TÉCNICO LISBOA

MEIC 23/24

Deep Learning

Homework 1

Both students contributed to answer the homework questions with the same commitment. The code development and theory questions have been produced by all members of the group.

Group 65

99286 Miguel Mano

99300 Pedro Rodrigues

Question 1

1. Breaking down the computation Z = SoftMax(Q K^T)V:

   -> Q, K, and V are matrices of size L×D.

   -> Computing Q K^T involves multiplying Q (L×D) by the transpose of K (D×L), resulting in a matrix of size L×L. This operation has a complexity of $O(L^2*D)$.

   -> The SoftMax operation is applied along the columns of the resulting matrix (L×L), which has a complexity of $O(L^2)$ since it involves exponentiation and normalization operations.

   -> The final multiplication between the SoftMax output (L×L) and V (L×D), results in a matrix of size L×D. This operation has a complexity of $O(L^2 * D)$.

   Therefore, the overall complexity is $O(L^2*D)$ and $O(L^2)$ in terms of L.

2. Given the first three terms of the McLaurin Series expansion, a suitable feature map (ϕ: $R^D \rightarrow R^M$) is $\phi(z) = [\ 1, z, z^2/\sqrt{2}]^T$ and the product of $\phi(q)^T\phi(K)$ is equals to $1 + (q^T)\ k + ½\ (q^T k)^2$.

   The dimensionality M of the feature space is 1x1 + 2D if you only use the first three terms, explained by the fact that the first entry of the matrix is the integer 1 and the last two are of dimension D, and if you use K terms M is 1x1 + D(K-1) because the first entry is the integer one and the rest of the entries are of dimension D excluding the first term of the expansion which is the reason why D is multiplied by K-1.

3. Based on the approximation of the last exercise, we approximate the SoftMax function to $softmax \sim \frac{\Phi(\mathrm{qi})^T \Phi(\mathrm{ki})}{\sum_{j=1}^{n} \Phi(\mathrm{qi})^T \Phi(\mathrm{kj})}$. We also know that $Diag\left(\sum_{j=1}^{n} \Phi(\mathrm{qi})^T \Phi(\mathrm{kj}) \ldots \sum_{j=1}^{n} \Phi(\mathrm{qn})^T \Phi(\mathrm{kj}))\right)$ is $Diag(\Phi(\mathrm{Q})\ \Phi(\mathrm{K})^{\mathrm{T}}\ 1\mathrm{L}) = D$. Therefore, $Z \sim D^{-1}\ \Phi(\mathrm{Q})\ \Phi(\mathrm{K})^{\mathrm{T}}\ V$.

4. We can exploit the above approximation by adding two linear projection matrices of size L*K when computing key and values helping us to project the original key and value layers (L*D) into a new projected layer (K*D). Using dot-product attention with the new layers we get another matrix (L*K) and we can

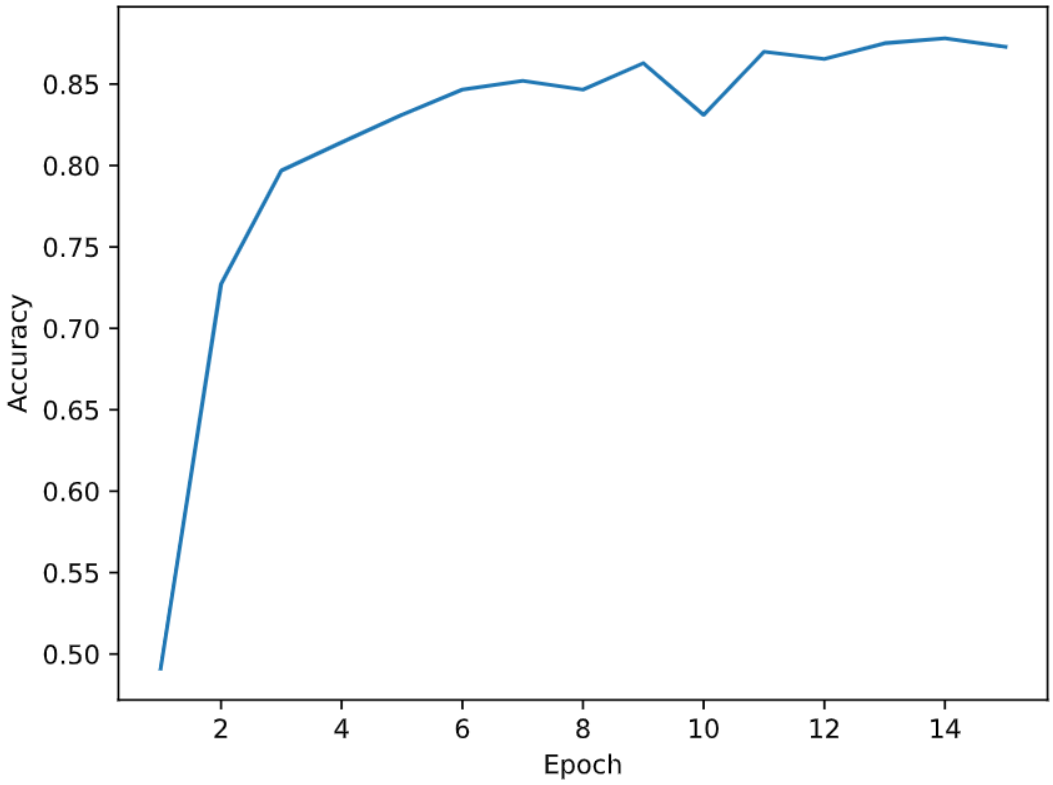
finally compute context embeddings in which operations only require O(L*K) time and space complexity. This means that the complexity no longer depends on M and L.

