DS-GA 1011 NLP HW1 - Sylvie Shao (MS6771)

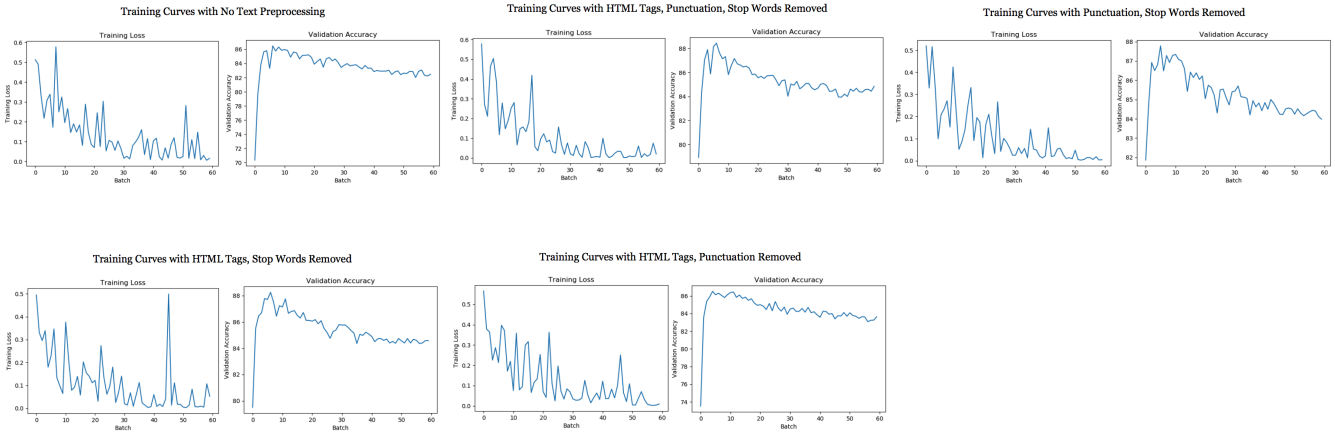
Github: https://github.com/sylvieshaonyu/DSGA-1011-NLP

1 Hyperparameter Tuning

1. Tokenization Schemes
   In order to fully understand how big impact different tokenization schemes could have on the model performance, I included three different tokenization schemes - removing HTML tags, removing punctuations and removing stopwords. Five combinations of these tokenization schemes were experimented. At the end, training with HTML tags, punctuation and stopwords removed achieved the highest validation accuracy. Indeed, those meaningless but frequent words, tags and punctuation can adds nonsense to the vocabulary and thus achieving suboptimal results.

Tokenization Schemes

ID	Remove HTML	Remove Punctuation	Remove Stopwords	Ngram	Vocabulary Size	Embedding Dimension	Learning Rate	Decreasing Learning Rate	Optimizer	Training Loss	Validation Accuracy
1	No	No	No	1	10000	100	0.01	No	Adam	0.1725	86.48%
2	Yes	Yes	Yes	1	10000	100	0.01	No	Adam	0.1176	**88.44%**
3	No	Yes	Yes	1	10000	100	0.01	No	Adam	0.1522	86.92%
4	Yes	No	Yes	1	10000	100	0.01	No	Adam	0.1795	87.78%
5	Yes	Yes	No	1	10000	100	0.01	No	Adam	0.2191	86.14%



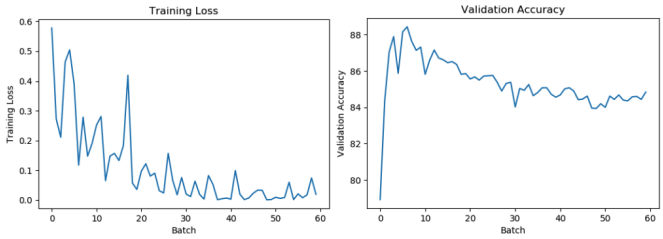
2. Model Hyperparameters

   (a) N-gram
   After playing with 1, 2, 3 and 4 grams, uni-gram achieved the best result. As the n-gram length increases, the amount of times any given n-gram shows up will decrease, which would lead to data sparsity. The more sparse the data set, the worse the model could possibly fit it. For this reason, despite that a higher-order n-gram model, in theory, contains more information about a word's context, it cannot easily generalize to other data sets because the number of events (i.e. n-grams) it has seen during training becomes progressively less as n increases.

