HW Word2vec

BOUZKKA Ismail , FADIL Anas , DAHMANI Salah

Introduction

This report outlines the implementation of the Word2Vec model from scratch. The Word2Vec model is a contrastive representation learning model that learns word embeddings. The implementation is based on the formulation provided in the homework guidelines.

Within this report, you will find answers to the questions asked in Section 2, some results obtained with plots, and a final reflective section discussing some possible considerations.

Answers to Preliminary Questions

Question 1: To compute the similarity between a word and a context, we use the similarity $\sigma(c \cdot w)$. Since we want to minimize the loss (1):

(a) For $c \in C^+$, should we maximize or minimize $\sigma(c \cdot w)$?

Answer: For words c in the positive context C^+, we want to **maximize** $\sigma(c \cdot w)$ that will maximize the probability that c and w appear together in the same context. In fact the first term of the loss function, $-1_{c \in C^+} \log(\sigma(c \cdot w))$, is minimized when $\sigma(c \cdot w)$ is maximized.

(b) Same question for $c \in C^-$.

Answer: Here, we should **minimize** $\sigma(c \cdot w)$ that will minimize the probability that c and w are contextually related, which increases $-\log(1-\sigma(c \cdot w))$, thus reducing the loss for negative examples (the function $-\log(1-\sigma(c \cdot w))$ is an increasing function of $\sigma(c \cdot w)$ so minimizing it minimizes the loss).

Geometrical Interpretation: Geometrically, we aim to maximize the similarity (reduce the angle or increase the dot product) between a word and its positive context. Conversely, we aim to minimize the similarity between a word and its negative context. $\sigma(c \cdot w)$ being high means the angle between vectors c and w is small, indicating that they are more aligned. For negative examples, minimizing $\sigma(c \cdot w)$ increases the angle between c and w, reflecting their dissimilarity.

Question 2: Reference to the article by Chopra, Hadsell, and LeCun on contrastive learning:

(a) Explain very simply what is contrastive learning.

Answer: Contrastive learning is a type of representation learning where the goal is to distinguish between similar and dissimilar examples. The objective is to learn mappings from raw inputs to a vector space where similar examples are closer, and dissimilar examples are farther apart.

(b) What is the analog of Y in our setup?

Answer: In the Word2Vec context, Y would analogously represent whether a context word c is from a positive (C^+) or negative (C^-) context of the word w. $Y = 1$ for positive contexts and $Y = 0$ for negative contexts.

(c) What is the analog of E_W?

Answer: The analog of E_W is the similarity score between a word and its context, which is computed as the dot product between the word's embedding vector and the context's embedding vector, followed by the application of the sigmoid function. The reason why E_W is not the same as the similarity score in Word2Vec is that Word2Vec uses a different approach to computing similarity, which involves two separate embedding tables for words and contexts.

(d) What are the analogs of L_G and L_I?

Answer: The analogs of L_G and L_I are the loss functions for positive and negative examples, respectively. In Word2Vec, the loss function for positive examples is given by the first term of the loss function in equation (1), which is the negative log likelihood of the similarity score between a word and its positive context. The loss function for negative examples is given by the second term of the loss function in equation (1), which is the negative log likelihood of one minus the similarity score between a word and its negative context.

Implementation

In the *extract_words_contexts* function, borders are managed through padding. the function pads the input list of IDs (ids_list) with zeros. The padding is applied at both the beginning and the end of the list (radius zeros are added to the start and to the end of the list). By padding the start of the list with zeros, we fill non-existent preceding slots with zeros, thereby maintaining the uniform size of the context window while acknowledging the absence of actual preceding words.Same for the end of the list padding

In the training step, we computed the loss separately for positive and negative samples and used *binary_cross_entropy* with the appropriate labels for positive and negative examples (= contrastive nature of the Word2Vec). We handled the different context IDs for positive and negative samples with Looping through context IDs within a batch and calculating losses individually. We evaluated the average loss and accuracy of the model on the validation set, the plot below describe the evolution of those metrics :