Question 2

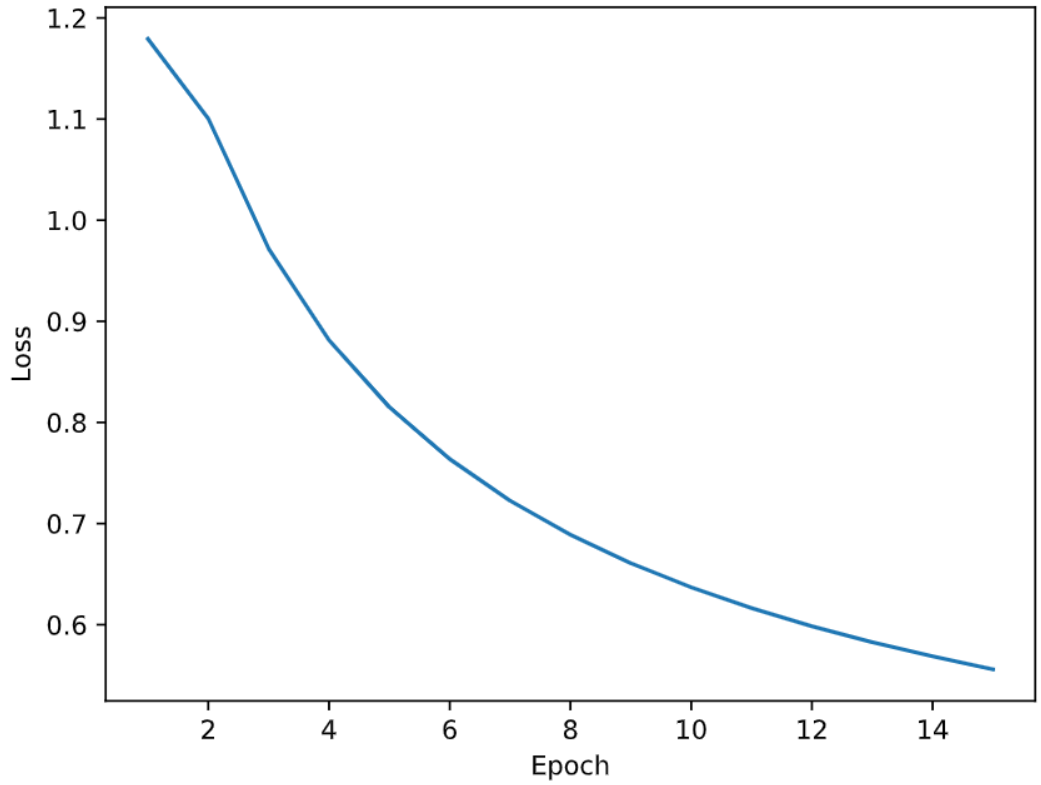
1. In terms of final validation accuracy, the model with best learning rate configuration for this task is **0.01**. – Final Accuracy Value is bigger.