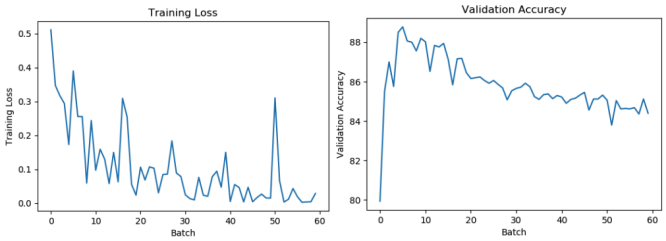
Tokenization Schemes

ID	Remove HTML	Remove Punctuation	Remove Stopwords	Ngram	Vocabulary Size	Embedding Dimension	Learning Rate	Decreasing Learning Rate	Optimizer	Training Loss	Validation Accuracy
2	Yes	Yes	Yes	1	10000	100	0.01	No	Adam	0.1176	**88.44%**
6	Yes	Yes	Yes	2	10000	100	0.01	No	Adam	0.1498	87.94%
7	Yes	Yes	Yes	3	10000	100	0.01	No	Adam	0.1209	88.14%
8	Yes	Yes	Yes	4	10000	100	0.01	No	Adam	0.1284	88.02%

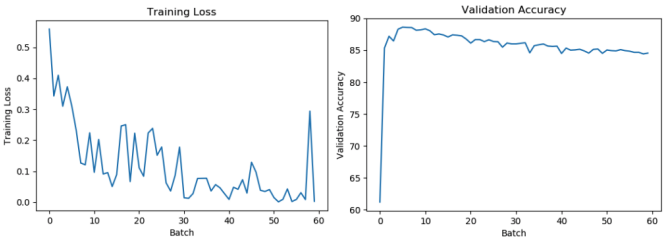
Training Curves with 1-Gram



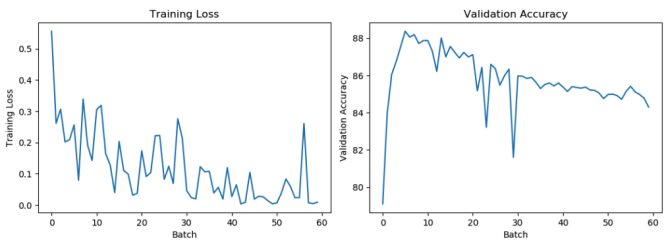
Training Curves with 2-Grams



Training Curves with 3-Grams



Training Curves with 4-Grams



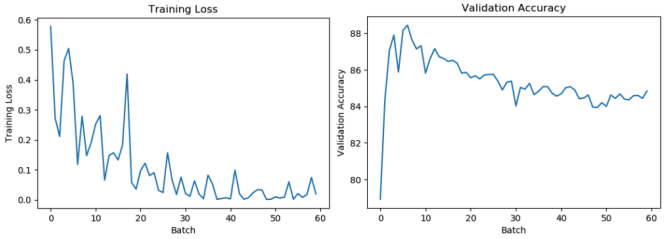
(b) Vocabulary Size

With 10k, 30k, 50k and 100k vocabulary size experimented, 30k achieved the highest validation accuracy. Having a relatively large vocabulary is beneficial for model training. However, with a vocabulary size being too large (i.e. 100k), it cannot easily generalize to other data sets because of the increasing number of less frequent n-grams in the vocabulary.

