Bag-of-Words based Natural Language Inference

Nimi Wang; Ruoyu Zhu; Xiao Tan

1 Training on SNLI

1.1 Feature Extraction and Model Structure

We used a Bow encoder to map each string of text to a fixed dimension vector(we used 100 here).We implemented two models for this task: Logistic Regression and a 2 hidden layer Neural Network. The structure of the neural network consists of 2 linear transformations and one ReLU activation functions in between.The hidden layer has a dimension of 20.

1.2 Hyper-parameter tuning

1.2.1 Hyper-parameter settings

We tuned three sets of hyper-parameters. (a) Method of combining the two vector representation of the two sentences. (b) Optimization hyper-parameters including optimizer (SGD vs Adam),the learning rate, and whether to use linear annealing of learning rate. (c)Different rates of dropout. For other parameters that were not tuned, we kept them fixed.

1.2.2 Performance on each model

Model 1: Two Layers Neural Net

Merge method	optimizer	learning rate	adjust learning rate	dropout rate
Concatenation	Adam	0.01	False	0
Element-wise product	Adam	0.01	False	0
Sum	Adam	0.01	False	0
Element-wise product	Adam	1	False	0
Element-wise product	Adam	0.01	False	0.2
Element-wise product	SGD	0.01	False	0

(a) Tuning the method of handling the vector representations of the two sentences

Concatenating the two sentences directly, we observed a validation accuracy around 64.8% and training accuracy around 70.5%. We observed from the plot that with small number of epochs, the validation accuracy fluctuates quite often. This is mostly due to that we're using only a subset of the whole dataset (limited data) and a very simple model, causing some overfit which results in the instability of validation. However, the training loss

reached to a plateau, so we believe that the model has converged. Overall, the performance is not bad with the given circumstances.

Computing element-wise product, we observed a validation accuracy around 65.9%, and a training accuracy around 82.3%. This model performed better compared to the one above in terms of validation accuracy. However, from the training loss plot, we see that it converges slower than the one above.

Computing the sum, we observed a validation accuracy around 57.9%, and a training accuracy around 65.5%. This is much lower than the previous two models, mainly given that using the sum, instead of concatenation and element-wise product, we don't really get any information from the relationship (distance) of the two sentences. Thus, doing the sum is nearly useless for the inference task.

To summerize, computing the element-wise product of the two sentences gave the best performance.(The results are shown in Figure 1.) Thus, we will use this parameter in the following tasks.

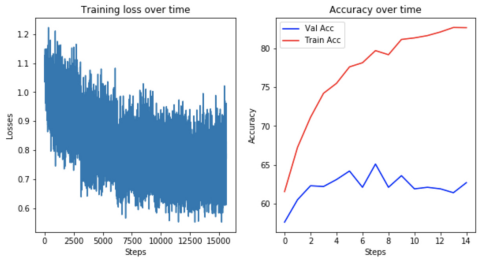

Figure 1: Training and validation losses and accuracy when using element-wise product (lr=0.01)

(b) Tuning optimization hyper-parameters

Previously, we all used a small learning rate to make sure the model converges. Here, we tried a higher learning rate of 1. We observed that the validation(33.4%) and training accuracy (33.4%) went down by a lot. This is because high learning rate causes undesirable divergent behavior in your loss function. From the training loss plot(figure 2), we see the model does not even converge, and training and validation accuracy are very unstable.

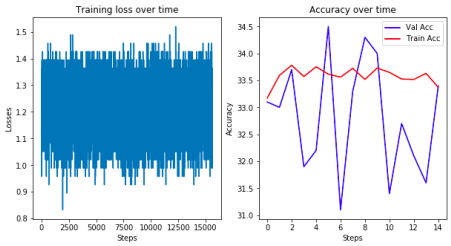

Figure 2: Training and validation losses and accuracy when using high learning rate (lr = 1)

Then we used a small learning rate and SGD as optimizer. We achieved a training and validation accuracy both around 40%. We see it converged much slower than Adam. This is because Adam Optimizer can help smooth gradients and speed up convergence, especially for sparse data like text.(Shown in Figure 3)

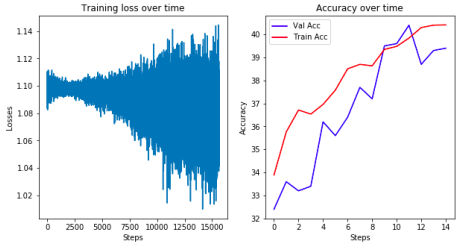

Figure 3: Training and validation losses and accuracy when using SGD

(c) Tuning dropout rates

When we added drop rates of 0.2, the validation accuracy becomes even higher to 65.8%. This is because we added more regularization to make sure the model does not overfit.

In conclusion, we used element-wise product method, learning rate 0.01, Adam optimizer, and 0.2 dropout rate as the hyper-parameters for our final neural network model.

Model 2: Logistic Regression

For logistic regression, we focused on using the learning rate annealing or not. Without learning rate schedule and a 0.01 learning rate, we achieved a training accuracy of 88.6% and validation accuracy of 65.5%.

