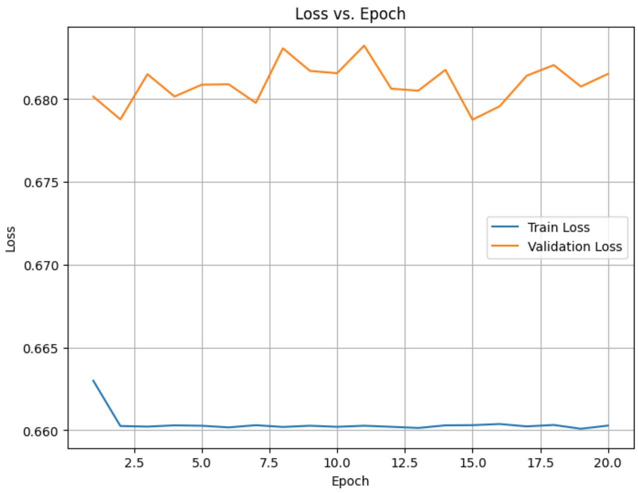
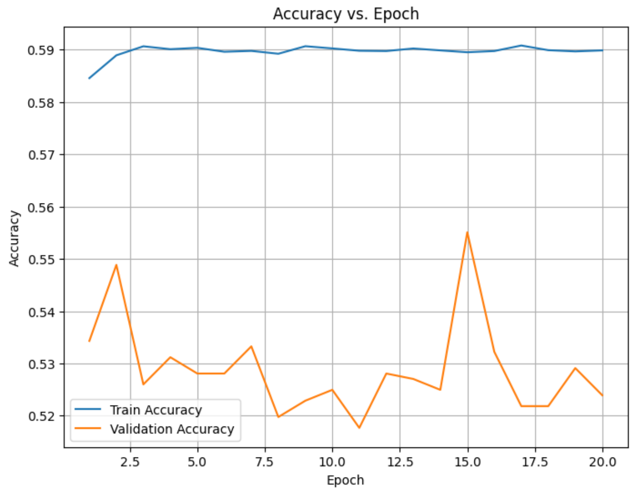
Gal Cesana 318510633
*Authorization has been granted to submit the assignment individually

Natural Language Processing – Exercise 4

Model 1 – Log Linear with One Hot:

Plots:

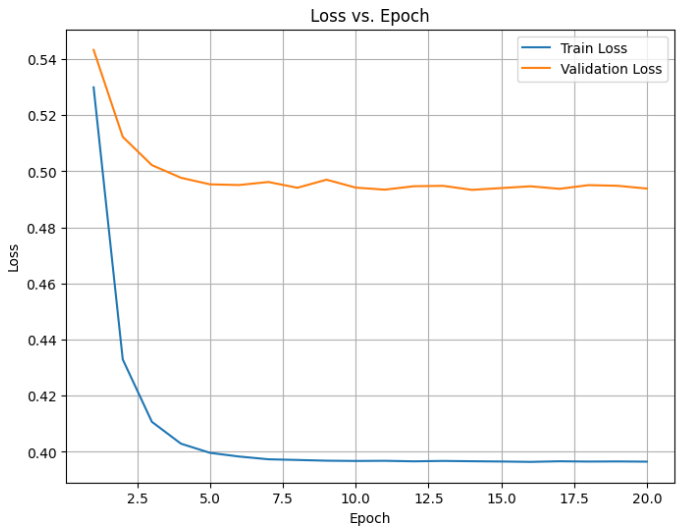


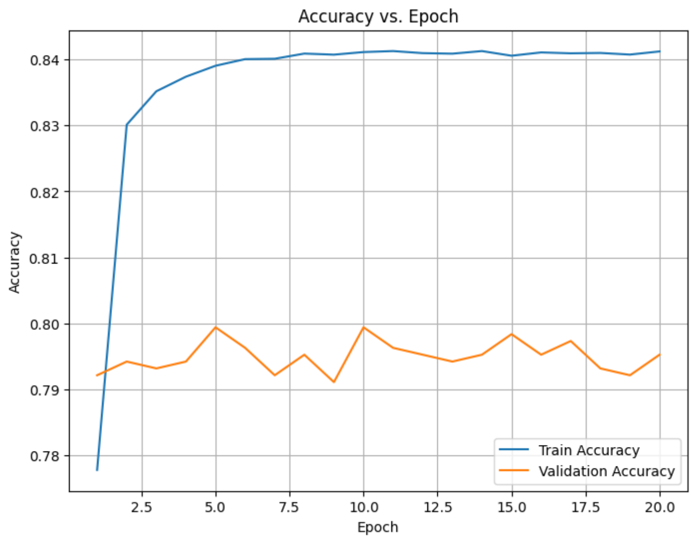


Final Test Performance -> Loss: 0.6791, Accuracy: 0.5229

Model 2 – Log Linear with Word2Vec:

Plots:

