Report

Name: Abhinav Reddy Boddu

Roll No: 2021101034

> Note: Had to reduce hidden dimensions to 64 when running the models for Ulysses corpus since memory was not sufficient. Also, for ulysses corpus, it was run for less no of epochs.

corpus	model	epochs	train perplexity	test perplexity
p	f_3n	5	61.52	1690.59
p	f_3n	10	43.90	536.53
p	f_3n	25	32.62	1844.18
p	f_5n	5	47.19	268.75
p	f_5n	10	31.24	563.62
p	f_5n	20	22.77	1095.94
p	r	5	166.19	200.96
p	r	10	104.08	162.72
p	r	25	58.13	171.36
p	r	50	32.41	277.74
p	l	5	204.63	218.32
p	l	10	124.16	156.87
p	l	25	65.40	147.73
p	l	50	35.88	252.97
u	f_3n	5	131.71	602.20
u	f_3n	10	108.87	1098.72
u	f_3n	15	99.91	1439.41
u	f_5n	5	112.38	879.71
u	f_5n	10	99.05	1901.80
u	f_5n	15	93.92	2390.09
u	r	5	3701.54	4424.86
u	r	15	1579.64	3364.25
u	r	25	963.13	3417.59
u	l	5	7250.89	7246.16

corpus	model	epochs	train perplexity	test perplexity
u	l	10	2699.79	3979.54
u	l	20	1252.08	3281.59

Observations:

1. **Effect of Corpus Complexity**

- **Smaller Corpus (p):**
  - The models achieve significantly **lower perplexity** on the smaller corpus, as expected since simpler datasets are easier to model.
  - Even with increased epochs, models like RNN and LSTM maintain reasonable perplexity, indicating stable learning.
- **Complex Corpus (u):**
  - **Perplexity scores are much higher** across all models, reflecting the increased difficulty in modeling complex data.
  - The feedforward models (f_3n and f_5n) show **explosive growth in test perplexity** with more epochs, suggesting **overfitting** to the training data.

---

2. **Feedforward Models (f_3n and f_5n)**

- **On p:**
  - Increasing the **n-gram size from 3 to 5** improves training perplexity, suggesting better context capture.
  - However, the **test perplexity increases sharply after 10 epochs**, especially for f_3n (1844.18) and f_5n (1095.94), again indicating **overfitting**.
- **On u:**
  - Similar trends, but **even worse generalization**. For example, f_5n with 15 epochs has **test perplexity of 2390.09**, despite relatively low training perplexity (93.92).
  - The model memorizes training data but fails on unseen data, hinting at the **lack of sequence modeling capability** in feedforward architectures.

---

3. **RNN Models (r)**

- **On p:**
  - **Consistent improvement** in both train and test perplexity as epochs increase.
  - Test perplexity remains relatively low (171.36 at 25 epochs), showing good generalization compared to feedforward models.
- **On u:**

- **Extremely high perplexity scores** (e.g., 4424.86 at 5 epochs) but a **decreasing trend with more epochs**, which is promising.
- Indicates that RNNs handle complexity better but still struggle without more advanced regularization or tuning.

---

4. **LSTM Models (`l`)**

- **On p:**
  - Shows **better stability** compared to RNNs, with steady improvement as epochs increase.
  - Slight increase in test perplexity at 50 epochs (252.97) could suggest **mild overfitting**, but overall, LSTMs generalize well.
- **On u:**
  - **Very high perplexity** initially (7246.16 at 5 epochs) but **improves significantly with more epochs** (3281.59 at 20 epochs).
  - **Best performance on complex data** among all models, confirming LSTM's strength in handling long-term dependencies.

