# Model Evaluation

July 11, 2019

```python
[48]: import numpy as np
      import pandas as pd

      import os
      print(os.listdir("."))
```

```
['.ipynb_checkpoints', 'dev_NLI_B.tsv', 'Model Evaluation.ipynb',
'test_ep_1.txt', 'test_ep_2.txt', 'test_ep_3.txt', 'test_ep_4.txt']
```

```python
[90]: test_orig = pd.read_csv('dev_NLI_B.tsv', sep='\t')
      test_orig.head()
```

```
[90]:       id                              sentence1  polarity  \
      0  1262    Tienda de Autoservicio. Siempre bien  Positive
      1  1262  Tienda de Autoservicio. Siempre bien.  Positive
      2  1262    Tienda de Autoservicio. Siempre bien      None
      3  1262  Tienda de Autoservicio. Siempre bien.      None
      4  1262    Tienda de Autoservicio. Siempre bien  Negative

                         context   target   aspect  label
      0  Tienda de Autoservicio  general  general      1
      1  Tienda de Autoservicio  general  general      1
      2  Tienda de Autoservicio  general  general      0
      3  Tienda de Autoservicio  general  general      0
      4  Tienda de Autoservicio  general  general      0
```

```python
[91]: from glob import glob

      test_models = [pd.read_csv(f, sep=' ', header=None, usecols=[0]) for f in␣
       ↪glob('test_ep_*.txt')]
      for i, t in enumerate(test_models):
          t.columns = ['label_pred_{0}'.format(i)]

      test_model = pd.concat(test_models, axis = 1)
      test_model.head()
```

```
[91]:    label_pred_0  label_pred_1  label_pred_2  label_pred_3
      0             1             1             1             1
```

```
1               1             1             1             1
2               0             0             0             0
3               0             0             0             0
4               0             0             0             0
```

```
[92]: test = pd.concat([test_model, test_orig], axis = 1)
      test.head()
```

```
[92]:    label_pred_0  label_pred_1  label_pred_2  label_pred_3    id  \
      0             1             1             1             1  1262
      1             1             1             1             1  1262
      2             0             0             0             0  1262
      3             0             0             0             0  1262
      4             0             0             0             0  1262

                                  sentence1  polarity                 context  \
      0    Tienda de Autoservicio. Siempre bien  Positive  Tienda de Autoservicio
      1  Tienda de Autoservicio. Siempre bien.  Positive  Tienda de Autoservicio
      2    Tienda de Autoservicio. Siempre bien      None  Tienda de Autoservicio
      3  Tienda de Autoservicio. Siempre bien.      None  Tienda de Autoservicio
      4    Tienda de Autoservicio. Siempre bien  Negative  Tienda de Autoservicio

          target   aspect  label
      0  general  general      1
      1  general  general      1
      2  general  general      0
      3  general  general      0
      4  general  general      0
```

```
[93]: test['y_real'] = np.select([(test['aspect'] == 'general') & (test['polarity']
      → == 'Positive') & (test['label'] == 0),
                                  (test['aspect'] == 'general') &
      → (test['polarity'] == 'Negative') & (test['label'] == 0),
                                  (test['aspect'] == 'general') &
      → (test['polarity'] == 'None') & (test['label'] == 0),
                                  (test['aspect'] == 'servicio') &
      → (test['polarity'] == 'Positive') & (test['label'] == 0),
                                  (test['aspect'] == 'servicio') &
      → (test['polarity'] == 'Negative') & (test['label'] == 0),
                                  (test['aspect'] == 'servicio') &
      → (test['polarity'] == 'None') & (test['label'] == 0),
                                  (test['aspect'] == 'ambiente') &
      → (test['polarity'] == 'Positive') & (test['label'] == 0),
                                  (test['aspect'] == 'ambiente') &
      → (test['polarity'] == 'Negative') & (test['label'] == 0),
                                  (test['aspect'] == 'ambiente') &
      → (test['polarity'] == 'None') & (test['label'] == 0),
```

```python
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'Positive') & (test['label'] == 0),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'Negative') & (test['label'] == 0),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'None') & (test['label'] == 0),
                                (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'Positive') & (test['label'] == 0),
                                (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'Negative') & (test['label'] == 0),
                                (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'None') & (test['label'] == 0),
                                (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'Positive') & (test['label'] == 0),
                                (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'Negative') & (test['label'] == 0),
                                (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'None') & (test['label'] == 0),

                                (test['aspect'] == 'general') &␣
↪(test['polarity'] == 'Positive') & (test['label'] == 1),
                                (test['aspect'] == 'general') &␣
↪(test['polarity'] == 'Negative') & (test['label'] == 1),
                                (test['aspect'] == 'general') &␣
↪(test['polarity'] == 'None') & (test['label'] == 1),
                                (test['aspect'] == 'servicio') &␣
↪(test['polarity'] == 'Positive') & (test['label'] == 1),
                                (test['aspect'] == 'servicio') &␣
↪(test['polarity'] == 'Negative') & (test['label'] == 1),
                                (test['aspect'] == 'servicio') &␣
↪(test['polarity'] == 'None') & (test['label'] == 1),
                                (test['aspect'] == 'ambiente') &␣
↪(test['polarity'] == 'Positive') & (test['label'] == 1),
                                (test['aspect'] == 'ambiente') &␣
↪(test['polarity'] == 'Negative') & (test['label'] == 1),
                                (test['aspect'] == 'ambiente') &␣
↪(test['polarity'] == 'None') & (test['label'] == 1),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'Positive') & (test['label'] == 1),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'Negative') & (test['label'] == 1),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'None') & (test['label'] == 1),
                                (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'Positive') & (test['label'] == 1),
```

```
                                    (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'Negative') & (test['label'] == 1),
                                    (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'None') & (test['label'] == 1),
                                    (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'Positive') & (test['label'] == 1),
                                    (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'Negative') & (test['label'] == 1),
