Assignment 2 - Q5

324827328, 213713522

April 2025

Method for Identifying the Top 5 Trigrams per Filter

To interpret the learned character-level CNN filters, we aim to find which character trigrams (sequences of three characters) most strongly activate each filter. The process is as follows:

1. **Extract Filter Weights:** For each filter in the convolutional layer, we obtain its weight matrix, which has dimensions corresponding to the character embedding size and the kernel size (typically 3 for trigrams).
2. **Enumerate All Possible Trigrams:** We generate all possible combinations of three characters from a predefined character set (e.g., lowercase English letters, hyphen, period, and apostrophe).
3. **Embed Each Trigram:** For each trigram, we look up the embedding vector for each character using the model's character embedding matrix. This results in a matrix of shape (*embedding dimension* $\times$ 3) for each trigram.
4. **Compute Filter Response:** For a given filter and trigram, we compute the filter's response by taking the element-wise product of the trigram's embedding matrix and the filter's weight matrix, and then summing all elements. Mathematically, for filter F and trigram embedding E, the response is:

$$\text{Response} = \sum_{i=1}^{d} \sum_{j=1}^{3} F_{i,j} \cdot E_{i,j}$$

   where d is the embedding dimension.
5. **Rank Trigrams:** For each filter, we compute the absolute value of the response for all possible trigrams and rank them in descending order.
6. **Select Top Trigrams:** The top 5 trigrams with the highest absolute response values are selected as those that most strongly activate the filter.

This method allows us to interpret each filter by identifying the specific character patterns it is most sensitive to, providing insight into the morphological or orthographic features the model has learned to recognize.

Analysis of NER Model Results: Effect of Number of Filters

Parameters

Learning rate: 0.0001
Batch size: 128
Hidden layer size: 100
Chars embedding size: 32
Dropout rate: 0.3
Optimizer: Adam
Early stopping patience: 10

Accuracy Trends

The first three plots display the NER dev accuracy (excluding the 'O' tag) over 100 epochs for each filter setting:

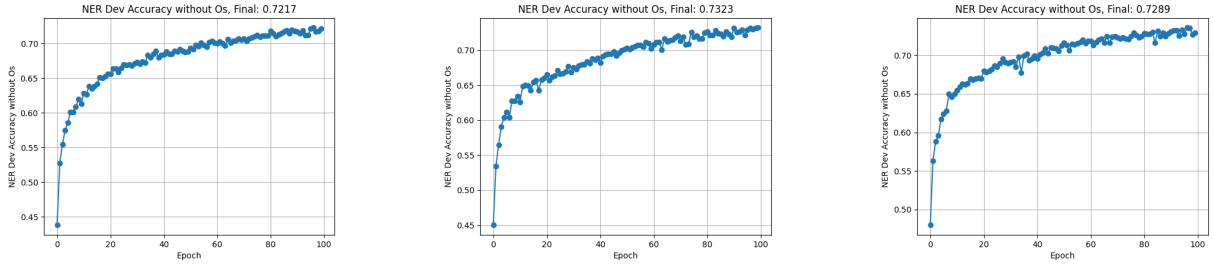


Figure 1: NER Dev Accuracy

- **20 filters:** Final dev accuracy ≈ 0.7217
- **30 filters:** Final dev accuracy ≈ 0.7323
- **40 filters:** Final dev accuracy ≈ 0.7289

All models show a rapid increase in accuracy during the initial epochs, followed by a slower, steady improvement as training progresses. The model with 30 filters achieves the highest final accuracy, while 20 and 40 filters yield slightly lower but comparable results.

Loss Trends

The next three plots show the dev loss for each filter setting:

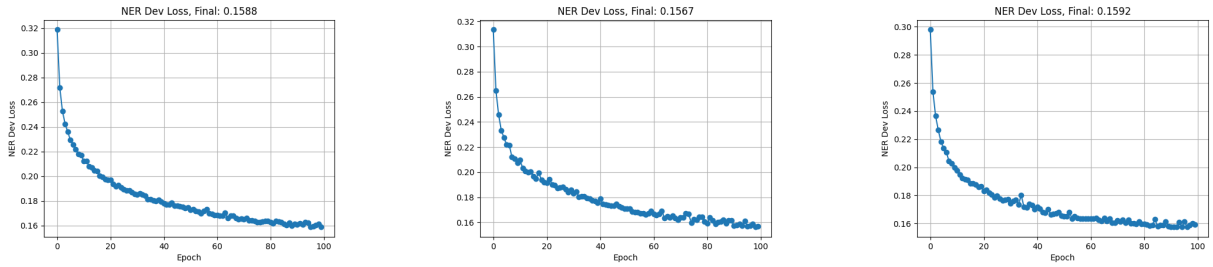


Figure 2: NER Dev Loss

- **20 filters:** Final dev loss ≈ 0.1588
- **30 filters:** Final dev loss ≈ 0.1567
- **40 filters:** Final dev loss ≈ 0.1592