Learning Rate of 0.01

Validation Accuracy



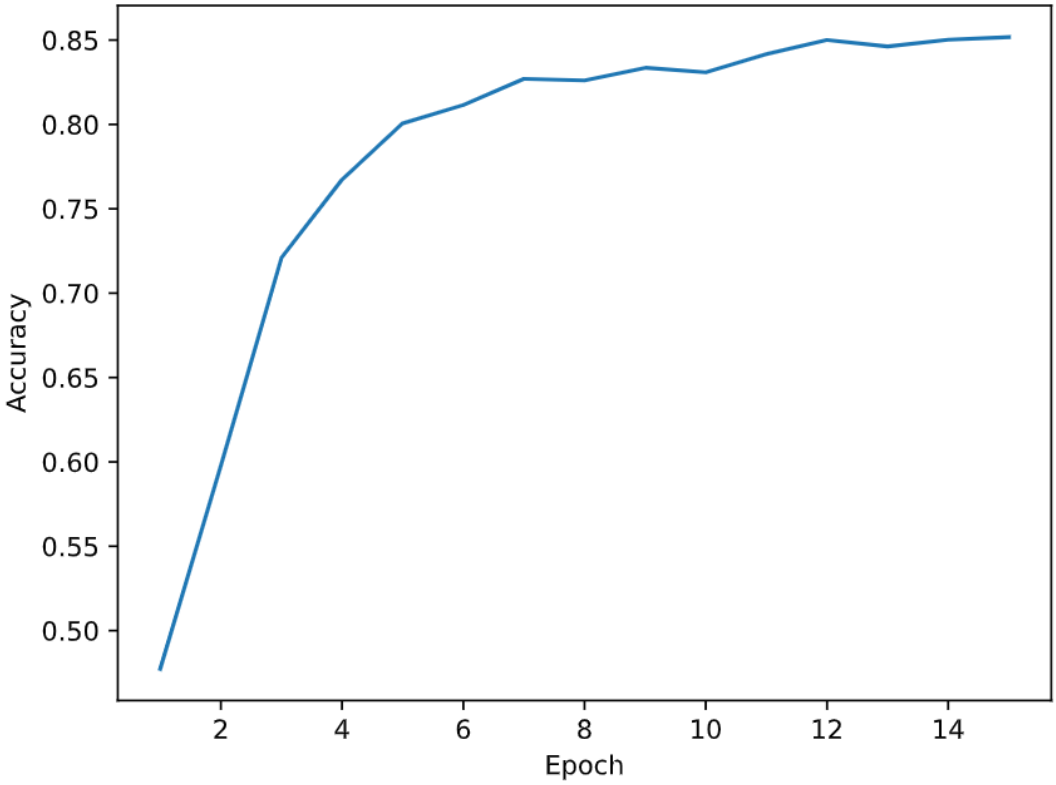
Training Loss



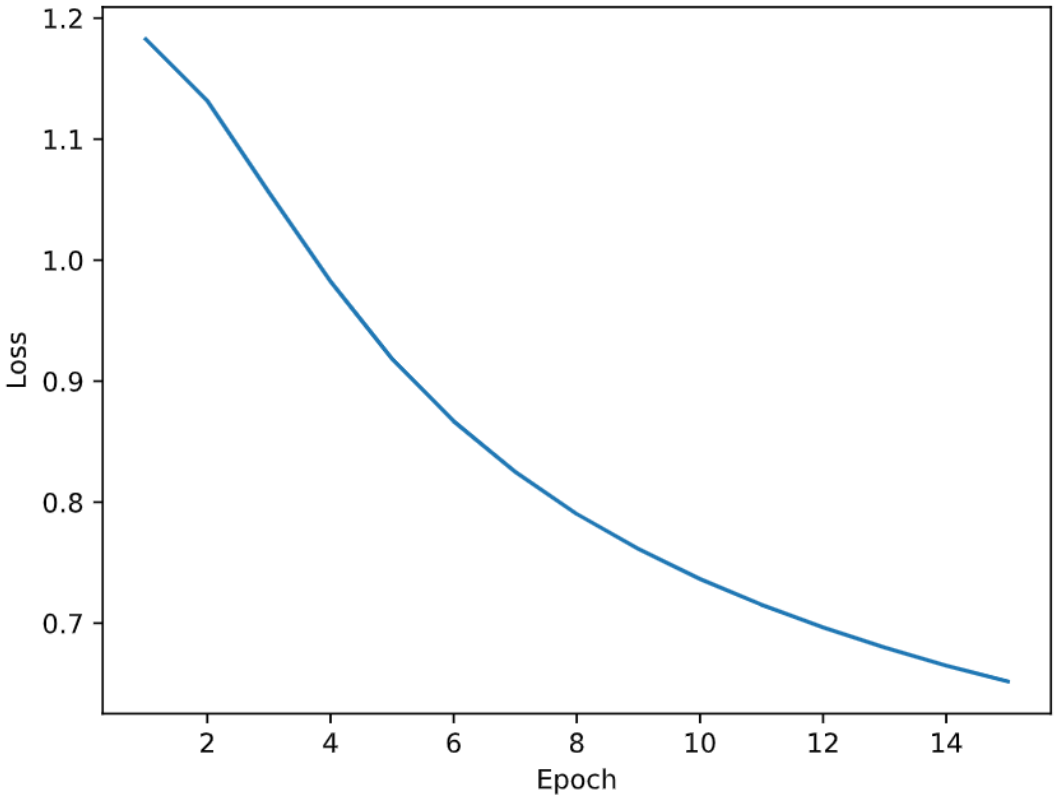
2.