Now we added learning rate annealing. We started with a higher learning rate of 0.1, and set the decay speed to be 0.1, we found that the model will achieve the same accuracy even with the high learning rate. This is because as loss approaching the minima, the learning rate gradually lowers. However, it is interesting to see that when we start with a small learning rate,and annealing, the model becomes much slower to converge. However, for simplicity purpose, we will choose learning rate 0.01 and no annealing adjustment for the final model.

To summarize, the values we chose for the parameters are: using element-wise multiplication, Adam optimizer, learning rate 0.01 with no learning rate annealing adjustment.

1.3 Analysis on prediction performances

Based on the three correct and incorrect examples extracted (see Appendices A), we observed that the model always have trouble understanding the complicated sentences, either these sentences are very long or they have complicated logic embedded. This shows that the model is not good at extracting complicated information, especially is not location-sensitive.

2 Evaluating on MultiNLI

Overall, the validation accuracy of MNLI dataset is lower compared to SNLI, with genre Telephone being the highest accuracy one using NeuralNet and genre Slate and Travel being the highest accuracy one using Logistic Regression. However, the validation accuracy do not vary much across genres. This is probably under the assumption that all genres appear with the same probability in the training dataset of SNLI. We did not specifically trained for one kind of genre. (See the first three columns of Table 1 below.)

3 Fine-tuning on MultiNLI

As we can see from the table above, after fine-tuning we see great improvement for every genre using either model. Especially we see the Travel genre improved the most greatly.

4 Pre-Trained Word Embedding

Repeat 3.1

With frozen pre-trained embeddings from fastText, for neuralnet work model, we achieved a training accuracy of 56.3% and validation accuracy of 55.0%. For logistic regression, training accuracy is 52.7% and validation accuracy is 49.8%. However with lower accuracy we conclude that pre-trained embeddings might not be more helpful than self-trained embedding

Table 1: Model with Self-trained Embeddings

Model	Genre	ValAcc	ValAcc (fine-tuned)
NeuralNet	Telephone	42.78	44.77
LogReg	Telephone	32.23	40.30
NeuralNet	Fiction	40.90	45.53
LogReg	Fiction	31.35	43.62
NeuralNet	Slate	40.02	46.51
LogReg	Slate	36.33	41.52
NeuralNet	Government	40.65	48.43
LogReg	Government	29.43	48.72
NeuralNet	Travel	40.02	79.79
LogReg	Travel	36.33	91.35

layer. Since the pre-trained embeddings was trained for a different task from our natural language inference task, it wouldn't help (much) necessarily, especially with such limited data.

From the training loss plot (see code), we observed that the model converged a bit faster with 5 epochs than previous best model. This is because this pre-trained embeddings work better as an initial weight than random weight values. However, if things might be different if we try embeddings like ELMo, BLUE or BERT which has been verified their ability to achieve better performance on multiple tasks. We might get better results using them.

Appendices B shows us the three correct and three incorrect examples. The possible reasons for the wrong examples are the same as above. However, it is worth noted that for incorrect example No.2, we can see the model is not good at differentiating action verbs.

Repeat 3.2

Table 2: Model with Pretrained Embeddings Evaluation on MNLI

Model	Genre	ValAcc
NeuralNet	Telephone	35.92
LogReg	Telephone	36.02
NeuralNet	Fiction	36.08
LogReg	Fiction	34.87
NeuralNet	Slate	36.93
LogReg	Slate	36.83
NeuralNet	Government	29.92
LogReg	Government	30.02
NeuralNet	Travel	36.93
LogReg	Travel	36.83

We achieved a lower validation accuracy with pretrained embeddings than with self-trained embedding layer. Likewise, there isn't much difference between different genres.

5 Bonus

For this problem, we used cosine distance to measure the similarity between two words, since it represents the distance between two vectors. The higher the cosine distance is, the more similar the two words are, and vice versa. The top 10 most similar words for each best model is indicated in the Appendices C and D. They are very different from the ones we used in section 3.4. It makes sense since the best models we have are trained on this specific training data. The words that have the most different embedding vectors from section 3.1 to 3.3 vary from genre to genre. For example, words 'protect' and 'treats' changes the most in government data-set, while it doesn't change that much in other genre. It makes sense since those two are very sensitive words regarding the government. However, some words, like 'act','shaping','caution' all change a lot across most of the genres.

Appendices

A Correct and Incorrect Examples for 3.1

```
Correct Examples:


Premise: a woman in a blue and black jacket looking at a sign by railroad tracks
Hypothesis: a woman is waiting for her train
Correct Label: neutral


Premise: a boy in sandals is looking at a camera in a busy street
Hypothesis: a boy is in a busy street wearing sandals looking at a camera
Correct Label: entailment


Premise: horses jumping a hurdle in a race
Hypothesis: horses are jumping over a hurdle in the last race of the day
Correct Label: neutral
--------------------------------------------------------------------------------------------------------------------------
Wrong Examples:


Premise: a woman wearing sunglasses holds two newspapers
Hypothesis: a woman wearing sunglasses holds two newspapers as she leaves the minimart and heads home
Correct Label: neutral
Predicted Label: entailment


Premise: two backpackers look at scenic mountains
Hypothesis: two guys are carrying things on their back
Correct Label: entailment
Predicted Label: neutral


Premise: two men on a basketball court one in white and red the other in blue the player in white is throwing the ball in
while teammates look on
Hypothesis: the men are playing basketball
Correct Label: entailment
Predicted Label: contradiction
```