                                    (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'None') & (test['label'] == 1),
                               ],
                                    ['GP0', 'GN0', 'G-0',
                                     'SP0', 'SN0', 'S-0',
                                     'AP0', 'AN0', 'A-0',
                                     '$P0', '$N0', '$-0',
                                     'CP0', 'CN0', 'C-0',
                                     'UP0', 'UN0', 'U-0',

                                     'GP1', 'GN1', 'G-1',
                                     'SP1', 'SN1', 'S-1',
                                     'AP1', 'AN1', 'A-1',
                                     '$P1', '$N1', '$-1',
                                     'CP1', 'CN1', 'C-1',
                                     'UP1', 'UN1', 'U-1'])
```

```
[103]: for k in test.keys():
           if 'label_pred_' in k:
               test['y_' + k] = np.select([(test['aspect'] == 'general') &␣
       ↪(test['polarity'] == 'Positive') & (test[k] == 0),
                                   (test['aspect'] == 'general') &␣
       ↪(test['polarity'] == 'Negative') & (test[k] == 0),
                                   (test['aspect'] == 'general') &␣
       ↪(test['polarity'] == 'None') & (test[k] == 0),
                                   (test['aspect'] == 'servicio') &␣
       ↪(test['polarity'] == 'Positive') & (test[k] == 0),
                                   (test['aspect'] == 'servicio') &␣
       ↪(test['polarity'] == 'Negative') & (test[k] == 0),
                                   (test['aspect'] == 'servicio') &␣
       ↪(test['polarity'] == 'None') & (test[k] == 0),
                                   (test['aspect'] == 'ambiente') &␣
       ↪(test['polarity'] == 'Positive') & (test[k] == 0),
                                   (test['aspect'] == 'ambiente') &␣
       ↪(test['polarity'] == 'Negative') & (test[k] == 0),
                                   (test['aspect'] == 'ambiente') &␣
       ↪(test['polarity'] == 'None') & (test[k] == 0),
```

```
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'Positive') & (test[k] == 0),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'Negative') & (test[k] == 0),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'None') & (test[k] == 0),
                                (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'Positive') & (test[k] == 0),
                                (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'Negative') & (test[k] == 0),
                                (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'None') & (test[k] == 0),
                                (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'Positive') & (test[k] == 0),
                                (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'Negative') & (test[k] == 0),
                                (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'None') & (test[k] == 0),

                                (test['aspect'] == 'general') &␣
↪(test['polarity'] == 'Positive') & (test[k] == 1),
                                (test['aspect'] == 'general') &␣
↪(test['polarity'] == 'Negative') & (test[k] == 1),
                                (test['aspect'] == 'general') &␣
↪(test['polarity'] == 'None') & (test[k] == 1),
                                (test['aspect'] == 'servicio') &␣
↪(test['polarity'] == 'Positive') & (test[k] == 1),
                                (test['aspect'] == 'servicio') &␣
↪(test['polarity'] == 'Negative') & (test[k] == 1),
                                (test['aspect'] == 'servicio') &␣
↪(test['polarity'] == 'None') & (test[k] == 1),
                                (test['aspect'] == 'ambiente') &␣
↪(test['polarity'] == 'Positive') & (test[k] == 1),
                                (test['aspect'] == 'ambiente') &␣
↪(test['polarity'] == 'Negative') & (test[k] == 1),
                                (test['aspect'] == 'ambiente') &␣
↪(test['polarity'] == 'None') & (test[k] == 1),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'Positive') & (test[k] == 1),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'Negative') & (test[k] == 1),
                                (test['aspect'] == 'precio') &␣
↪(test['polarity'] == 'None') & (test[k] == 1),
                                (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'Positive') & (test[k] == 1),
```

```
                                     (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'Negative') & (test[k] == 1),
                                     (test['aspect'] == 'comida') &␣
↪(test['polarity'] == 'None') & (test[k] == 1),
                                     (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'Positive') & (test[k] == 1),
                                     (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'Negative') & (test[k] == 1),
                                     (test['aspect'] == 'ubicaciÃn') &␣
↪(test['polarity'] == 'None') & (test[k] == 1),
                                  ],
                                       ['GP0', 'GN0', 'G-0',
                                        'SP0', 'SN0', 'S-0',
                                        'AP0', 'AN0', 'A-0',
                                        '$P0', '$N0', '$-0',
                                        'CP0', 'CN0', 'C-0',
                                        'UP0', 'UN0', 'U-0',

                                        'GP1', 'GN1', 'G-1',
                                        'SP1', 'SN1', 'S-1',
                                        'AP1', 'AN1', 'A-1',
                                        '$P1', '$N1', '$-1',
                                        'CP1', 'CN1', 'C-1',
                                        'UP1', 'UN1', 'U-1'])
```

```python
[120]: from sklearn.metrics import confusion_matrix
       from sklearn.utils.multiclass import unique_labels
       import matplotlib.pyplot as plt
       from matplotlib.pyplot import figure
       import math
       from matplotlib.pyplot import figure
       import seaborn as sns
       sns.set(style='darkgrid')

       def plot_confusion_matrix(y_true, y_pred, classes, title="", cmap=plt.cm.Blues,␣
        ↪clean=False, figsize=(20, 16), dpi=300):

           cm = confusion_matrix(y_true, y_pred)
           cm_norm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] * 100

           if clean:
               # indexes of No's '0'  and  None's '-'
               indexes = [i for i, c in enumerate(classes) if c.endswith('0') or  '-'␣
        ↪in c]

               cm = np.delete(cm, indexes, axis=0)
               cm = np.delete(cm, indexes, axis=1)
```

```python
        cm_norm = np.delete(cm_norm, indexes, axis=0)
        cm_norm = np.delete(cm_norm, indexes, axis=1)

        classes = np.delete(classes, indexes, axis=0)

    fig, ax = plt.subplots(figsize=figsize, dpi=dpi)

    im = ax.imshow(cm_norm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    ax.grid(False)

    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           xticklabels=classes,
           yticklabels=classes,
           ylabel='True label',
           xlabel='Predicted label',
           title="Precisión promedio = {0:.2f} %".format(np.mean(cm.
