

Homework 2

New Attempt

- Due Wednesday by 11:59pm
- Points 56
- Submitting a file upload
- File Types pdf, py, and txt
- Available Feb 10 at 12am - Mar 5 at 11:59pm

Learning objectives

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#learning-objectives

After completing this assignment, students will be able to:

- Demonstrate how weights adjust to better fit training input in stochastic gradient descent
- Implement a text classification system using both feature-based and neural network approaches
- Identify informative features in a feature-based text classification system
- Analyze errors in an NLP system

Part 1: Learning weights in logistic regression

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#part-1-learning-weights-in-logistic-regression

You are training a classifier for reviews of a new product recently released by a company. You design a couple of features, x_1 and x_2 . You will be using logistic regression. With an initialization of the weights w_1 , w_2 and b (the bias) all set = 0 and a learning rate $\eta = 0.2$, calculate the weights after processing each of the following 3 inputs:

1.	$x_1 = 2$	$x_2 = 1$	$y = 1$
2.	$x_1 = 1$	$x_2 = 3$	$y = 0$
3.	$x_1 = 0$	$x_2 = 4$	$y = 0$

During calculations, keep at least 3 significant digits for values. Points will not be taken off for slight differences due to rounding.

Deliverables for Part 1


https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#deliverables-for-part-1

In the report:

- Give weights after training on each data point (3 total weight changes, one after each timestep/data point).
- Show your work in calculating the values of the weights after training on each data point.
- Briefly comment on any shift in weights from positive to negative or negative to positive and why this was the case.

Part 2: Implement a deception classifier



https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#p2-implement-a-deception-classifier


In this portion, you will design and implement a program to classify if a comment from a player of the [Diplomacy](https://en.wikipedia.org/wiki/Diplomacy_(game))  game is truthful or not. You can use any packages you want for this (scikit-learn, spaCy, NLTK, Gensim, code from Homework 1, etc). Any packages used should be specified in the `README.txt` file, along with version numbers for Python and all packages. If you will be using a language other than Python, please let us know before submitting. Your script should be able to take the filename of a dataset as a single keyword argument.

Dataset

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#dataset

Here is the dataset that you should download for this assignment:

- [diplomacy_cv.csv](https://drive.google.com/file/d/1xGq8i8uaUzHLVM9rDzWHX2TYFAi68I5Z/view?usp=sharing) . This dataset has a variety of fields, but the most important are:
 - `text`: the text of the comment
 - `intent`: 0 for truth, 1 for lie
- [diplomacy_test.csv](https://drive.google.com/file/d/1Wa9LrkH0mxNLD34nNb9jA31RNpipqmhF/view?usp=sharing)  (only necessary if participating in the optional challenge). This data has the same fields as the training data. You will use this in the optional challenge competition hosted on Kaggle.

This dataset is from a recording of online players of Diplomacy, as presented in [Peskov et al. 2020](https://aclanthology.org/2020.acl-main.353/)  <https://aclanthology.org/2020.acl-main.353/>. Negotiation and back-stabbing are key elements of the Diplomacy game.

2.1 Feature-based logistic regression models

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#feature-based-logistic-regression-models

In this section, you will build a logistic regression model based on bag-of-word features and/or features of your own design. You can do whatever preprocessing you see fit. You will report performance using 5-fold cross-validation on the `diplomacy_cv.csv` dataset, which you will set up. Make sure to just extract features (bag-of-words, etc) from the training set and not the test folds within cross-validation.

Tasks for section 2.1

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#tasks-for-section-2.1

Implement and try the following feature and model combinations:


- *Logistic regression with bag-of-words (unigram) features.* Build a logistic regression classifier that uses bag-of-words (unigram) features.
- *Logistic regression with your own features/change in preprocessing.* Design and test at least two modifications (custom features or preprocessing changes) to unweighted unigram features. Note that these features can be used in conjunction with bag-of-words features or by themselves. Possible features/changes to add and test include:
 - Tf-idf transformed bag-of-words features
 - Changing count bag-of-words features to binary 0 or 1 for the presence of unigrams
 - N-gram features (sequences of words) beyond the single words used for the bag-of-words features
 - Different preprocessing (stemming, different tokenizations, stopword removal)
 - Reducing noisy features with feature selection
 - Counts or added weight from custom word lists
 - Static word embeddings of your choice (do not use any contextualized word embeddings, such as BERT, for this part)
 - Any other custom-designed feature (such as length of input, number of capitalized words, etc)

You will thus have 3 total logistic regression models: one using bag-of-word features and 2 with your own selected features or preprocessing changes.

