Natural Language Processing **Tel Aviv University**

Assignment 2: Language Models

Due Date: *December 28, 2025* Lecturer: Dr. Tal Wagner, TA: Idan Tarshish

Preliminaries

Submission Instructions This assignment includes both theoretical and coding questions. Your submission should be one zip file that includes:

1. A single PDF file in digital format (we recommend using Overleaf, but you may use any digital editor you prefer). The PDF should contain:
   (a) Answers to Q1a+b+d.
   (b) Answer to Q2a and the plot from Q2c
   (c) Answers to all three sub questions of Q3.
   (d) Answers to Q4a+b.
   (e) Answers to all seven sub questions of Q5.
   (f) AI disclosure & Reflection (Q6)
2. The directory `q1_2_3`, including all its files, with the **completed** `.py` files, the `saved_params_40000.npy` and the **completed** `q2_char_rnn_generation`
3. The **completed** notebook `q4_transformer`.
4. Your new notebook with the code to Section 5 named `q5_sst.ipynb`.

Submit your solution via Moodle. Your submission should be a single zip file named `<id1>_<id2>_<id3>.zip`, where `id1` is the ID of the first student. The zip file should include all of the above components.

Notes

- Only one student needs to submit.
- Your implementation should be efficient and vectorized whenever possible (i.e., use numpy matrix operations rather than `for` loops). A non-vectorized implementation will not receive full credit!
- Your code should follow coding standards (well-documented, self-explained, meaningful naming, PEP8).

1 Word-Level Neural Bi-gram Language Model (20 pt)

In this question, you will implement and train neural language model, and evaluate it on using perplexity.

(a) Derive the gradient with respect to the input of a softmax function when cross entropy loss is used for evaluation, i.e., find the gradients with respect to the softmax input vector $\boldsymbol{\theta}$, when the prediction is made by $\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{\theta})$. Cross entropy and softmax are defined as:

$$\text{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_i y_i \cdot \log(\hat{y}_i)$$

$$\text{softmax}(\boldsymbol{\theta})_i = \frac{\exp(\theta_i)}{\sum_j \exp(\theta_j)}$$

The gold vector $\boldsymbol{y}$ is a one-hot vector, and the predicted vector $\hat{\boldsymbol{y}}$ is a probability distribution over the output space.

(b) Derive the gradients with respect to the input $\boldsymbol{x}$ in a one-hidden-layer neural network (i.e., find $\frac{\partial J}{\partial \boldsymbol{x}}$, where J is the cross entropy loss $\text{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}})$). The neural network employs a sigmoid activation function for the hidden layer, and a softmax for the output layer. Assume a one-hot label vector $\boldsymbol{y}$ is used. The network is defined as:

$$\boldsymbol{h} = \sigma(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1),$$
$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{h}\boldsymbol{W}_2 + \boldsymbol{b}_2).$$

The dimensions of the vectors and matrices are $\boldsymbol{x} \in \mathbb{R}^{1\times D_x}, \boldsymbol{h} \in \mathbb{R}^{1\times D_h}, \hat{\boldsymbol{y}} \in \mathbb{R}^{1\times D_y}, \boldsymbol{y} \in \mathbb{R}^{1\times D_y}$. The dimensions of the parameters are $\boldsymbol{W}_1 \in \mathbb{R}^{D_x\times D_h}, \boldsymbol{W}_2 \in \mathbb{R}^{D_h\times D_y}, \boldsymbol{b}_1 \in \mathbb{R}^{1\times D_h}, \boldsymbol{b}_2 \in \mathbb{R}^{1\times D_y}$.

(c) Implement the forward and backward passes for a neural network with one sigmoid hidden layer. Fill in your implementation in `q1c_neural.py`. Sanity check your implementation with `python q1c_neural.py`.