 diagonal()))) if clean else title)

    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
 rotation_mode="anchor")

    fmt = 'd'
    fmt_norm = '.2f'

    thresh = 50

    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            if cm[i, j] == 0:
                continue
            ax.text(j, i, '\n' + format(cm[i, j], fmt), fontsize=8,
                    ha="center",  va="top",
                    color="white" if cm_norm[i, j] > thresh else "black")

            if not math.isnan(cm_norm[i, j]):
                ax.text(j, i, format(cm_norm[i, j], fmt_norm) + '%', fontsize=8,
                        ha="center",  va="bottom",
                        color="white" if cm_norm[i, j] > thresh else "black")


    fig.tight_layout()
    return ax
```
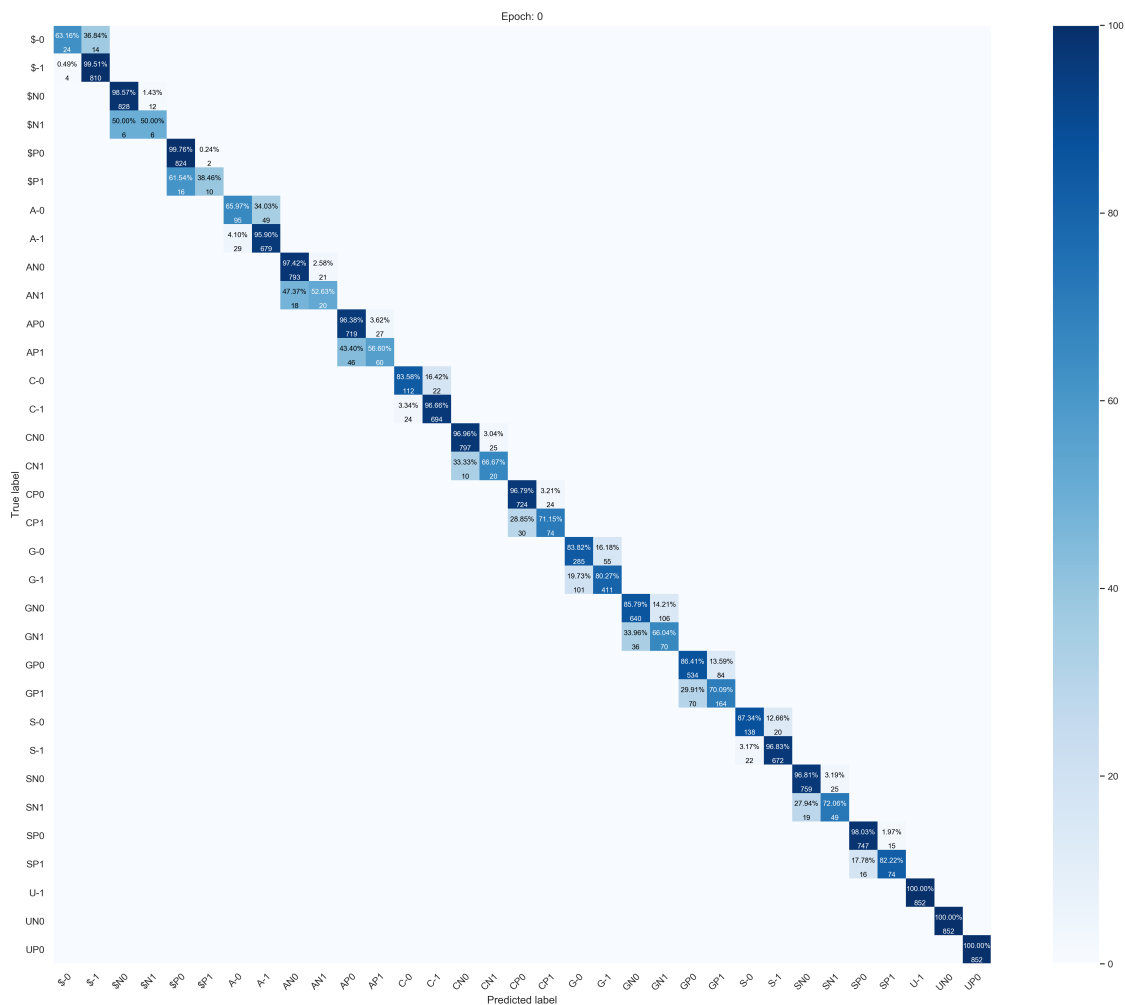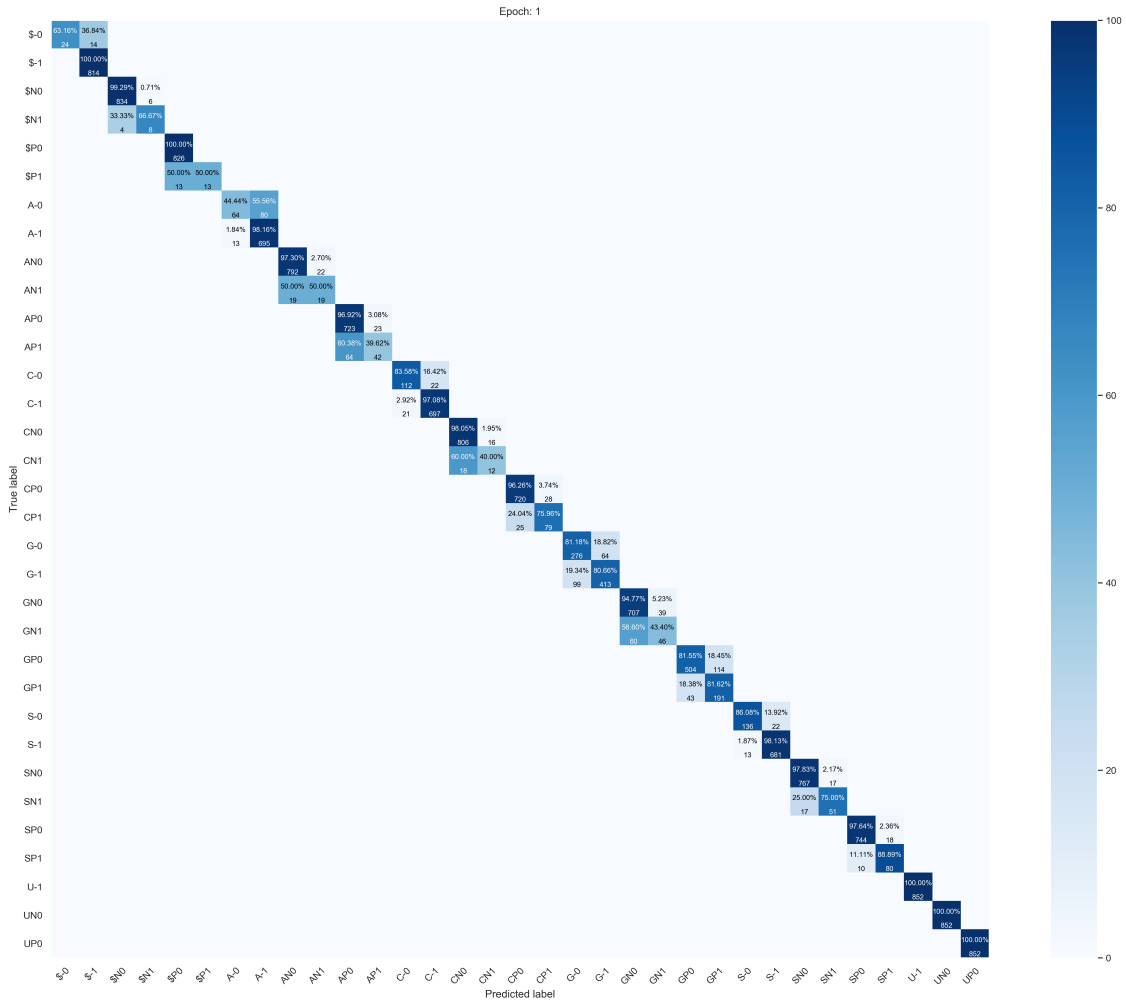
```
[107]: y_real = test['y_real'].values
       y_preds = {}

       for k in test.keys():
           if 'y_label_pred_' in k:
               y_preds[k] = test[k].values
```
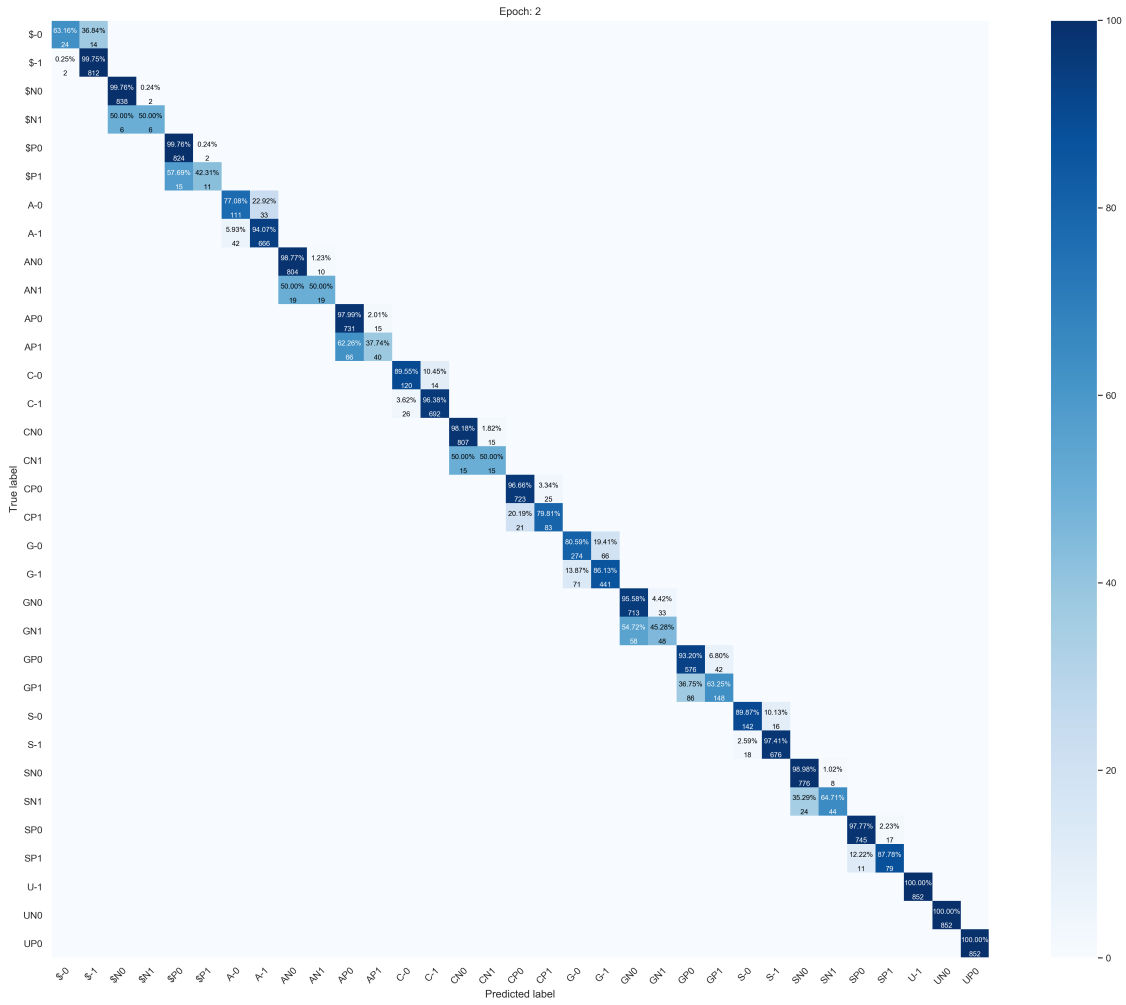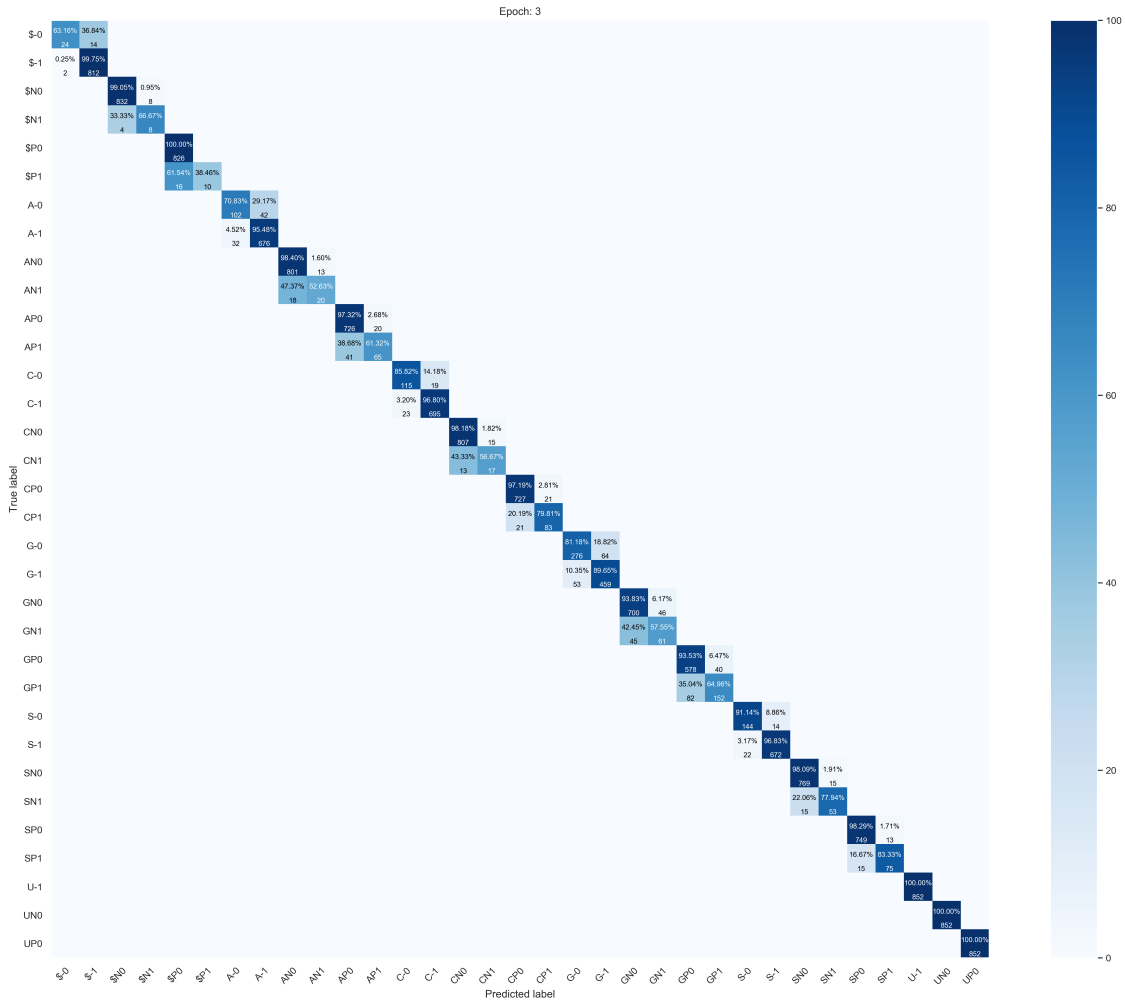
```
[126]: for k in test.keys():
           if 'y_label_pred_' in k:
               y_pred = y_preds[k]
               k = k.replace('y_label_pred_', '')
               plot_confusion_matrix(y_real, y_pred, classes=unique_labels(y_pred),␣
