entropy_distributions

August 27, 2020

1 Analysis of borrowing – entropy method

1.1 Based on word tables and using recurrent neural networks

[1]:
```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:80% !important; }</style>"))

# Do this once in the kernel.
import sys
sys.path.append('/src/pybor')

# Set to automatically reload.
%load_ext autoreload
%autoreload 2

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

```
<IPython.core.display.HTML object>
```

[2]:
```
import tensorflow as tf
from pybor.data import LexibankDataset
import pybor.entropies as entropies
import pybor.evaluate as evaluate
import pybor.neural as detector
import pybor.wold as wold
```

2 Duplicate plot routines for notebook control

[3]:
```
from matplotlib import pyplot as plt

def plot_model_measures(
        measures=None,
        title=None,
```

```
        output_path=None,
        ):
    ## From Keras example
    print('measures', measures.keys())
    # Plot training & validation cross_entropy values
    plt.plot(measures['categorical_crossentropy'][1:])
    if 'val_categorical_crossentropy' in measures.keys():
        plt.plot(measures['val_categorical_crossentropy'][1:])
    if title:
        plt.title(title)
    plt.ylabel('Model Crossentropy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')
    plt.show()

    # Plot training & validation accuracy values
    plt.plot(measures['categorical_accuracy'][1:])
    if 'val_categorical_accuracy' in measures.keys():
        plt.plot(measures['val_categorical_accuracy'][1:])
    if title:
        plt.title(title)
    plt.ylabel('Model Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper right')
    plt.show()

    if output_path is not None:
        plt.savefig(output_path, dpi=600)

    plt.close()
```

[4]:
```
from matplotlib import pyplot as plt
import statistics
import math
import numpy as np

def plot_entropies(
    entropies1=None,
    entropies2=None,
    output_path=None,
    title='',
    label1='',
    label2='',
    graph_limit=None):


    cnt1 = f"{len(entropies1):6d}"
```

```
    avg1 = f"{statistics.mean(entropies1):6.3f}"
    std1 = f"{statistics.stdev(entropies1):6.3f}"
    cnt2 = f"{len(entropies2):6d}"
    avg2 = f"{statistics.mean(entropies2):6.3f}"
    std2 = f"{statistics.stdev(entropies2):6.3f}"

    # Drop most extreme values to determine graph limit.
    entropies = sorted(entropies1 + entropies2)
    upper_limit = graph_limit if graph_limit is not None else math.
↪ceil(entropies[-4])
    lower_limit = math.floor(entropies[0])

    # Set frame horizontal for this measure.
    bins = np.linspace(lower_limit, upper_limit, 60)
    plt.figure(figsize=(8, 5))
    plt.hist(
        entropies1,
        bins,
        alpha=0.65,
        label=label1
        + r"$(n="
        + cnt1
        + ", \mu="
        + avg1
        + ", \sigma="
        + std1
        + ")$",
        color="blue",
    )
    plt.hist(
        entropies2,
        bins,
        alpha=0.65,
        label=label2
        + r"$(n="
        + cnt2
        + ", \mu="
        + avg2
        + ", \sigma="
        + std2
        + ")$",
        color="red",
    )
    plt.grid(axis="y", alpha=0.8)
    plt.legend(loc="upper right")

    plt.xlabel("Entropies")
```

```
    plt.ylabel("Frequency")
    plt.title(title)

    if output_path:
        plt.savefig(output_path, dpi=600)

    plt.show()
    plt.close()
```

3 Plot entropy distributions corresponding to neural detection model.

[5]:
```
def plot_entropy_distributions(neural=None):
    # Get corresponding data and models from pybor.

    # Plot training data from native and loan - native basis.
    native_train_tokens_ids = neural.native_data.get_data_tokens_ids(neural.
↪native_data.training)
    loan_train_tokens_ids = neural.loan_data.get_data_tokens_ids(neural.
↪loan_data.training)
    native_train_entropies_native = neural.native_model.
↪calculate_entropies(native_train_tokens_ids)
    loan_train_entropies_native = neural.native_model.
↪calculate_entropies(loan_train_tokens_ids)
    plot_entropies(native_train_entropies_native, loan_train_entropies_native,
                   title="Inherited versus borrowed training entropies -␣
↪Inherited basis",
                   label1='Inherited training entropies', label2='Borrowed␣
↪training entropies',
                   output_path = "./output/train-"+neural.native_model.model.
↪name)


    #Plot testing data from native and loan - native basis.
    native_test_tokens_ids = neural.native_data.get_data_tokens_ids(neural.
↪native_data.testing)
    loan_test_tokens_ids = neural.loan_data.get_data_tokens_ids(neural.
↪loan_data.testing)
    native_test_entropies_native = neural.native_model.
↪calculate_entropies(native_test_tokens_ids)
    loan_test_entropies_native = neural.native_model.
↪calculate_entropies(loan_test_tokens_ids)
    plot_entropies(native_test_entropies_native, loan_test_entropies_native,
```

```
                    title="Inherited versus borrowed testing entropies -␣
↪Inherited basis",
                    label1='Inherited testing entropies', label2='Borrowed␣
↪testing entropies',
                    output_path = "./output/test-"+neural.native_model.model.
↪name)


    if neural.loan_model is None: return  # Not dual model.

    native_train_entropies_loan = neural.loan_model.
↪calculate_entropies(native_train_tokens_ids)
    loan_train_entropies_loan = neural.loan_model.
↪calculate_entropies(loan_train_tokens_ids)

    native_test_entropies_loan = neural.loan_model.
↪calculate_entropies(native_test_tokens_ids)
    loan_test_entropies_loan = neural.loan_model.
↪calculate_entropies(loan_test_tokens_ids)

    plot_entropies(native_train_entropies_native, native_train_entropies_loan,
                    title="Inherited training entropies - Inherited versus␣
↪borrowed basis",
                    label1='Inherited basis', label2='Borrowed basis',
                    output_path = "./output/train-native-"+neural.native_model.
↪model.name+
                    neural.loan_model.model.name)


    plot_entropies(native_test_entropies_native, native_test_entropies_loan,
                    title="Inherited testing entropies - Inherited versus␣
↪borrowed basis",
                    label1='Inherited basis', label2='Borrowed basis',
                    output_path = "./output/test-native-"+neural.native_model.
↪model.name+
                    neural.loan_model.model.name)


    plot_entropies(loan_train_entropies_native, loan_train_entropies_loan,
                    title="Borrowed training entropies - Inherited versus␣
↪borrowed basis",
                    label1='Inherited basis', label2='Borrowed basis',
                    output_path = "./output/train-loan-"+neural.native_model.
↪model.name+
                    neural.loan_model.model.name)
```

```
    plot_entropies(loan_test_entropies_native, loan_test_entropies_loan,
                   title="Borrowed testing entropies - Inherited versus␣
↪borrowed basis",
                   label1='Inherited basis', label2='Borrowed basis',
                   output_path = "./output/test-loan-"+neural.native_model.
↪model.name+
                   neural.loan_model.model.name)

    # *** Entropy difference distributions.  Only this matters! ***

    loan_delta_train = [n - l for n, l in␣
↪zip(loan_train_entropies_native,loan_train_entropies_loan)]
    native_delta_train = [n - l for n, l in␣
↪zip(native_train_entropies_native,native_train_entropies_loan)]
    loan_delta_test = [n - l for n, l in␣
↪zip(loan_test_entropies_native,loan_test_entropies_loan)]
    native_delta_test = [n - l for n, l in␣
↪zip(native_test_entropies_native,native_test_entropies_loan)]

    plot_entropies(loan_delta_train, native_delta_train,
                   title="Training entropy deltas - Inherited versus borrowed␣
↪basis",
                   label1='Borrowed word deltas', label2='Inherited word␣
↪deltas',
                   output_path = "./output/train-delta-"+neural.native_model.
↪model.name+
                   neural.loan_model.model.name)

    plot_entropies(loan_delta_test, native_delta_test,
                   title="Testing entropy deltas - Inherited versus borrowed␣
↪basis",
                   label1='Borrowed word deltas', label2='Inherited word␣
↪deltas',
                   output_path = "./output/test-delta-"+neural.native_model.
↪model.name+
                   neural.loan_model.model.name)
```

4 Custom version of examples/neural_example.py

[11]:
```
import pickle
from pathlib import Path

import attr
```

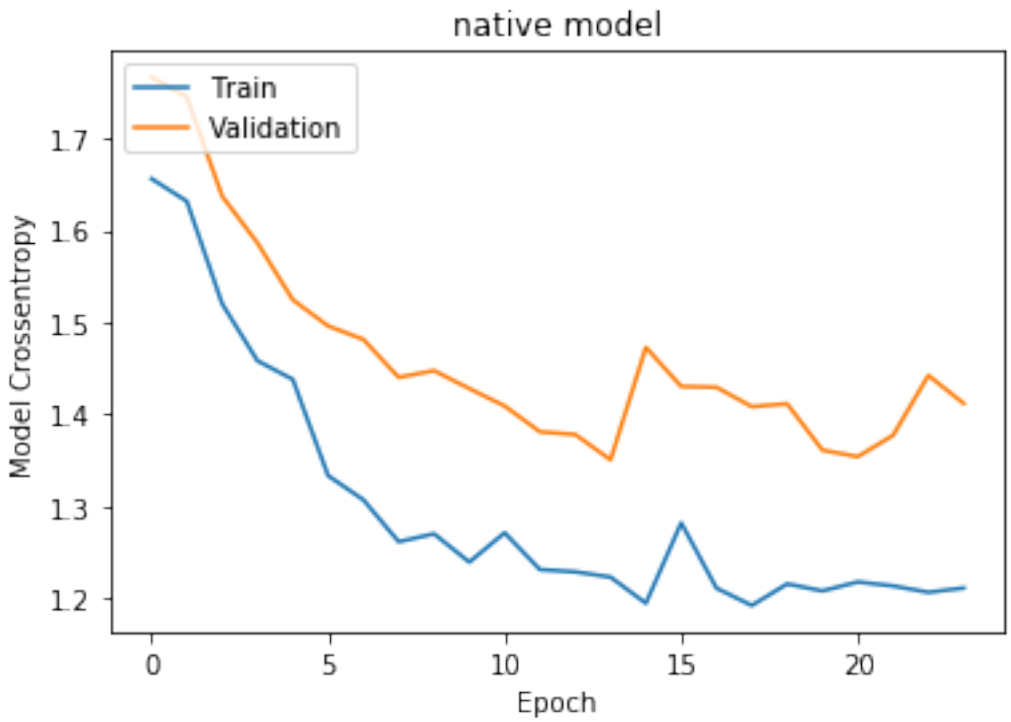
```
import pybor.config as config
import pybor.util as util

output_path = Path(config.BaseSettings().output_path).resolve()


def evaluate_prediction(model=None, data=None, title=''):
    print(f"Evaluate {title} dataset.")
    predictions = model.predict_data(data)
    train_metrics = evaluate.evaluate_model(predictions, data)


def evaluate_borrowing_detection_train_test(language=None,
                                            train=None,
                                            test=None,
                                            detect_type=None,
                                            model_type=None,
                                            val_split=None,
                                            settings=None):

    print(f'*** Evaluation for {language}, detect type is {detect_type}, neural␣
 ↪model type is {model_type}')

    if detect_type == 'native':
        neural = detector.NeuralNative(training=train,
                               testing=test,
                               language=language,
                               series='devel',
                               model_type=model_type,
                               val_split=val_split,
                               settings=settings)


    else:
        neural = detector.NeuralDual(training=train,
                              testing=test,
                              language=language,
                              series='devel',
                              model_type=model_type,
                              val_split=val_split,
                              settings=settings)

    neural.train()

    evaluate_prediction(neural, train, 'Training')
    evaluate_prediction(neural, test, 'Testing')
```

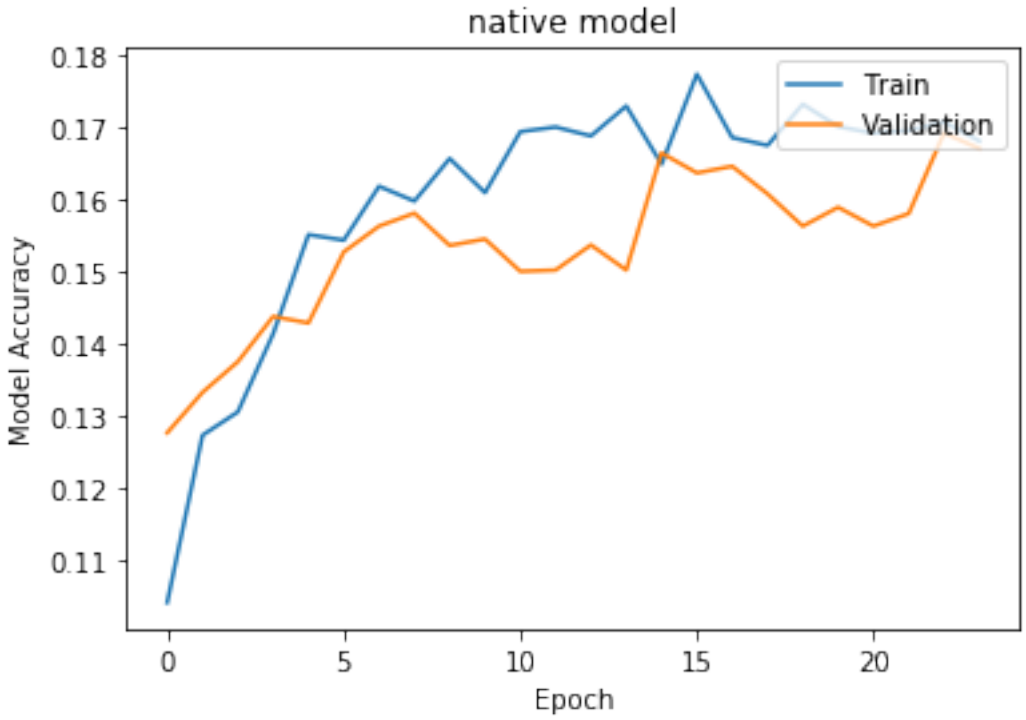
```
    plot_model_measures(neural.native_history, 'native model')
    if detect_type == 'dual':
        plot_model_measures(neural.loan_history, 'loan model')

    plot_entropy_distributions(neural)


def perform_detection_by_language(languages=None,
                                  form=None,
                                  detect_type=None,
                                  model_type=None,
                                  test_split=None,
                                  settings=None):

    wolddb = wold.get_wold_access()
    languages = wold.check_wold_languages(wolddb, languages)