In the report, please provide:

1. A table of 5-fold cross-validation performance scores for models trained on each set of features. Include accuracy as well as precision, recall, and f1-score for the positive (lying) class. Please

average these scores across the 5 folds for each evaluation metric (there is no need to include scores for each fold).

2. For each feature or change in input text processing:
 1. Describe your motivation for including the feature
 2. Discussion of results: Did it improve performance or not? (Either result is fine. It is not necessary to beat logistic regression with unigram features. This is a very difficult task.)
3. For a feature-based model of your choice (not a neural model):
 1. Extract and discuss the most informative features that are mostly strongly positively and negatively associated with deception. Report the 5 features with the highest weights and 5 features with the lowest (negative) weights. Discuss how these may or may not make sense for this task. You may adapt code provided in the Naive Bayes (a simple generative classification model) example (notebook [here](https://colab.research.google.com/drive/1J-FcHTFYXTsNgcEB19kQcqJnE8CsbSe7?usp=sharing) , use another source online, or write your own. Give specific informative features, such as particular words (e.g. “actually”) for bag-of-words features, instead of sets of features like “tf-idf unigram features”.
 2. Do an error analysis. Provide a confusion matrix, sample examples from both false negatives and false positives and present a few of them in the report. Do you see any patterns in these errors? How might these errors be addressed with different features or if the system could understand something else? (You don’t have to implement these, just speculate.)

2.2 Neural network-based approaches

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#neural-network-based-approaches

In this section, you will build and evaluate neural network-based classifier on deception classification. For example, you could implement a feedforward neural network that uses pre-trained static word embeddings (word2vec, GloVe, FastText, etc) as input. You can download these pre-trained word embeddings from wherever you like (you don’t have to train your own). To represent the document, you could take the average word embeddings of the input sentence or choose another function. You can choose which activation function to use and other hyperparameters. You are also welcome to try other methods we haven’t yet covered in class, such as LSTMs, convolutional neural networks, BERT, or other LLMs. As long as the technique uses neural networks at some point in its architecture and involves some sort of training or fine-tuning of a model, it will be accepted. Simply prompting a pre-trained LLM to classify the instances (“zero-shot” or “in-context” learning) will not be sufficient. You can also incorporate any non-text metadata features in this part. If you have questions about what is acceptable, ask the instructor or TA.

You will again use 5-fold cross validation on the dataset. There is no need for this model to outperform the logistic regression model you made.

Tasks for section 2.2

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#tasks-for-section-2.2

Implement a neural network-based deception classifier, such as a feedforward neural network with static word embeddings as input.

In the report, please provide:

- Performance scores for this model. Include accuracy as well as precision, recall, and f1-score for the positive (polite) class. This can be an additional row in the table with other performance scores. It does not need to outperform the logistic regression model.
- Discuss the motivation for any choices you made as far as neural classifier, word embedding types, pretraining dataset, and/or how you represented the document, or if you experimented with multiple of these options.
- Discuss the motivation for any choices you made as far as network architecture (number and dimensions of hidden layers) or hyperparameters (learning rate, number of epochs, etc). Note if you experimented with any of these options.

2.3 (Optional) Submit your classifier in the class challenge



https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#submit-your-classifier-in-the-class-challenge



Optionally, you can submit your classifier to run on a hidden held-out test set as part of a class competition. Bonus points will be awarded in the competition as follows, as measured by accuracy on our held-out test set.

- 4 bonus points for the best-performing logistic regression classifier
- 4 bonus points for the best neural network classifier
- 2 bonus points for the 2nd best-performing logistic regression classifier
- 2 bonus points for the 2nd best-performing neural network classifier
- 1 bonus point for submitting any system (either logistic regression or neural network)

How to submit your classifier

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2.html#how-to-submit-your-classifier

There are two Kaggle competitions. See these pages for instructions on how to submit:

- [Logistic regression Kaggle competition](https://www.kaggle.com/competitions/hw-2-lr-deception-classification-cs-2731-spri-2025/models)  (<https://www.kaggle.com/competitions/hw-2-lr-deception-classification-cs-2731-spri-2025/models>)
- [Neural network Kaggle competition](https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.kaggle.com%2Fcompetitions%2Fhw-2-for-cs-2731-spring-2025&data=05%7C02%7CXIANGLLI%40pitt.edu%7Cd862dd65d3344174b78108dd4af97713%7C9ef9f489e0)  (<https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.kaggle.com%2Fcompetitions%2Fhw-2-for-cs-2731-spring-2025&data=05%7C02%7CXIANGLLI%40pitt.edu%7Cd862dd65d3344174b78108dd4af97713%7C9ef9f489e0>)

You will need to create a Kaggle account to submit. Please provide your Kaggle username used in the competition in your report so we can assign any bonus points. Note that this username will be visible in a leaderboard to other challenge competition participants.