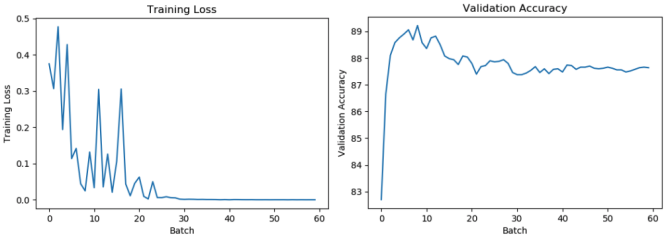
Tokenization Schemes

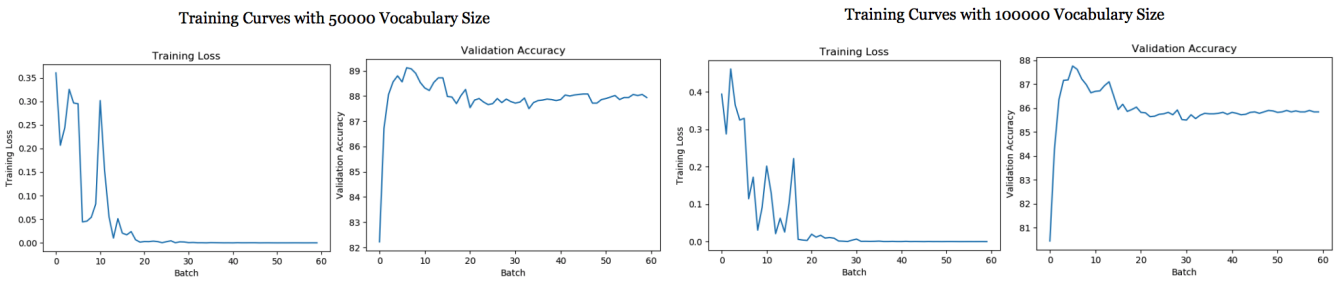
ID	Remove HTML	Remove Punctuation	Remove Stopwords	Ngram	Vocabulary Size	Embedding Dimension	Learning Rate	Decreasing Learning Rate	Optimizer	Training Loss	Validation Accuracy
2	Yes	Yes	Yes	1	10000	100	0.01	No	Adam	0.1176	88.44%
9	Yes	Yes	Yes	1	30000	100	0.01	No	Adam	0.0247	**89.22%**
10	Yes	Yes	Yes	1	50000	100	0.01	No	Adam	0.0444	89.12%
11	Yes	Yes	Yes	1	100000	100	0.01	No	Adam	0.0624	87.10%

Training Curves with 10000 Vocabulary Size



Training Curves with 30000 Vocabulary Size



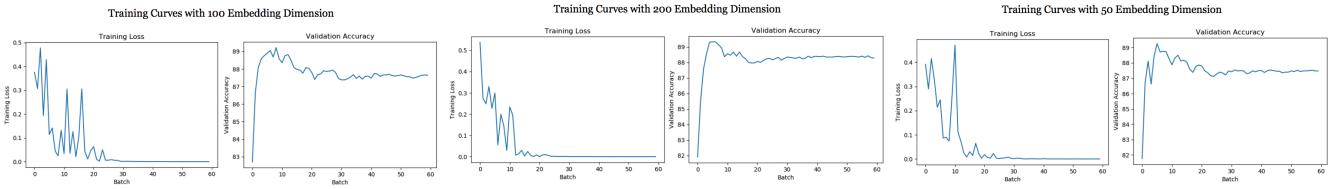


(c) Embedding Size

Longer embedding size don't add enough information and smaller ones don't represent the semantics well enough. In this experiment, 100 seems to be best embedding dimension.

Tokenization Schemes

ID	Remove HTML	Remove Punctuation	Remove Stopwords	Ngram	Vocabulary Size	Embedding Dimension	Learning Rate	Decreasing Learning Rate	Optimizer	Training Loss	Validation Accuracy
9	Yes	Yes	Yes	1	30000	100	0.01	No	Adam	0.0247	**89.22%**
12	Yes	Yes	Yes	1	30000	200	0.01	No	Adam	0.0899	88.76%
13	Yes	Yes	Yes	1	30000	50	0.01	No	Adam	0.1624	83.88%



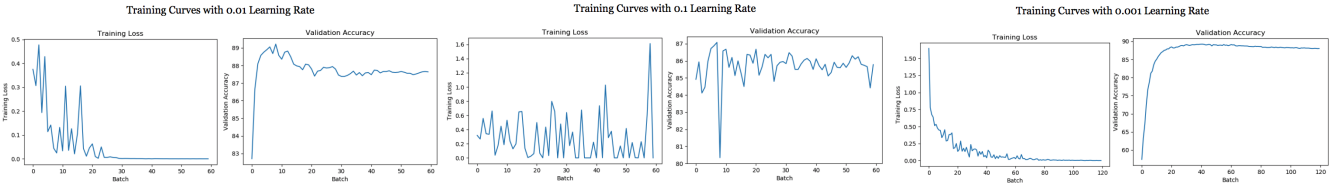
3. Optimization Hyperparameters

(a) Fixed Learning Rate

Learning rate is very important for optimization. Choosing a learning rate too large would lead to the failure of convergence (shown in the middle graph). Choosing a learning rate too small would take longer and more epochs to converge (shown in the right graph). Therefore, an appropriate 0.01 learning rate proves to work the best for this case.