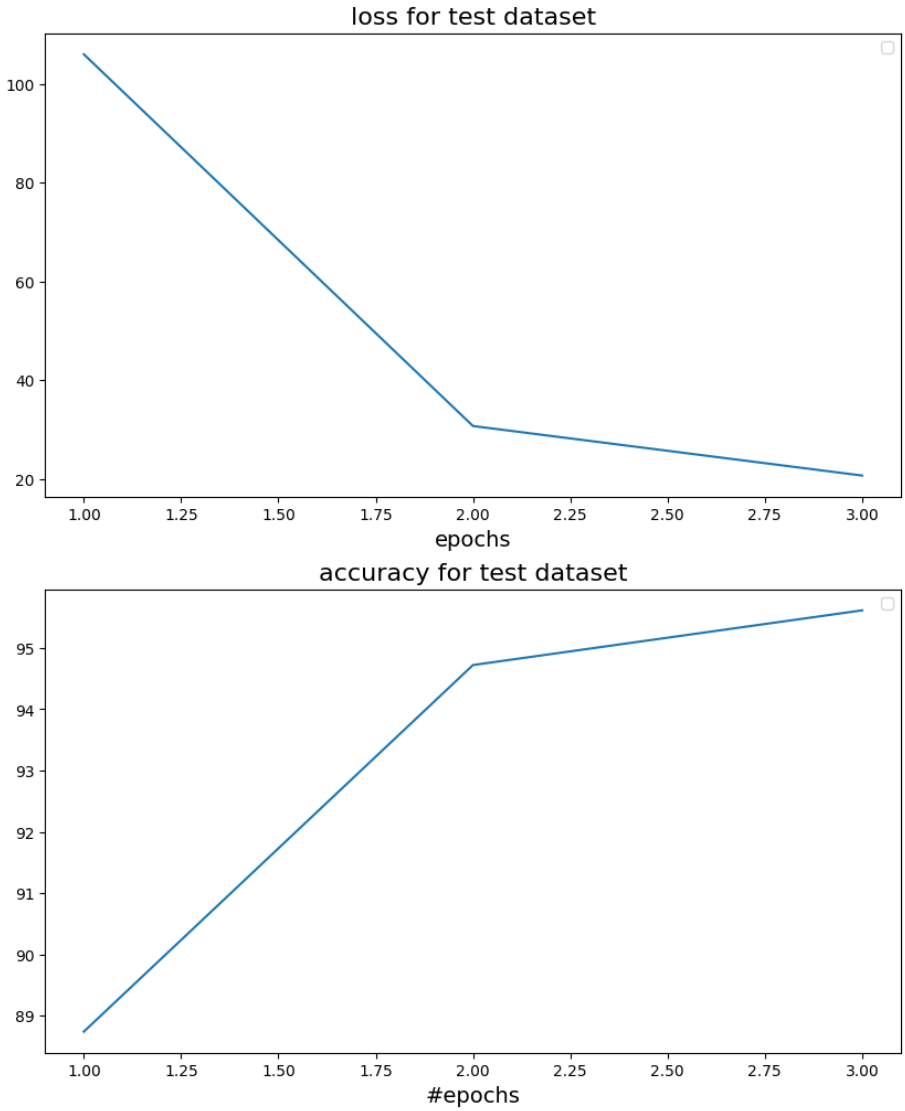

Figure 1: Loss and accuracy evolution on validation set

Possible considerations

In our Word2Vec training implementation, we iterate through each context vector individually (via pos_context_ids[:, i] and neg_context_ids[:, i]), which slows down training. We can consider modifying the forward method so it can handle batches of context vectors, computing dot products for entire batches at once, which reduces the need to loop over individual context IDs. For the Loss Calculation, instead of calculating the loss for each context word pair individually, we might try computing the losses for all positive and negative in a single batch which can make the training more fast.

Conclusion

In this homework, we implemented the Word2Vec model from scratch using PyTorch. We also applied the model to a classification task. We learned the importance of contrastive learning in representation learning and the geometric interpretation of the Word2Vec model. Overall, this homework helped us gain a deeper understanding of word embeddings and their applications in natural language processing tasks.