Language Models

Advanced NLP : Assignment 1

Varun Edachali (2022101029)

December 11, 2024

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

1 Common

1.1 Data Processing

The data is pushed to lowercase and stripped. Punctuation and other special characters (such as quotes, hyphens) are replaced with spaces. Word and sentence tokenisation is done using nltk.

1.2 Dataset

The *ModelDataset* contains the corpus, vocabulary (as a dict from word to index in embeddings attribute) and embeddings as attributes. The *__getitem__* function returns the context (stacked embeddings), the index of the target (used to compare using *NLLLoss* as this index is expected to be '1' or max probability in the output) as well as the sentence itslf (context + target as a string). This sentence is used to build the output (sentence and matching perplexity) as it may get shuffled and lost during batch execution.

A static collate function is used to pad the sequences of variable length in RNNs and Transformer-Decoders.

Note: FastText embeddings were used as provided by the *torchtext* module. Unknown words (outside the train set) were treated using an "unknown" embedding, here "$< UNK >$".

1.3 train, evaluate, calculate_nll

The implementation is mostly trivial. It is consciously general across models. Note that in the final calculation of *negative log loss* per sentence we do not perform any reduction because we want the loss corresponding to every sentence.

Model Selection was done by training the model over a fixed number of epochs (usually, 10) and selecting the model with the least validation loss across these epochs.

Note that perplexities are calculated as mentioned here:

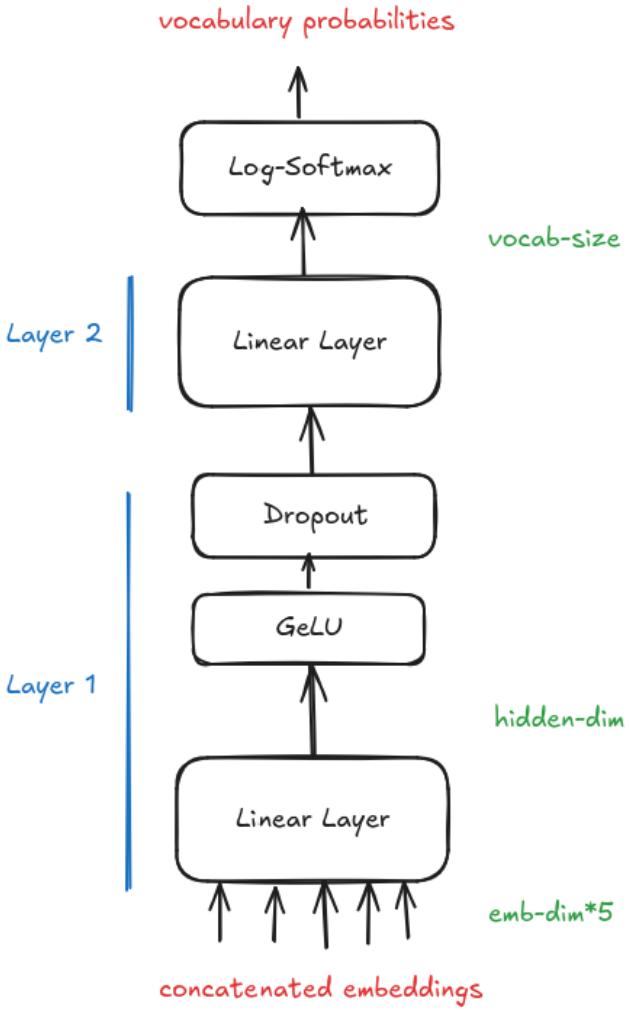
$$PPL(X) = exp\{-\frac{1}{t}\sum_{i}^{t} logp_\theta(x_i|x_{<i}\}$$

1.4 Evaluation

The master function that trains the model, evaluates it, and saves the perplexities is also highly generic.

2 Neural Network Language Models

2.1 Model



The *embedding dimension* is usually 300 (from FastText) and the *hidden state* was also 300 by default. The *GeLU* introduces some non-linearity and the *dropout* layer introduces some non-linearity. The *log-softmax* is used instead of softmax due to similar computational results at a faster rate than regular *softmax*.

