

COMP 484/584 Homework 1

Your Name:

2026

In this homework, there are two problems that must be solved by hand and one coding implementation problem. In addition, there is an extra coding implementation problem, which is a bonus for undergraduate students and mandatory for graduate students.

Before you start, please make sure that you know how to run a Jupyter notebook (`.ipynb`) file. A brief tutorial is provided below.

How to run the notebook (`.ipynb`). You may choose either:

- **Option 1 (Local Jupyter).** Install Anaconda and launch JupyterLab (you can search for installation tutorials online), then open `HW1.ipynb`.
- **Option 2 (Google Colab).** Open <https://colab.research.google.com/> and upload `HW1.ipynb`.

To launch JupyterLab locally from a terminal, run:

```
jupyter lab
```

Problem 1 (20pt): Text Representation (BoW and TF-IDF) In this problem, you will compute Bag-of-Words (BoW) and TF-IDF representations by hand. We use a fixed vocabulary with an explicit `<unk>` token for out-of-vocabulary words.

Vocabulary (fixed order):

$\mathcal{V} = [\text{lecture}, \text{homework}, \text{exam}, \text{student}, \text{grade}, \text{<unk>}].$

We consider the following three short documents (already lowercased):

- d_1 : lecture homework homework quiz
- d_2 : exam grade grade student
- d_3 : student lecture exam

Tokenization rule: split on whitespace. Any token not in \mathcal{V} is mapped to `<unk>`. Each occurrence of an out-of-vocabulary token contributes *one* count to `<unk>`.

Part I (12 pt): Bag-of-Words (Binary vs. Count)

In lecture and quizzes, we have primarily used *binary* bag-of-words features, where each vocabulary item is represented by a 1 if it appears in a document and 0 otherwise. In practice, bag-of-words features may also use *counts*, where a word that appears multiple times in a document contributes proportionally to the feature value. In this problem:

- **Binary BoW** indicates whether a token appears at least once in the document.
- **Count BoW** records the number of times each token appears in the document.

Both representations use the same vocabulary and differ only in how repeated words are handled.

1. (4pt) Compute the **binary BoW** vector for document d_1 with respect to \mathcal{V} .
2. (4pt) Compute the **count BoW** vector for document d_1 with respect to \mathcal{V} (including `<unk>`).
3. (4pt) Compute the **count BoW** vector for document d_2 with respect to \mathcal{V} .

Part II (8 pt): TF-IDF Let N be the number of documents in the corpus (here $N = 3$). Recall in the class, for a token $t \in \mathcal{V}$:

- Term frequency (TF): $\text{tf}(t, d)$ is the **raw count** of t in document d .
- Document frequency (DF): $\text{df}(t)$ is the number of documents in which t appears at least once.
- Inverse document frequency (IDF):

$$\text{idf}(t) = \ln \left(\frac{N}{\text{df}(t)} \right).$$

- TF-IDF:

$$\text{tfidf}(t, d) = \text{tf}(t, d) \cdot \text{idf}(t).$$

1. (4pt) Compute $\text{df}(t)$ for each token $t \in \mathcal{V}$.
2. (4pt) Compute the TF-IDF vector for document d_1 with respect to \mathcal{V} . You may leave your answer in terms of $\ln(\cdot)$.

Problem 2 (30 pt): Classification In this problem, we study binary classification models trained using *negative log-likelihood (NLL)* loss. Unless otherwise stated, assume the sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Throughout this problem, consider a single training example (\mathbf{x}, y) with label $y \in \{0, 1\}$.

Part I (12 pt): A Simple Two-Layer MLP We first consider a simple two-layer neural network for binary classification.

The model is defined as:

$$\mathbf{h} = W^{(1)}\mathbf{x}, \quad z = \mathbf{w}^{(2)\top}\mathbf{h} + b, \quad \hat{y} = \sigma(z),$$

where $\mathbf{x} \in \mathbb{R}^2$, $W^{(1)} \in \mathbb{R}^{2 \times 2}$, $\mathbf{w}^{(2)} \in \mathbb{R}^2$, and $b \in \mathbb{R}$.

The loss for one data point is the binary cross-entropy (negative log-likelihood):

$$\mathcal{L} = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})].$$

1. (5 pt) Explain why logistic regression and neural networks are trained by maximizing the *log-likelihood* instead of the likelihood.
2. (7 pt) Assume $y \sim \text{Bernoulli}(\hat{y})$. Derive the negative log-likelihood loss for a single training example and show that it reduces to the binary cross-entropy loss above.

Part II (4 pt): Forward Pass Consider a binary classifier with multiple linear paths and no bias terms. Also for simplicity, we assume no intermediate non-linearity (e.g., $\sigma(x)$) unless specified. Given an input vector $\mathbf{x} \in \mathbb{R}^d$, the model is defined as follows:

$$\begin{aligned} \mathbf{a}_1 &= W_1 \mathbf{x}, \\ \mathbf{a}_{21} &= W_{21} \mathbf{a}_1, \quad \mathbf{a}_{22} = W_{22} \mathbf{a}_1, \\ \mathbf{a}_2 &= \mathbf{a}_{21} + \mathbf{a}_{22}, \\ a_3 &= \mathbf{w}_3^\top \mathbf{a}_2, \quad a_u = \mathbf{u}^\top \mathbf{x}, \\ z &= a_3 + a_u, \quad \hat{y} = \sigma(z), \end{aligned}$$

where $\sigma(\cdot)$ denotes the sigmoid function.

