NLP Assignment 3: Word Embeddings and Sequence Models

(Group assignment – groups must be the *same* as your Final Project teams)

Fall-2025
Instructor: Ayesha Enayet

Overview. This assignment has two major questions. In **Q1**, you will implement Word2Vec Skip-Gram with *full softmax* from scratch and visualize the learned embeddings using t-SNE. In **Q2**, you will build an Urdu sentiment classifier using *RNN-based models* (Simple RNN, LSTM, or BiLSTM). Overall, the focus is on correct implementation, analysis quality, and clear illustrations (t-SNE, unigram/bigram frequencies). Cite all resources you use. During the viva, each member must be able to explain the entire codebase.

Important Note: *Failure to satisfactorily explain your code or analysis during the viva may result in a score of* ***zero*** *for the entire assignment, regardless of the written submission. Ensure every group member fully understands all parts of the implementation and report.*

Question 1 — Word2Vec (Skip-Gram with Softmax) + t-SNE Visualization (50 pts)

Corpus

Use the provided small corpus:
`https://drive.google.com/drive/folders/1I5nqVXGIPFqXMNqNf1TjPdfXX1c5WVWj?usp=sharing`
(vocabulary ≈ 75; ≈ 42k tokens).

Tasks

1. **Implement Skip-Gram with full softmax (no negative sampling).** You may use NumPy/PyTorch/TensorFlow , but do *not* use pre-built word2vec implementations.
   - Preprocess text: tokenization, vocabulary building, word–index mapping.
   - Generate (center, context) pairs for a fixed window.
   - Compute probabilities with softmax; optimize cross-entropy loss via SGD/Adam.
   - Train and save embeddings.
2. **Analyze & visualize.**
   - 2-D visualization using t-SNE; label points.
   - Plot unigram and bigram frequency distributions for the corpus; discuss effects of frequent words.
   - Implement `most_similar(word)` (cosine similarity) and qualitatively evaluate neighborhoods.
3. **Improve model performance.** Tune embedding dimension, learning rate/optimizer, context window, epochs; compare runs and discuss how changes affect semantic clustering.

Deliverables (Q1)

- Code (well-structured and commented).

- Plots: labeled t-SNE; unigram/bigram frequencies; (optional) loss curve.
- Report (2–3 pages) describing implementation, analysis of clusters, challenges, and improvements.

Suggested resources (Q1)

- Jay Alammar, *The Illustrated Word2Vec*: https://jalammar.github.io/illustrated-word2vec/
- Mikolov et al. (2013), *Efficient Estimation of Word Representations*: https://arxiv.org/abs/1301.3781
- Chris McCormick, *Word2Vec Tutorial: The Skip-Gram Model*: https://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
- Softmax implementation (NumPy example): https://machinelearningmastery.com/softmax-activatio
- t-SNE in scikit-learn (API): https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
- Distill, *How to Use t-SNE Effectively*: https://distill.pub/2016/misread-tsne/

Question 2 — Urdu Sentiment Classification with RNNs (Competition-Based, 50 pts)

Objective

Train a sentiment classifier for Urdu text using **Simple RNN, LSTM, or BiLSTM**. This is a **competition-based** task: you will report metrics and be ranked; the top-performing group (highest F1) gets full marks, others receive relative scores. You are **allowed to use frameworks and built-in libraries such as TensorFlow, Keras, or PyTorch** for this question. The focus will be on the quality of preprocessing, model design, and analysis rather than coding everything from scratch.

Dataset

Use the public Urdu Sentiment Corpus:
https://github.com/MuhammadYaseenKhan/Urdu-Sentiment-Corpus

Tasks

1. **Preprocessing.** Explore labels; normalize Urdu text, tokenize, split into train/validation/test.
2. **Modeling.** Build an RNN-based classifier (Simple RNN / LSTM / BiLSTM) in PyTorch/TensorFlow/Keras. You may use:
   - Pretrained embeddings (e.g., FastText/Word2Vec for Urdu),
   - Self-trained embeddings (e.g., from Q1 or trained on the corpus),
   - One-hot encoded embeddings.
3. **Evaluation.** Compute and report **Accuracy**, **Precision**, **Recall**, and **F1-score** on the final test split.
4. **Competition reporting.** Before submission, record your metrics (Accuracy, Precision, Recall, F1) in the following shared sheet https://docs.google.com/spreadsheets/d/1TXr-uldSY6Fe3pprJX edit?usp=sharing. Ranking is based primarily on F1, precision, and recall.

Deliverables (Q2)

- Code (with comments) and a brief model summary (architecture; hyperparameters).
- Tables/plots of evaluation metrics (train/validation/test) and a confusion matrix.
- 2–3 page report: data preparation, model choices, training details, results, and reflections.

Guidelines (Both Q1 & Q2)

- **Group policy:** Work in the *same groups as your Final Project teams.* Submit one codebase and one report per group.
- **Use of resources:** You may consult online tutorials, papers, and code, but you must **cite all sources clearly**. Uncredited code will be penalized.
- **Viva:** Every member must be able to explain all parts of the implementation and analysis; marks may be deducted individually.

Submission & Packaging

Submit a single archive containing:

- `word2vec/` (Q1 code + plots) `sentiment_classifier/` (Q2 code + plots)
- `report.pdf` (2–3 pages for each question, combined in one 4–6 pages)
- Any additional resources (e.g., saved embeddings)

Evaluation Rubric (Total 100 pts)

Q1 — Word2Vec (Skip-Gram + Softmax)	
Correct implementation	20
Visualization & analysis (t-SNE, frequencies)	20
Code clarity & documentation	10
Q2 — Urdu Sentiment Classification (RNN)	
Preprocessing & model design	10
Evaluation metrics & performance	20
Report analysis & interpretation	10
Relative ranking (competition)	10
Total	**100**

Credit: ChatGPT has been used to assist with the editing and formatting of this assignment.

Reminder: The better the training, the more semantically coherent neighborhoods you will observe in t-SNE for Q1. For Q2, tuning embeddings and architecture (e.g., BiLSTM with pretrained vectors) typically improves F1. Do your best.