(d) GloVe (Global Vectors) embeddings are a type of word embeddings that represent words as vectors in a high-dimensional space, based on the co-occurrence statistics of words in a corpus. They are related to the skip-gram embeddings you saw in class in that they both aim to capture the semantic and syntactic relationships between words, but GloVe embeddings incorporate global corpus-level information in addition to local context information. In this section you will be using GloVe embeddings to represent the vocabulary.

Use the neural network to implement a bigram language model in `q1d_neural_lm.py`. Use GloVe embeddings to represent the vocabulary (`data/lm/vocab.embeddings.glove.txt`). Implement the `lm_wrapper` function, that is used by `sgd` to sample the gradient, and the `eval_neural_lm` function that is used for model evaluation. **Report the dev perplexity in your written solution**. Don't forget to save `saved_params_40000.npy` and include it in your submission zip!

2 Generating Shakespeare with a Char-level LM (15 pt)

In the previous section we dealt with word-level language models. But looking again at section 1, there is nothing that constraints us to using words as the basic elemnents in our model. The model we analyzed in section 1 could just as well be character-based - just replace "word" with "character", and you are good to go. In this section we will train a small character-based language model that will help us generate Shakespearean-like (emphasis on the like...) texts.

(a) Can you think of an advantage a character-based language model could have over a word-based language model? And what about the other way around: can you think of an advantage a word-based language model could have over a character-based language model? **Answer in 3-6 sentences**.

(b) Follow the instructions and complete the code in the attached notebook:

`q2_char_rnn_generation.ipynb`.

(c) Plotting the losses that were computed during training can provide a further indication that the network was indeed learning. Add your plot to the wirtten solution.

3 Perplexity (20 pt)

(a) Show that perplexity calculated using the natural logarithm $\ln(x)$ is equal to perplexity calculated using $\log_2(x)$. i.e:

$$2^{-\frac{1}{M}\sum_{i=1}^{M}\log_2 p(s_i|s_1,\ldots,s_{i-1})} = e^{-\frac{1}{M}\sum_{i=1}^{M}\ln p(s_i|s_1,\ldots,s_{i-1})}$$

(b) In this section you will be computing the perplexity of your previous trained models on two different passages - Wikipedia text and Shakespearean Text. You may and should add the code for doing so to the code you wrote in the previous sections - You can use the `saved_params` files and the `load_saved_params` modalities in Q1 (You don't need to train a new model). You can add cells to Q2's notebook as well.

All the data you need is in the assignment .zip. `shakespeare_for_perplexity.txt` contains a subset from the Shakespeare dataset, and `wikipedia_for_perplexity.txt` contains a certain passage from Wikipedia. For your convenience, both passages are also provided in a POS-tagged format, similar to the datasets used for training and testing in Section 1.

Please compute the perplexity of the bi-gram LM (from Section 1) and the character-level LM (from Section 2) on both these passages. Report your results in a table of this format:

Model-Data	Wikipedia Text	Shakespearean Text
Word-Level Neural Bi-gram (Section 1)	XX	XX
Character-level RNN (Section 2)	XX	XX

(c) Try to explain the results you have got. Particularly, why there might be large gaps in perplexity, while looking at different passages. **Answer in 3-6 sentences.**

4 Implementing a Transformer Model (20 pt)

In this question we will implement a full functioning transformer, using toy data and building the architecture from the building blocks. **Open the attached notebook (`q4_transformer.ipynb`) and fill in the missing code parts according to the instructions.**

After completing and running the notebook, answer these theoretical questions in your submitted pdf file:

(a) Observe the assertion of

$$assert\ d\_model\ \%\ num\_heads == 0$$

at the beginning of the '`__init__`' function in the Multiple Head Attention Class (Part 1 of the notebook) . Why is this assertion needed?

(b) As mentioned throughout the notebook, our implemented Transformer is autoregressive. What changes would be required to adapt it into a Masked Language Model (MLM)?

 (i) Which components should be added or removed from the architecture?

 (ii) How would the masking strategy change, both for attention (e.g., no-peek masking) and for token masking?

