In this lab, we'll work on a basic Neural Network for Language Modelling, which will predict the next word to come given a context of two previous words.

Model Architecture

The neural network used in this assignment is comprised by an embedding layer, 2 hidden linear layers and a final log-softmax layer which outputs the probability for each word being the "next word" given a context.

The embedding layer can be represented as a matrix where each row corresponds to a vector representation of a word on different dimensions. These vectors are initialized with random values and will be updated when training our network. Each word has an index which corresponds to its row in the embedding matrix (word a_n is represented by the n-th row). In our case there are 17 words, and we use 5 dimensions to represent them, so each embedding is a 1x5 vector, on a 17x5 matrix.

Then our first linear layer takes as input the embeddings for the 2 words in the context and joins them into 1 vector (each embedding has 5 values, which combined make 10). This layer uses a linear transformation of the form $f(x_1, \ldots, x_k) = b + a_1x_1 + \ldots + a_kx_k$.

These values are passed through a ReLU function of the form $f(x) = \max(0, x)$, which modifies the values of the vector for the next layer

The second linear layer takes these ReLU transformed values from the previous layer and does a new linear transformation, outputting a vector of size 17 (same as the vocabulary size)

The last layer uses a log softmax function to transform these 17 values into log probabilities for each word. The calculated probability is the likelihood of the word being the next word to come given a context.

The following table summarizes the layer architecture:

Type	Input	Output	Trainable Params
Embdding Layer	17	5	85
Linear 1 Layer	10	128	1280
ReLU activation	128	128	128
Linear 2 Layer	128	17	2176
Log-Softmax activation	17	17	17

Sanity Check

In order to do a sanity check on our language model, a prediction function takes as an input a context (two words) and predicts the probability of next word to come. The output is a tensor with the probabilities for each word in the vocabulary. So for the most likely word we just need to select the one with the highest log probability.

The model was trained with several combinations of epochs, embedding size and learning rate, and it got confused sometimes with too many epochs or too big embeddings, so the decision was to keep it smaller, given that the vocab is pretty small

In the end our model used a embedding size of 5, a learning rate of 0.02 and 100 epochs. This combination predicted succesfully the next word for each of the context in the sentence "The mathematician ran to the store".

The model predicts mathmatician instead of physicist because it learns that for the context (START, The) the most likely word is mathmatician, given that it has seen it 3 times in the dataset, compared to just 1 from physicist.

Fill the gap

Given the sentence "The _____ solved the open problem." we would like to know which word between "physicist" or "philosopher" our model thinks it is more likely to fill the gap.

In order to answer this question we can use the log probabilities our model returns for a determined context.

We will compute the probabilities for $p\left(philosopher \mid Start, The\right)$, $p\left(solved \mid The, philosopher\right)$, $p\left(the \mid philosopher, solved\right)$ and by adding them we will get the joint probability of the phrase "The philosopher solved the".

We repeat the same process with the pairs $p\left(physicist \mid Start, The\right)$, $p\left(solved \mid The, physicist\right)$, $p\left(the \mid physicist, solved\right)$ and get the joint probability for "The physicist solved the".

We did not consider the following part "open problem END" just because the probabilities of those context and words do not depend on our words to "fill", so they will be the same for both cases.

After trying different parameters for our network, the combination of an embedding size of 5, a learning rate of 0.02 and 100 epochs, our network correctly predicts that the word **physicist** is more likely to fill the gap than **philosopher**.

We checked this by calculating the cosine similarity between the physicist embedding and the mathematician embedding, and the philosopher and the mathematician embedding. As expected the cosine similarity tells us that physicist is more similar to mathematician than philosopher. This is learned by the model because in the training data the mathematician and the physicist ran to the store, so they have some similarity.

This result would not be possible to obtain with our bigram model in Lab2, because our training data does not contain the bigram **(philosopher, solved)** or **(physicist, solved**) thus even if we used smoothing, the model would not be able to decide between the two options (same probability).