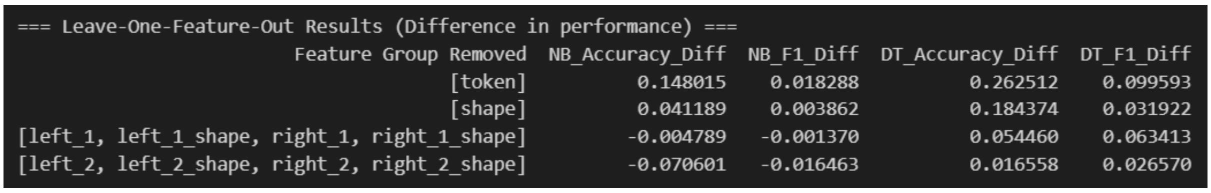
Leave-One-Feature-Out Results (Difference in performance)



```
=== Leave-One-Feature-Out Results (Difference in performance) ===
                      Feature Group Removed  NB_Accuracy_Diff  NB_F1_Diff  DT_Accuracy_Diff  DT_F1_Diff
                                    [token]          0.148015    0.018288          0.262512    0.099593
                                    [shape]          0.041189    0.003862          0.184374    0.031922
[left_1, left_1_shape, right_1, right_1_shape]      -0.004789   -0.001370          0.054460    0.063413
[left_2, left_2_shape, right_2, right_2_shape]      -0.070601   -0.016463          0.016558    0.026570
```

Q1. Which is the best machine learning algorithm (classifier) for this task (for both the baseline and the improved classifier)?

Based on the evaluation results, Decision Tree is the better-performing classifier for both the baseline and the improved systems. In the improved model, the Decision Tree achieved higher scores in all performance metrics compared to Naive Bayes. Specifically, its F1-score and accuracy on both the validation and test sets were consistently superior. Also, it showed greater sensitivity to feature ablation, indicating it makes more nuanced use of the available features.

Q2. Which attributes contributed the most to each of the performance metrics for your improved model? Which contributed the least?

For Decision Tree, the feature that contributed the most was the token, with the largest drop in both accuracy (0.2625) and F1-score (0.0996) when removed. This makes sense, as the token itself carries the primary lexical information.

In contrast, left_2 and right_2 context features contributed the least for Decision Tree, with only marginal performance degradation (0.0166 accuracy, 0.0266 F1), and similarly for Naive Bayes. Interestingly, for Naive Bayes, removing the left_2 group even improved performance slightly, suggesting an overfitting.

Q3. How good is your feature set for this task (for each algorithm)?

The feature set is well optimized for the Decision Tree classifier, as shown by the strong baseline performance and the clear drop when key features are removed. The combination of token, shape, and immediate context forms a robust set of attributes that helps the model learns complex decision boundaries.

For Naive Bayes, the feature set is still effective but less impactful, likely due to the independence assumptions of the model. The model did not benefit as much from wider context or token shape, and showed a weaker correlation between feature removal and performance drop.

Q4. If you had more time to work on this problem and do it more efficiently (in terms of performance), which features/text representation would you choose?

If I had more time and computational resources, I would explore character-level and subword features, such as affixes, suffixes, and prefixes, which helps capture morphological patterns like tense, case, or gender. Additionally, I would consider using pretrained word embeddings, which encode semantic information and generalize better across different contexts.