Write \hat{y} explicitly as a function of \mathbf{x} , W_1 , W_{21} , W_{22} , \mathbf{w}_3 , and \mathbf{u} .

Part III (7 pt): Backward Pass Using the same network definition, assume the binary cross-entropy loss

$$\mathcal{L} = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})].$$

Derive symbolic expressions (no numerical computation required) for:

$$\frac{\partial \mathcal{L}}{\partial W_1} \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{u}},$$

in terms of $\mathbf{x}, W_1, W_{21}, W_{22}, \mathbf{w}_3, \hat{y}$, and y .

Part IV (7 pt): Backward Pass cont. Suppose we modify the model by inserting an additional linear layer W_{super} after W_1 , such that

$$\mathbf{a}_{21} = W_{21} W_{\text{super}} \mathbf{a}_1, \quad \mathbf{a}_{22} = W_{22} W_{\text{super}} \mathbf{a}_1,$$

and the rest of the model remains unchanged.

1. Derive the gradient $\frac{\partial \mathcal{L}}{\partial W_{\text{super}}}$.
2. Derive the gradient $\frac{\partial \mathcal{L}}{\partial W_1}$.

Problem 3 (50 pt): Text Classification with Bag-of-Words (Coding)

In this problem, you will implement text classification models using Bag-of-Words (BoW) representations and a neural network classifier. This problem extends the toy BoW and TF-IDF computations in Problem 1 to a real dataset. You will complete this problem using the provided Jupyter notebook `HW1.ipynb`.

Dataset. We use the **AG News** dataset, which consists of news articles labeled into four categories:

$$\{\text{World, Sports, Business, Sci/Tech}\}.$$

This is a 4-class classification problem.

Provided Components. In the notebook, the following components are already implemented for you:

- Dataset download and train/dev/test split
- Word-level tokenizer
- Vocabulary construction (training data only)
- Evaluation code for Accuracy and Macro-F1

You should *not* modify these components.

Part I (10 pt): Binary Bag-of-Words In this part, you will implement a *binary* Bag-of-Words representation.

Given a document and a vocabulary of size V , the binary BoW representation is a vector $\mathbf{x} \in \mathbb{R}^V$ such that

$$x_j = \begin{cases} 1 & \text{if token } j \text{ appears at least once in the document,} \\ 0 & \text{otherwise.} \end{cases}$$

1. (5 pt) Implement binary BoW vectorization in the notebook.
2. (5 pt) Construct binary BoW feature matrices for the training, development, and test sets.

Part II (10 pt): Count Bag-of-Words In this part, you will implement a *count*-based Bag-of-Words representation.

For a document and vocabulary of size V , the count BoW representation is a vector $\mathbf{x} \in \mathbb{R}^V$ such that

$$x_j = \text{number of times token } j \text{ appears in the document.}$$

1. (5 pt) Implement count BoW vectorization in the notebook.
2. (5 pt) Construct count BoW feature matrices for the training, development, and test sets.

Part III (22 pt): Neural Network Classifier In this part, you will implement a two-layer neural network classifier using PyTorch.

The classifier has the following form:

$$\mathbf{h} = \text{ReLU}(W_1 \mathbf{x}), \quad \mathbf{z} = W_2 \mathbf{h},$$

where \mathbf{x} is a BoW feature vector and $\mathbf{z} \in \mathbb{R}^4$ are the output logits.

1. (8 pt) Implement the two-layer neural network

$$\text{Linear} \rightarrow \text{ReLU} \rightarrow \text{Linear}.$$

2. (14 pt) Implement the training loop using PyTorch autograd. You may perform backpropagation either by calling `loss.backward()`, or by using `torch.autograd` (e.g., `torch.autograd.grad`).

You should use the cross-entropy loss for multi-class classification.

Part IV (8 pt): Comparison and Analysis Train the neural network classifier using:

- Binary Bag-of-Words features
- Count Bag-of-Words features

Report the following metrics on the development and test sets:

- Accuracy
- Macro-averaged F1 score

1. Compare the performance of binary and count BoW representations.
2. Briefly explain why one representation may outperform the other on the AG News dataset.

Your explanation should be written in the designated markdown cell in the notebook.

Problem 4 (10 pt): TF-IDF (Graduate Required / Undergraduate Bonus) In this part, you will extend the count Bag-of-Words representation to TF-IDF.

Using the training data only, define:

$$\text{idf}(t) = \log \left(\frac{N}{\text{df}(t)} \right),$$

and construct TF-IDF feature vectors.

1. Implement TF-IDF vectorization in the notebook.
2. Train the *same* neural network classifier that you implemented in Problem 3 using TF-IDF features.
3. Compare TF-IDF with binary and count BoW representations.

Briefly explain whether TF-IDF performs better or worse in this setting, and why. Your explanation should be written in the designated markdown cell in the notebook.

Submission. You should submit:

- A PDF report containing your answers and analysis. For Problems 1–2, you may choose either handwritten work or L^AT_EX.
- The completed Jupyter notebook with your implementation `HW1.ipynb`.
- Please submit your homework (a PDF + a .ipynb) as **one single .zip file** named with your NetID:

`NLP_HW1_<NETID>.zip`

Please make sure that your report is **exported as a PDF**. Do **not** submit Word documents or LaTeX source files.

You must submit the .zip file to Canvas before the deadline.