    print(f'Languages {languages}.')
    for language in languages:
        table = wolddb.get_table(
                    language=language, form=form, classification='Borrowed'
                    )

        train, test = util.train_test_split(table, split=test_split)
        val_split = (test_split if test_split is None or test_split >= 1.0
                                else test_split/(1-test_split))

        evaluate_borrowing_detection_train_test(language=language,
                                                train=train,
                                                test=test,
                                                detect_type=detect_type,
                                                model_type=model_type,
                                                val_split=val_split,
                                                settings=settings)
```

5 Datasets selected from WOLD in format [ident, [str], status]

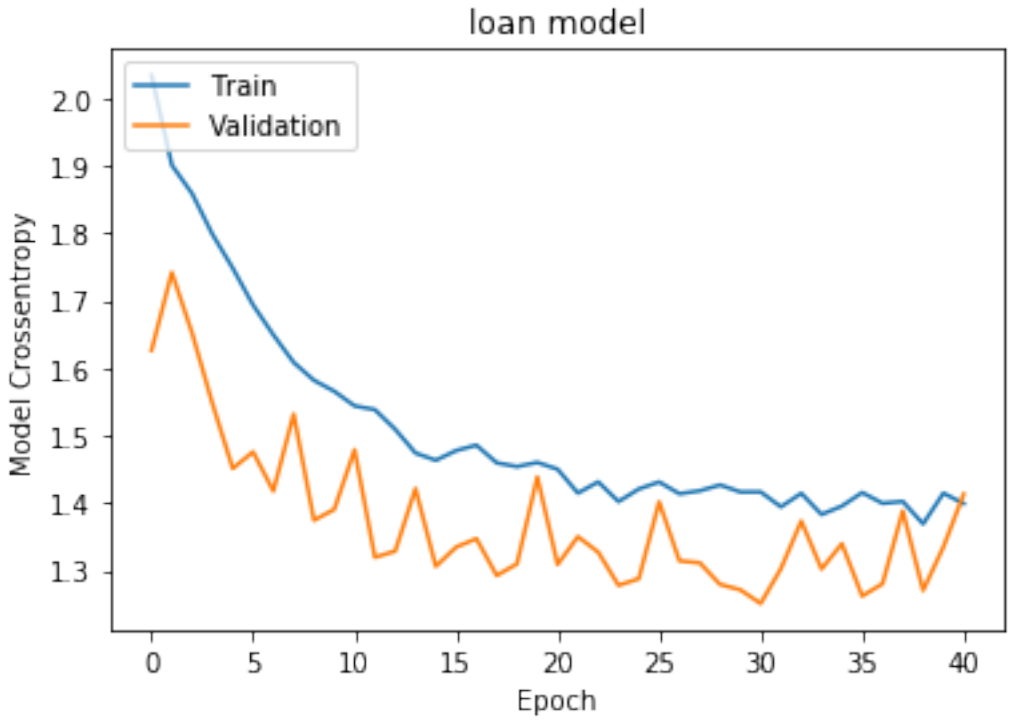
[12]:
```
settings = config.RecurrentSettings(learning_rate = 0.00333)
perform_detection_by_language(languages='English',
                              form='Tokens',
                              detect_type='dual',
                              model_type='recurrent',
                              test_split=0.15,
                              settings=settings)
```

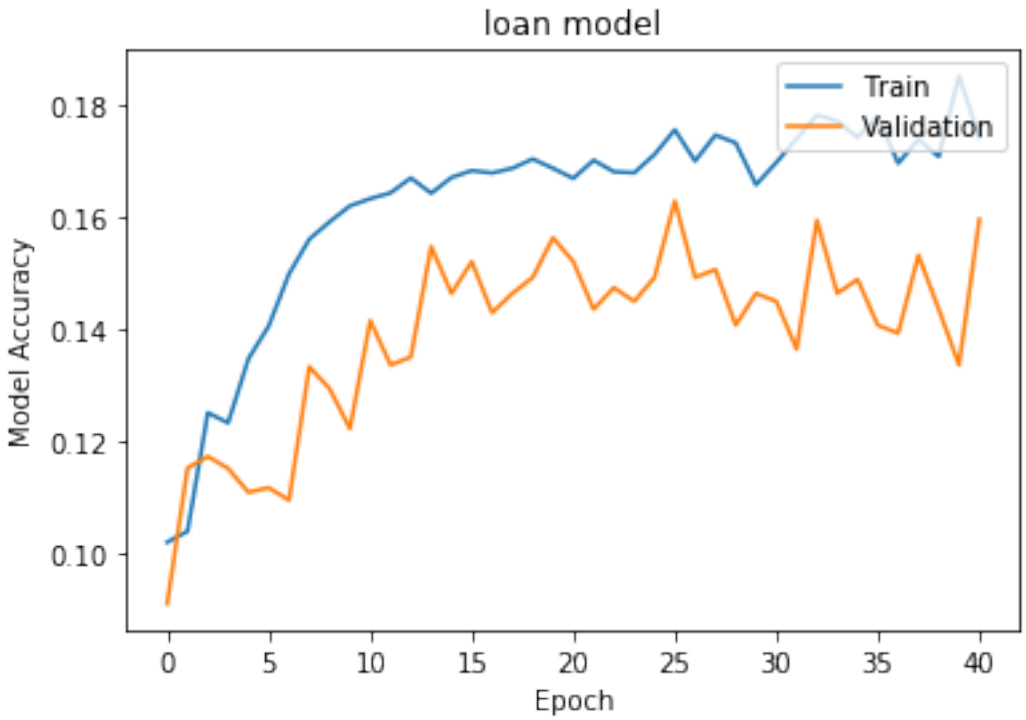
```
loading data: 64289it [00:22, 2816.13it/s]