5 Sentiment Analysis (practical) (20 pt)

In this exercise, you will be asked to browse solutions for specific downstream tasks in NLP. The goal of this exercise is to help you become comfortable with using other researchers' code, public benchmarks, and common model repositories. We hope that this practical experience will aid you in your final project. We encourage you to continue diving deeper with critical thinking and creative ideas!

Hugging Face is currently the largest open platform for machine learning models, datasets, and research papers. You can browse state-of-the-art solutions according to tasks, examine benchmark datasets, compare models using different evaluation metrics, and often directly access fully functional code and model weights for use in your own projects.

In this exercise, you will explore the task of sentiment analysis on text. We will use the Hugging Face ecosystem, which now hosts model cards, datasets, and connections to research papers. You will look at the best-performing models available for sentiment analysis, verify whether their code and weights are publicly accessible, and practice loading and evaluating one of them in a Colab Notebook.

Include the code you used for this section in a Colab Notebook (create a new one yourself), and answer the questions in the PDF.

(a) In HF Papers page, search and download the article "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". It's OK if you don't fully understand the meaning of this article and its solution! You will learn more about it during the course. Find the link to their GitHub repository and copy it here. Once you've opened their GitHub, answer the following questions:

Which model have they made publicly available? Which dataset is used to benchmark sentiment analysis? How do you evaluate success in this task?

(b) In the article, T5 was pre-trained on multiple datasets and fine-tuned for specific downstream tasks. The original repository only published the weights for the pre-trained version (and not for the fine-tuned version used in the sentiment analysis task).

Search for a T5 model fine-tuned on the SST2 dataset on the HuggingFace platform, and use it as demonstrated in the Hugging face platform tutorial. LLMs as you may know, come in different sizes. Use the **T5-Small** pre-trained version.

Please state in your written solution exactly which model will you use and load.

(c) Now, use the model to predict the sentiment of following sentences (print each result in a different cell in the Colab Notebook):

- "This movie is awesome"
- "I didn't like the movie so much"

- ”I’m not sure what I think about this movie.”
- ”Did you like the movie?”

(d) Load the SST2 dataset and evaluate the accuracy of this model on it. Report the accuracy in the written solution.

(e) Is the data in the SST2 dataset balanced? Why is this an important question when evaluating the model’s performance with respect to this dataset? **Answer in 2-4 sentences.**

(f) Can you think on some properties of sentiment analysis evaluation that human-evaluators may notice but are not considered when evaluating the accuracy of this dataset?

(g) Imagine you are consulting with friends who work in a healthcare organization that has recently approved the use of an alternative supplier for one of their drugs. They want to monitor client satisfaction with both suppliers, ensuring they are equally satisfactory to patients. To achieve this, they have collected a dataset consisting of summaries from visits to a family doctor. They have confirmed that the dataset includes patients treated with each of the two drug suppliers. Now, they would like to classify the visit summaries as positive or negative to analyze the results. Since this dataset is quite unique, they seek your advice on how to approach this task.

Please suggest some general high-level approaches to solve the task. Explain to them which factors they should take into account and what the advantages and disadvantages are of the different approaches. **Write no longer than half a page.**

6 AI Disclosure & Reflection (5 pt)

The following question is graded only for its completion, not its content, nor for whether you used AI or not. Furthermore, its content does not impact the grading and evaluation of the rest of your submission. The goal is to promote transparency and reflection on LLM and AI usage.

Answer the following parts briefly (1-2 paragraphs):

a) Did you use AI in the work on this assignment? If yes, please name the specific tools/models used (e.g., ChatGPT—model name, Copilot, Claude, etc.).

b) How and why did you use it – which parts of the work and for what purpose (e.g., brainstorming, outlining, code generation, debugging, editing, evaluation).

c) Reflection: What helped or hindered? What is your takeaway from this experience?

If you did not use AI, write: “No AI used”, and add a paragraph explaining why.