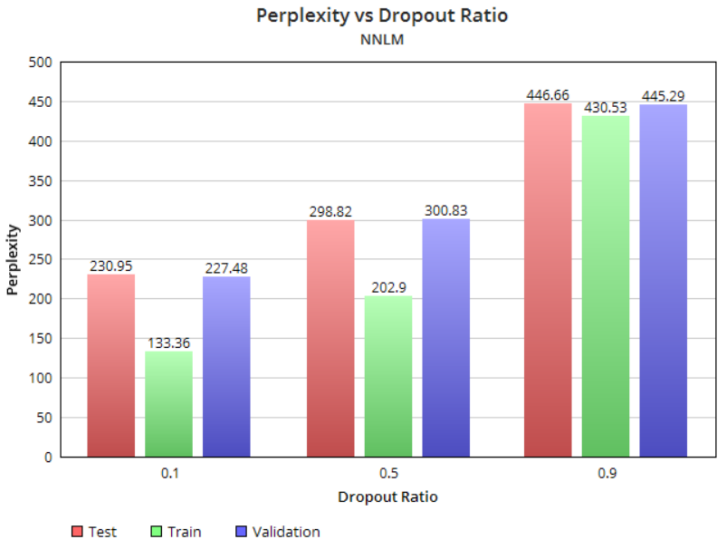
The default optimiser used in most calculations was *Adam*, and the default loss function was *NLLLoss*, or negative-log-likelihood, with a reduction of "sum". This is to ensure we correctly calculate the average, summing over all data-points and then

dividing by the number of them present.

2.2 Analysis

2.3 Hyper-Parameters

2.3.1 Dropout Ratio

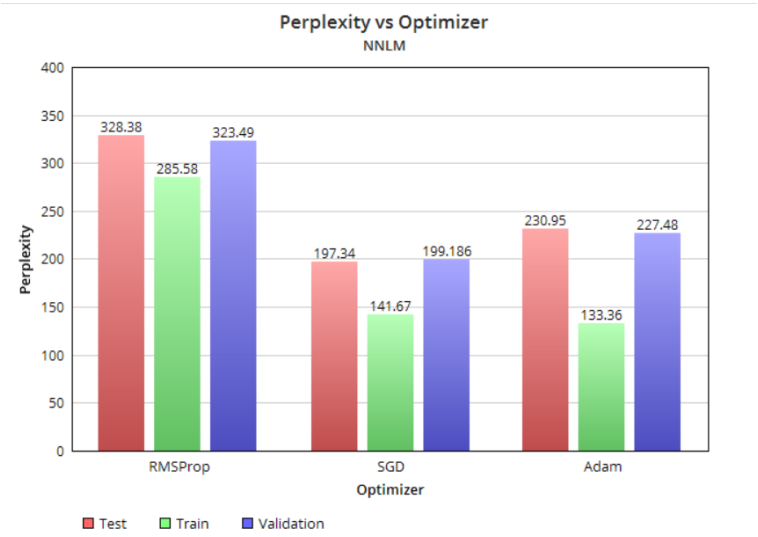


Three dropout ratios were tested - one small, one medium and one large.

- A dropout ratio of just 0.1 could provide excellent, accurate results. There is some over-fitting, however, with the train perplexity being far less than the test and validation perplexities.
- Increasing the dropout ratio to something like 0.5 yields no discernible advantage. It is worse than 0.1 in all three sets while not preventing overfitting.
- A higher dropout ratio reduces over-fitting on the training data, as is expected as 90% of the neurons are effectively not considered in a step. But, this is at the consequence of a much higher perplexity value for all three sets.

Overall, we see that a lower dropout ratio yields better results in this case, unless we want to stringently prevent over-fitting.

2.3.2 Optimizer

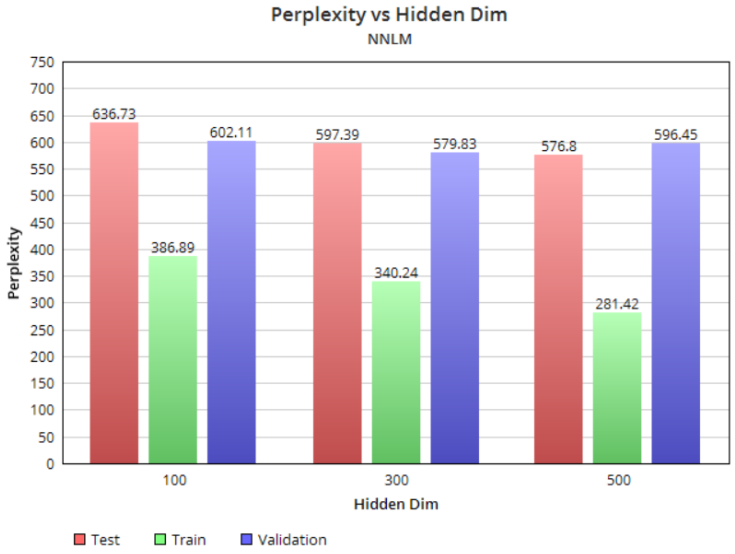


All three optimizers were tested with a *learning rate* of $1e-3$ and a *weight decay* of $1e-5$.

- **SGD** or Stochastic Gradient Descent provided the best results by far. After 10 epochs, both the train and validation losses were still decreasing too, so it is likely that this could yield even better results with more time.
- **Adam** provided mediocre results. The optimizer was decreasing the training loss throughout, but the validation loss began increasing after a few epochs.
- **RMSProp** or Root Mean Squared Propogation was the worst among these choices. Both the train and validation losses increased almost throughout the epochs, yielding unfavourable results.

An added bonus of SGD was that it also seemed to yield the fastest pipeline, although this could be because of external factors. Thus, SGD is the best optimizer, but Adam was also sufficient.

2.3.3 Hidden Dimension



All three were tested with 1000 sentences due to computational constraint (I ran out of memory to the point that images were being saved as text files by my system lol).

- A lower dimension favours less over-fitting.
- A higher dimension favours better overall perplexity, with more robust representations or features. But, this is also prone to overfitting.

Thus, 300 is an accurate balance between the two.

2.4 Perplexity

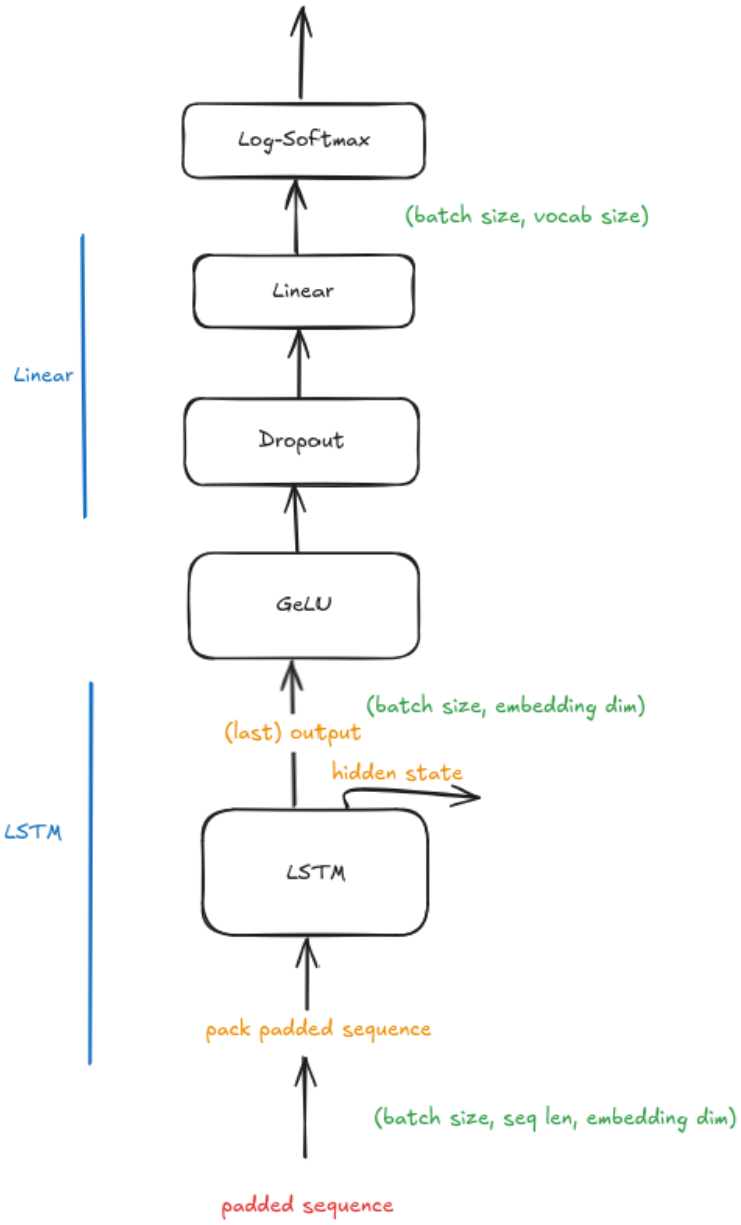
A con of the Neural-Network Language Model is the in-feasibility of running the model on variable length input with ease. Thus, the analysis was performed purely on 5-gram and not sentence-length contexts.