Languages ['English'].
*** Evaluation for English, detect type is dual, neural model type is recurrent
pybor.neural - INFO - Train length: 789, fit length: 649, val length: 140, test
length: 142.
pybor.neural - INFO - Train length: 499, fit length: 410, val length: 89, test
length: 86.
pybor.neural - INFO - training native model
pybor.entropies - INFO - Training neural <class
'pybor.entropies.NeuralWordRecurrent'> model.
pybor.entropies - INFO - Using per step learning rate decay 0.0026
pybor.entropies - INFO - Available quality measures: dict_keys(['loss',
'categorical_accuracy', 'categorical_crossentropy', 'val_loss',
'val_categorical_accuracy', 'val_categorical_crossentropy']).
pybor.entropies - INFO - Best epoch: 14 of 25.
pybor.entropies - INFO - Statistics from TensorFlow:
pybor.entropies - INFO - Train dataset: loss=1.2524, accuracy=0.1729,
cross_entropy=1.2234.
pybor.entropies - INFO - Validate dataset: loss=1.3760, accuracy=0.1502,
cross_entropy=1.3509.
pybor.neural - INFO - training loan model
pybor.entropies - INFO - Training neural <class
'pybor.entropies.NeuralWordRecurrent'> model.
pybor.entropies - INFO - Using per step learning rate decay 0.0044
pybor.entropies - INFO - Available quality measures: dict_keys(['loss',
'categorical_accuracy', 'categorical_crossentropy', 'val_loss',
'val_categorical_accuracy', 'val_categorical_crossentropy']).
pybor.entropies - INFO - Best epoch: 31 of 42.
pybor.entropies - INFO - Statistics from TensorFlow:
pybor.entropies - INFO - Train dataset: loss=1.4412, accuracy=0.1697,
cross_entropy=1.4166.
pybor.entropies - INFO - Validate dataset: loss=1.2662, accuracy=0.1449,
cross_entropy=1.2512.
Evaluate Training dataset.
|   Precision |   Recall |   F-score |   Accuracy |
|------------:|---------:|----------:|-----------:|
|       0.762 |    0.788 |     0.774 |      0.822 |
Evaluate Testing dataset.
|   Precision |   Recall |   F-score |   Accuracy |
|------------:|---------:|----------:|-----------:|
|       0.619 |    0.698 |     0.656 |      0.724 |
measures dict_keys(['loss', 'categorical_accuracy', 'categorical_crossentropy',
'val_loss', 'val_categorical_accuracy', 'val_categorical_crossentropy'])
```

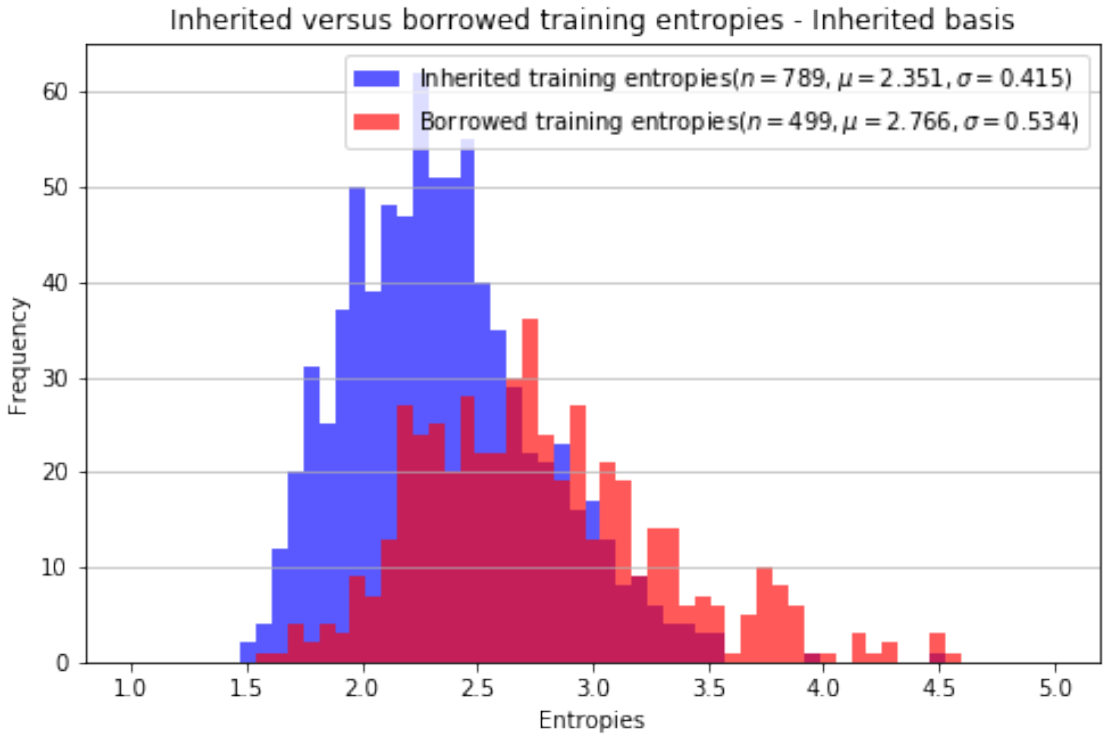

native model
Train
Validation
Model Crossentropy
1.7
1.6
1.5
1.4
1.3
1.2
0
5
10
15
20
Epoch

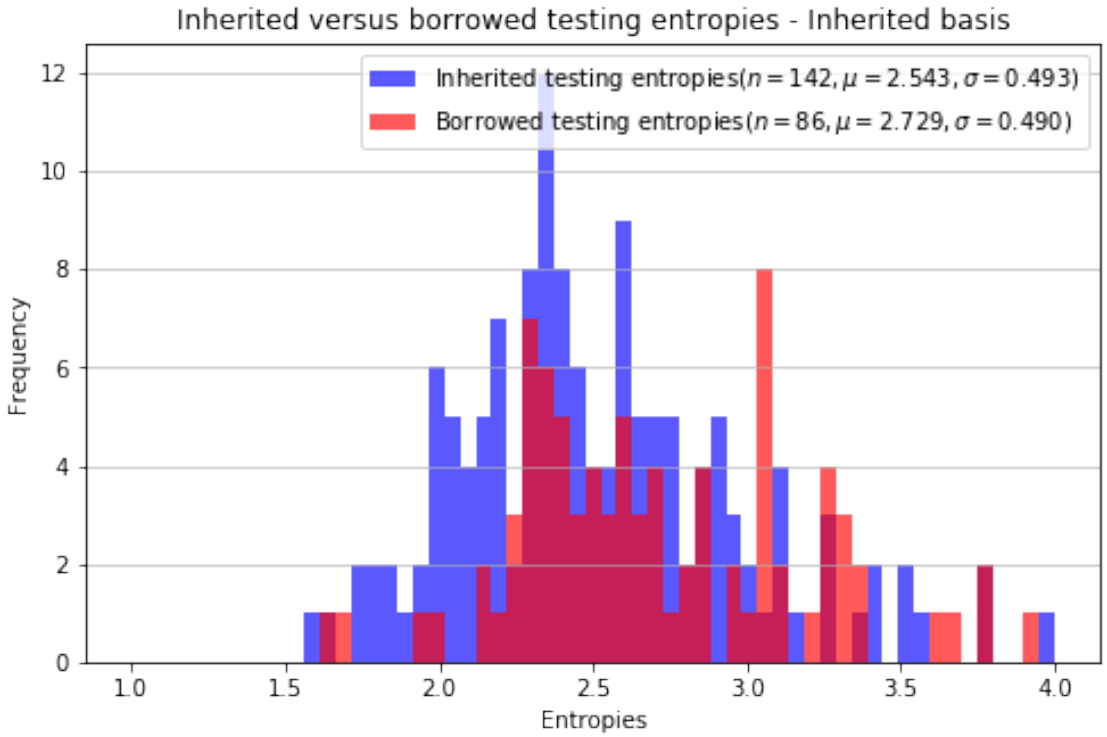

native model
Train
Validation
Model Accuracy
0.18
0.17
0.16
0.15
0.14
0.13
0.12
0.11
0
5
10
15
20
Epoch

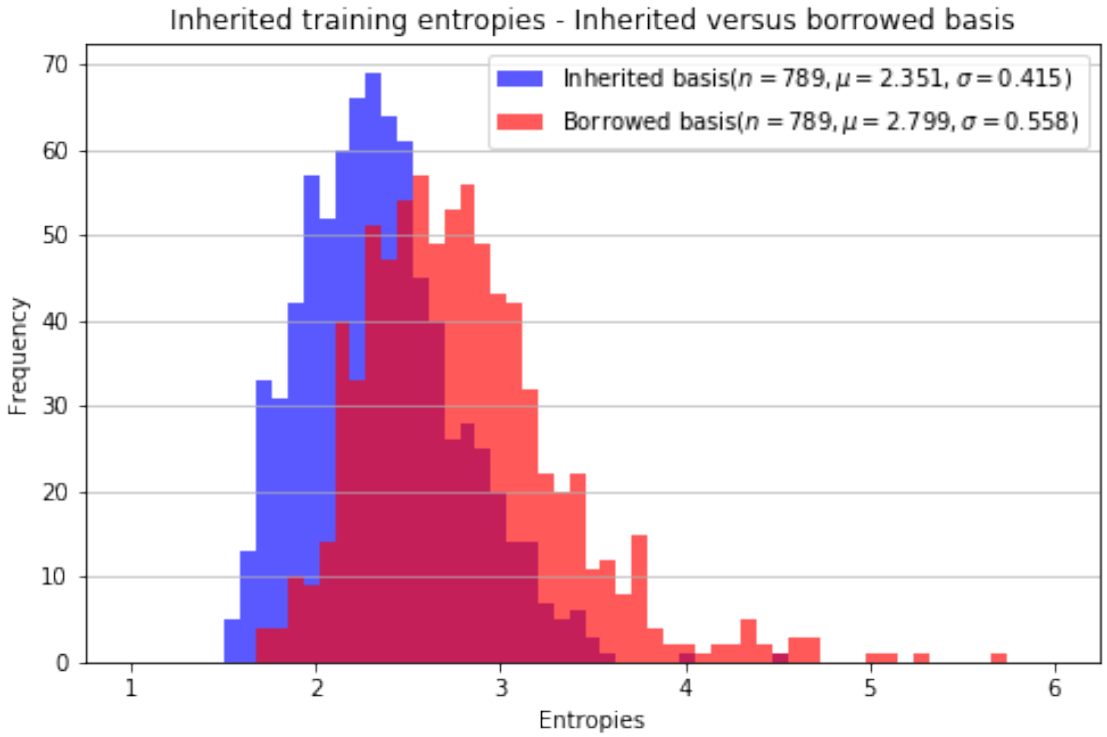
```
measures dict_keys(['loss', 'categorical_accuracy', 'categorical_crossentropy',
'val_loss', 'val_categorical_accuracy', 'val_categorical_crossentropy'])
```

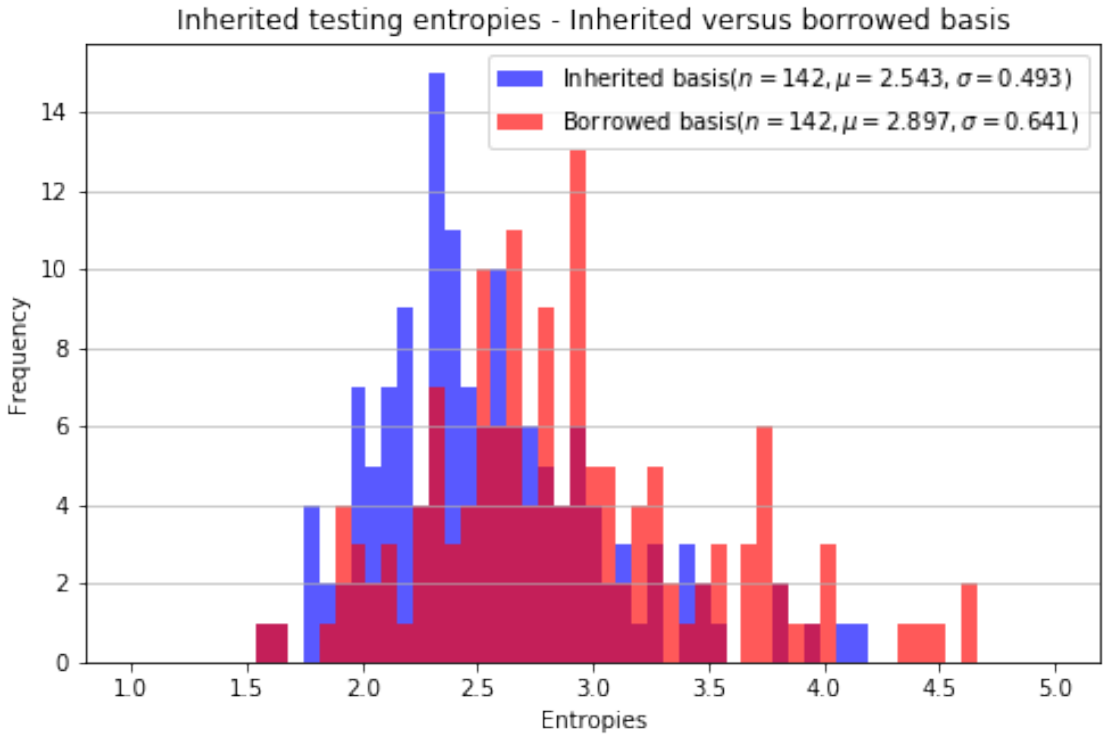


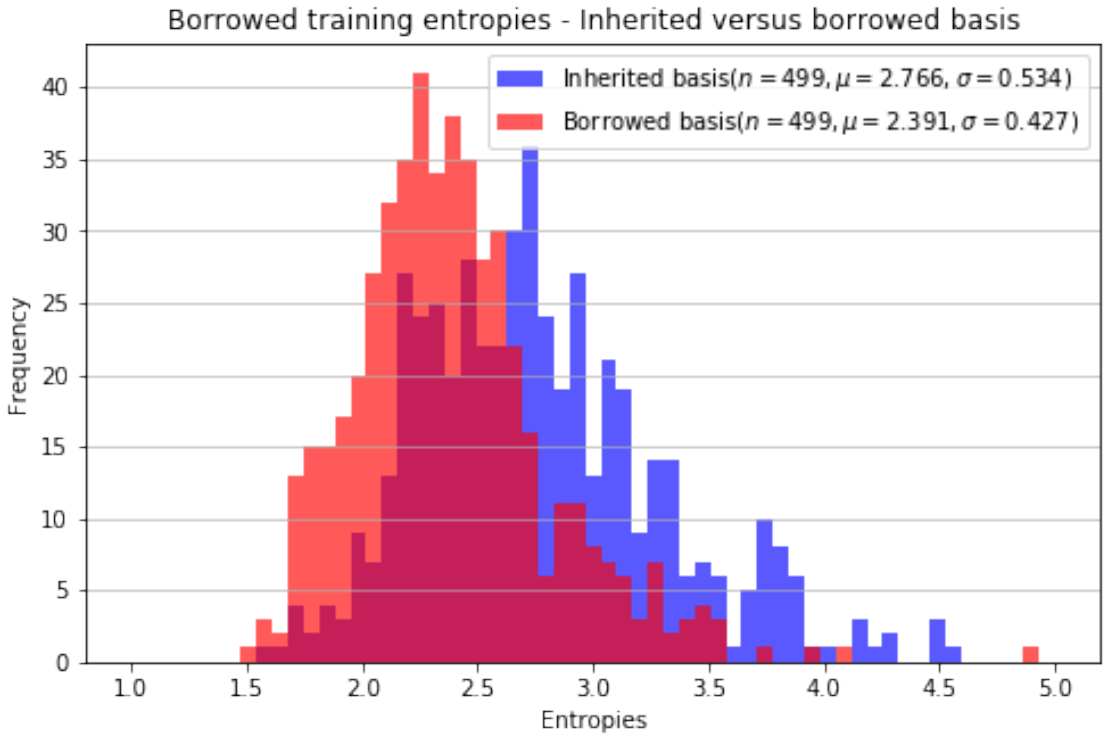


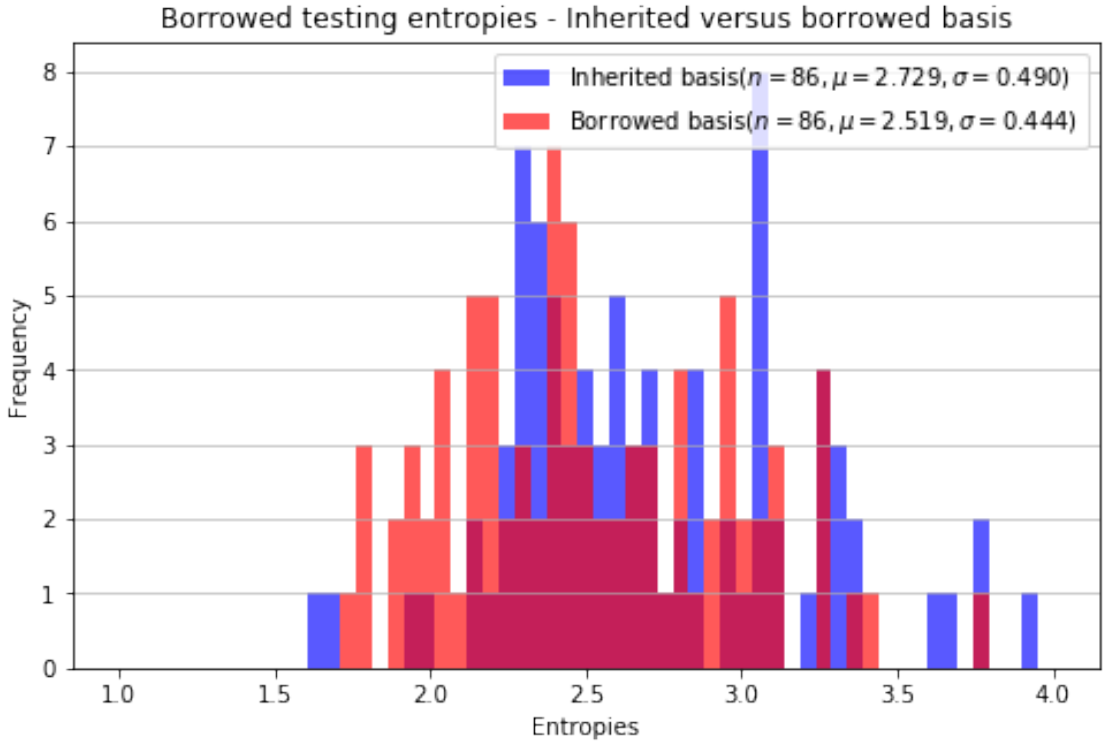

Inherited versus borrowed training entropies - Inherited basis
Inherited training entropies($n = 789, \mu = 2.351, \sigma = 0.415$)
Borrowed training entropies($n = 499, \mu = 2.766, \sigma = 0.534$)
Frequency
Entropies

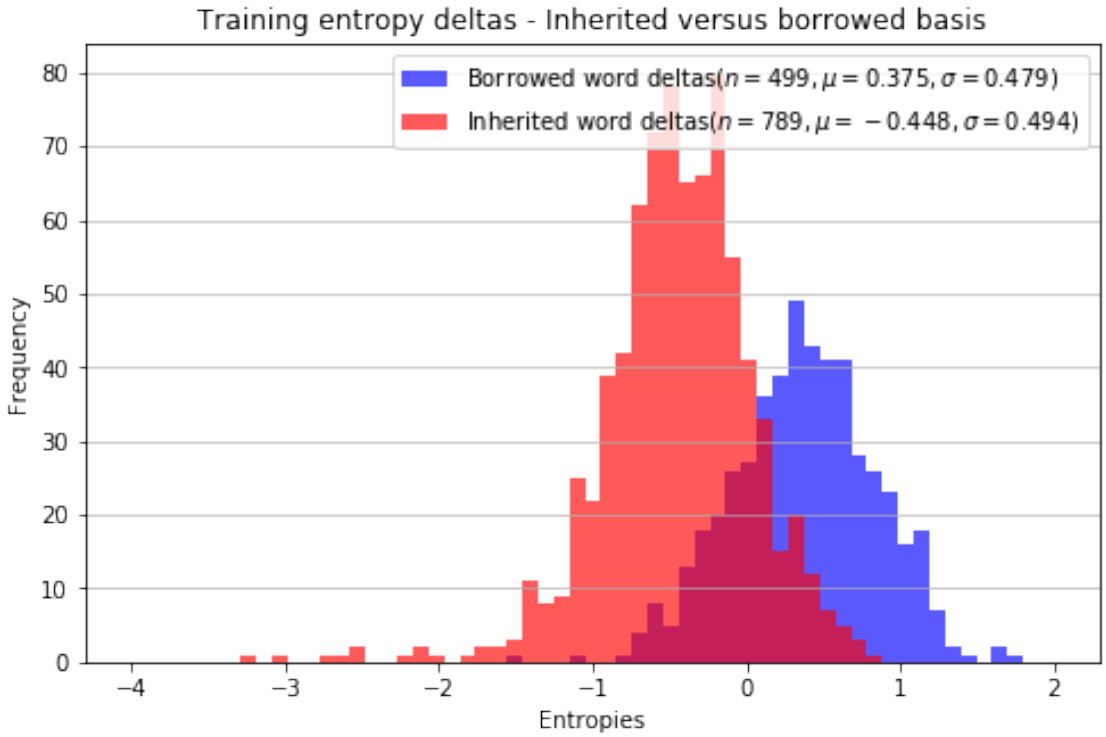

Inherited versus borrowed testing entropies - Inherited basis
Inherited testing entropies($n = 142, \mu = 2.543, \sigma = 0.493$)
Borrowed testing entropies($n = 86, \mu = 2.729, \sigma = 0.490$)
Frequency
Entropies

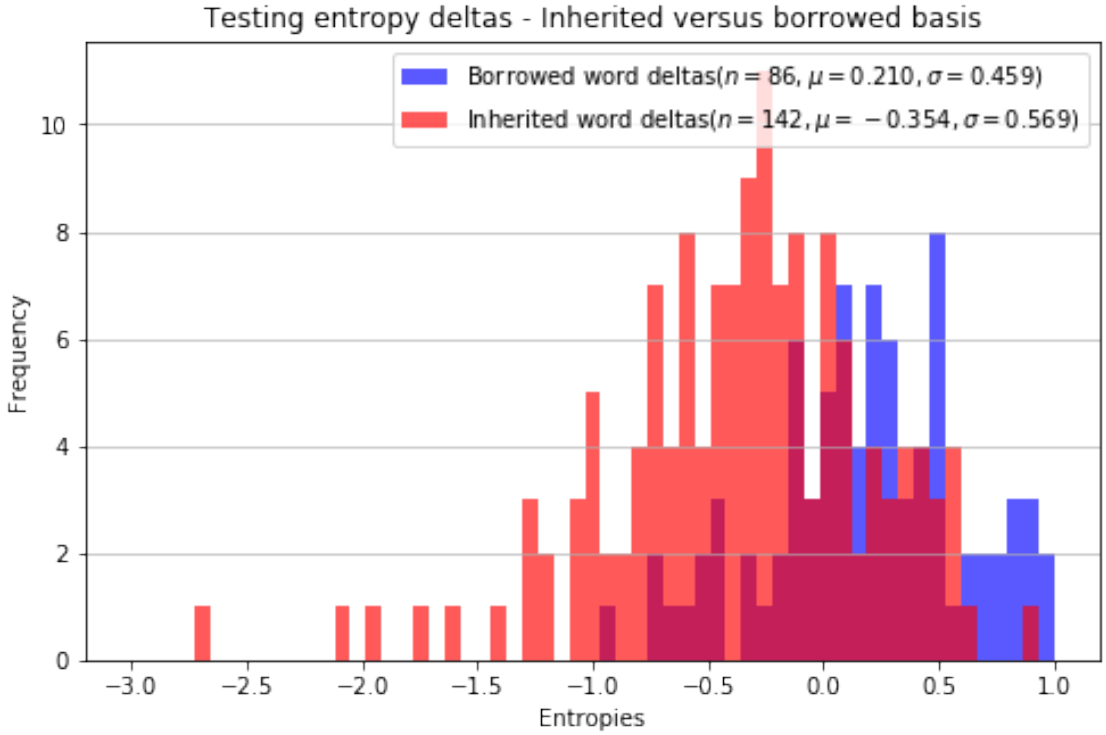

Inherited training entropies - Inherited versus borrowed basis
Inherited basis($n = 789, \mu = 2.351, \sigma = 0.415$)
Borrowed basis($n = 789, \mu = 2.799, \sigma = 0.558$)
Frequency
Entropies
0
10
20
30
40
50
60
70
1
2
3
4
5