       ↪title="Epoch: {0}".format(k))
```
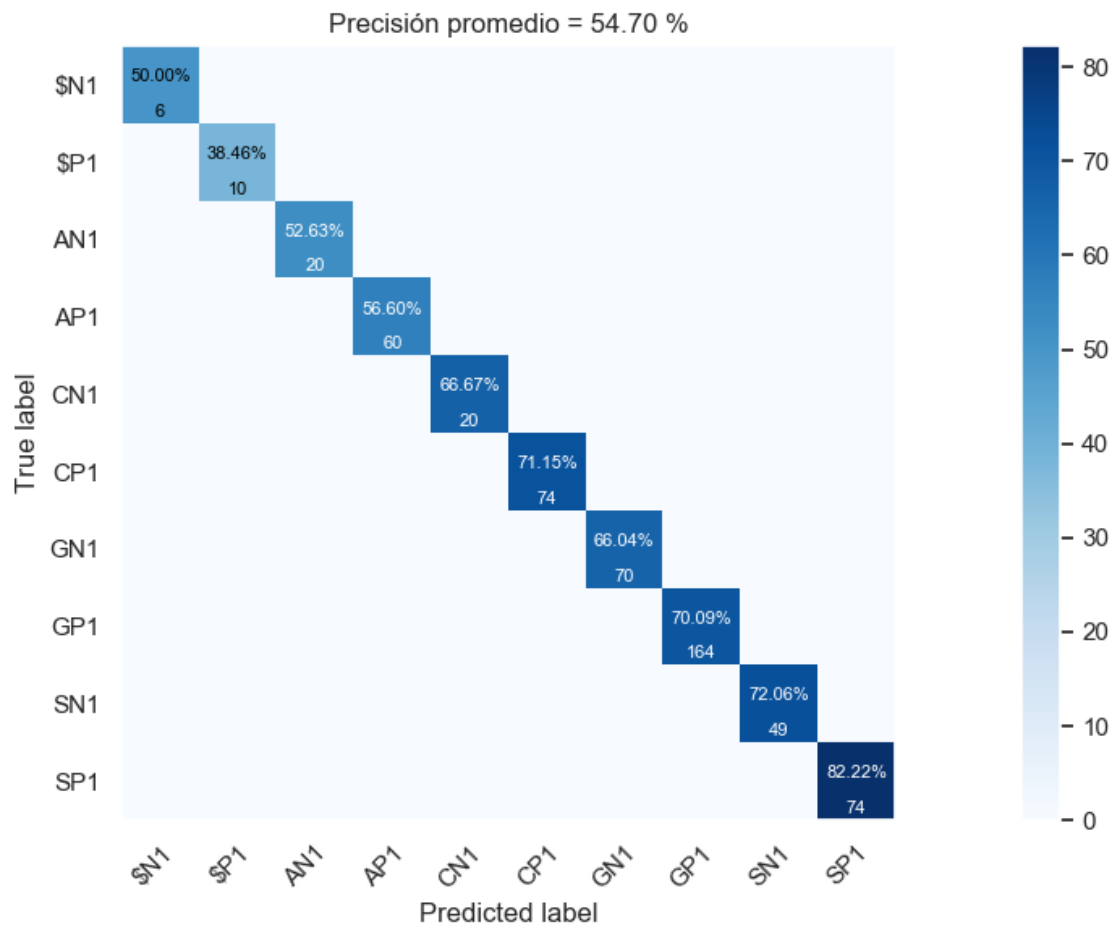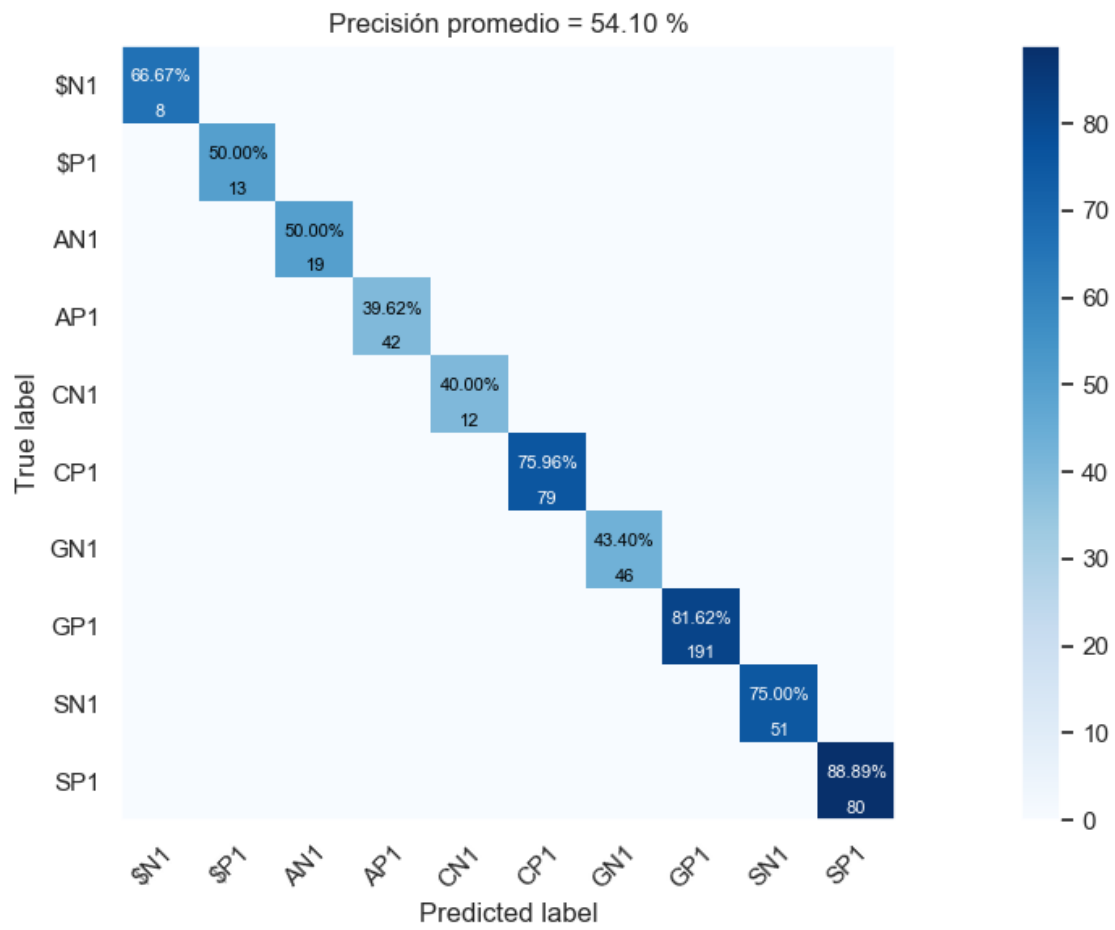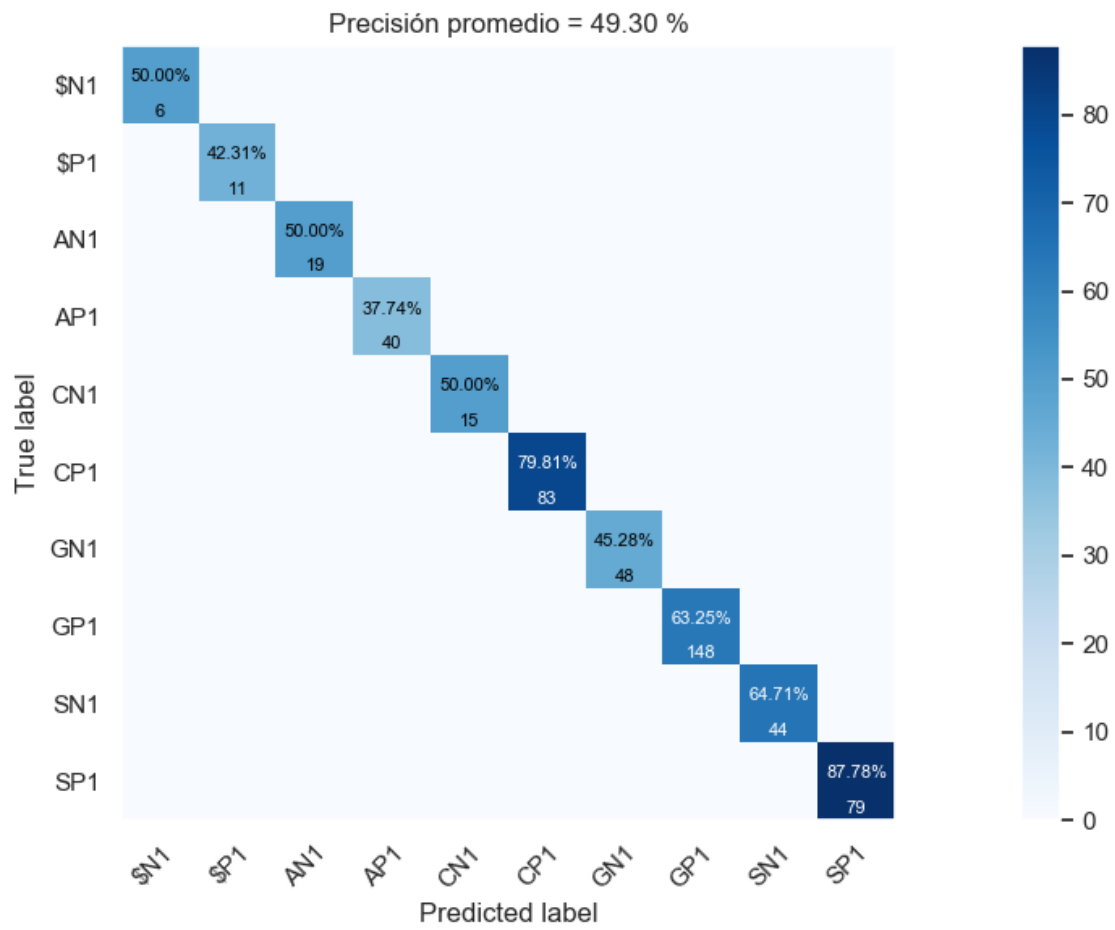
Epoch: 1

Epoch: 2

Epoch: 3
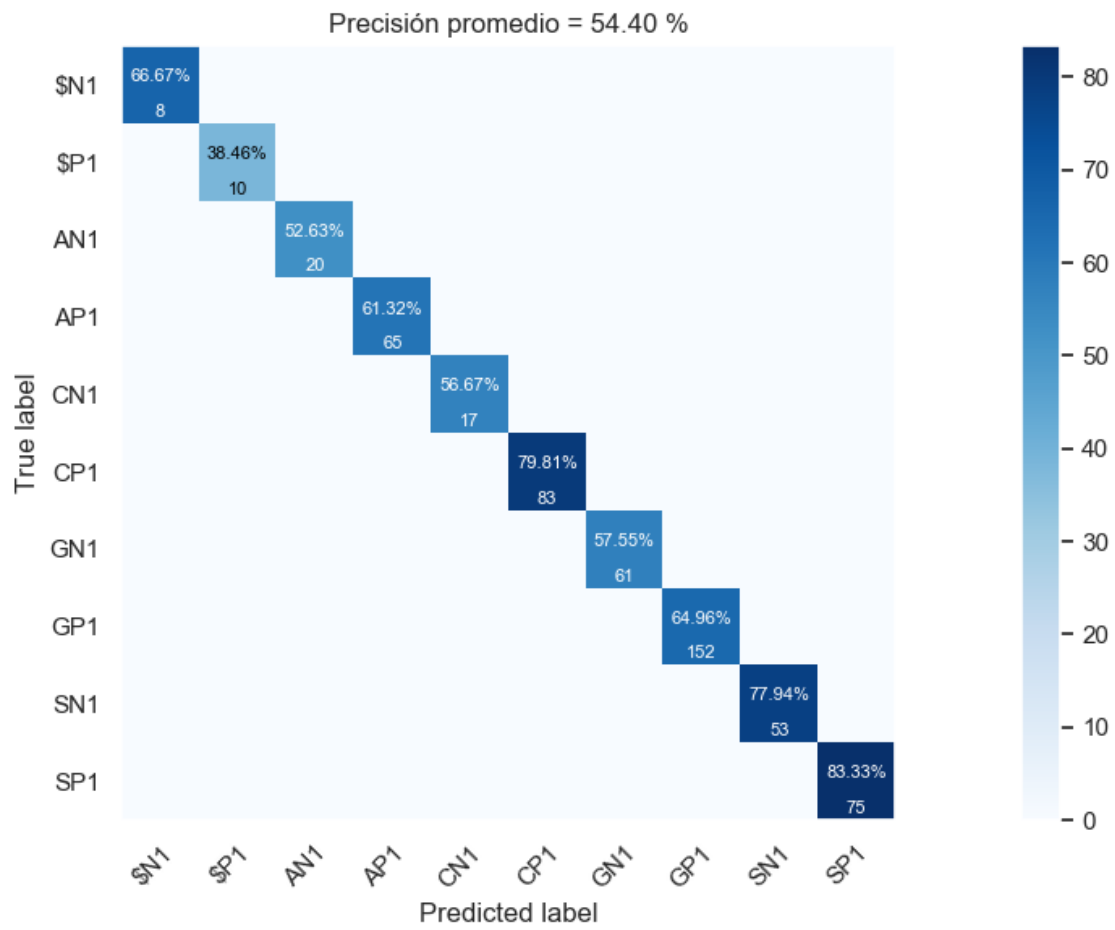
```
[127]: for k in test.keys():
           if 'y_label_pred_' in k:
               y_pred = y_preds[k]
               k = k.replace('y_label_pred_', '')
               plot_confusion_matrix(y_real, y_pred, classes=unique_labels(y_pred),␣
       ↪clean=True, figsize=(16, 6), dpi=100)
```

Precisión promedio = 54.70 %

Precisión promedio = 54.10 %

Precisión promedio = 49.30 %

Precisión promedio = 54.40 %

[ ]: