NLP - Ex 4

Guy Kornblit, 308224948; Mohammed Salama, 318983384

Here we compare different models for sentiment analysis problem, analyzing results over sub-phrases in sentences within the Sentiment Treebank dataset by Stanford. First, we examine each model by its learning process, then we can compare and analyze advantages and disadvantages of each model.

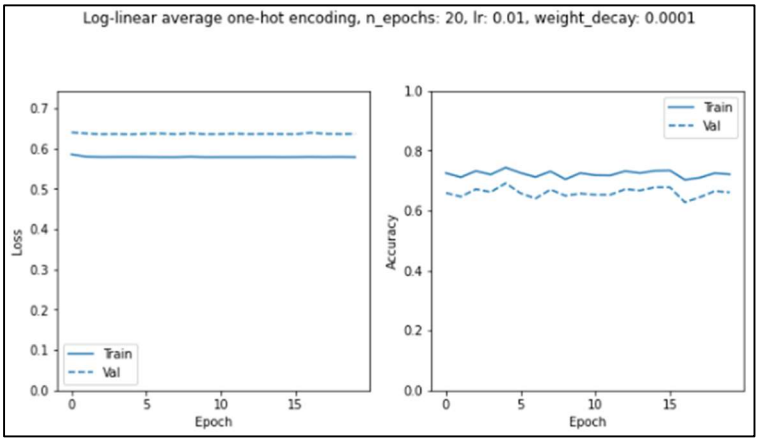
Implementation notes

- We measured the loss with "binary cross entropy with logits" (without activating sigmoid after the affine transformation), and the binary accuracy on the final prediction (after sigmoid and round).
- Loss for every epoch is calculated over the entire data, using evaluate. Although it causes computational overhead, using GPU we were able to measure more accurate results.

Log-Linear model

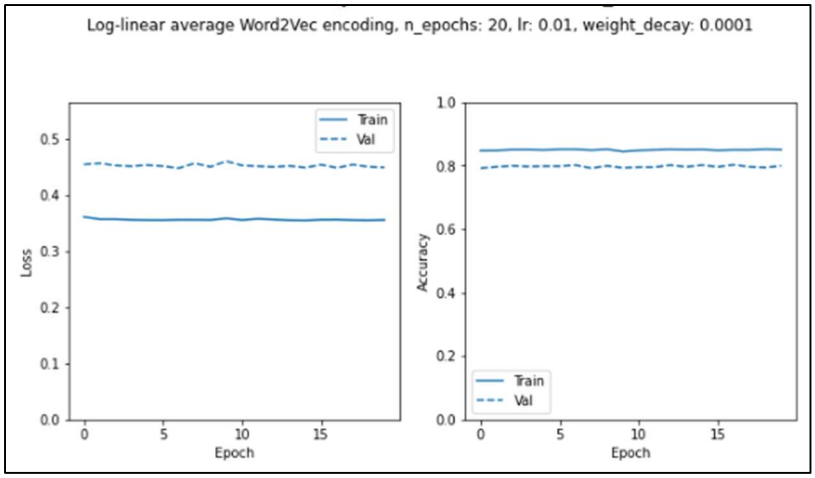
This model uses a single affine transformation and a sigmoid activation to get a probability score in [0,1]. Predictions are made using rounding that probability to the nearest integer. We have tested this model using two different embedding techniques.

Average one-hot encoding - we embed each sentence a single vector of dimension $|V| \approx 16k$, where the coordinates are an average of the one-hot embedded words in the sentence. We have set the hyperparameters of the model as shown in the training figure and trained the model for 20 epochs. We can observe that the model failed to learn and improve significantly, and quickly converges on a local minimum (loss of ~0.58). Correspondingly, the accuracy of the model stays rather fixed on ~63%.



