```python
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from scipy.sparse import vstack

# 1. Load and prepare data
data = pd.DataFrame({
    'email_text': [
        'Free money offer just for you',
        'Earn cash quickly and easily',
        'Meeting scheduled with boss',
        'Project deadline is tomorrow',
        'Cheap loans available now',
        'Dinner plans with family tonight',
        'Limited offer: Get rich fast!',
        'Quarterly report submission',
        'Click here for a surprise gift',
        'Review your class notes',
    ],
    'sender': [
        'unknown@random.com',
        'cash@offer.com',
        'manager@company.com',
        'team@company.com',
        'loan@finance.com',
        'mom@family.com',
        'fast@rich.com',
        'boss@company.com',
        'gift@spam.com',
        'teacher@school.com',
    ],
    'label': [1, 1, 0, 0, 1, 0, 1, 0, 1, 0]
})

# Extract sender domain
data['sender_domain'] = data['sender'].apply(lambda x: x.split('@')
[1])

# 2. Feature extraction
vectorizer_text = CountVectorizer()
X_text = vectorizer_text.fit_transform(data['email_text'])

vectorizer_sender = CountVectorizer()
X_sender = vectorizer_sender.fit_transform(data['sender_domain'])

y = data['label'].values
```

```
# 3. Split into labeled and unlabeled (simulate semi-supervised
setting)
X_text_labeled, X_text_unlabeled, X_sender_labeled,
X_sender_unlabeled, y_labeled, y_unlabeled = train_test_split(
    X_text, X_sender, y, test_size=0.6, random_state=42, stratify=y
)

# 4. Initialize classifiers
clf_text = MultinomialNB()
clf_sender = MultinomialNB()

clf_text.fit(X_text_labeled, y_labeled)
clf_sender.fit(X_sender_labeled, y_labeled)

# 5. Co-training loop
new_X_text, new_X_sender, new_y = [], [], []

for i in range(X_text_unlabeled.shape[0]):
    p1 = clf_text.predict_proba(X_text_unlabeled[i])[0]
    p2 = clf_sender.predict_proba(X_sender_unlabeled[i])[0]

    # If both classifiers agree and are confident
    if np.argmax(p1) == np.argmax(p2) and max(p1) > 0.9 and max(p2) >
0.9:
        label = np.argmax(p1)
        new_X_text.append(X_text_unlabeled[i])
        new_X_sender.append(X_sender_unlabeled[i])
        new_y.append(label)

# Add pseudo-labeled data
if new_X_text:
    X_text_combined = vstack([X_text_labeled] + new_X_text)
    X_sender_combined = vstack([X_sender_labeled] + new_X_sender)
    y_combined = np.concatenate([y_labeled, new_y])

    # Retrain classifiers
    clf_text.fit(X_text_combined, y_combined)
    clf_sender.fit(X_sender_combined, y_combined)

    print(f"{len(new_y)} unlabeled samples were pseudo-labeled and
added to training.")
else:
    print("No confident agreement found between classifiers.")

# 6. Evaluate on remaining unlabeled set
y_pred_text = clf_text.predict(X_text_unlabeled)
y_pred_sender = clf_sender.predict(X_sender_unlabeled)

print("Text Classifier Accuracy on Unlabeled Data:",
accuracy_score(y_unlabeled, y_pred_text))
```

```python
print("Sender Classifier Accuracy on Unlabeled Data:",
accuracy_score(y_unlabeled, y_pred_sender))
```

```
No confident agreement found between classifiers.
Text Classifier Accuracy on Unlabeled Data: 0.6666666666666666
Sender Classifier Accuracy on Unlabeled Data: 0.5
```