The loss curves decrease smoothly and consistently for all filter settings, indicating stable training and effective learning. The lowest final loss is observed for the model with 30 filters, which aligns with the highest accuracy observed in the previous section.

Effect of Varying the Number of Filters

- **Fewer filters (20):** The model is less expressive and may not capture all relevant character-level patterns, leading to slightly lower accuracy and higher loss compared to the optimal setting.
- **More filters (40):** Increasing the number of filters beyond the optimal point does not further improve performance and may even slightly degrade it. This could be due to overfitting, increased model complexity, or redundancy among filters.
- **Optimal number (30):** The model with 30 filters achieves the best balance, capturing sufficient morphological diversity without unnecessary complexity. This setting yields the highest dev accuracy and lowest dev loss.

Filter analysis for ther ner trained model using window size of 30

Top 5 trigrams for filter 7:

- ovx Response: -4.06
- 'vx Response: -3.92
- tvx Response: -3.71
- ovr Response: -3.67
- onx Response: -3.63

The filter appears to find a set of characters where x is the last char or v is the middle char.

Top 5 trigrams for filter 10:

- er' Response: -4.43
- es' Response: -4.42
- gr' Response: -4.12
- gs' Response: -4.10
- wr' Response: -4.10

The filter appears to find a set of characters where ' is the last char or r is the middle char.

Top 5 trigrams for filter 0:

- t'i Response: -4.36
- t'. Response: -4.29
- tji Response: -4.08
- t'z Response: -4.05
- tj. Response: -4.01

The filter appears to find a set of characters where t is the first char.

Top 5 trigrams for filter 9:

- vzo Response: -5.32
- zzo Response: -5.09
- -zo Response: -4.95
- bzo Response: -4.88

- .zo Response: -4.79

The filter appears to find a set of characters where z is the middle char.

Top 5 trigrams for filter 25:

- oxg Response: 3.81
- ovg Response: 3.74
- ohg Response: 3.66
- vyq Response: -3.55
- odg Response: 3.49

The filter appears to find a set of characters where i is the first char or g is the last char.

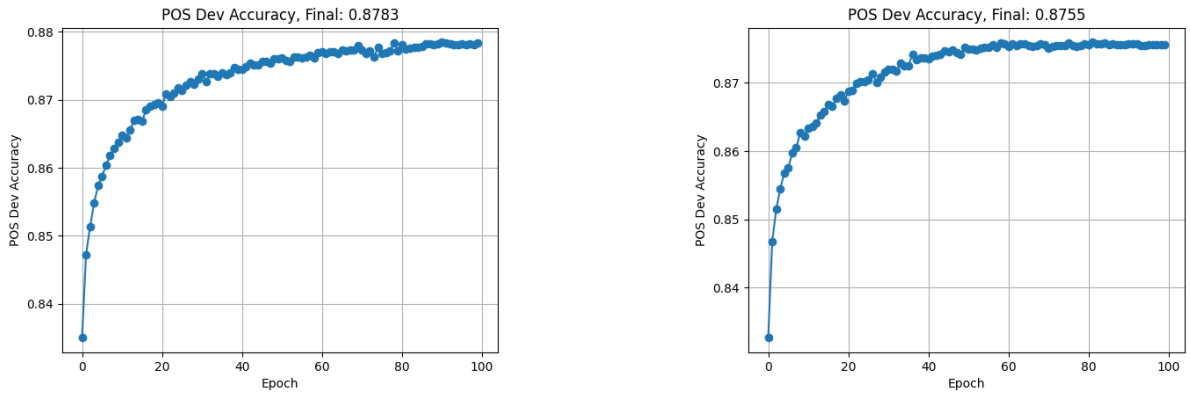
Analysis of POS Model Results: Effect of Number of Filters

Parameters

Learning rate: 0.0001
Batch size: 128
Hidden layer size: 100
Chars embedding size: 32
Dropout rate: 0.3
Optimizer: Adam
Early stopping patience: 10