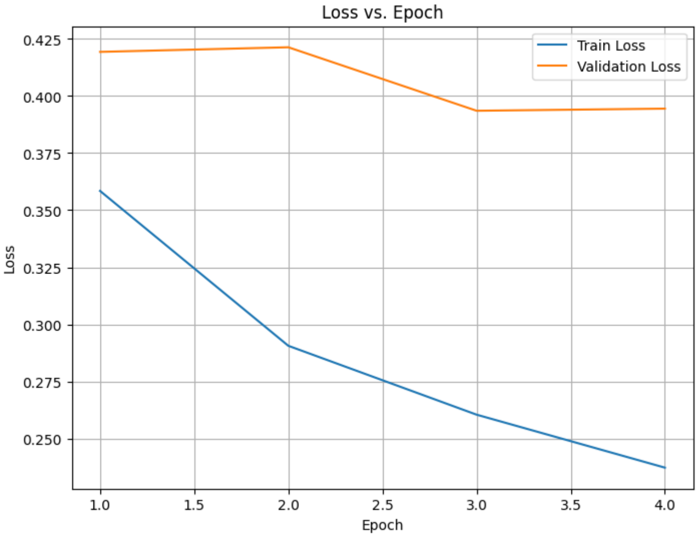


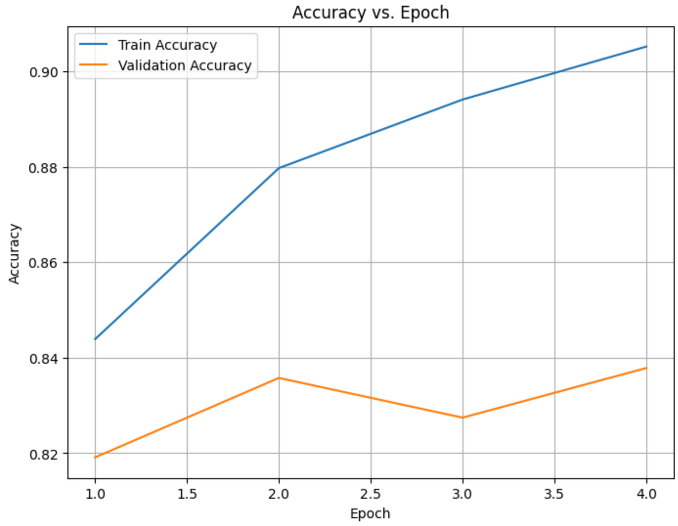


Final Test Performance -> Loss: 0.4686, Accuracy: 0.8212

Gal Cesana 318510633
*Authorization has been granted to submit the assignment individually

Model 3 — LSTM with Word2Vec:





Final Test Performance -> Loss: 0.3432, Accuracy: 0.8586

Model 4 — Transformer:

Final Test Performance -> Loss: 0.2043, Accuracy: 0.9116

Putting it all together:

Model	Validation Loss	Validation Accuracy	Test Loss	Test Accuracy	Negated Polarity Loss	Negated Polarity Accuracy	Rare Words Loss	Rare Words Accuracy
One-Hot Log-Linear	0.6815	0.5239	0.6791	0.5229	0.7005	0.4839	0.7411	0.28
2v Log-Linear	0.4939	0.7952	0.4686	0.8212	0.7507	0.5806	0.6442	0.76
w2v LSTM	0.3945	0.8378	0.3432	0.8586	0.8394	0.6774	0.5597	0.78
Transformer	0.2415	0.9064	0.2043	0.9116	0.4782	0.7903	0.411	0.84

Gal Cesana 318510633
*Authorization has been granted to submit the assignment individually

$Q1$: Compare the results (test accuracy, validation accuracy) you've received for the simple log-linear model, and the Word2Vec log-linear model. Which one performs better? Provide a possible explanation for the results you have.

Validation Accuracy:

- One-Hot Log-Linear: 52.39%
- Word2Vec Log-Linear: 79.52%

Word2Vec log-linear performs significantly better in validation accuracy.

Test Accuracy:

- One-Hot Log-Linear: 52.29%
- Word2Vec Log-Linear: 82.12%

Word2Vec log-linear also outperforms the One-Hot model on test accuracy.

Possible Explanation for the Results:

- **Representation of Features**:
  The One-Hot encoding used in the One-Hot Log-Linear model represents words as sparse vectors, which fail to capture semantic relationships between words. Each word is treated as completely distinct, leading to poorer generalization.

In contrast, the Word2Vec log-linear model uses pre-trained Word2Vec embeddings, which are dense vector representations that capture semantic and contextual relationships between words. This richer representation helps the model understand the meaning and context of words better.

- **Dimensionality**:
  One-Hot vectors are high-dimensional, which can lead to overfitting or difficulty in learning meaningful patterns with limited data. Word2Vec embeddings reduce the dimensionality and represent meaningful relationships in a more compact and effective way.
- **Transfer Learning**:
  Word2Vec embeddings benefit from pre-training on large corpora, which provides the model with prior knowledge about language. This advantage helps the model perform better on downstream tasks.

In summary, the Word2Vec log-linear model leverages more informative word embeddings, leading to its superior performance compared to the simpler One-Hot Log-Linear model.

$Q2$: Compare the latter results with the results of the LSTM and Transformer models. Which perform better? Provide an explanation for the results you received.

Performance Comparison:

1. Validation Accuracy:

- **Word2Vec Log-Linear**: 79.52%
- **LSTM**: 83.78%
- **Transformer**: 90.64%

Ranking: Transformer > LSTM > Word2Vec Log-Linear

2. Test Accuracy:

- **Word2Vec Log-Linear**: 82.12%
- **LSTM**: 85.86%
- **Transformer**: 91.16%

Ranking: Transformer > LSTM > Word2Vec Log-Linear

The **Transformer** model achieves the best performance, followed by the **LSTM**, and then the **Word2Vec Log-Linear** model.

Both the LSTM and Transformer significantly outperform the Word2Vec Log-Linear model in validation and test accuracy.

Explanation for the Results:

1. Sequential Data Processing:

- The LSTM and Transformer are **sequence models**, meaning they are designed to capture dependencies between words over time or position, which is crucial for tasks like language modeling or text classification.
- The Word2Vec Log-Linear model does not explicitly account for the sequence of words. While it uses word embeddings, it lacks mechanisms to model the relationships between words in a sequence.

2. Long-Term Dependencies:

- The LSTM is specifically designed to capture **long-term dependencies** in sequential data using mechanisms like forget gates and memory cells. This enables it to learn better contextual relationships compared to the Word2Vec Log-Linear model.
- However, the LSTM can struggle with very long sequences or tasks requiring complex contextual understanding.

3. Transformers and Attention Mechanisms:

- The Transformer uses a **self-attention mechanism**, which allows it to directly model relationships between all words in a sequence, regardless of their distance.

This is a major advantage over LSTMs, which process sequences step-by-step and may lose context over long distances.

- The ability of Transformers to capture both local and global dependencies efficiently leads to their superior performance.

4. Pre-trained Representations:

- Both the LSTM and Transformer likely benefit from dense word embeddings (like Word2Vec). However, the Transformer's architecture makes better use of these embeddings by modeling complex relationships between words.
- The LSTM's performance may be slightly lower due to its stepwise processing and reliance on recurrent mechanisms.

$Q3$: Last, compare the results that all the models had on the 2 special subsets of sentences we've provided you. For each subset, state the model that has the highest result (and the lowest result) and provide a possible explanation for these results.

1. Negated Polarity Subset:

Accuracy:

- **Highest**: **Transformer (79.03%)**
- **Lowest**: **One-Hot Log-Linear (48.39%)**

Loss:

- **Lowest (Best)**: **Transformer (0.4782)**
- **Highest (Worst)**: **One-Hot Log-Linear (0.7005)**

Explanation:

Negated polarity sentences are complex because they require understanding negation words like "not," "never," or "no," and how they modify the sentiment or meaning of the sentence.

1. **Transformer (Best)**:
    - The Transformer excels due to its **self-attention mechanism**, which effectively captures relationships between negation words and the words they modify, regardless of their position in the sentence. This global context-awareness makes it highly capable of handling negation complexities.
2. **One-Hot Log-Linear (Worst)**:
    - The One-Hot Log-Linear model treats words as independent features and lacks any mechanism to understand contextual or syntactical relationships. This makes it unable to correctly interpret the effect of negation, resulting in poor performance.

3. **Intermediate Performers**:
    - The **Word2Vec Log-Linear** and **LSTM** models show better results than the One-Hot model but underperform compared to the Transformer. Word2Vec embeddings provide some semantic understanding, and LSTMs add sequence modeling, but they lack the comprehensive context modeling of the Transformer.

---

2. Rare Words Subset:

Accuracy:

- **Highest**: **Transformer (84%)**
- **Lowest**: **One-Hot Log-Linear (28%)**

Loss:

- **Lowest (Best)**: **Transformer (0.411)**
- **Highest (Worst)**: **One-Hot Log-Linear (0.7411)**

Explanation:

Handling rare words requires either pre-trained embeddings with broad vocabulary coverage or the ability to infer meaning from context.

1. **Transformer (Best)**:
    - The Transformer achieves the highest accuracy because of its **ability to infer meaning from surrounding words** through self-attention. Even if a rare word is not well-represented in embeddings, the Transformer can rely on contextual cues from the sequence to interpret its role.
2. **One-Hot Log-Linear (Worst)**:
    - The One-Hot Log-Linear model performs poorly because rare words are represented as isolated dimensions in the sparse vector space. If these words are not adequately represented in the training data, the model cannot generalize to them effectively.
3. **Intermediate Performers**:
    - The **Word2Vec Log-Linear** and **LSTM** models benefit from embeddings (like Word2Vec), which provide meaningful representations for many rare words. However, if the embeddings for rare words are not well-trained, their performance will still lag behind the Transformer.