Learning Rate of 0.01

Validation Accuracy



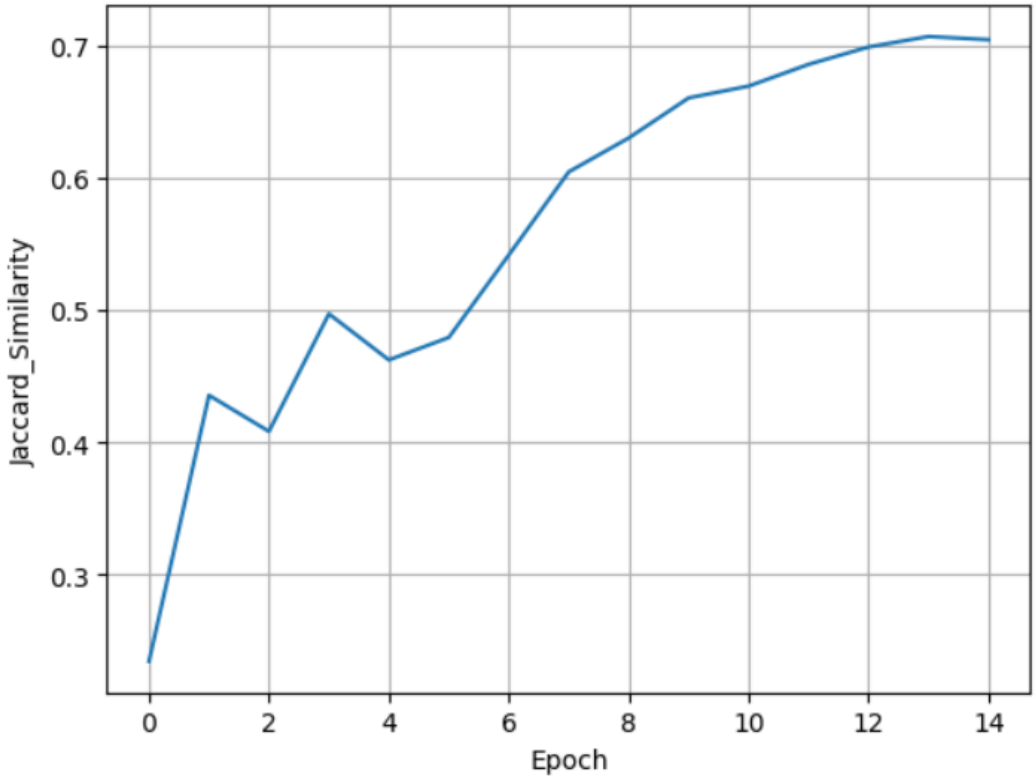
Training Loss

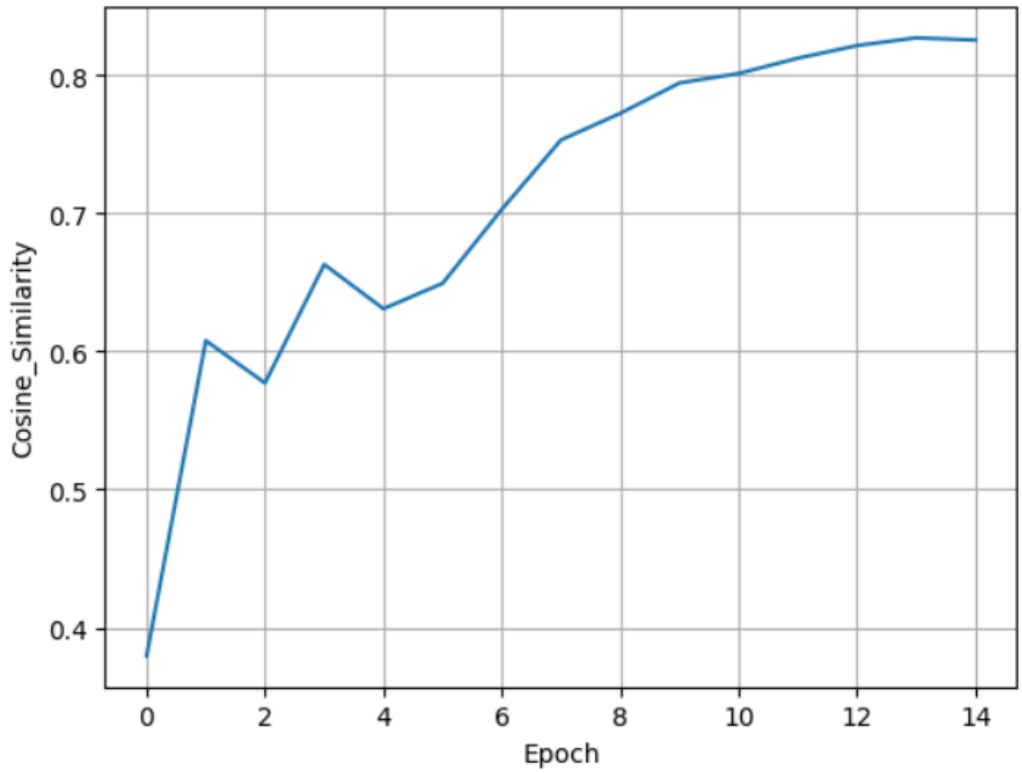


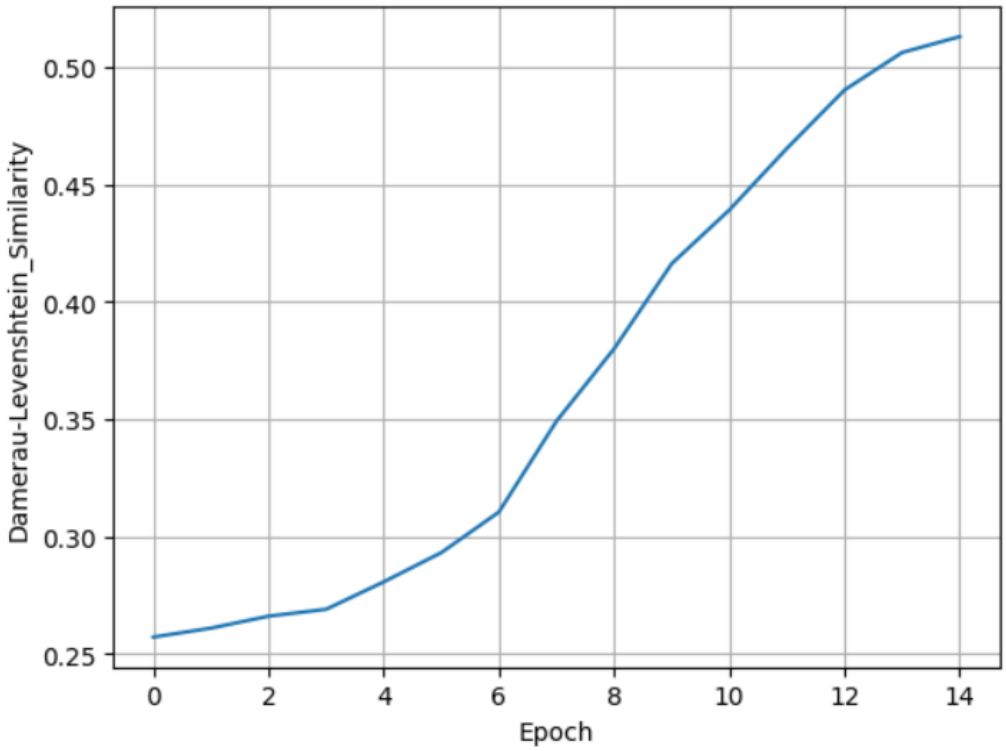
3. The number of trainable parameters is the same (225618) for both models, with and without maxpool variable activated. It indicates that the architectural differences (presence of max-pooling layers) might not be hardly influencing the network's capacity to learn features from the data. Max-pooling layers can contribute to translation invariance and reduce the spatial dimensions of the input, affecting the network's ability to capture hierarchical features. One important point to note is that maxpooling can contribute to making the neural network less sensitive to small variations in the input data and, to some extent, help avoid getting stuck in local minima (possible evaluation due to decrease on epoch 8/9/10).

