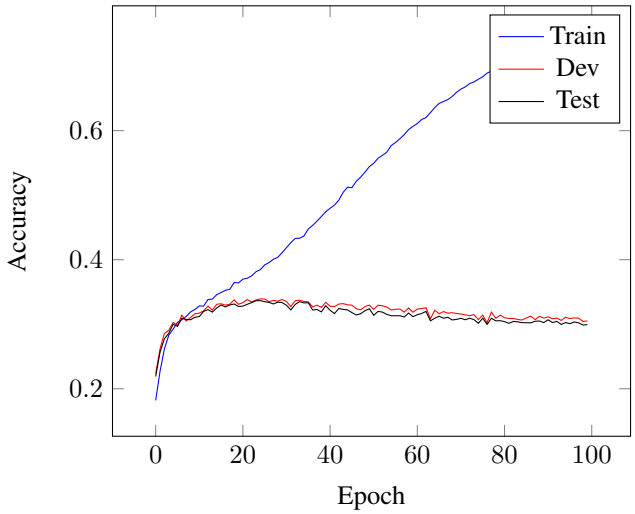


Figure 1: Language Model - LogLoss

TTIC 31210 HOMEWORK 2
SPRING 2017

Hao Jiang

1 LANGUAGE MODELING

In all the experiments mentioned in this section, I use Adam with learning rate $\eta = 0.001$ to update parameters. The hyper-parameters follow the default value suggested in the original paper, that is, $\beta_1 = 0.9, \beta_2 = 0.999$. I use a mini-batching of size 50.

1.1 IMPLEMENTATION AND EXPERIMENTATION

The source code for implementing language model using log loss in in `lm_logloss.py`. The experiment result is demonstrated in Figure 1. The best test accuracy 33.66% is obtained at epoch 24 , with training accuracy 38.46% and dev accuracy 33.93%.

1.2 ERROR ANALYSIS

Table 1 lists the top 20 error made by log loss on dev dataset. They can be generally categorized into the following types:

1. **Name and Pronoun**.

   Example: Bob - He, Bob - She, Bob - Sue, Bob - They.

   These words appear at the same position of a sentence and is interchangable. Context is generally needed to decide which one is better in the sentence. Thus it is understandable that the model cannot distinguish between them.

2. **Was and Other Verbs**

   Example: was - decided, was - had, was - didn, was - 's, was - went

Predict	GroundTruth	Count
Bob	He	158
Bob	Sue	98
Bob	She	96
was	had	46
.	and	41
.	to	35
and	.	32
to	.	31
the	his	31
Bob	The	28
was	decided	28
Bob	They	25
was	didn	23
was	's	22
his	the	22
was	went	22
Bob	But	19
Bob	His	19
Bob	When	19
a	the	18

Table 1: Top 20 Error Made by LogLoss

The subjunctive verbs "was", as the word with highest frequency in training set, is used by the model to replace other verbs that may appear at the same position.

3. **Period and Conjunctions**.

   Example: . - and

   The model may fail to predict whether a sentence comes to an end or is followed by another subsentence connected by "and".

4. **Definite Article and Indefinite Article**

   Example: a - the

   They are both article grammarly and in most of the time interchangable.

1.3 Hinge Loss Implementation

The source code for implementing language model using log loss in in `lm_hingeloss.py`.

1.4 Hinge Loss Experiments

a,b) In Figure 2, we compare the test accuracy of Log Loss and Hinge Loss with different configurations. The first thing we can notice is that LogLoss have a much higher accuracy than any of the Hinge Loss variations and achieve the peak value fast.

When NEG equals the entire vocabulary and use $emb = emb_i$. It can be seen that HingeLoss has a lower best test accuracy, and takes more epochs to converge to the best result (around 25 more epochs). When $emb \neq emb_i$, Hinge Loss is able to achieve a slightly higher test accuracy (31.42% vs. 30.48%) and takes less epochs to reach it (10 epochs faster).

c) Figure 2 also shows the experiments with different negative sample size (all vocabularies, e.g., 1498, 100 and 10). It can be seen that although a larger negative sample size brings higher accuracy, the difference is not that obvious. Specifically, using negative sample size 100 almost achieve same level of accuracy with the case when using all vocabulary as negative sample.

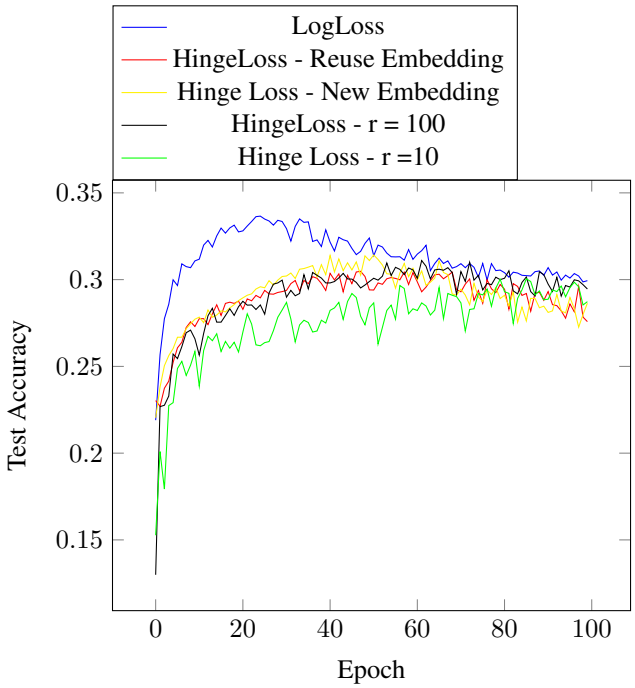


Figure 2: Test Accuracy - Hinge Loss vs. Log Loss

Loss	Log Loss	Hinge Loss - All Vocab	Hinge Loss - r = 10
Value	431.1	503	603.6

Table 2: Comparison of #Sentences / Sec

1.5 Loss Function Comparison and Analysis

a) Table 2 shows the #sents/sec values for the three losses. This number is computed using the following formula:

$$\frac{\text{size of training set}}{\text{average training time}}$$

b,c,d)

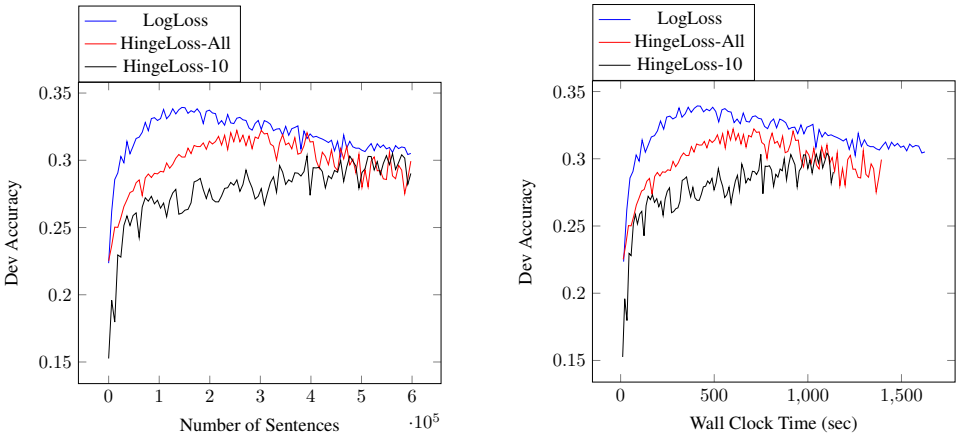
Figure 3a demonstrates how Dev accuracy varies with number of sentences processed and Figure 3b shows how dev accuracy improves with wall clock time. From these figures, it can be observed that LogLoss takes both shorter wall clock time and less number of sentences to achieve its peak dev accuracy. In addition, a smaller negative sample in Hinge Loss has negative impact to both the training time needed and epochs needed to reach the peak dev accuracy.

e) The relationship between dev loss and dev accuracy in Log Loss is demonstrated in Figure 4a. It can be seen that when dev loss reaches its minimum, dev accuracy reaches it's maximum.

f) For Hinge Loss, the result is shown in Figure 4b. Different from the observation in Log Loss, now the dev accuracy occurs at totally different time point from the dev loss. The peak value of dev accuracy appears much slower than the minimum of dev loss.

g, h) From Figure 4a and Figure 4b, it can be seen that when training with Log Loss, after reaching the optimal point, training more epochs does not help improving the dev accuracy and reduce the loss. Instead, the dev accuracy begins to drop and dev loss begins to increase. The same thing happens to the case when using Hinge Loss. This is caused by the overfitting on training data.

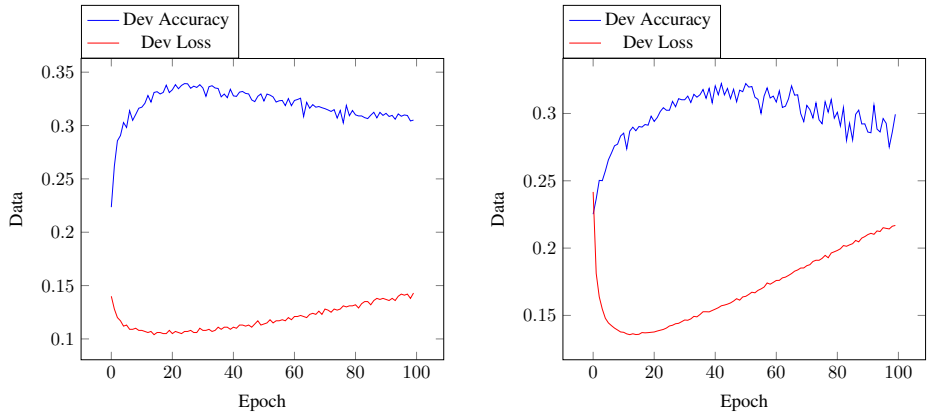
i) If I have a very large training set, I will use Log Loss. Although Hinge Loss takes less time for each epoch, it takes more epochs and thus longer time to achieve its best result. On the contrast, Log



(a) Dev Accuracy with # of Sent

(b) Dev Accuracy with Wall Clock Time

Figure 3: How Dev Accuracy with different factors



(a) Log Loss - Dev Accuracy with Dev Loss

(b) Hinge Loss - Dev Accuracy with Dev Loss

Figure 4: How Dev Loss and Dev Accuracy Varies in different Loss

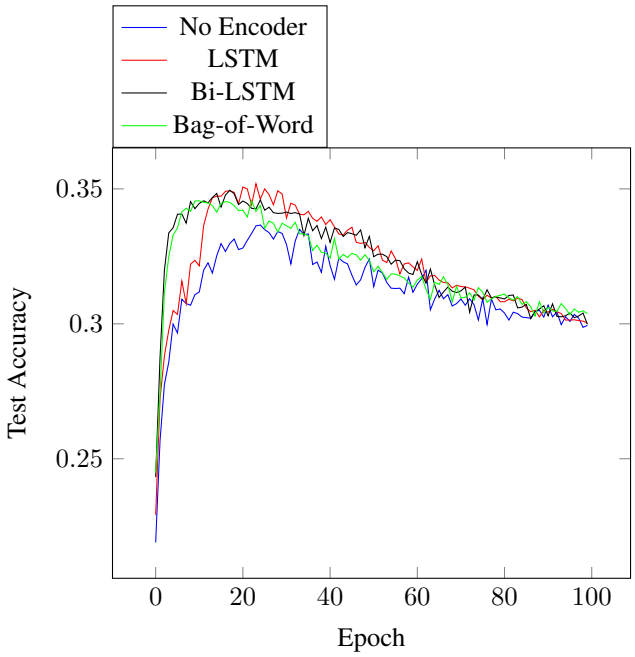


Figure 5: Use Encoders to improve Test Accuracy

Loss takes slightly longer time for each epoch, but use much less epochs to reach its optimum. Thus Log Loss should be a better choice.

2 Sequence-to-Sequence Models

In this section, all the experiments use a similar hyper parameter setting as is in Language model. In addition, we use log loss for all the decoder evaluation.

2.1 Encoder Implementation and Empirical Comparison

a,b,c)

The code of using forward LSTM encoder is shown in `s2s_lstm.py`, the code of using Bi-directional LSTM encoder is shown in `s2s_bilstm.py`, and the code of using Bag-of-Word encoder is shown in `s2s_bow.py`.

Figure 5 shows the test accuracy when using this model and the log loss LSTM with no encoder model. It can be seen the two model have similar convergence trends. They both use 20-25 epochs to reach the best test accuracy. However, LSTM with encoder performs much better than the one with no encoder, and is able to achieve 36% test accuracy. Bi-directional LSTM does not improve test accuracy, however, it reduce the number of epochs needed to train the model. It uses around 5 less epochs to achieve the best result.

In my Bag-of-Word implementation, I simply use the average of previous words as both the hidden state and cell state for the decoder. Interestingly, although this model is much similar than the previous two encoders, it performs almost equally well, while using only 2/3 of the time of the previous two (Average time per epoch: LSTM 36 secs, Bi-LSTM 38 secs, Bag-of-Word 22 secs).

2.2 Error Analysis

The top 20 most frequent error made by forward-LSTM model is shown in Table 3.

The following error categories can be observed.

Predict	GroundTruth	Count
to	.	33
.	and	27
.	to	26
the	his	26
He	Bob	26
Bob	He	25
and	.	24
a	the	21
Sue	Bob	20
the	a	18
was	had	17
for	.	17
her	the	17
had	was	17
the	her	17
his	the	16
a	his	15
.	!	15
.	for	14
.	,	14

Table 3: Top 20 Error Made by Forward-LSTM

1. **Name and Pronoun** This category also appears in Log Loss error.
2. **Was and other Verb** This category also appears in Log Loss error. However, here "was" is no longer dominating. There are cases ground truth is "was" but the model generates "had".
3. **Definite and Indefinite Article** This category also appears in Log Loss error.
4. **Period, Conjunctions, Preposition and Other Symbol**

   Example: "." - "!" , "." - ",". "." - for

   This is a new category. In previous case we notice period is mistaken with conjunctions such as "and". But here we have seen period mistaken with exclamation marks, commas, and other prepositions such as "for" and "to". Exclamation mark and period is in most of the time interchangeable (some people use exclamation mark to replace all periods, especially on Internet) and is hard to distinguish. In addition, generating a period when comma and other conjective words are expected shows that the model tends to generate shorter sentence.

When comparing the number of errors in each case to the Log Loss result(Table 1), it can be noticed that the absolute number of errors is much smaller with forward-LSTM encoder model. For example, in previous model, top 5 errors count in total 439 errors, while in this model, all top 20 errors together count for only 405 errors. Especially, forward-LSTM encoder greatly reduce the number of errors between name and pronouns. In the old model, top errors all belong to this category, while in new model, the error in this category is reduced by over 80%.

2.3 Sentence Embeddings and Story Generation

a,b)

In this experiment, I use the cell state from encoder as the encoded value. The original sentence and top 10 nearest neighbor sentences generated by bi-directional LSTM and bag-of-word is demonstrated below. The code is in `s2s_neighbor.py`.

In bi-directional LSTM case, the most obvious similarity captured is the first several words. For example, for the original sentence ”He said it was the most fun he ’s had in a long time . ”, top 10 neighbors are

He landed on his arm , and it broke under him .
He pulled out a shopping list he made to check what to get .
The new lady took him by the hand and walked towards the door .
He came in and stole the only chair we had .
He ran out of time before he had to eat with the family .
His dad missed , and the baseball hit him right in the face .
He could not leave the house for another week after he got out of bed .
He opened the door and decided he didn ’t need his coat .
His mother told him that they must go to the doctor .
Since it well on the side with the cheese , it was no longer good .

and for ”Bob spent hours working for money for the project . ”, top 10 neighbors are

Bob was 8 and really wanted a pet .
Bob was an older man with a large family .
Bob was disappointed but kept waiting for the call .
Bob decided to go to his grandmother ’s house .
Bob went to the store and bought a pool .
Bob tried to talk to a girl at the bar .
Bob was on a camping trip with his friends .
Bob wanted to get a cat to keep him company .
Bob was excited and nervous at the same time .
Bob wanted to learn how to play the piano .

In bag-of-word case, the similarity seems focus more on the entire content of the sentence. For example, with original sentence ”Bob proposed to her at dinner . ”, top 10 neighbors are

Bob proposed to her at dinner .
Bob proposed to her that night at dinner .
She wanted to get her mind of Bob from school .
Sue wanted Bob to ask her on a date .
Bob took Sue to her favorite restaurant .
When Bob spoke to Sue in class she spoke back .
Bob decided to make her dinner .
Bob met her but was too nervous to say anything .
During math class , Sue passed a note to Bob .
Sue and Bob wanted to go to the movies .

Here sentences starts ”Bob”, ”Sue”, and ”Her” all have chance to be chosen, which enables higher possibility to find related sentences.

c)

I use bi-directional LSTM to generate next sentences. The code can be found in `s2s_nextsentence.py`. The result is shown in Table 4. It can be seen that all generated sentences are properly formatted (end with period) and are meaningful English sentences. Most of them have a close meaning to the query sentences.

2.4 EXTRA CREDIT

For extra credit, I implement attention based on bi-directional LSTM and use it to perform both next word prediction and story generation. The code can be seen in `s2s_attention.py`

Original Sentence	Generated
He was accepted !	Bob was very happy that he did not have to go alone .
Sue was the last one in the house .	She was nervous .
Then Bob caught up .	He was nervous .
His mom and dad told him that soon he would be a big brother .	Bob was saved !
Her dentist decided to pull the teeth to speed things up .	Sue was thrilled .
Sue knew she would be in trouble if she woke him up .	She told Sue that they would have to give the dog away .
He searched every store and no one carried fresh milk .	Bob was mad .
A past due notice arrived in the mail , but Bob ignored it .	Bob told his teacher the story .
It was a close race but Bob pulled ahead at the end .	The old job took Bob back happily .
She found some cute black ones .	Sue was thrilled .

Table 4: Using forward-LSTM to generate story

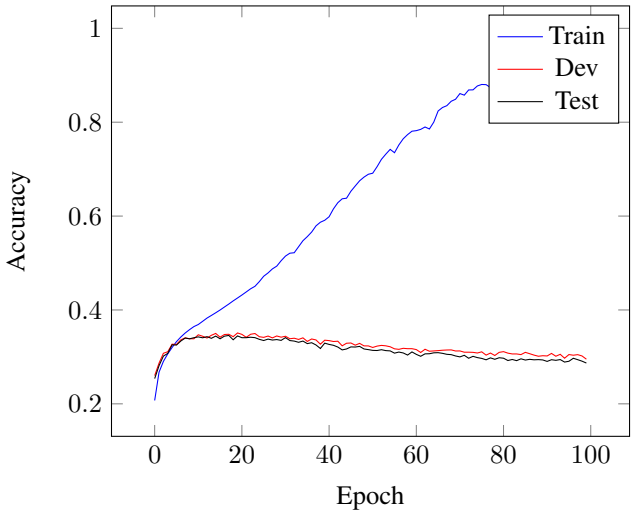


Figure 6: Bi-directional LSTM with Attention

My implementation follows the structure described in (1), that is, the decoder's current state at step $t, c_t^{(d)}$ is computed as a weighted sum of encoder's hidden states $h_i^{(e)}$, where the weight α_{ti} for $h_i^{(e)}$ is the softmax of dot product of $h_i^{(e)}$ and decoder's previous hidden state $h_{t-1}^{(d)}$.

Figure 6 shows the experiment result. Unfortunately, although it takes longer time to train, the best test accuracy (34.16%) is slightly worse than bi-directional LSTM without attention (34.85%).

When the attention model is used to generate sentences, we did see some new longer sentences such as "Bob decided to go to the store to get to work . " and "The dog would not go away , but laid down on Sue 's mother 's ' feet . " But in general, no obvious improvement has been observed from adding attention to the model.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.