Comparison of Perplexity Scores Between Assignments

Model	Dataset p	Dataset u
l_1_train	338.63	461.62
l_1_test	317.75	393.52
l_3_train	1030.62	3213.13
l_3_test	1657.73	4707.75
l_5_train	1469.79	4547.52
l_5_test	2738.4	7938.85
g_1_train	340.18	470.62
g_1_test	317.8	395.91
g_3_train	5.91	15.81
g_3_test	7335.91	57963.17
g_5_train	4.1	7.21
g_5_test	134658.53	4868188.28
i_1_train	338.87	464.79
i_1_test	317.17	393.42
i_3_train	8.5	11.94
i_3_test	151.13	280.43

Model	Dataset p	Dataset u
i_5_train	2.16	5.58
i_5_test	845.41	1183.44

📊 **Overall Trends:**

1. **Current Neural Models (RNN, LSTM, FFNN):**
    - **Lower perplexity on simpler corpus (p)**, but **struggle with complex corpus (u)**, especially the feedforward models.
    - **LSTMs outperform RNNs and FFNNs** on complex data, thanks to their ability to capture long-term dependencies.
2. **Previous Assignment (N-gram Models with Smoothing Techniques):**
    - **Very high perplexity for larger n-grams** in Good-Turing (`g_5_test` ~ 4.86 million) and Laplace (`l_5_test` ~ 7,938), especially on complex data.
    - **Linear Interpolation (i)** consistently performs better than both Laplace and Good-Turing, with reasonable perplexity even for higher n-grams.

🏆 Model Ranking (Best to Worst Performance):

1. **LSTM Model (Current Assignment)**
    - Strong performance on complex data (`u` corpus).
    - Handles longer sentences effectively due to memory capabilities.
2. **RNN Model (Current Assignment)**
    - Decent performance but suffers on complex data compared to LSTM.
3. **Linear Interpolation (i) N-gram Model**
    - Low perplexity, especially for small n-grams.
    - **Best generalization** for both datasets for shorter sentences.
4. **Feedforward Neural Network (FFNN, Current Assignment)**
    - Prone to **overfitting**, especially on complex corpus (`u`).
5. **Good-Turing N-gram Model (Previous Assignment)**
    - **Severe overfitting** for higher n-grams, causing test perplexity to skyrocket.
6. **Laplace Smoothing N-gram Model (Previous Assignment)**
    - Better than Good-Turing for higher n-grams but still suffers from high test perplexity.

Detailed Analysis

Performance Differences & Potential Reasons:

- **Neural Models (RNN, LSTM):**
  - **Advantage:** Capable of capturing **sequential dependencies** beyond fixed window sizes (n-grams).
  - **Issue:** Require more data and careful tuning; FFNNs especially struggle with longer contexts.
- **N-gram Models (Laplace, Good-Turing, Linear Interpolation):**
  - **Advantage:** Simpler to implement, perform well on small datasets with proper smoothing (linear interpolation shines here).
  - **Issue: High sparsity** with large n-grams leads to unreliable probabilities and high perplexity.

Performance for Longer Sentences:

- **Better Performer: LSTM**
  - LSTM can maintain information over long sequences due to **gating mechanisms**, reducing the vanishing gradient problem that affects RNNs and making it superior to n-gram models that are limited by the fixed context window.

Effect of N-gram Size on FFNN Performance:

- **Smaller N-grams (3-gram):** FFNN performs moderately well, capturing short dependencies effectively.
- **Larger N-grams (5-gram):**
  - FFNN shows signs of **overfitting** as model complexity increases but doesn't gain much in generalization.
  - **Reason:** FFNNs lack sequence-awareness beyond the input window, unlike RNNs/LSTMs, leading to poor adaptation for longer contexts.

Key Takeaways:

- **Linear Interpolation** (N-gram) remains highly effective for simpler datasets.
- **Overfitting** is a major issue for feedforward models, especially as the number of epochs increases.
- **RNNs** perform better on sequential data but struggle with complex datasets unless fine-tuned.
- **LSTM** is the **best neural model** for complex data and longer sentences.
- **FFNNs** need regularization or additional context-handling mechanisms to improve generalization.
- **Model choice should be based on data complexity** and the nature of the sequences being modeled.
- **Model choice depends on data complexity:**
  - For simple data: Feedforward models can suffice with proper regularization.
  - For complex data: RNNs or LSTMs are necessary, with LSTMs preferred for long-term dependencies.