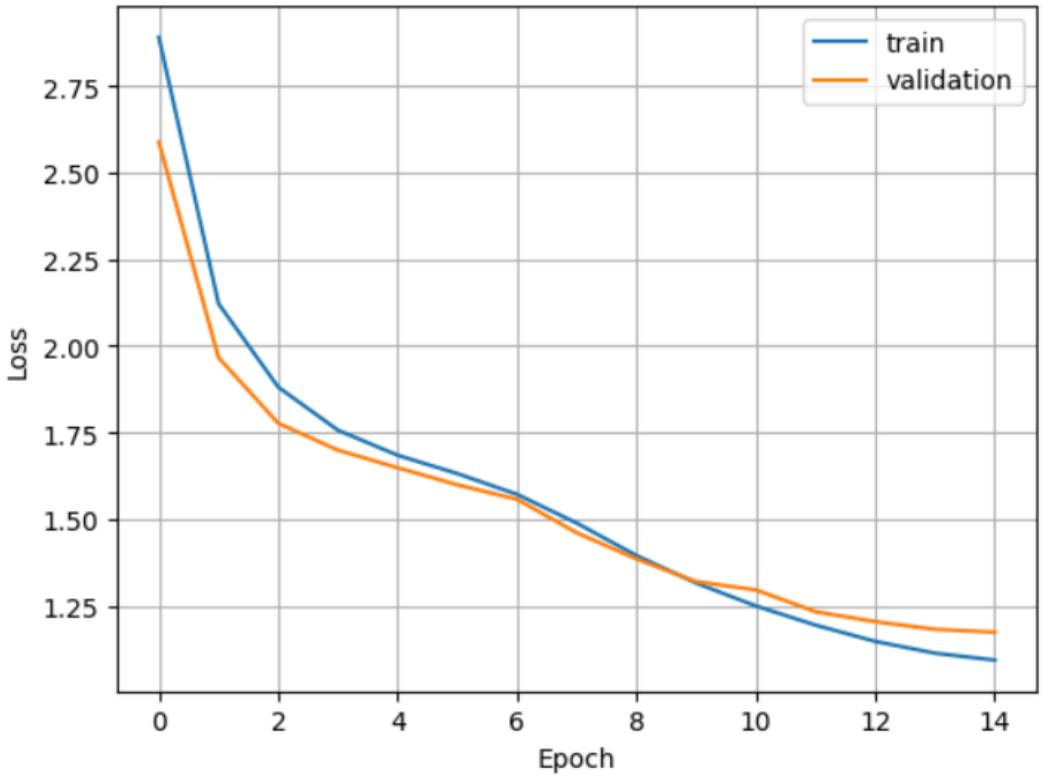
Question 3

1.


Jaccard_Similarity
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8
10
12
14
Epoch

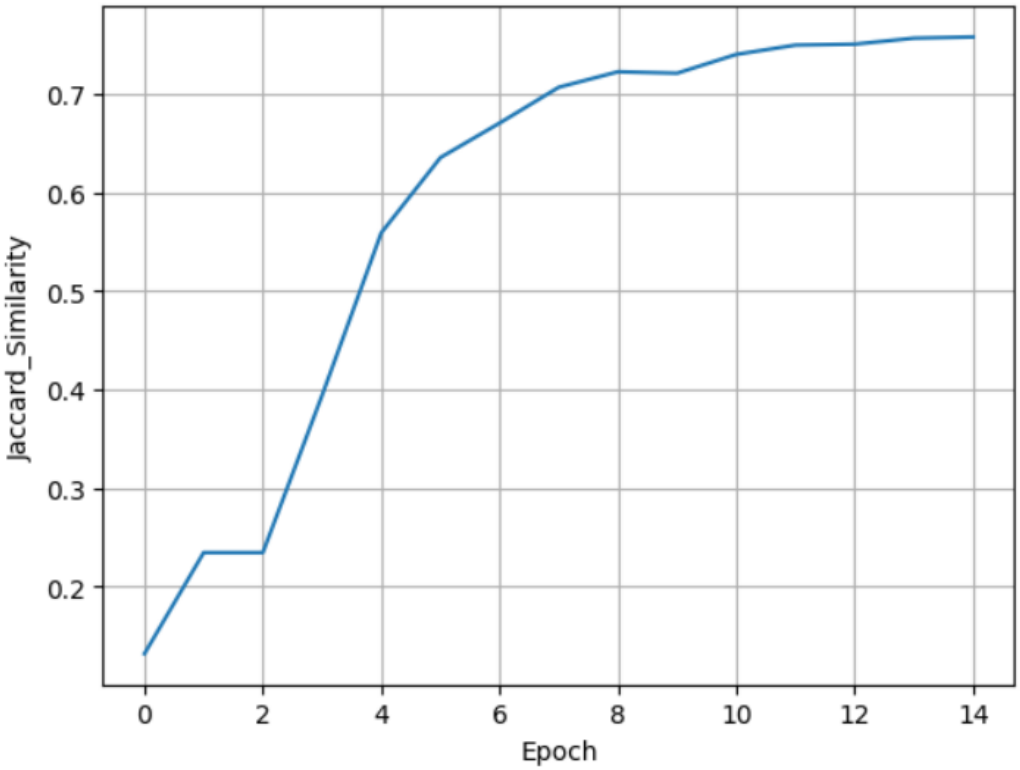

Cosine_Similarity
0.8
0.7
0.6
0.5
0.4
0
2
4
6
8
10
12
14
Epoch

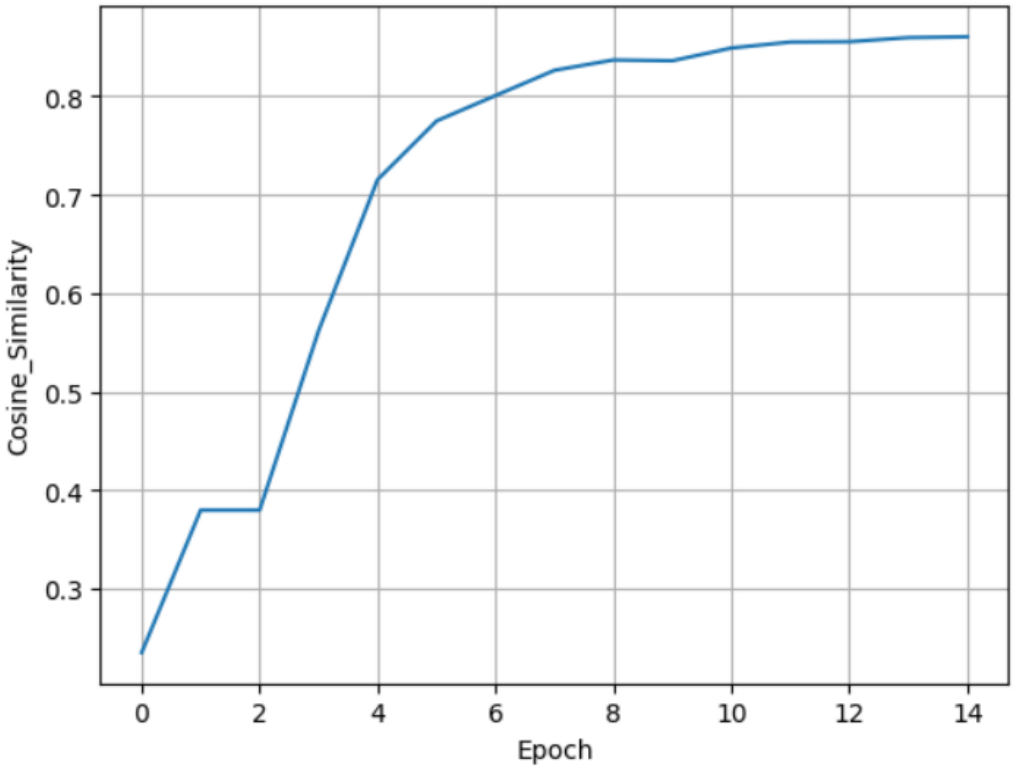


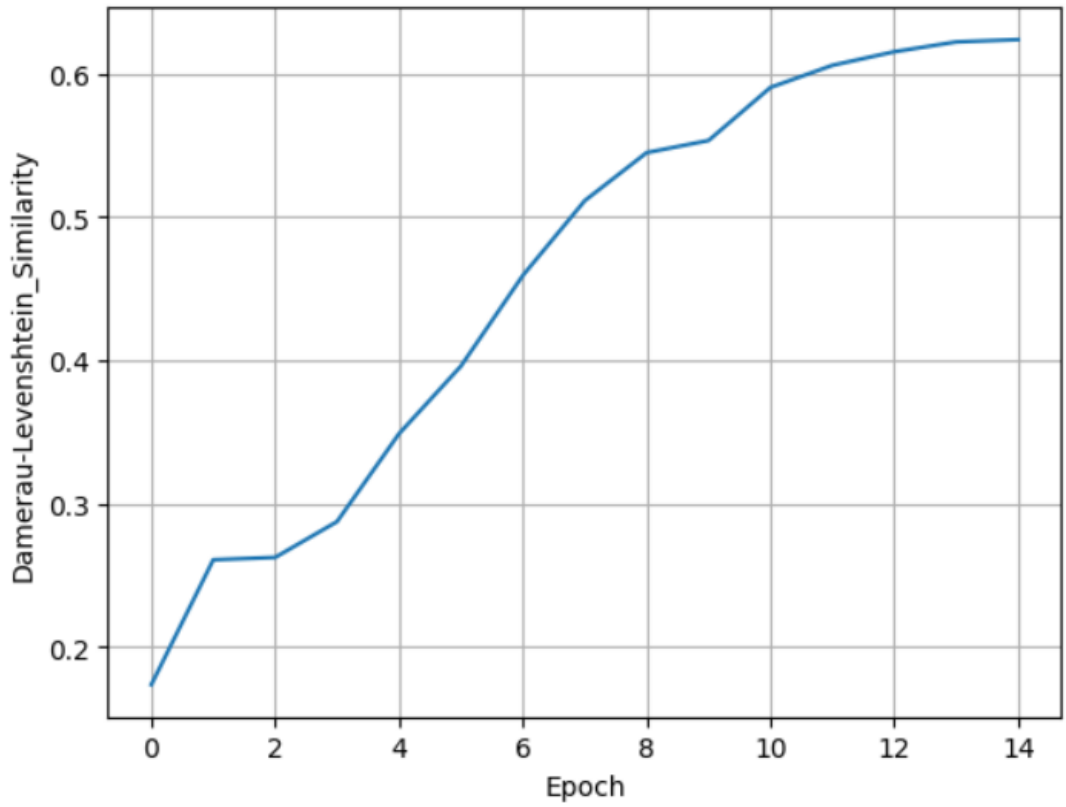


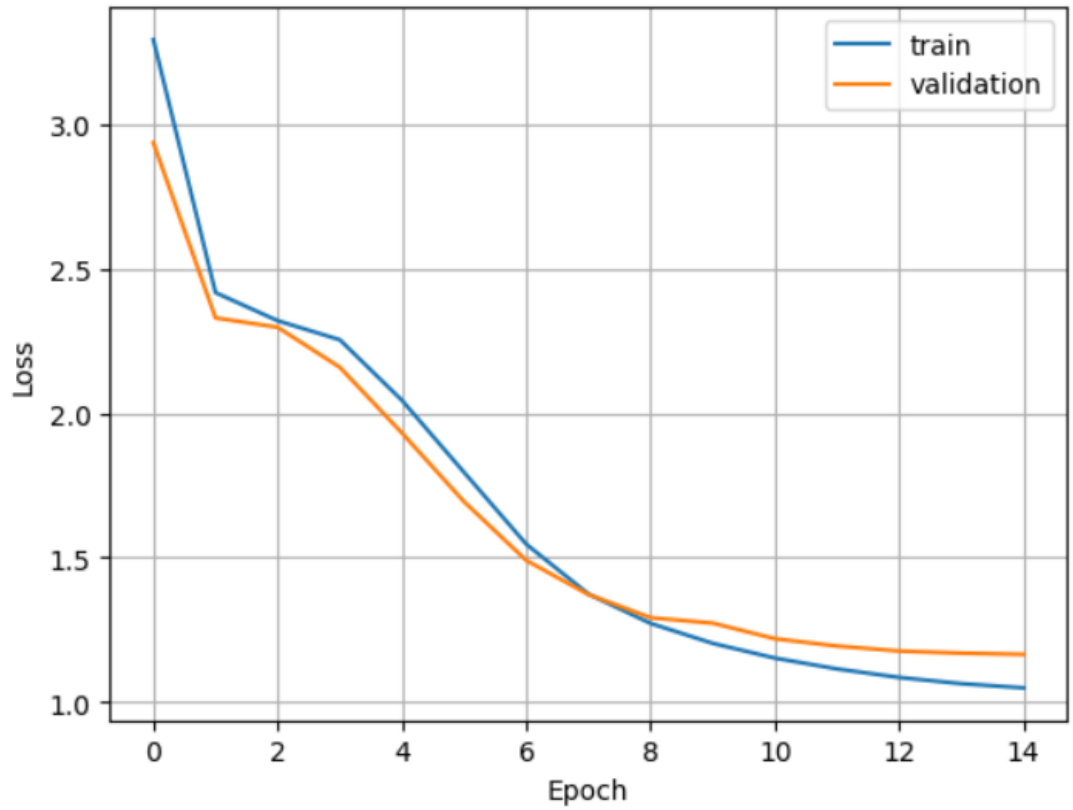
test:
{'jaccard_similarity': **0.7149009519408625**,
'cosine_similarity': **0.8323995590722496**,
'damerau-levenshtein_similarity': **0.5086990875939703**,
'loss': **1.1828287709050063**}

2.


