CSC2042S 2025
Assignment 3
Logistic Regression News Classification

Total marks: 30

In this assignment you will implement a multinomial logistic regression model for news category classification. You will train your model on the MasakhaNEWS dataset. It is available on Amathuba under Resources > Datasets > A3-dataset or you could download it from here: https://github.com/masakhane-io/masakhane-news/tree/main, where you will also find a detailed description of the dataset.

MasakhaNEWS contains real news articles, each of which is labelled with one news category, (*business*, *entertainment*, *health*, *politics*, *religion*, *sport*, *technology*). It is actually a multilingual collection of datasets, covering 16 languages spoken across Africa. Not all languages include articles for all seven categories listed above – some have a more limited set of classes. For details about the dataset, see this paper (Table 2 on p4 shows the full dataset statistics).

For this assignment, you will train and evaluate separate models for two languages: English and isiXhosa. Each language covers a different subset of news categories, so the number of classes will not be constant. They also differ linguistically and in the size/quality of their datasets, which may affect the difficulty of the classification task and the role of feature extraction. You are expected to experiment with hyperparameters and modelling decisions to analyse how they affect performance for each language.

Tasks

1. Data processing (3 marks)

Load the MasakhaNEWS dataset for the three target languages. The data is already split into train/dev/test. Implement a pipeline for transforming the article text into vector representations.

- Apply basic text cleaning (e.g. lowercasing, handling punctuation, etc.) and tokenisation. You do not have to experiment with different preprocessing settings, but do motivate your decisions since they can affect vocabulary size and model performance.
- Convert the headline + text (stored in the full_text column) into numerical vectors using feature extraction methods (see Section 5 for details).

2. Multinomial logistic regression implementation (3 marks)

Implement a **`MultinomialLogisticRegression`** class that defines a multinomial logistic regression model with a softmax output layer. You are required to use an object-oriented design in PyTorch. Your class should include at least three methods:

- **`forward`**: process a batch of inputs during training.
- **`compute_probabilities`**: compute probabilities for a batch of inputs.
- **`predict`**: return the predicted classes for a batch of inputs.

3. Training (2 marks)

Your model should be trained with a cross-entropy loss for multi-class classification. Implement a training loop in PyTorch. At each epoch, shuffle the training data and cycle through it in mini-batches, computing the loss, gradients, and updating parameters with gradient descent. Motivate your choice of stopping criteria (e.g. fixed number of epochs or early stopping based on validation accuracy).

4. Hyperparameter tuning (2 marks)

For each language, you should individually tune the learning rate and batch size. Discuss how these hyperparameter values influence convergence, stability, and overall performance. Provide evidence from your experiments (e.g. results from a grid search and validation loss curves).

5. Feature extraction (4 marks)

In addition to hyperparameters, also investigate different feature extraction methods for text. Compare the following three methods (available in scikit-learn) for converting text into vectors:

- **Bag-of-words** (count-based) representations
- **Binary presence features** (word occurs / does not occur)
- **TF–IDF features** (to weight frequent vs informative terms)

In each case, experiment with vocabulary size control – either by setting a maximum number of features or by applying frequency cutoffs to ignore very rare words. Discuss how these settings change training times and results.

Tune the hyperparameters (learning rate and batch size) to maximise validation accuracy, but do this separately for each language and feature extraction method. For each case, identify the best configuration based on validation accuracy, and then evaluate this configuration on the test set. In your report, discuss whether different feature types or vocabulary settings proved more effective for English vs isiXhosa.

6. isiXhosa training decisions (4 marks)

The isiXhosa dataset of MasakhaNEWS is smaller and more imbalanced than the English dataset. For isiXhosa, conduct additional experiments to investigate the following approaches (separately, you don't have to combine them):

- **Regularization**: Compare both L1 and L2 penalties, tuning the regularisation coefficients and analysing how the techniques influence overfitting, stability, and generalisation.
- **Handling class imbalance**: Experiment with strategies such as upsampling minority classes or downsampling majority classes, and discuss their impact on performance.

You may reuse the best isiXhosa hyperparameter settings and feature extractor based on earlier experiments without re-tuning them in combination with these techniques. Focus instead on experimenting with different regularisation coefficients and sampling rates. Report whether either of these approaches improve performance. For these experiments, only compare models on the validation set. Once you have identified the best approach, evaluate it on the test set and compare its performance against your previously best isiXhosa model without these techniques.

7. Evaluation (4 marks)

In all your test evaluations, report the following metrics overall: Accuracy, Precision, Recall, F1. In reporting the overall performance (across all classes), report both micro- and macro-averaged results (both are available in scikit-learn). Micro-averaging gives more weight to majority classes, while macro-averaging treats all classes equally, which is especially relevant for imbalanced datasets like isiXhosa. Discuss any differences between the two.

Use a confusion matrix to visualise how performance differs across news categories – which categories are classified most reliably and which categories are most often misclassified.

8. Weight analysis (2 marks)

For one language of your choice (one you understand), analyse the learned softmax weights. Identify whether specific keywords or groups of words strongly contribute to certain categories. Provide examples where the model has learned plausible associations, or cases where it may be overfitting or learning incorrect associations.

9. Code quality and reproducibility (3 marks)

Your code should be well-structured and reproducible. Use fixed random seeds where appropriate, so we can rerun your code to replicate your results, and organise your notebook with clear functions, classes, and documentation.

10. Final report (3 marks)

Alongside your notebook, submit a short PDF report (maximum 5 pages). This should describe your OOP design (the classes you used), describe your hyperparameter tuning process, and present your key results and insights.

Submission Requirements

Submit a single compressed archive named as your student number, e.g. GWRBRA001.tar.xz. The code should be submitted as a Jupyter notebook with clear documentation. Version control with git is recommended but not required. The code should be able to run with only the provided data and standard libraries such as Pillow, numpy, scikit-learn, matplotlib, and PyTorch (check with a tutor if you are unsure about using other libraries). A README file should include setup instructions and a description of each submitted file. Do not submit any data.

Also include a report of at most 5 pages (submitted as a PDF document) that describes your design decisions, reports and discusses your results, including visualizations. Marks will be given primarily based on showing understanding of the concepts being applied and demonstrating ability to develop appropriate models and to analyse the results. Tasks 1–8 will be marked based on both your report and the code in your notebook.

Please ensure that your tarball works and is not corrupt (you can check this by trying to downloading your submission and extracting the contents of your tarball - make this a habit!). **Corrupt or non-working tarballs will not be marked - no exceptions. A 10% penalty will be incurred for a submission that is one day late (handed in within 24 hours after the deadline). Any submissions later than 1 day will be given a 0% mark.**

Academic integrity

All code and analysis must be your own work. You may use standard libraries (in line with the specification of what is permitted for the different components of the assignment) and reference documentation, but must cite all sources. You may discuss your work with other students, but code sharing is prohibited. Use of AI tools must be declared and limited to debugging assistance only. AI assistance is not permitted for writing the report.

Marking rubric

Category	**Marks**
Data processing	3
Logistic regression implementation	3
Training	2
Hyperparameter tuning	2
Modelling decisions	4
isiXhosa training decisions	4

Evaluation	4
Weight analysis	2
Code quality	3
Report quality	3
Total	**30**