Introduction to Computational Linguistics 2026
PA1 - Text Classification

Due 6 Feb 2026

Instructions

- This is an individual assignment. You may discuss the tasks with classmates or the TA, but you must write your own code independently.
- This programming assignment consists of two parts:
    - Part 1 - You will implement the inference algorithm for logistic regression. Submit logistic_regression.py on MyCourseVille.
    - Part 2 - You will use sklearn library to train a logistic regression classifier and use an LLM library to implement a zero-shot LLM classifier.

Part 1 - Inference for Logistic Regression

Starter code

We will implement the TextClassifier class, which loads a parameter matrix for a bag-of-words model. Here, every word found in the training set is used as a feature. For simplicity, this model does not include a bias (intercept) term in its calculations.

The constructor will load the parameter matrix stored in a CSV file, where each row represents a word and its corresponding weights for different labels. The leftmost column contains words and is named "word". The remaining columns store weights, and their headers are the label names (these depend on the given CSV and are not always fixed as A, B, C—nor limited to just 3 labels).

Example:

word	A	B	C
money	0.2	-0.5	0.3
finance	0.4	0.3	0.2
dust	-0.2	-0.5	0.1
hate	0.1	0.1	0.1
mob	-0.3	0.6	-0.2

From this table, the weights for the word "dust" under labels A, B, and C are -0.2, -0.5, and 0.1, respectively.

In the provided code, the constructor will read the CSV file and store it in an attribute called self.model_parameters, with each row indexed by the word in the "word" column. This allows for easy row selection. You can test this out yourself using “self.model_parameters.loc['money']

Functions to Implement:

1. get_all_possible_features

Returns: A list of strings containing all features (words) in the model.
Example: model.get_all_possible_features() → ['money', ‘finance’, 'dust', 'hate', 'mob']

Hint: Each row’s index in self.model_parameters.index corresponds to a word.

2. get_all_possible_labels

Returns: A list of strings of all possible labels (column headers except "word").

Example: model.get_all_possible_labels() → ['A', 'B', 'C']

3. compute_probability

Tokenize the text using NLTK’s tokenizer. Use the parameter matrix to classify the text into one of the labels from the CSV. Calculate and return a dictionary of probabilities for each label, where:

Keys: Label names, Values: Corresponding probabilities

Ex. I hate dust → [‘I’, ‘hate’, ‘dust’] (tokenization)

Label A: 0.1 (hate) - 0.2 (dust) = -0.1

Label B: 0.1 (hate) - 0.5 (dust) = -0.4

Label C: 0.1 (hate) + 0.1 (dust) = 0.2

(Note: "I" is excluded as it doesn’t exist in the parameter matrix.)

Probability of label A = exp(-0.1) / sum (exp(-0.1) + exp(-0.4) + exp(0.2)) = 0.32
Probability of label B = exp(-0.4) / sum (exp(-0.1) + exp(-0.4) + exp(0.2)) = 0.23
Probability of label C = exp(0.2) / sum (exp(-0.1) + exp(-0.4) + exp(0.2)) = 0.43

As such, the function returns: {'A': 0.32, 'B': 0.23, 'C': 0.43}

Hint 1: Fetch weights directly from self.model_parameters (no feature vector needed as most values come out to zero).

Hint 2: We can use this pseudocode to score the labels.

```
for label in labels:
      for feature in token_list:
            d.loc[feature][label] # weight for this feature-label pair
            update scores
```

4. classify

Predict the highest-probability label for the input text.

Example: For the probabilities {'A':0.32, 'B':0.23, 'C':0.43}, return 'C'

Part 2 - Model Comparison

The goal of this part is to compare the evaluation results from a logistic regression model and a zero-shot LLM classifier. The dataset (training and test sets) that we will use for this part is the financial news data, available for download on mycourseville.

You are required to answer Questions 1-5 and submit your work on mycourseville as pa-part2.pdf

2.1 Logistic Regression

Train a logistic regression model using bag-of-word features on the training set and evaluate the model of the test set. (You might have done this in class already).

Question 1: Report the precision, recall, F1-score for each label, as well as the overall accuracy.

Question 2: Identify the two best-performing labels. Examine the top 10 highest-weighted features for each of these labels and explain why the classifier performs well for them.

Question 3: Identify the two worst-performing labels. Examine the top 10 highest-weighted features for each of these labels and explain why the classifier does not perform well for them.

2.2 Zero-shot LLM Classifier

Implement a zero-shot LLM classifier for the same task and evaluate the performance.

1. **Prompt design**: Write a templatized prompt (i.e. a prompt with a placeholder for the input text) that instructs an LLM to classify a news text string into one of the categories (labels) from the training set. The prompt should:
    - Define the role of the LLM in this task
    - Clearly explain each label
    - Specify the desired output format
    - Place the input text at the end of the prompt (this helps reduce token usage through context caching)

    You should test your prompt on the web-based LLM such as ChatGPT, Gemini, or Claude to make sure that it works decently well.

2. **Implementation**: Write a function that takes a list of news text, inserts it into the prompt, calls an LLM API through a client library, and returns a label string.
    The `typhoon-v2.5-30b-a3b-instruct` is recommended as it gives unlimited free api tokens (within rate limit). However, you may use other models if you prefer
    - You will need to create a free API key here: https://playground.opentyphoon.ai/settings/api-key
    - The TA recommends Deepseek if you want to try another model *(optional, not free)* https://platform.deepseek.com/api_keys
      (It is fairly cheap, and the minimum payment is smaller than most other providers at around 65 baht as of Jan 2026)

Question 4: What is the prompt that you end up using?

3. **Evaluation**: Run the function on the test set, collect the predicted labels, and evaluate them against the gold-standard labels.

Question 5: Does this zero-shot LLM classifier work better or worse than the logistic regression that you trained in Part 2.1? Give a possible explanation for this result.

Note if you have trouble running zero-shot LLM classifiers on the entire test set, sample only 100 (or more) rows and evaluate both logistic regression model and LLM classifier on the same 100 rows to answer Question 5.