Jaccard_Similarity
0.2
0.3
0.4
0.5
0.6
0.7
0
2
4
6
8
10
12
14
Epoch


Cosine_Similarity
0.3
0.4
0.5
0.6
0.7
0.8
0
2
4
6
8
10
12
14
Epoch

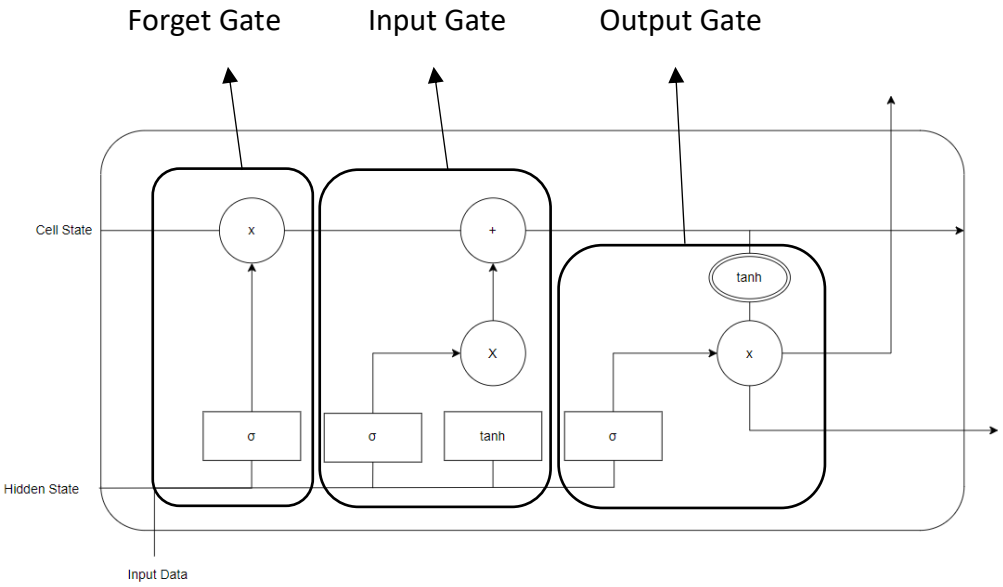




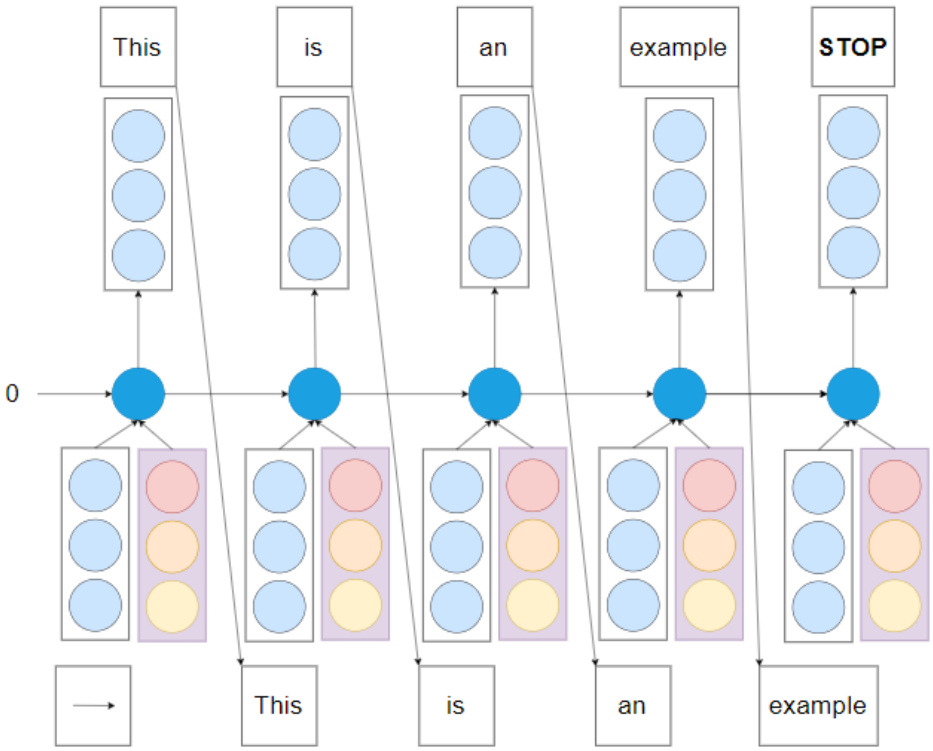
test:
{'jaccard_similarity': **0.7634892710593816**,
'cosine_similarity': **0.8645833813979843**,
'damerau-levenshtein_similarity': **0.6317842661080755**,
'loss': **1.1608208868561722**}

3.

LSTM



ATTENTION MECHANISM



The attention mechanism's dynamic focus on relevant information likely contributes to the observed improvements in test results compared to the LSTM in

Question 1. The attention mechanism's ability to capture context and relationships between input and output sequences might be key factors in the performance differences.

4. The Jaccard Similarity score measures the similarity between two sets by calculating the ratio of their shared elements to the total number of distinct elements. This metric is particularly useful when assessing the similarity between two sets of items without considering the order in which they appear. The score ranges from 0 to 1, with 0 indicating no common elements and 1 indicating identical sets. In the context of text comparison, the Jaccard Similarity is valuable for evaluating the presence or absence of words in two texts, providing insights into set-based relationships. Cosine Similarity, on the other hand, is a metric commonly used in natural language processing and information retrieval. It calculates the cosine of the angle between two vectors, treating each element in the vectors as dimensions in a high-dimensional space. Cosine Similarity is suitable for capturing the semantic similarity of text, as it focuses on the directionality of vectors. The score ranges from 0 to 1, with 1 indicating identical vectors and 0 indicating orthogonality. This metric excels in scenarios where the order of elements matters less than their overall semantic meaning. The last one, Damerau-Levenshtein Similarity assesses the similarity between two strings by measuring the minimum number of operations required to transform one string into the other. These operations include insertions, deletions, substitutions, and transpositions. Unlike the previous two metrics, Damerau-Levenshtein is not bounded within a fixed range. Higher values generally indicate greater similarity, as they represent a lower number of necessary operations for transformation. So, we can conclude that question 2, the attention mechanism, provide us better scores in every string similarity scores tested.