6


Inherited testing entropies - Inherited versus borrowed basis
Inherited basis($n = 142, \mu = 2.543, \sigma = 0.493$)
Borrowed basis($n = 142, \mu = 2.897, \sigma = 0.641$)
Frequency
Entropies
0
2
4
6
8
10
12
14
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Borrowed training entropies - Inherited versus borrowed basis
Inherited basis($n = 499, \mu = 2.766, \sigma = 0.534$)
Borrowed basis($n = 499, \mu = 2.391, \sigma = 0.427$)
Frequency
Entropies


Borrowed testing entropies - Inherited versus borrowed basis
Inherited basis($n = 86, \mu = 2.729, \sigma = 0.490$)
Borrowed basis($n = 86, \mu = 2.519, \sigma = 0.444$)
Frequency
Entropies

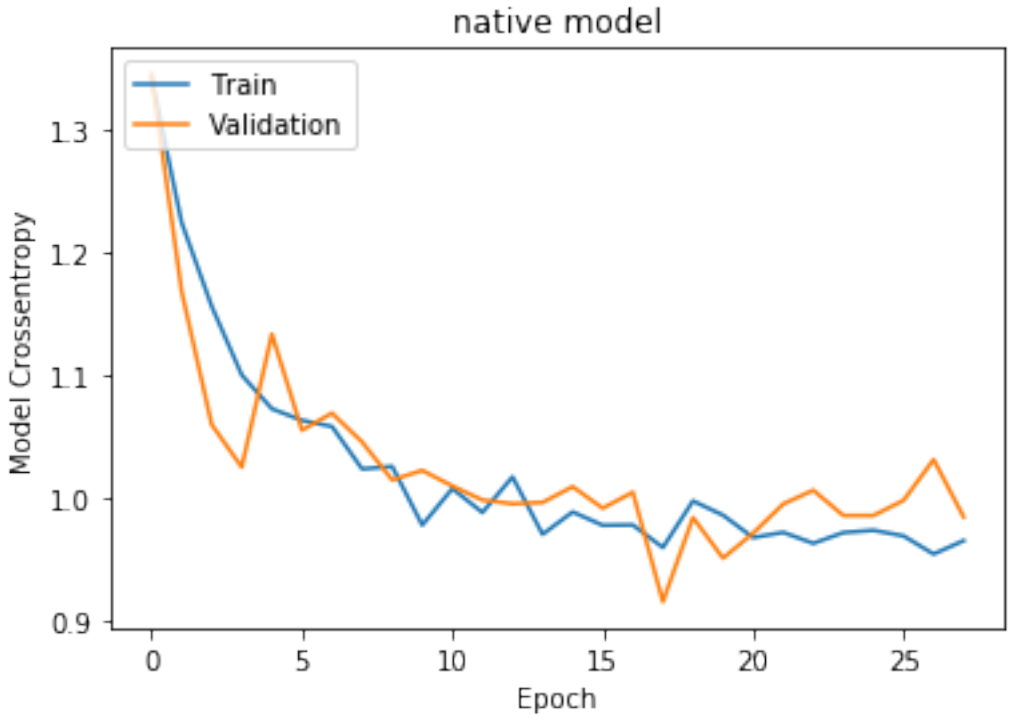

Training entropy deltas - Inherited versus borrowed basis
Borrowed word deltas($n = 499, \mu = 0.375, \sigma = 0.479$)
Inherited word deltas($n = 789, \mu = -0.448, \sigma = 0.494$)
Frequency
Entropies
0
10
20
30
40
50
60
70
80
−4
−3
−2
−1
0
1
2

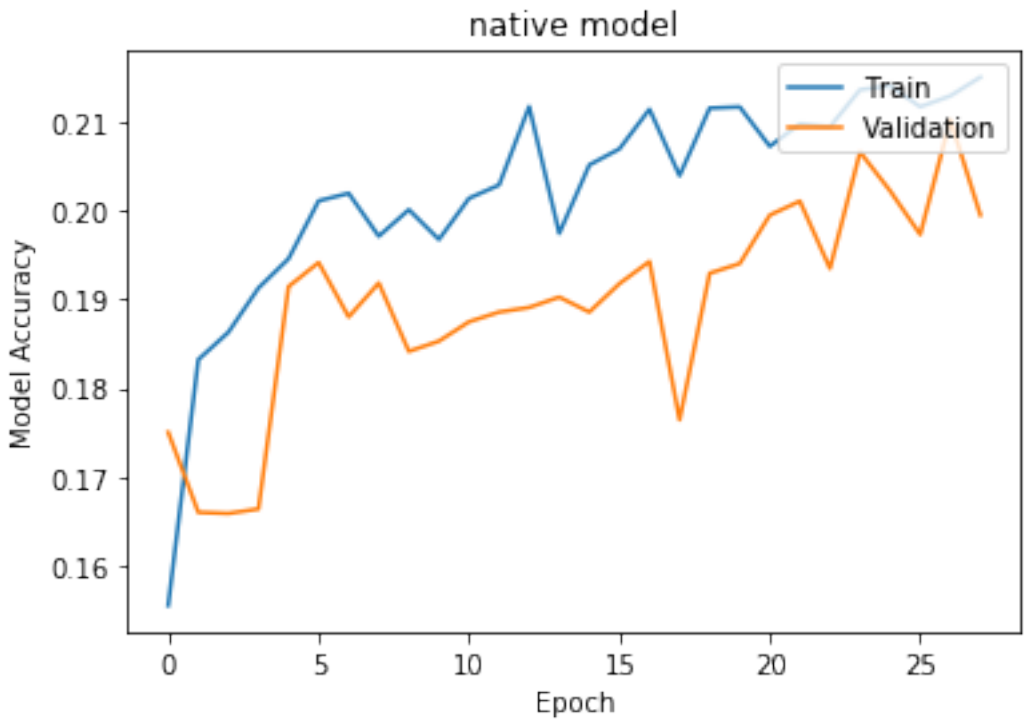

Testing entropy deltas - Inherited versus borrowed basis
Borrowed word deltas($n = 86, \mu = 0.210, \sigma = 0.459$)
Inherited word deltas($n = 142, \mu = -0.354, \sigma = 0.569$)
Frequency
Entropies
0
2
4
6
8
10
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0

[14]:
```
settings = config.RecurrentSettings(learning_rate = 0.00333)
perform_detection_by_language(languages='Imbabura Quechua',
                              form='Tokens',
                              detect_type='dual',
                              model_type='recurrent',
                              test_split=0.15,
                              settings=settings)
```

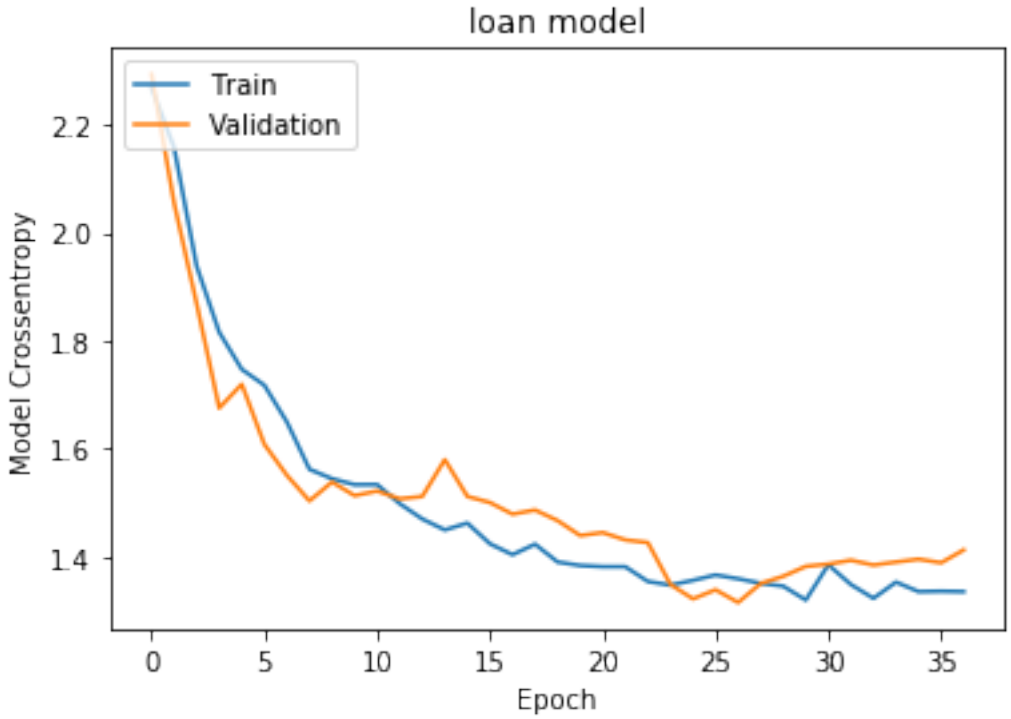
```
Languages ['Imbabura Quechua'].
*** Evaluation for Imbabura Quechua, detect type is dual, neural model type is
recurrent
pybor.neural - INFO - Train length: 821, fit length: 676, val length: 145, test
length: 129.
pybor.neural - INFO - Train length: 300, fit length: 247, val length: 53, test
length: 69.
pybor.neural - INFO - training native model
pybor.entropies - INFO - Training neural <class
'pybor.entropies.NeuralWordRecurrent'> model.
pybor.entropies - INFO - Using per step learning rate decay 0.0025
pybor.entropies - INFO - Available quality measures: dict_keys(['loss',
'categorical_accuracy', 'categorical_crossentropy', 'val_loss',
'val_categorical_accuracy', 'val_categorical_crossentropy']).
pybor.entropies - INFO - Best epoch: 18 of 29.
pybor.entropies - INFO - Statistics from TensorFlow:
pybor.entropies - INFO - Train dataset: loss=0.9809, accuracy=0.2040,
cross_entropy=0.9599.
pybor.entropies - INFO - Validate dataset: loss=0.9696, accuracy=0.1764,
cross_entropy=0.9159.
pybor.neural - INFO - training loan model
pybor.entropies - INFO - Training neural <class
'pybor.entropies.NeuralWordRecurrent'> model.
pybor.entropies - INFO - Using per step learning rate decay 0.0075
pybor.entropies - INFO - Available quality measures: dict_keys(['loss',
'categorical_accuracy', 'categorical_crossentropy', 'val_loss',
'val_categorical_accuracy', 'val_categorical_crossentropy']).
pybor.entropies - INFO - Best epoch: 27 of 38.
pybor.entropies - INFO - Statistics from TensorFlow:
pybor.entropies - INFO - Train dataset: loss=1.3758, accuracy=0.2346,
cross_entropy=1.3594.
pybor.entropies - INFO - Validate dataset: loss=1.3282, accuracy=0.2031,
cross_entropy=1.3155.
Evaluate Training dataset.
|   Precision |   Recall |   F-score |   Accuracy |
|------------:|---------:|----------:|-----------:|
|       0.835 |    0.843 |     0.839 |      0.913 |
```