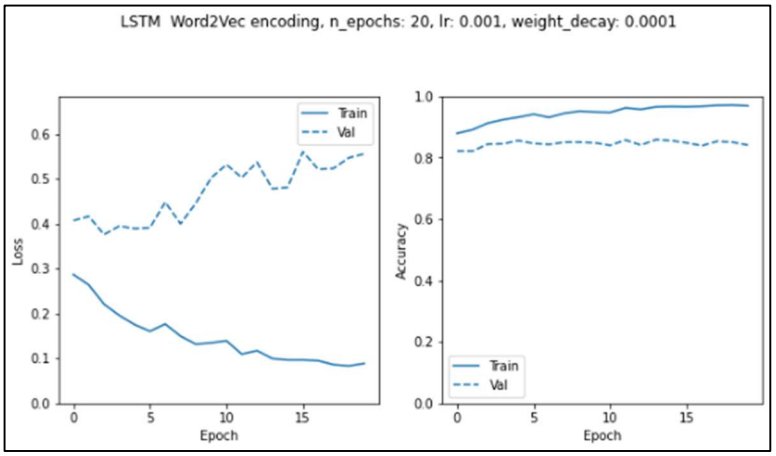
Average word2vec embedding - every sentence embedding is the average of word2vec encodings of the words that are contained in it. We average only on known words to the encoding. Since the word2vec embedding uses 300 dimensions, each sentence embedding is of that dimension as well. After training for 20 epochs, it seems that the model again reached a local minimum on loss values of ~0.35, and validation accuracy of ~0.8 which is much better than the average-one-hot embedding version.

In both models, we suggest that the **quick convergence caused by the simplicity of the net**, that restrict its expressiveness. As for the difference between the embedding methods, its clear that the **w2v-based encoding was able to capture more complicated aspects of the text** (with fewer dimensions), which explains the better results.



Bidirectional LSTM model

This model uses bidirectional LSTM that receives as input the Word2Vec embedding of a word in the input sentence. When the number of layers in each direction is greater than 1, we use dropout with probability of 0.5 on every hidden state that passes to the next layer. Lastly, we concatenate the hidden states from both directions, apply FC layer to map to a single neuron, and apply sigmoid activation and rounding to get the final prediction. We trained the model with n_layers=1, with learning rate of 0.001 for 20 epochs. Additionally, we normalized (padding or truncation) each sentence by a fixed number of words (52), so it would correlate in batch learning. We can see that the model overfits the data significantly in terms of loss, but not enough to cause major generalization error, as seen by the accuracy graph.



Model comparison

We examine the model by their performance on the entire loss, and on two subsets:

a. Negated polarity - sentences that have subphrase with opposite sentiment.
b. Rare words - how does our model handles sentences with unknown words?

Results:

Model+ Hyperparameters	Loss	Accuracy	Negated Polarity Accuracy	Rare Words Accuracy
Log-Linear+ Avg-one-hot + optimizer=Adam +lr=0.01 + decay=0.001 + n_epochs=20	0.628	0.66	0.564	0.4
Log-Linear+ Average-w2v + optimizer=Adam +lr=0.01 + decay=0.001 + n_epochs=20	0.413	0.825	0.58	0.78
Bi-LSTM + Seq-W2V + Optimizer=Adam+ lr=0.001+ decay=0.0001+ n_layers=1+ n_epochs=20	0.542	0.836	0.693	0.72

Analysis-

1. As said before, we see that the log linear models behave similarly for both embedding, but clearly Word2Vec achieves better results in all scores, since it encodes semantic properties of the sentence which correspond to the task, where average-one-hot encodes only vocabulary so its semantical encoding is lacking.
2. Regarding Bi-LSTM, the results are much better than the log-linear model (one-hot) and slightly better than the word2vec variation. To our understanding, this model can encapsulate past words in the sentence, and can find relations between different words of the input sentence (negation etc.). This ability is well suited to sentiment analysis task.

   We can think of averaging word2vec as finding a new vector that "points" to a subspace that correlates with a semantic property of the entire sentence. So, LSTM basically use this idea, but also tweaks the final vector to better fit the wanted property (sentiment). The small gap from the log-linear avg-w2v can be the result of a not complex enough net.
3. Regarding the negated-polarity subset, we see that LSTM has the highest accuracy and Log-linear with avg-one-hot has the lowest accuracy. We suggest that negation is a construct that is better captured by a sequential method.
4. Regarding the rare-words subset, we can see that LSTM and log-linear with avg-word2vec has similarly high result, and log-linear avg-one-hot has the lowest accuracy. In average-one-hot embedding, rare words correspond to entries of the embedding that was not learned for the entire dataset (its weight is not significant), that is why we think that rare words didn't have much effect in the result of the log-linear with one-hot embedding. As for the word2vec embedding, rare words are similar to words that have the same context, so this embedding treats rare words better.