The (best) perplexities yielded are:

	Test	Train	Validation
5-gram	230.95	133.86	227.48

3 Recurrent Neural Network Language Model

3.1 Model



3.2 Perplexities

The Recurrent Neural Network was run on both 5-grams and on sentence length context.

Here, sentence length context refers to using the last word as the target and all of the preceding words as the context. Earlier, I was training by using windows, i.e., if "A B C" is my sentence, I was creating samples as "A -¿ B", "A B -¿ C" and so on, but a TA explicitly asked to do the former, so I had to redo my training at the last minute.

The (best) perplexities yielded are:

	Test	Train	Validation
Sentence	1640.89	508.40	1825.34
5-gram	172.88	100.81	171.34

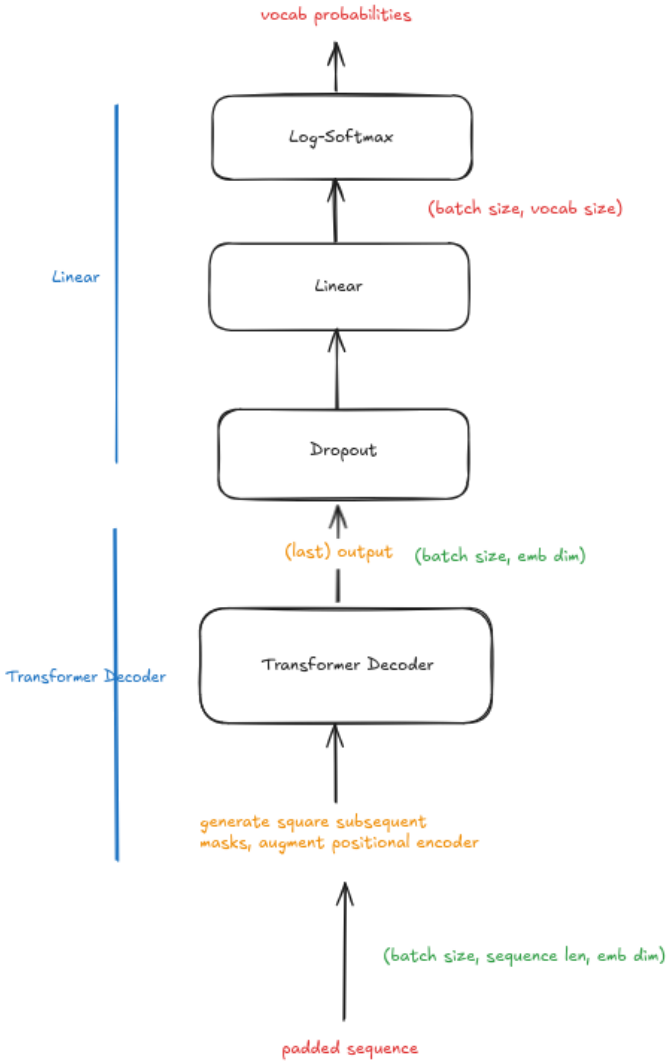
Because of the versatility of variable sequence lengths and much superior performance on 5-grams (with a larger range of dropout ratios yielding similar results, unlike NNLM), I believe that the RNN model is more capable of LM than NNLMs.

4 Transformer Language Model

4.1 Model

Note that the code was largely inspired by the tutorial by debuggercafe. I modified the tutorial to work for a language-model like setting (allowing variable sequence lengths, changing the target to the last word in the sentence with a context of preceding words, and so on).

Number of layers was chosen arbitrarily. 3 Heads was chosen to be divisible by the embeddings dim (300).



4.2 Perplexities

The transformer performed near-flawlessly on 5-grams, with the perplexity approaching 1. On sentence-context however, the loss was not spectacular. This is likely because sentence context provides at most training samples as there are sentences in the corpus, which is disappointingly low in our case. It absolutely blows the transformer out of the water for 5-grams case, leading me to conclude that it is the best model when there is sufficient data.

The (best) perplexities yielded are:

	Train	Test	Validation
Sentence	2641.16	1565.68	2639.36
5-gram	1.02	1.006	1.02