity suggests good generalization, whereas a large gap may indicate **overfitting**.

4. Context Size Impact

- **Smaller Context (e.g., 3 words):**
 - Provides limited information for prediction.
 - Resulted in **slightly higher perplexity** in both datasets, indicating that a shorter context was not always sufficient for accurate predictions.
- **Larger Context (e.g., 5 words):**
 - Offers more context, leading to better predictions.
 - **Perplexity was slightly lower for context size 5 compared to context size 3 in both *Pride and Prejudice* and *Ulysses*, confirming that LSTMs benefit from longer sequences.**

This result aligns with the **strength of LSTMs in capturing long-range dependencies**, where additional context provides **more meaningful information** for next-word prediction.

5. Results

The LSTM model was evaluated on both datasets, and the results showed:

- **Better performance with context size 5 across both corpora**, reinforcing the model's ability to handle longer dependencies.
- **Lower perplexity in *Pride and Prejudice* compared to *Ulysses***, likely due to the more structured and predictable nature of Austen's writing.
- A relatively small gap between training and test perplexity, suggesting good generalization.

6. Conclusion

The LSTM-based language model **outperforms standard RNNs** in handling sequential dependencies, making it a strong choice for language modeling tasks. Key findings include:

- **A longer context (5 words) improves predictions**, leading to slightly lower perplexity compared to a 3-word context in both datasets.
- **LSTMs effectively model complex text structures**, as seen in the better performance on *Ulysses* compared to simpler models.

Overall, the results demonstrate **LSTM's capability to learn and generalize from diverse text sources**, making it a valuable approach for next-word prediction and other NLP tasks.

Predictions with LSTM:

- **Pride & Prejudice Corpus-**

```
python3 ffnm_ankita.py -l "/home/ankita/Downloads/Pride and  
Prejudice - Jane Austen.txt" 10
```

Input sentence: The man was

Word: the | Probability: 0.1335

Word: <unk> | Probability: 0.0793

Word: colonel | Probability: 0.0642

Word: always | Probability: 0.0604

Word: long | Probability: 0.0412

Word: letter | Probability: 0.0343

Word: has | Probability: 0.0196

Word: tell | Probability: 0.0188

Word: turned | Probability: 0.0172

Word: come | Probability: 0.0156

- **Ulysses Corpus-**

```
python3 ffnn_ankita.py -l "/home/ankita/Downloads/Ulysses -  
James Joyce.txt" 10
```

Input sentence: The man was

Word: <unk> | Probability: 0.1119

Word: the | Probability: 0.1001

Word: our | Probability: 0.0598

Word: got | Probability: 0.0224

Word: same | Probability: 0.0203

Word: my | Probability: 0.0176

Word: is | Probability: 0.0146

Word: about | Probability: 0.0140

Word: when | Probability: 0.0113

Word: what | Probability: 0.0109