```
Evaluate Testing dataset.
|   Precision |   Recall |   F-score |   Accuracy |
|------------:|---------:|----------:|-----------:|
|       0.841 |    0.841 |     0.841 |      0.889 |
measures dict_keys(['loss', 'categorical_accuracy', 'categorical_crossentropy',
'val_loss', 'val_categorical_accuracy', 'val_categorical_crossentropy'])
```

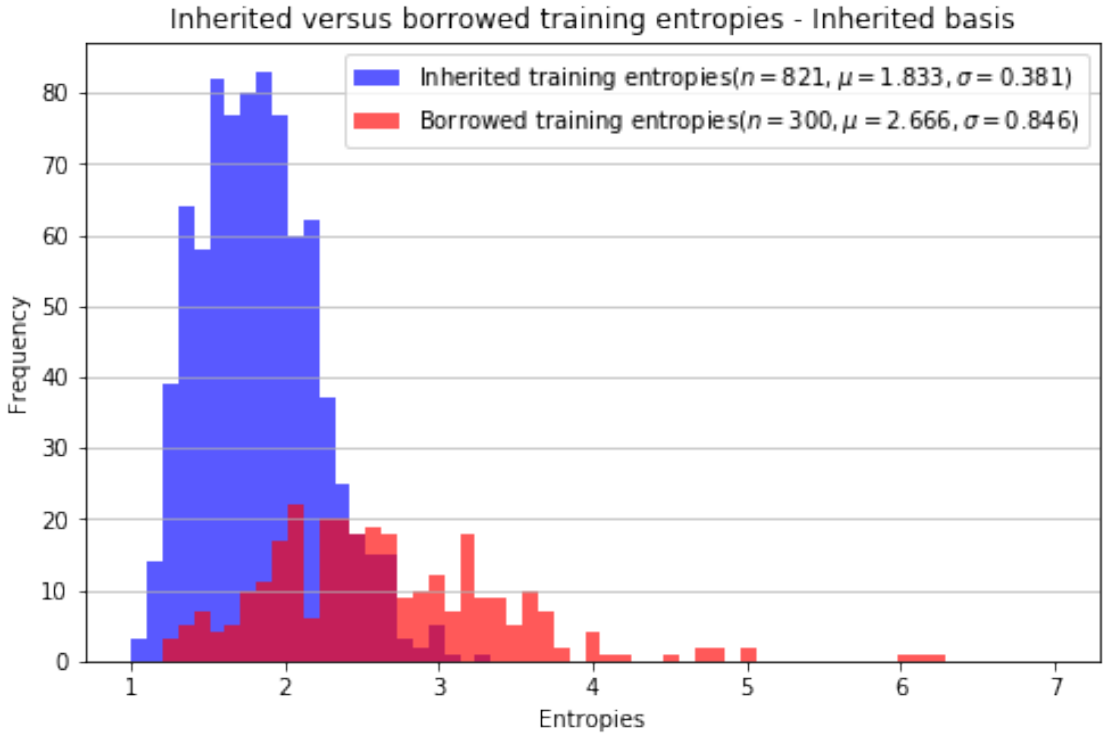


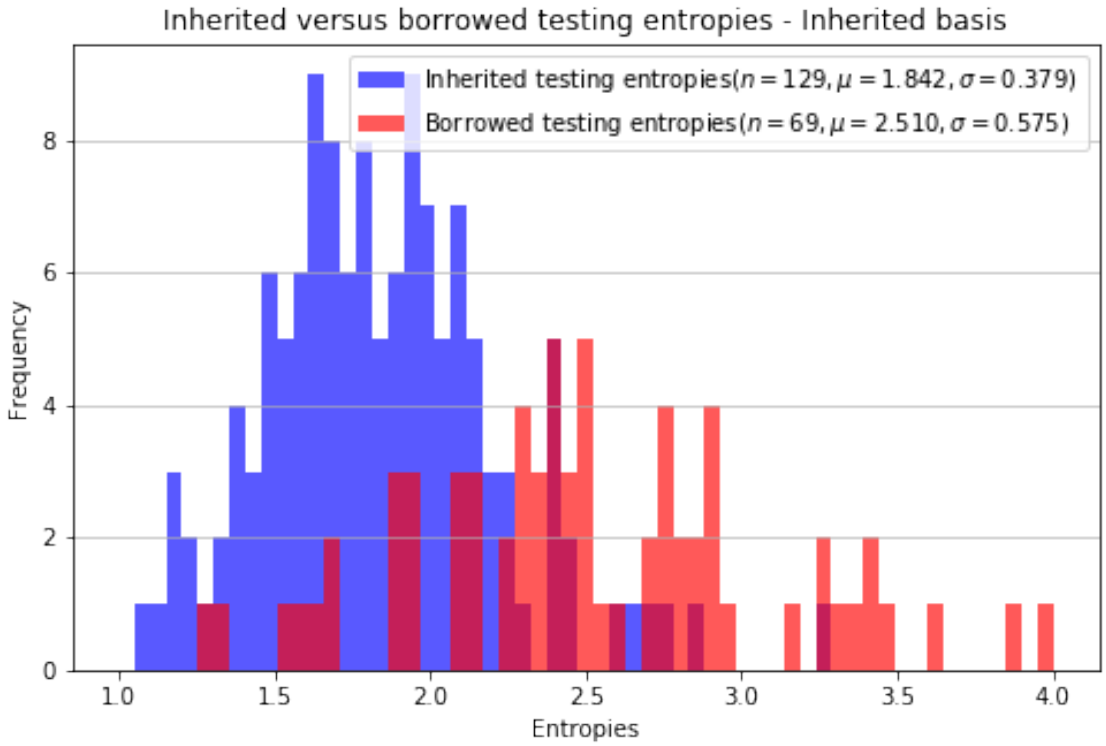


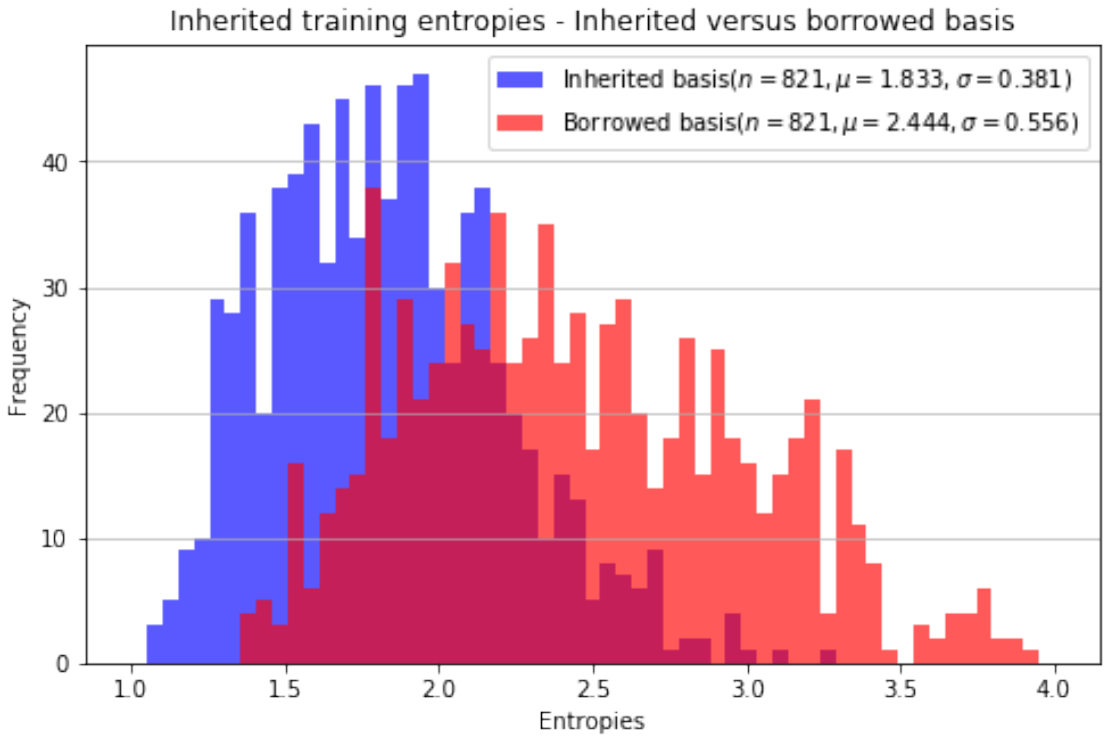

```
measures dict_keys(['loss', 'categorical_accuracy', 'categorical_crossentropy',
'val_loss', 'val_categorical_accuracy', 'val_categorical_crossentropy'])
```

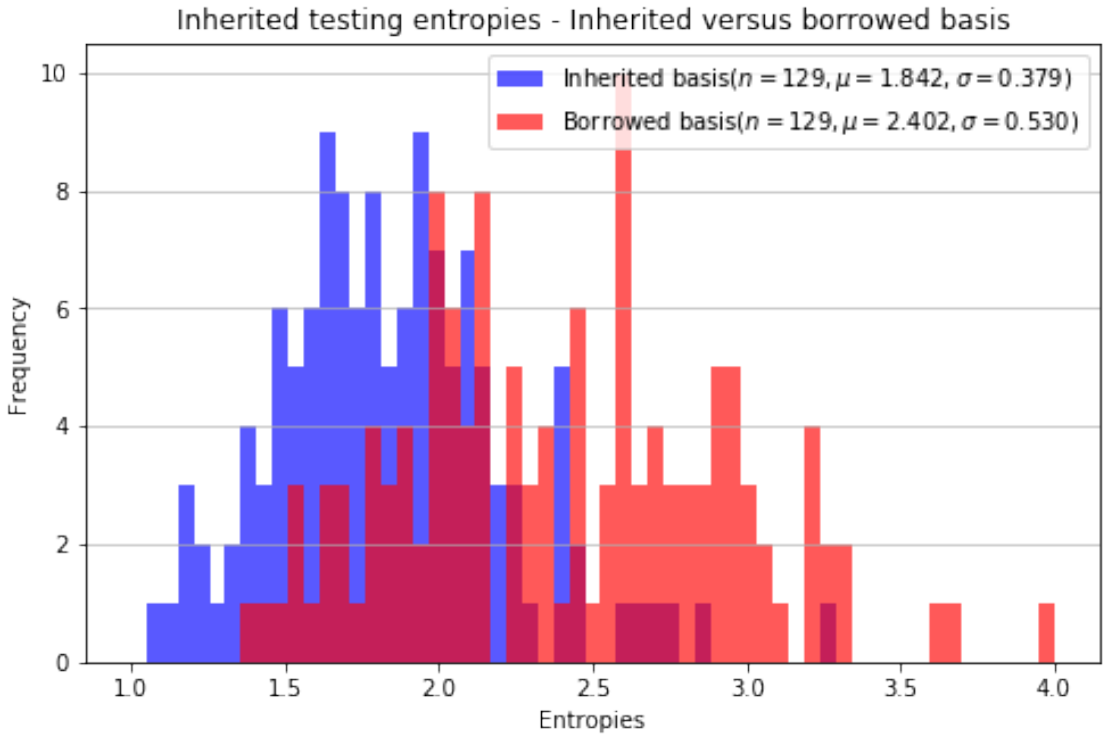


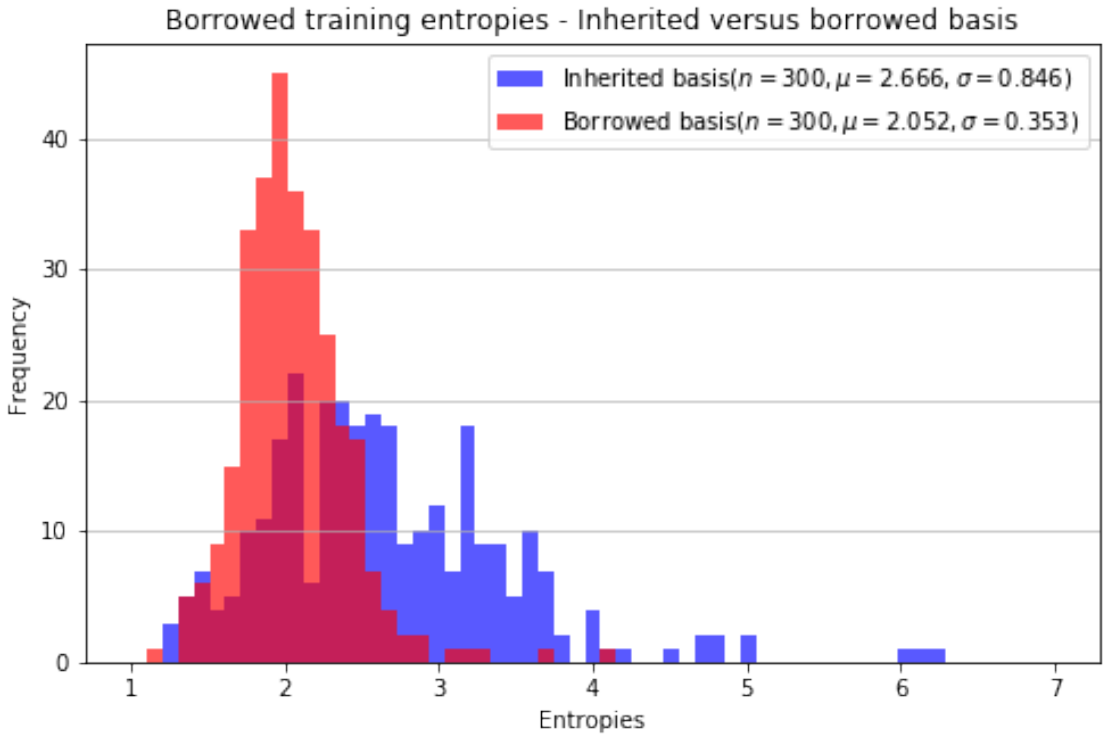

loan model
Train
Validation
Model Accuracy
0.24
0.22
0.20
0.18
0.16
0.14
0.12
0
5
10
15
20
25
30
35
Epoch

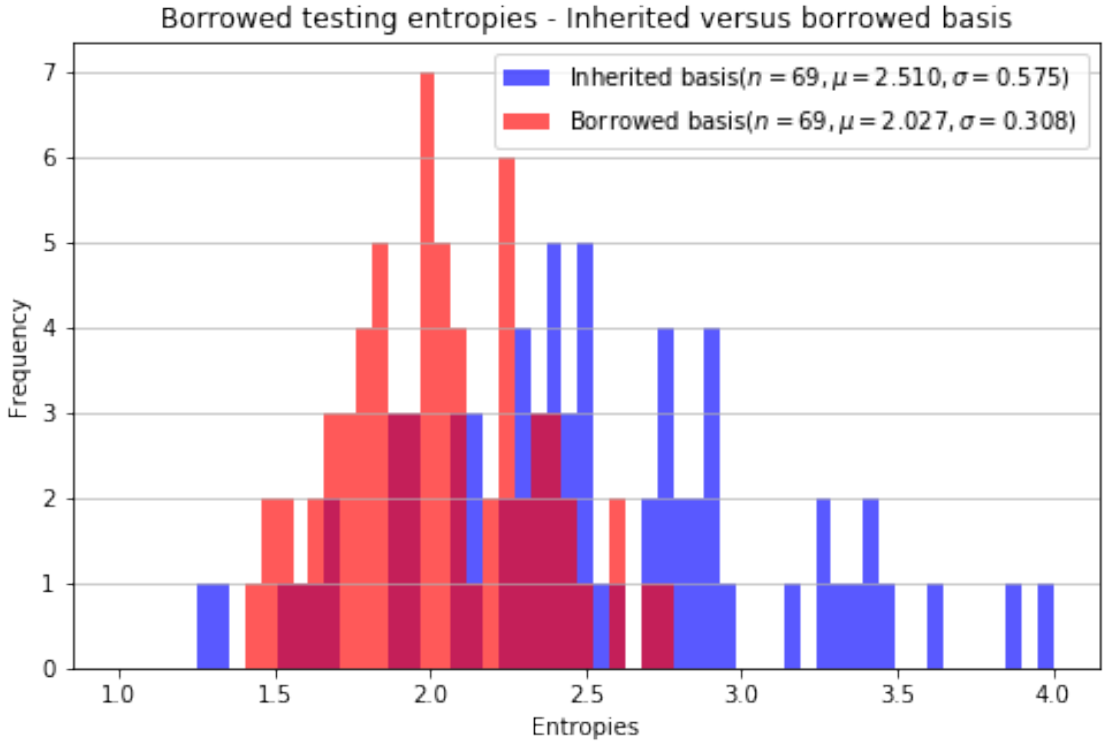

Inherited versus borrowed training entropies - Inherited basis
Inherited training entropies($n = 821, \mu = 1.833, \sigma = 0.381$)
Borrowed training entropies($n = 300, \mu = 2.666, \sigma = 0.846$)
Frequency
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
Entropies

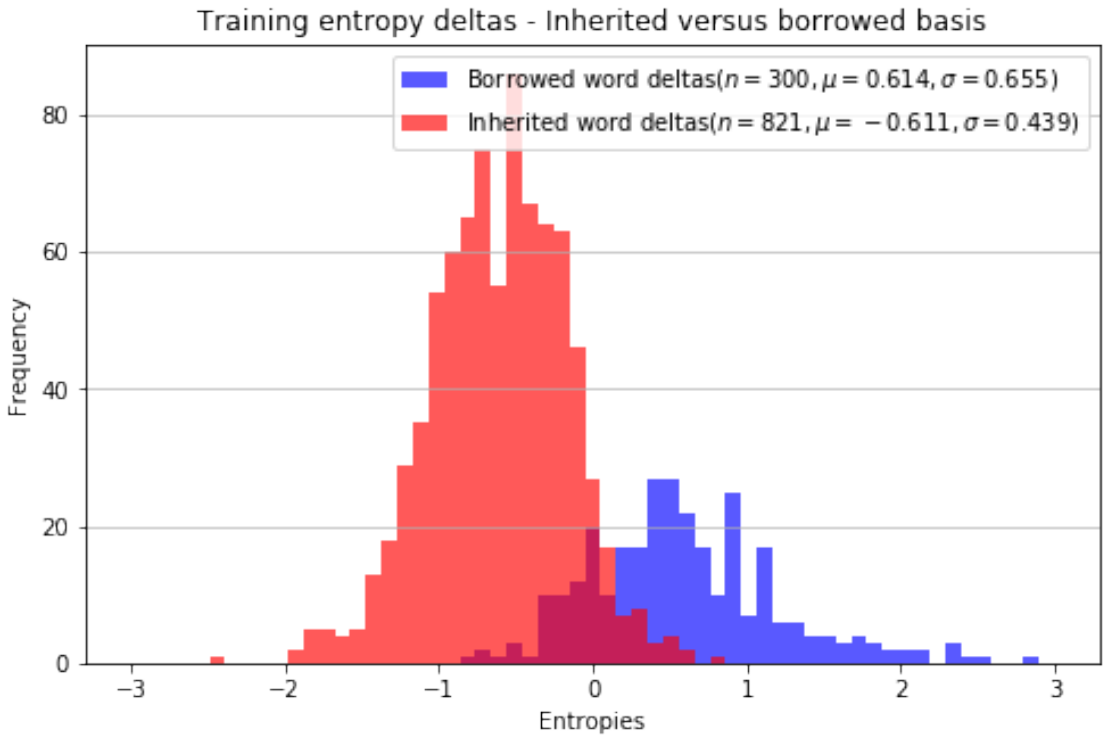

Inherited versus borrowed testing entropies - Inherited basis
Inherited testing entropies($n = 129, \mu = 1.842, \sigma = 0.379$)
Borrowed testing entropies($n = 69, \mu = 2.510, \sigma = 0.575$)
Frequency
Entropies
0
2
4
6
8
1.0
1.5
2.0
2.5
3.0
3.5
4.0

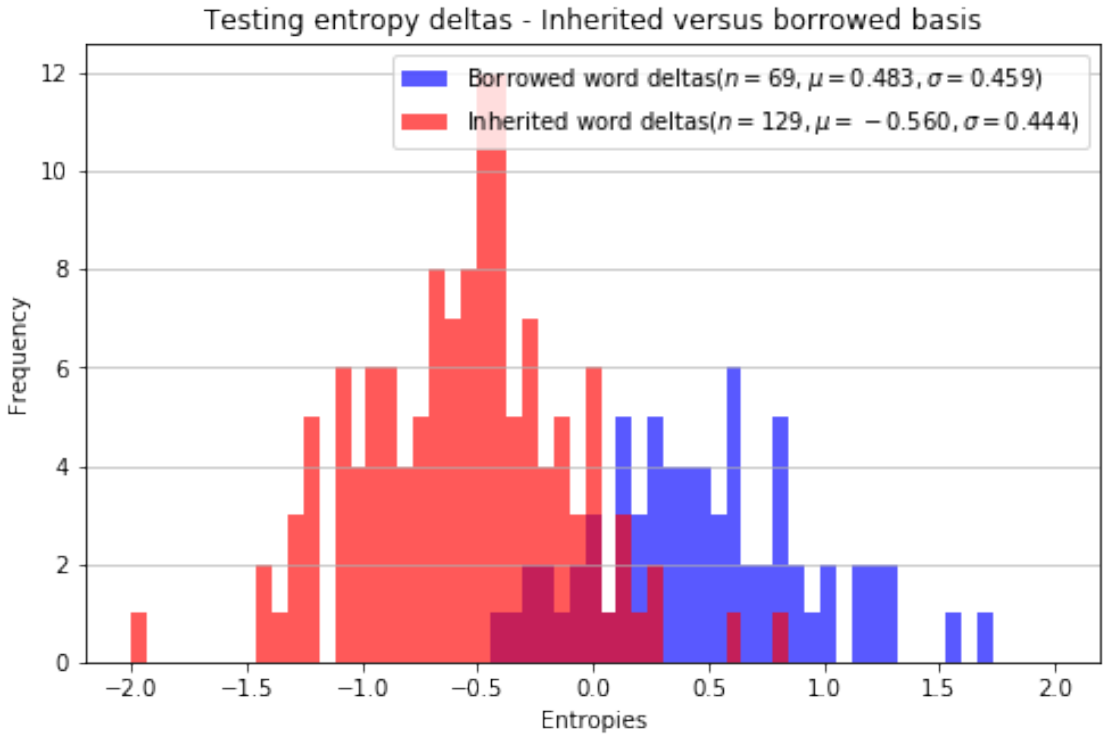

Inherited training entropies - Inherited versus borrowed basis
Inherited basis($n = 821, \mu = 1.833, \sigma = 0.381$)
Borrowed basis($n = 821, \mu = 2.444, \sigma = 0.556$)
Frequency
Entropies
0
10
20
30
40
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Inherited testing entropies - Inherited versus borrowed basis
Inherited basis($n = 129, \mu = 1.842, \sigma = 0.379$)
Borrowed basis($n = 129, \mu = 2.402, \sigma = 0.530$)
Frequency
Entropies
0
2
4
6
8
10
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Borrowed training entropies - Inherited versus borrowed basis
Inherited basis($n = 300, \mu = 2.666, \sigma = 0.846$)
Borrowed basis($n = 300, \mu = 2.052, \sigma = 0.353$)
Frequency
Entropies
0
10
20
30
40
1
2
3
4
5
6
7


Borrowed testing entropies - Inherited versus borrowed basis
Inherited basis($n = 69, \mu = 2.510, \sigma = 0.575$)
Borrowed basis($n = 69, \mu = 2.027, \sigma = 0.308$)
Frequency
Entropies
0
1
2
3
4
5
6
7
1.0
1.5
2.0
2.5
3.0
3.5
4.0

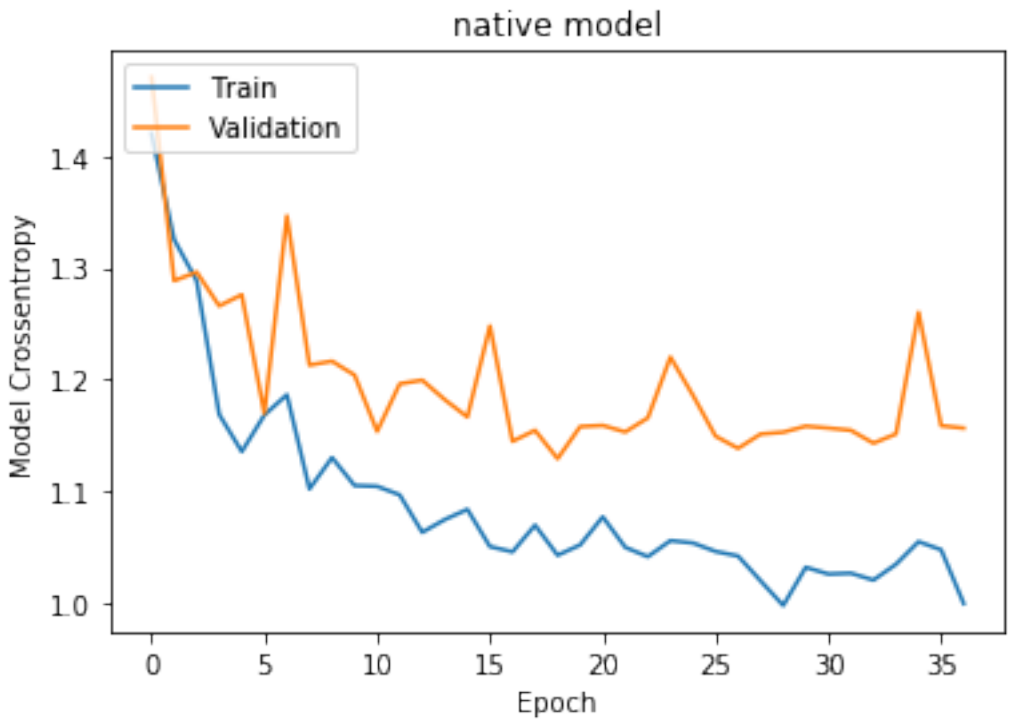

Training entropy deltas - Inherited versus borrowed basis
Borrowed word deltas($n = 300, \mu = 0.614, \sigma = 0.655$)
Inherited word deltas($n = 821, \mu = -0.611, \sigma = 0.439$)
Frequency
Entropies
0
20
40
60
80
−3
−2
−1
0
1
2
3

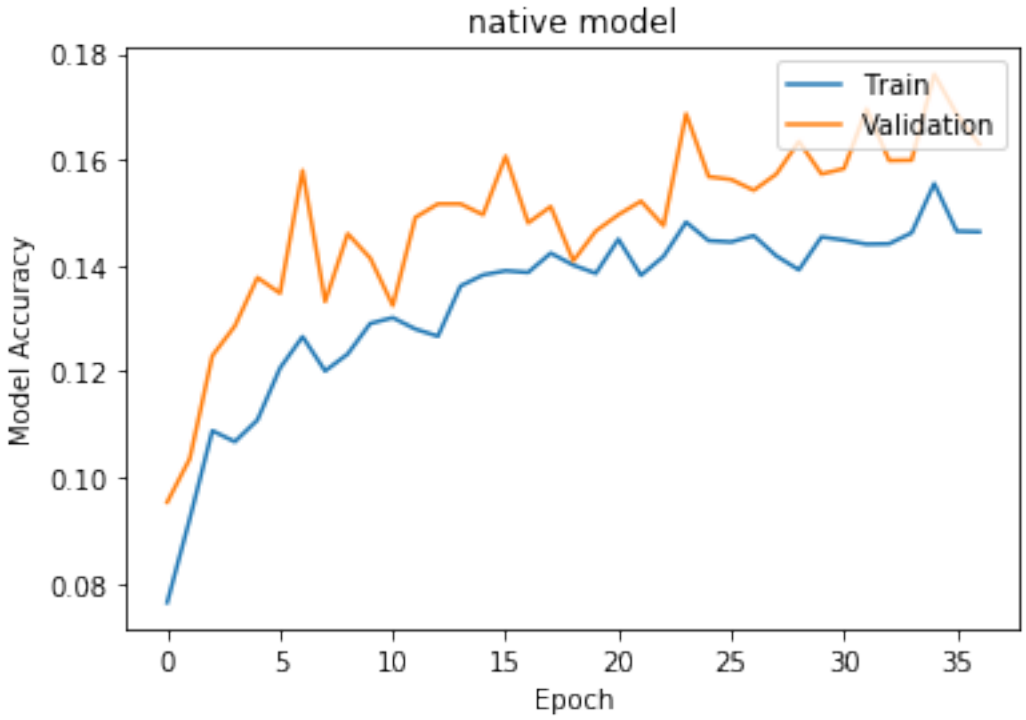


[15]:
```
settings = config.RecurrentSettings(learning_rate = 0.00333)
perform_detection_by_language(languages='Oroqen',
                              form='Tokens',