Accuracy Trends

The first two plots display the POS tagging development accuracy over 100 epochs for each filter setting:



(a) 30 filters

(b) 40 filters

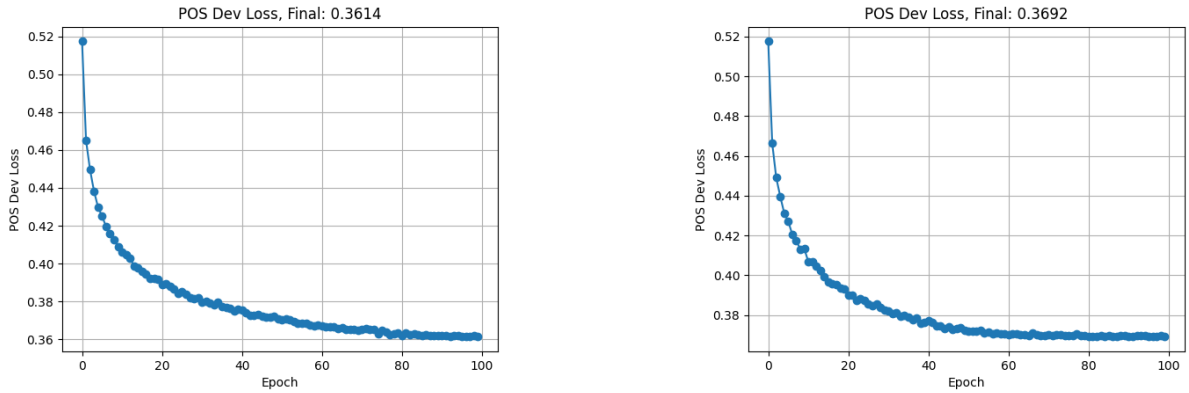
Figure 3: POS Dev Accuracy

- **30 filters (baseline):** Final dev accuracy = 0.8783
- **40 filters:** Final dev accuracy = 0.8755

Both models show similar learning patterns with rapid improvement in the early epochs (particularly in the first 20 epochs), followed by a more gradual increase and eventual plateau. The model with 30 filters achieves a slightly higher final accuracy (0.8783) compared to the model with 40 filters (0.8755), indicating that increasing the number of filters doesn't necessarily improve performance for this task.

Loss Trends

The next two plots show the development loss for each filter setting:



(a) 30 filters

(b) 40 filters

Figure 4: POS Dev Loss

- **30 filters (baseline):** Final dev loss = 0.3614
- **40 filters:** Final dev loss = 0.3692

Both loss curves show a steep decrease in the initial epochs, with the rate of improvement slowing as training progresses. The model with 30 filters achieves a slightly lower final loss (0.3614) compared to the model with 40 filters (0.3692), consistent with the accuracy results.

Effect of Varying the Number of Filters

- **Baseline (30 filters):** This configuration provides the optimal balance for the POS tagging task, achieving both higher accuracy and lower loss. It captures sufficient character-level patterns without introducing unnecessary complexity.
- **Increased filters (40):** Despite having more capacity to represent character-level features, the model with 40 filters performs slightly worse in terms of both accuracy and loss. This suggests that:
  - The additional filters may introduce redundancy in feature representation
  - The increased parameter count could lead to slight overfitting
  - The optimal representation capacity for POS tagging may be achieved with fewer filters

Comparison with NER Task

Unlike the NER task where 30 filters was optimal (compared to 20 or 40), for POS tagging we only compared 30 vs. 40 filters. However, we can observe that:

- POS tagging generally achieves higher accuracy (0.87-0.88) compared to NER (0.72-0.73), indicating that POS tagging is a comparatively easier task

- For both tasks, increasing filters beyond 30 did not improve performance, suggesting that there may be a sweet spot in model complexity regardless of the specific sequence labeling task
- The learning curves for POS tagging appear slightly smoother than those for NER, potentially indicating that POS tagging has more consistent patterns to learn

Conclusion

For the POS tagging task, the baseline model with 30 filters outperforms the variant with 40 filters, achieving a higher dev accuracy (0.8783 vs. 0.8755) and lower dev loss (0.3614 vs. 0.3692). This suggests that increasing model complexity by adding more convolutional filters does not necessarily lead to performance improvements. The character-level patterns relevant for POS tagging appear to be adequately captured with 30 filters, and additional filters may introduce unnecessary complexity without providing additional discriminative power. This finding aligns with the principle of parsimony in model design, where simpler models are preferred when they perform equally well or better than more complex alternatives.