Tokenization Schemes

ID	Remove HTML	Remove Punctuation	Remove Stopwords	Ngram	Vocabulary Size	Embedding Dimension	Learning Rate	Decreasing Learning Rate	Optimizer	Training Loss	Validation Accuracy
9	Yes	Yes	Yes	1	30000	100	0.01	No	Adam	0.0247	**89.22%**
14	Yes	Yes	Yes	1	30000	100	0.1	No	Adam	0.0378	86.88%
15	Yes	Yes	Yes	1	30000	100	0.001	No	Adam	0.0284	89.08%



(b) Linear Annealing of Learning Rate

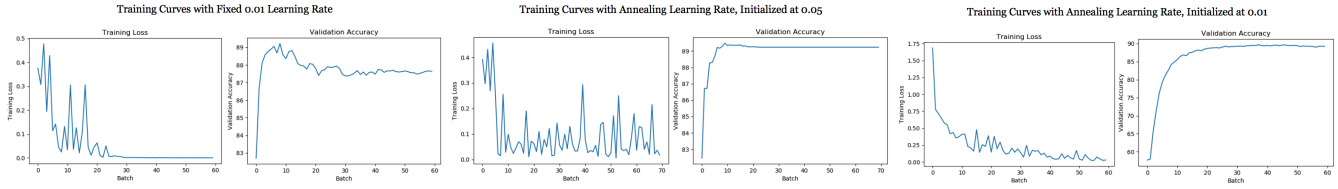
Adapting the learning rate for optimization procedure can increase performance and reduce training

time. It has the benefit of making large changes at the beginning of the training procedure when larger learning rate values are used, and decreasing the learning rate such that a smaller rate and therefore smaller training updates are made to weights later in the training procedure. This has the effect of quickly learning good weights early and fine tuning them later. Indeed, annealing learning rate with initialized at a good starting point (i.e. right graph) resulted in the best result.

In fact, removing HTML tags, punctuations and stopwords and using Adam optimizer with an annealing learning rate with initialized at 0.01 (ID 17) achieved the best validation accuracy during the whole experiment, which resulted in a test accuracy of 86.77%.

Tokenization Schemes

ID	Remove HTML	Remove Punctuation	Remove Stopwords	Ngram	Vocabulary Size	Embedding Dimension	Learning Rate	Decreasing Learning Rate	Optimizer	Training Loss	Validation Accuracy
9	Yes	Yes	Yes	1	30000	100	0.01	No	Adam	0.0247	89.22%
16	Yes	Yes	Yes	1	30000	100	0.05	Yes	Adam	0.0282	89.48%
17	Yes	Yes	Yes	1	30000	100	0.01	Yes	Adam	0.1054	**89.66%**



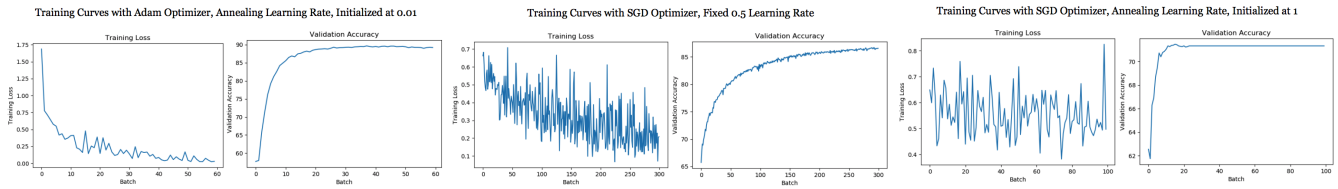
(c) Optimizer

From my experiment, SGD seemed to converge slower, and was unable to achieve the same result as Adam. For SGD, choosing a proper learning rate seems to matter a lot. However, choosing a proper learning rate can be difficult. A learning rate that is too small leads to painfully slow convergence, while a learning rate that is too large can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge. Another key challenge of minimizing highly non-convex error functions common for neural networks is avoiding getting trapped in their numerous suboptimal local minima.

Adam on the other hand is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average of past gradients similar to momentum, which empirically works better.

Tokenization Schemes

ID	Remove HTML	Remove Punctuation	Remove Stopwords	Ngram	Vocabulary Size	Embedding Dimension	Learning Rate	Decreasing Learning Rate	Optimizer	Training Loss	Validation Accuracy
17	Yes	Yes	Yes	1	30000	100	0.01	Yes	Adam	0.1054	**89.66%**
18	Yes	Yes	Yes	1	30000	100	0.5	No	SGD	0.1173	86.68%
19	Yes	Yes	Yes	1	30000	100	1	Yes	SGD	0.6403	71.46%



* Please refer to the jupyter notebook for 3 correct and 3 incorrect predictions.