Large Language Model - Homework2

Zhe HUANG, Linghao ZENG

November 2023

1 Preliminary Questions

1.1 Loss

To compute the similarity between a word and a context, we use the similarity $\sigma(\boldsymbol{c} \cdot \boldsymbol{w})$. We want to minimize the loss. Thus:

(a) We should maximize $\sigma(\boldsymbol{c} \cdot \boldsymbol{w})$ for $c \in C^+$. A positive context implies that the word w and the word c should be close to each other, and by maximizing the Sigmoid of their dot product, we are bringing their vectors closer.

(b) We should minimize $\sigma(\boldsymbol{c} \cdot \boldsymbol{w})$ for $c \in C^-$. A negative context implies that the word w and the word c should be far from each other. By minimizing the Sigmoid of their dot product, we are pushing their vectors farther apart.

Geometrical interpretation:

The dot product measures similarity: positive if vectors point in the same direction, negative if opposite. The Sigmoid function maps the dot product to a range between 0 and 1. In the positive context ($c \in C^+$), we aim for a value close to 1, indicating similarity; in the negative context ($c \in C^-$), we aim for a value close to 0, indicating dissimilarity. Figure 1 provides a geometrical representation of the similarity measure $\sigma(\boldsymbol{c} \cdot \boldsymbol{w})$ in a 2D space.

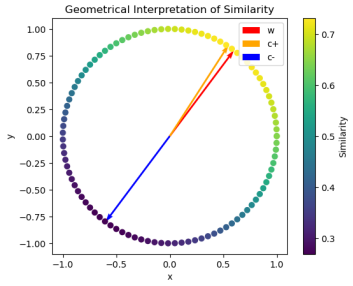


Figure 1: Geometrical interpretation of the similarity measure $\sigma(\boldsymbol{c} \cdot \boldsymbol{w})$.

1.2 Contrastive Learning

One of the first application to contrastive learning has been presented by Chopra, Hadsell, and LeCun [CHL05].

(a) Contrastive learning is a type of machine learning where we train a model to learn the differences between data points. It involves teaching the model to distinguish between positive pairs (similar data points) and negative pairs (dissimilar data points). In simple terms, contrastive learning helps the model understand what makes two things similar or dissimilar.

In page 3 of LeCun's article, we can find the expression:

$$L(W, (Y, X_1, X_2)^i) = (1 - Y) L_G(E_W(X_1, X_2)^i) + Y L_I(E_W(X_1, X_2)^i)$$

(b) In our setup, the analog of Y would be the binary label that indicates whether a pair of data points are similar (positive pair) or dissimilar (negative pair). It's a way to tell the model whether the data points in the pair should be considered a match (Y=1) or not (Y=0).

(c) The analog of E_W in our setup would be the similarity/embedding function that measures the similarity between two data points.

(d) The analogs of L_G and L_I would be the loss functions used to optimize the model during training. L_G corresponds to the loss function used for positive pairs (similar data points), and L_I corresponds to the loss function used for negative pairs (dissimilar data points).

2 Implementation Summary

2.1 Data Loading and Preprocessing

The IMDb dataset was loaded with 50,000 entries. A BERT tokenizer processed the reviews, and a subset of 5,000 shuffled samples was used. The tokenized data were split into 80% training and 20% validation sets. Next, we followed the implementations in the guidelines, where the negative contexts are randomly sampled from the whole vocabulary set.

2.2 Word2Vec

We generated word embeddings using Word2Vec, considering the following aspects:

- Model architecture: The model includes two embedding layers for encoding the target words and their context. A custom loss function was defined to calculate the error based on the difference between the predicted and actual context words.

- Training: We then trained the model with gradient descent, using the Adam optimizer over several epochs, by feeding forward word IDs and context IDs into the model.

- Validation: We validated the model on a separate dataset to assess its performance, ensuring it generalized well to new data without adjusting its parameters during this phase. The average loss from the validation provided a measure of the model's accuracy.