Notes

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#n

- Don't feel like you need to write things from scratch; use as many packages as you want. Google and Stack Overflow and NLP/ML software documentation are your friend! Adapting and consulting other approaches is fine and should be noted in comments in the code and/or in the `README.txt`. Just don't use complete, fully-formed implementations for this (including from generative AI tools). Use all resources as aids, not as a final product.
- Optionally, you may incorporate any form of regularization that you like
- This homework is designed to be able to be run on a laptop with CPUs, not GPUs. Let the instructor/TA know if you are having difficulty completing it with the resources you have.

Deliverables

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#d

- Your report with results and answers to questions in Part 1 and Part 2, named `hw2_{your pitt email id}.pdf`. No need to include @pitt.edu, just use the email ID before that part. For example: `report_xianglli_hw2.pdf`.
 - If participating in the challenge, the Kaggle username you used to submit your predictions
 - If participating in the challenge, your code used for that in a file named `hw2_{your pitt email id}_test.py`.
- Your code used to train models and estimate performance with cross-validation in a file named `hw2_{your pitt email id}_cv.py`.
- A `README.txt` file explaining
 - how to run the code you used to train your models and estimate cross-validation performance
 - the computing environment you used, including the names and versions of programming languages and packages used, in case we replicate your experiments. A `requirements.txt` file for setting up the environment is useful if there are many packages.
 - any additional files needed to run the code, such as the names and versions of pretrained embeddings
 - any additional resources, references, or web pages you've consulted

- any person with whom you've discussed the assignment and describe the nature of your discussions
- any generative AI tool used, and how it was used
- any unresolved issues or problems

Please submit all of this material on Canvas. Do **not** zip all files. We will grade your report and look over your code.

Acknowledgments

https://michaelmilleryoder.github.io/cs2731_fall2024/hw2#a

This assignment is inspired from a homework assignment by Prof. Diane Litman. Data is from [Peskov et](#)

Homework 2

Criteria	Ratings		Pts
Part 1: all work is shown	3 pts Full Marks	0 pts No Marks	3 pts
Part 1: Correct weights are given after each timestep	4 pts Full Marks	0 pts No Marks	4 pts
Part 1: Discusses sign flips in weight changes during training positive to negative weights after seeing negative examples with those features	3 pts Full Marks	0 pts No Marks	3 pts
Part 2: Explanation of environment in README	3 pts Full Marks	0 pts No Marks	3 pts
Part 2: Code is provided	3 pts Full Marks	0 pts No Marks	3 pts
Part 2.1: Implements LR with unigrams with proper cross-validation feature extraction and model training only on the training portions of each fold	3 pts Full Marks	0 pts No Marks	3 pts
Part 2.1: Implements at least 2 modifications to LR with unigrams change in features or preprocessing count	6 pts Full Marks	0 pts No Marks	6 pts
Part 2.1: Describes motivation for modifications and discusses results	4 pts Full Marks	0 pts No Marks	4 pts
Part 2.1: Provides a table with performance of LR models tried (accuracy, precision, recall, f1)	4 pts Full Marks	0 pts No Marks	4 pts
Part 2.1: Lists at least 5 informative features for an LR classifier, discusses	4 pts Full Marks	0 pts No Marks	4 pts
Part 2.1: Error analysis with discussion of patterns, how errors might be addressed	4 pts Full Marks	0 pts No Marks	4 pts

Criteria	Ratings		Pts
Part 2.2: Implements a neural approach with proper cross-validation	8 pts Full Marks	0 pts No Marks	8 pts
Part 2.2: Provides a table with performance (accuracy, precision, recall, f1)	4 pts Full Marks	0 pts No Marks	4 pts
Part 2.2: Discusses motivation for model choices	3 pts Full Marks	0 pts No Marks	3 pts
(Optional) Kaggle competition points +1 for entering either or both LR or NN competitions +2 for 2nd-best performing LR +2 for 2nd-best performing NN +4 for best LR +4 for best NN	0 pts Full Marks	0 pts No Marks	0 pts
Total Points: 56			