B Correct and Incorrect Examples for 3.4

```
Correct Examples:


Premise: man in overalls with two horses
Hypothesis: a man in overalls with two horses
Correct Label: entailment


Premise: a woman in a blue shirt talking to a baby
Hypothesis: a young woman talks to her baby at the park
Correct Label: neutral


Premise: a man in a red shirt entering an establishment
Hypothesis: lucy was at the library
Correct Label: contradiction
--------------------------------------------------------------------------------------------------------------------------
Wrong Examples:


Premise: a man in a dark shirt and overalls is squatting down on gravel and working on a railroad car
Hypothesis: several men wearing overalls are working on a railroad car while knelling on gravel
Correct Label: entailment
Predicted Label: neutral


Premise: a man in a green shirt and black hat is smoothing concrete
Hypothesis: the man is drilling into concrete
Correct Label: contradiction
Predicted Label: entailment


Premise: a woman in a black coat is riding her bike on a street as a man in shorts and a striped top looks at her as he pa
sses her
Hypothesis: a man is checking out a woman as she rides by on her red bike
Correct Label: entailment
Predicted Label: neutral
```

C Results for similar words for Bonus Questions

C.1 For best neural network

```
tall and sad have 0.7447402477264404 of the similarity score
sitting and sits have 0.7426776885986328 of the similarity score
no and nothing have 0.7397510409355164 of the similarity score
cats and nothing have 0.7117267847061157 of the similarity score
no and frowning have 0.702558696269989 of the similarity score
favorite and joyously have 0.6931725740432739 of the similarity score
no and nobody have 0.6906219124794006 of the similarity score
alone and nobody have 0.6891882419586182 of the similarity score
because and sad have 0.6856215596199036 of the similarity score
no and cats have 0.6824354529380798 of the similarity score
no and only have 0.673235297203064 of the similarity score
empty and cats have 0.671751856803894 of the similarity score
sleeping and nothing have 0.6712570190429688 of the similarity score
only and frowning have 0.6707717776298523 of the similarity score
no and n't have 0.6704748868942261 of the similarity score
```

C.2 For best Logistic Regression

```
no and nobody have 0.8180833458900452 of the similarity score
sad and joyously have 0.7885147929191589 of the similarity score
sleeping and asleep have 0.763414740562439 of the similarity score
nothing and silently have 0.7508991956710815 of the similarity score
no and nothing have 0.74535071849823 of the similarity score
siblings and joyously have 0.732618510723114 of the similarity score
sitting and sit have 0.729503870010376 of the similarity score
nothing and aliens have 0.7275766730308533 of the similarity score
nothing and actually have 0.7245376110076904 of the similarity score
sleeping and sleeps have 0.7238885760307312 of the similarity score
brothers and joyously have 0.7237866520881653 of the similarity score
nobody and never have 0.7237042784690857 of the similarity score
tall and sad have 0.7101843953132629 of the similarity score
no and cats have 0.7075361013412476 of the similarity score
nobody and quietly have 0.7056280374526978 of the similarity score
```

D Results for words that change the most for each genre for Bonus Questions

D.1 Telephone data

```
"acts" has very different embedding vectors before and after on Tel dataset
"shaping" has very different embedding vectors before and after on Tel dataset
"caution" has very different embedding vectors before and after on Tel dataset
"a" has very different embedding vectors before and after on Tel dataset
"stamp" has very different embedding vectors before and after on Tel dataset
```

D.2 Fiction data

```
"acts" has very different embedding vectors before and after on Fic dataset
"caution" has very different embedding vectors before and after on Fic dataset
"shaping" has very different embedding vectors before and after on Fic dataset
"stamp" has very different embedding vectors before and after on Fic dataset
"seeks" has very different embedding vectors before and after on Fic dataset
```

D.3 Slate data

```
"caution" has very different embedding vectors before and after on Sla dataset
"teeter" has very different embedding vectors before and after on Sla dataset
"acts" has very different embedding vectors before and after on Sla dataset
"seeks" has very different embedding vectors before and after on Sla dataset
"companion" has very different embedding vectors before and after on Sla dataset
```

D.4 Travel data

```
"boys" has very different embedding vectors before and after on Tra dataset
"observes" has very different embedding vectors before and after on Tra dataset
"supplies" has very different embedding vectors before and after on Tra dataset
"brand" has very different embedding vectors before and after on Tra dataset
"landed" has very different embedding vectors before and after on Tra dataset
```

D.5 Government data

```
"protects" has very different embedding vectors before and after on Gov dataset
"choosing" has very different embedding vectors before and after on Gov dataset
"seeks" has very different embedding vectors before and after on Gov dataset
"treats" has very different embedding vectors before and after on Gov dataset
"began" has very different embedding vectors before and after on Gov dataset
```