We had a Word2Vec model to generate word embeddings, then we saved the trained model parameters to a file for future use.

2.3 Classification Task

We loaded the checkpoint and transferred the embeddings directly to a convolutional model, which is the same as provided in the first lab, equipped with text feature extraction layers (embedding, convolutional, linear) and dropout. Then, we trained the text classification model using the Adam optimizer and compared the results with models without this initialization.

A more detailed ablation study is given in the next section.

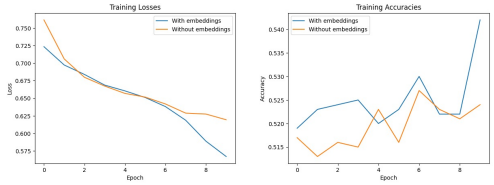
3 Results and Analysis with Ablation Study

The use of Word2Vec embeddings significantly influences the classification performance. We compare the results of the classifier with and without the pre-trained Word2Vec embeddings.

As shown in Figure 2, the classifier with Word2Vec embeddings outperforms the one without these embeddings, indicating the effectiveness of transfer learning in natural language processing tasks. The data suggests that the model using word2vec embeddings for training starts with a higher initial loss but improves more rapidly and achieves better accuracy on both the training and validation sets compared to the model without word2vec embeddings. This indicates that pre-trained word embeddings can provide a significant advantage in learning, likely due to the embeddings capturing useful semantic information from a large corpus that aids in the model's generalization.

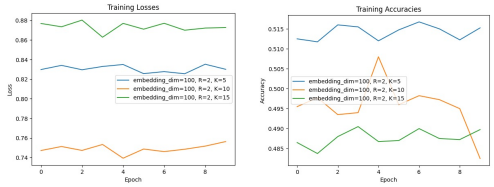
We further investigate the effect of various hyperparameters, such as the radius, negative sample ratio K, and embedding size, on the performance of our classifier.

As shown in Figure 3, The radius parameter in our convolutional layers affects the context window size for feature extraction. A larger radius R provides a wider context but may also introduce noise; The negative sample ratio K determines the level of feature map reduction after convolution operations. An optimal value of K is crucial for balancing model complexity and feature expressiveness; The dimensionality of the embedding vectors is another critical hyperparameter. We observe that the higher embedding size of a model is, the faster its training loss decreases and the higher its accuracy becomes.

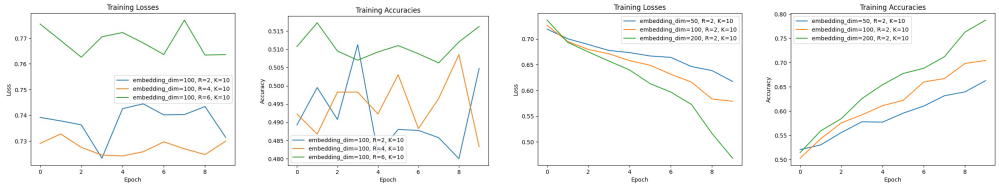


(a) Training losses (b) Training accuracies

Figure 2: Training losses and accuracies with and without Word2Vec embeddings



(a) Training losses of different K (b) Training accuracies of different K



(c) Training losses of different R (d) Training accuracies of different R (e) Training losses of different embedding dimension (f) Training accuracies of different embedding dimension

Figure 3: Hyperparameters impact study results

4 Conclusion of What We've Learned

What we've learned through this homework can be summarized as below:

- Reviewed essential NLP preprocessing steps like tokenization and data cleaning;
- Engaged in the implementation of the Word2Vec model, enhancing the understanding of the similarities calculation;
- Conducted ablation studies on different hyperparameters, presenting the impact of different Word2Vec models on the classification task;
- Understood the geometrical interpretation of the loss function used in the training of Word2Vec model;
- Read the Contrastive Learning paper and understood the idea of more advanced loss function design in unsupervised learning.

References

[CHL05] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005.