Tightly Coupled Worksheets and Homework Assignments for NLP

Laura Biester and Winston Wu
lbiester@middlebury.edu, wswu@hawaii.edu

Motivation

<u>Problem:</u> Computer Science students are accustomed to being able to validate the correctness of their solutions. However, in Natural Language Processing (NLP) and Machine Learning (ML) courses, programs often output a metric like accuracy, and students may ask the question *how good is good enough?*

We propose pairing in-class worksheets with test cases for assignments to:
- Encourage students to ensure that they **understand the algorithm** prior to programming
- Provide a **starting point for debugging**, both for students working on their own and for instructors who are helping students

Office hours for an NLP assignment

Development and Approach

	Group #1	Group #2
Students per Section	13	25
Sections	3	2
Age	High School (completed 9th, 10th, or 11th grade)	College (Sophomores, Juniors, and Seniors)
Prerequisites	None, some students had programming experience	Data Structures, Discrete Math, some students had taken ML or Deep Learning
Assignments	#1 only	#1, #2, and #3
Assignments were…	In-class supervised labs with the instructor and a TA	Homework, optionally completed with a partner

Development of Assignments

- **Assignment #1** was developed when working with **Group #1**, high school students taking an NLP course at Carleton College's Summer Liberal Arts Institute
- Students struggled to connect theory from the lecture to training a model on a large dataset
- The approach was adapted and more assignments were developed for **Group #2**, upper-level undergraduates at Middlebury College

Approach

- A paper worksheet is distributed in class
- Students practice the **step-by-step approach** of executing the algorithm on a small dataset
- In a tightly-coupled homework assignment, students are given a `test_mini.py` file, which executes the student's code with the example from the worksheet as input data
- Students are also given `test.py`, which trains the model on a larger dataset and reports metrics like accuracy on a test set

Assignments

Assignment #1: Language Identification with Naive Bayes

Paper Worksheet

Naive Bayes and Classifier Evaluation

1 Naive Bayes

You are building a naive bayes model for language detection using character bigrams, with the following training data:

Language	Sentence	Character Bigrams
english	learn	le ea ar rn
english	ablaze	ab bl la az ze
spanish	hablo	ha ab bl lo

1.1 N-Gram Counts (raw)

Record the counts of each character n-gram for each language below:

	ab	ar	az	bl	ea	ha	la	le	lo	rn	ze
spanish											
english											

1.2 N-Gram Counts (smoothed)

Now in the table below, convert your raw n-gram counts to smoothed versions, using Laplace (add-one) smoothing.

	ab	ar	az	bl	ea	ha	la	le	lo	rn	ze
spanish											
english											

1.3 Classify!

You want to classify a new single-word sentence, *s*: `able`.
Calculate $P(spanish|s)$ and $P(english|s)$. You should ignore unknown character bigrams, and use add-one smoothing.

$P(spanish|s) =$

$P(english|s) =$

test.py

```
from scoring import accuracy_score, confusion_matrix
from util import LANGUAGES, load_data, print_confusion_matrix
from model import NBLangIDModel

def main():
    """
    Use this test script to test your model on a larger data set.
    To test on the full data set with character bigrams, run:
        python test.py data/train.tsv data/test.tsv
    """
    ...
    # load data
    train_sentences, train_labels = load_data(
        args.train_file_path,
        avg_samples_per_language=args.avg_samples_per_language)
    test_sentences, test_labels = load_data(
        args.test_file_path,
        avg_samples_per_language=args.avg_samples_per_language)
    # train model and get predictions
    model = NBLangIDModel(ngram_size=args.ngram_size)
    model.fit(train_sentences, train_labels)
    predictions = model.predict(test_sentences)
    # evaluate model
    print(accuracy_score(test_labels, predictions))
    print_confusion_matrix(confusion_matrix(
        test_labels, predictions, LANGUAGES), LANGUAGES)

if __name__ == "__main__":
    main()
```

test_mini.py

```
import math
from model import NBLangIDModel

def main():
    """
    Use this test script to test your model on the "mini" data set,
which includes
    the example from class.
    """
    # data (yes, these are single words, not actual sentences)
    train_sentences = ["ablaze", "hablo", "learn"]
    train_labels = ["eng", "spa", "eng"]
    test_sentences = ["able"]
    # train model and predict language for the one test sentence
    model = NBLangIDModel(ngram_size=2)
    model.fit(train_sentences, train_labels)
    results = model.predict_one_log_proba(test_sentences[0])
    # convert from log probabilities
    print({lang: math.e ** log_prob
           for lang, log_prob in results.items()})

if __name__ == "__main__":
    main()
```

Assignment #2: Part-of-Speech Tagging with HMMs
- Worksheet: fill out Viterbi table
- Homework: write code to implement Viterbi and optimize smoothing parameters

Assignment #3: Beam Search for Text Generation
- Worksheet: execute beam search with the help of a Colab notebook to interact with GPT-2
- Homework: implement beam search with GPT-2 model (no `test.py` script in this assignment, as there is no training!)

Student Reception

- Students **relied heavily on `test_mini.py`** scripts and used them to debug during office hours
- Ground-truth values on worksheets allowed students to **isolate problems** in their code
- Students gave **positive unsolicited feedback** about the structure of assignments in one-on-one conversations and mentioned the structure in course evaluations

Acknowledgements

- The Hidden Markov Model for part-of-speech tagging assignment was adapted from an assignment created by Rada Mihalcea for an undergraduate NLP course at the University of Michigan
- Thanks to students in CS099 at Carleton College and CSCI0457 at Middlebury College for completing and providing feedback on the assignments
- Thanks to the anonymous reviewers for their feedback and suggestions

assignments are publicly available!