                              detect_type='dual',
                              model_type='recurrent',
                              test_split=0.15,
                              settings=settings)
```

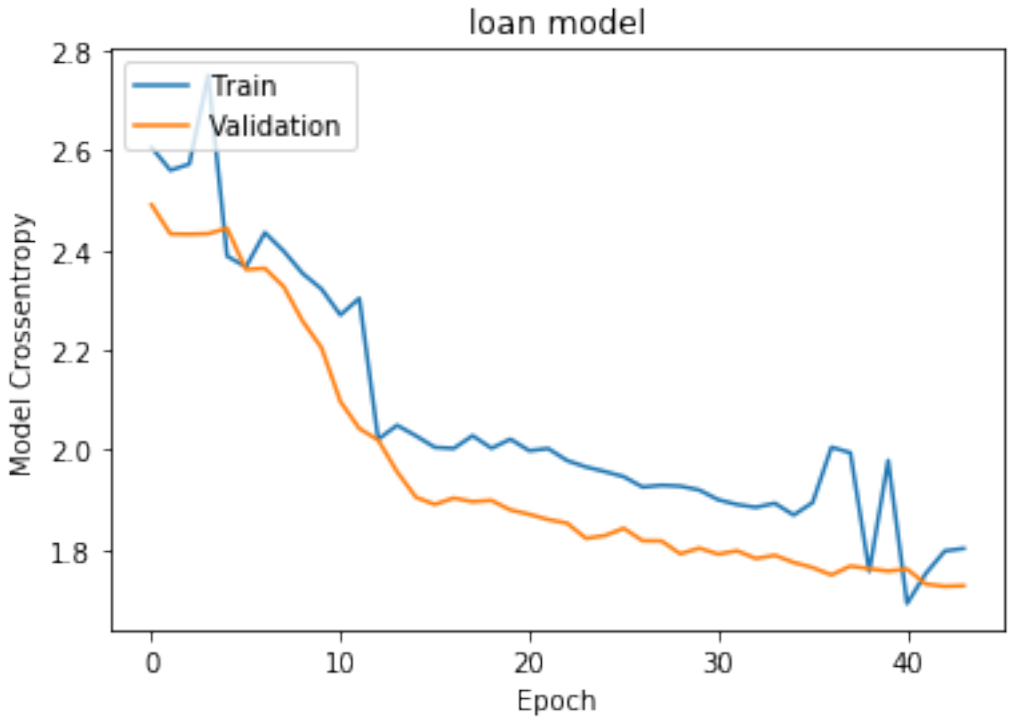
```
Languages ['Oroqen'].
*** Evaluation for Oroqen, detect type is dual, neural model type is recurrent
pybor.neural - INFO - Train length: 942, fit length: 775, val length: 167, test
length: 169.
pybor.neural - INFO - Train length: 82, fit length: 67, val length: 15, test
length: 12.
pybor.neural - INFO - training native model
pybor.entropies - INFO - Training neural <class
'pybor.entropies.NeuralWordRecurrent'> model.
pybor.entropies - INFO - Using per step learning rate decay 0.0022
pybor.entropies - INFO - Available quality measures: dict_keys(['loss',
'categorical_accuracy', 'categorical_crossentropy', 'val_loss',
'val_categorical_accuracy', 'val_categorical_crossentropy']).
```

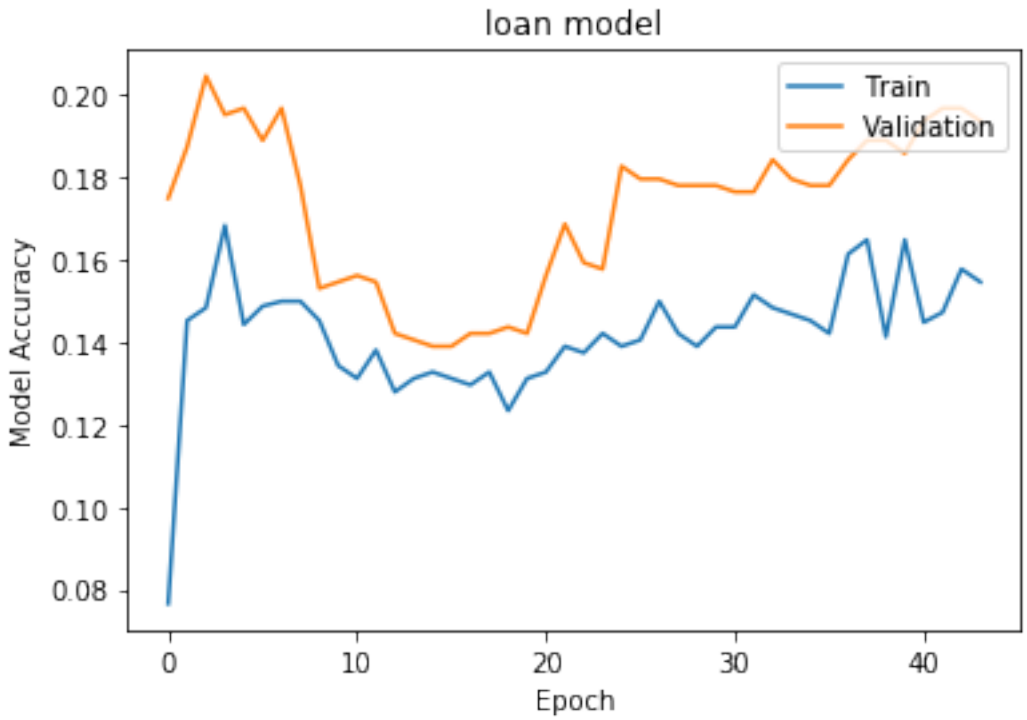
```
pybor.entropies - INFO - Best epoch: 27 of 38.
pybor.entropies - INFO - Statistics from TensorFlow:
pybor.entropies - INFO - Train dataset: loss=1.0963, accuracy=0.1456,
cross_entropy=1.0415.
pybor.entropies - INFO - Validate dataset: loss=1.1630, accuracy=0.1542,
cross_entropy=1.1381.
pybor.neural - INFO - training loan model
pybor.entropies - INFO - Training neural <class
'pybor.entropies.NeuralWordRecurrent'> model.
pybor.entropies - INFO - Using per step learning rate decay 0.0263
pybor.entropies - INFO - Available quality measures: dict_keys(['loss',
'categorical_accuracy', 'categorical_crossentropy', 'val_loss',
'val_categorical_accuracy', 'val_categorical_crossentropy']).
pybor.entropies - INFO - Best epoch: 43 of 45.
pybor.entropies - INFO - Statistics from TensorFlow:
pybor.entropies - INFO - Train dataset: loss=1.8164, accuracy=0.1578,
cross_entropy=1.7972.
pybor.entropies - INFO - Validate dataset: loss=1.7403, accuracy=0.1969,
cross_entropy=1.7265.
Evaluate Training dataset.
|   Precision |   Recall |   F-score |   Accuracy |
|------------:|---------:|----------:|-----------:|
|       0.588 |    0.488 |     0.533 |      0.932 |
Evaluate Testing dataset.
|   Precision |   Recall |   F-score |   Accuracy |
|------------:|---------:|----------:|-----------:|
|       0.286 |    0.333 |     0.308 |      0.901 |
measures dict_keys(['loss', 'categorical_accuracy', 'categorical_crossentropy',
'val_loss', 'val_categorical_accuracy', 'val_categorical_crossentropy'])
```

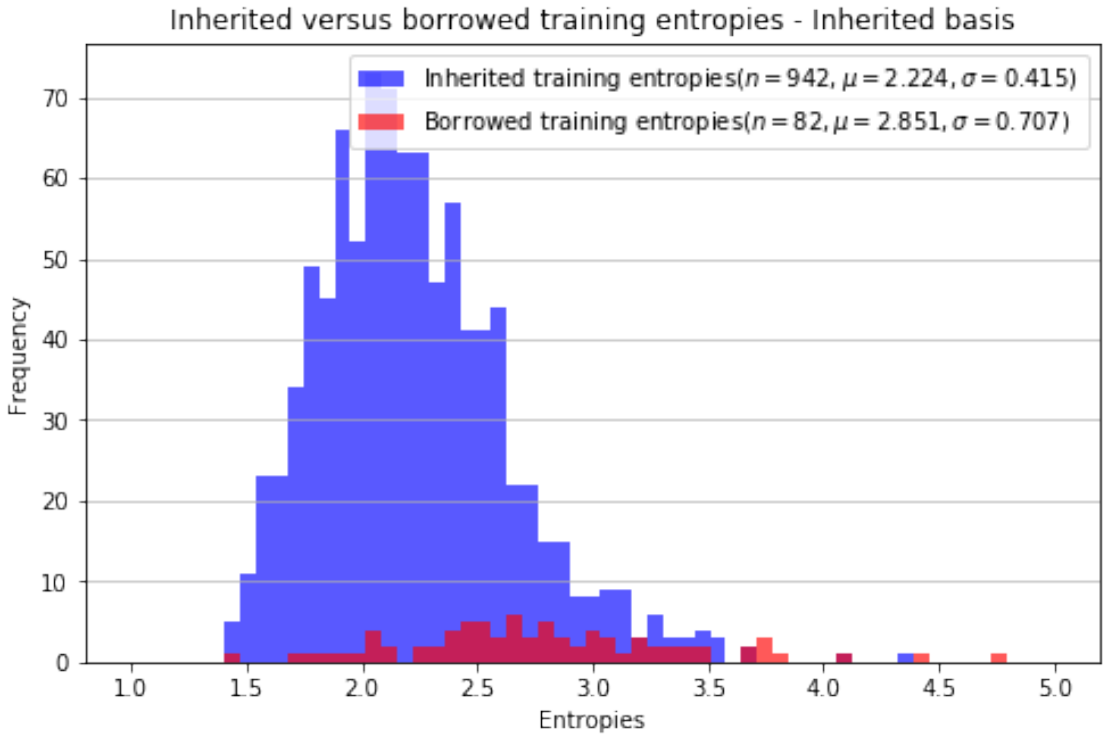

native model
Train
Validation
Model Crossentropy
1.4
1.3
1.2
1.1
1.0
0
5
10
15
20
25
30
35
Epoch

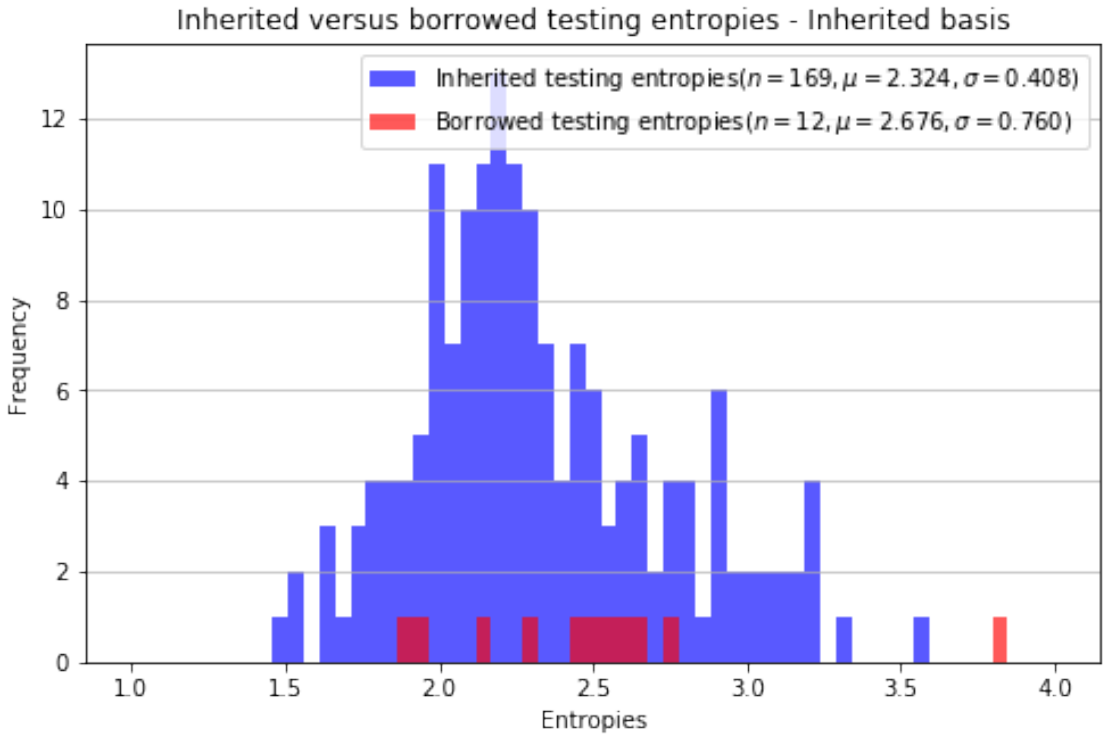

native model
Train
Validation
Model Accuracy
0.18
0.16
0.14
0.12
0.10
0.08
0
5
10
15
20
25
30
35
Epoch

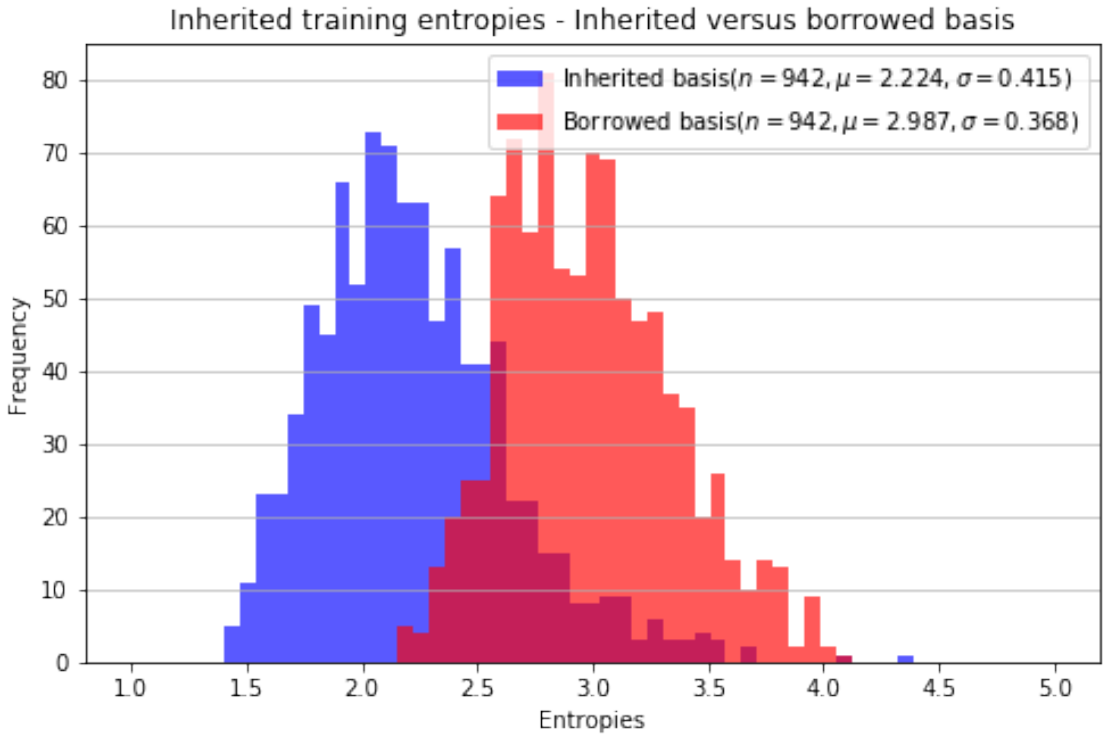
```
measures dict_keys(['loss', 'categorical_accuracy', 'categorical_crossentropy',
'val_loss', 'val_categorical_accuracy', 'val_categorical_crossentropy'])
```

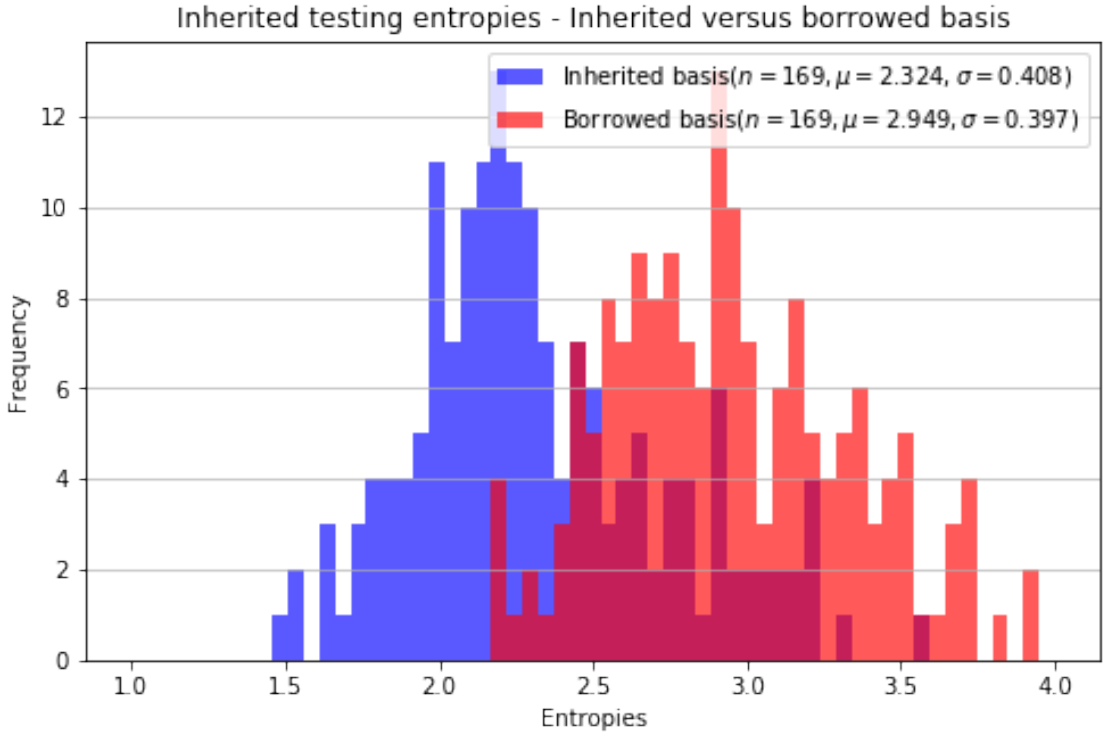


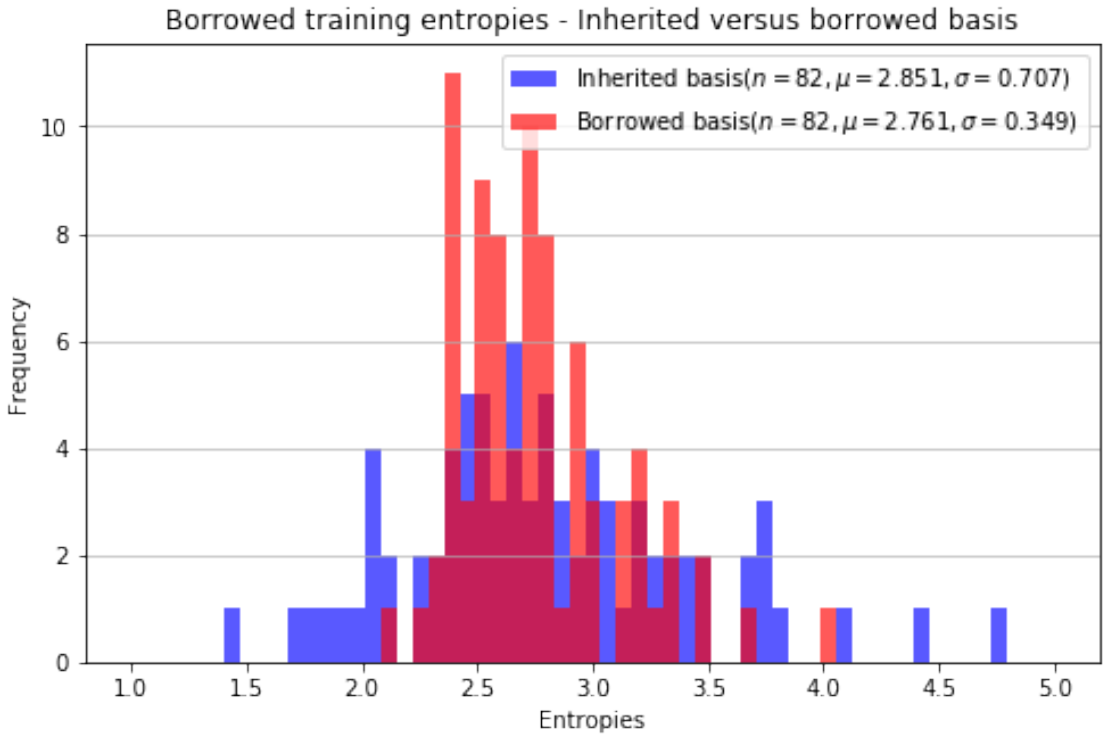


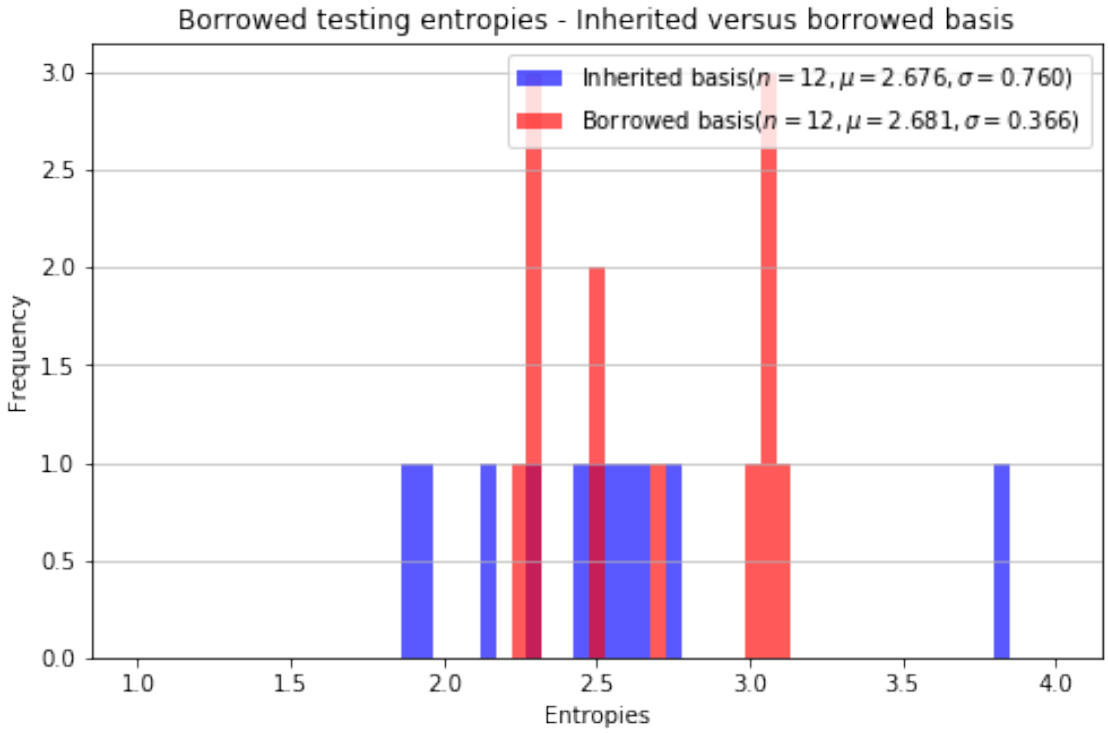

Inherited versus borrowed training entropies - Inherited basis
Inherited training entropies($n = 942, \mu = 2.224, \sigma = 0.415$)
Borrowed training entropies($n = 82, \mu = 2.851, \sigma = 0.707$)
Frequency
Entropies

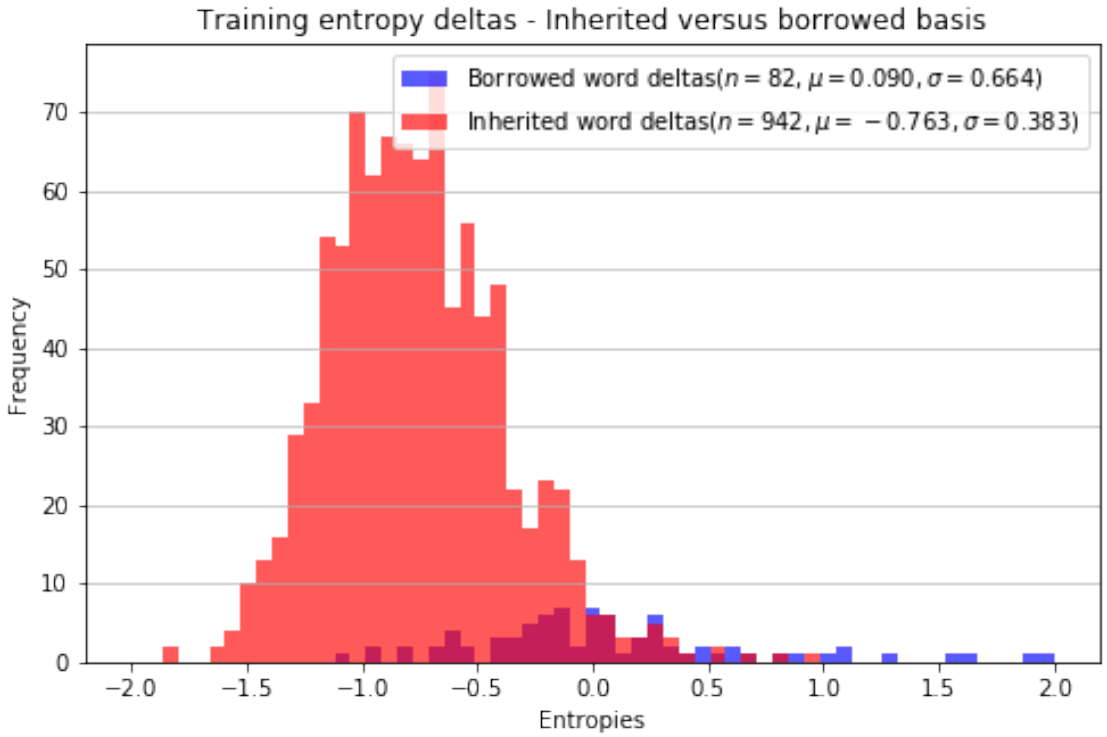

Inherited versus borrowed testing entropies - Inherited basis
Inherited testing entropies($n = 169, \mu = 2.324, \sigma = 0.408$)
Borrowed testing entropies($n = 12, \mu = 2.676, \sigma = 0.760$)
Frequency
Entropies

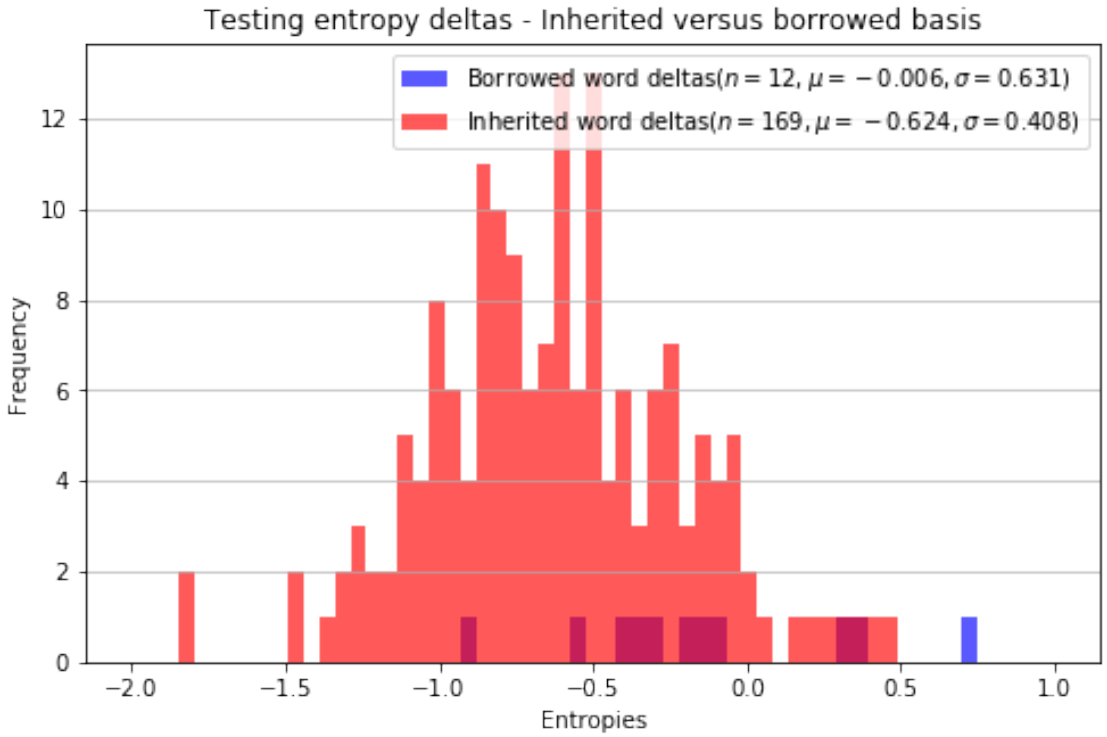

Inherited training entropies - Inherited versus borrowed basis
Inherited basis($n = 942, \mu = 2.224, \sigma = 0.415$)
Borrowed basis($n = 942, \mu = 2.987, \sigma = 0.368$)
Frequency
Entropies
0
10
20
30
40
50
60
70
80
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Inherited testing entropies - Inherited versus borrowed basis
Inherited basis($n = 169, \mu = 2.324, \sigma = 0.408$)
Borrowed basis($n = 169, \mu = 2.949, \sigma = 0.397$)
Frequency
Entropies
0
2
4
6
8
10
12
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Borrowed training entropies - Inherited versus borrowed basis
Inherited basis($n = 82, \mu = 2.851, \sigma = 0.707$)
Borrowed basis($n = 82, \mu = 2.761, \sigma = 0.349$)
Frequency
Entropies


Borrowed testing entropies - Inherited versus borrowed basis
Inherited basis($n = 12, \mu = 2.676, \sigma = 0.760$)
Borrowed basis($n = 12, \mu = 2.681, \sigma = 0.366$)
Frequency
Entropies


Training entropy deltas - Inherited versus borrowed basis
Borrowed word deltas($n = 82, \mu = 0.090, \sigma = 0.664$)
Inherited word deltas($n = 942, \mu = -0.763, \sigma = 0.383$)
Frequency
Entropies


Testing entropy deltas - Inherited versus borrowed basis
Borrowed word deltas($n = 12, \mu = -0.006, \sigma = 0.631$)
Inherited word deltas($n = 169, \mu = -0.624, \sigma = 0.408$)
Frequency
Entropies

[ ]: