Bag-of-Words based Natural Language Inference

Cody Fizette cf2372@nyu.edu

Lekha Iyengar li471@nyu.edu

Ying Jin yj1461@nyu.edu

Haonan Tian ht1151@nyu.edu

Abstract

In this assignment we use a Bag of Words model for natural language inference on Stanford Natural Language Inference (SNLI)[1] and Multi-Genre Natural Language Inference (MNLI)[2] dataset. We train the model on SNLI dataset and finetune it on MNLI dataset.

1 Model

Classifying the premise and hypothesis as an entailment, contradiction or neutral is a 3 class classification problem. For this task we use a bag of words encoder to get an embedding of the premise and hypothesis. These embeddings are then combined and given as input to a classifier. The output of the encoder serves as the input for the classifier. The encoder and the classifier are trained end to end. We test two classifiers here, Logistic Regression classifier (linear layer followed by log softmax) and a fully connected 3 layer Neural Network classifier.

1.1 Bag of Words Encoder

In a bag of words encoder, the input is represented as a bag of words ignoring syntax and word order but keeping multiplicity. We keep a count of total occurrences of the most frequent words. It is a simple lookup table that stores embeddings of a fixed dictionary and size. We retrieve the word embedding using the index of the word. The *nn.Embedding* module is used is used to implement this.

2 Training and Hyperparameter Tuning on SNLI

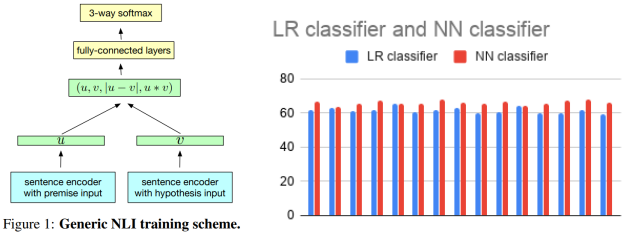

Figure 1: **Generic NLI training scheme.**

Figure 1: Left: Generic NLI training Scheme. Right: Some examples of Neural network and Logistic regression performances on SNLI.

The neural network outperforms logistic regression for the same set of hyperparameters. The best validation accuracy we get for our neural network is **69.4%** for vocabulary size 10k, embedding dimension 500 and element-wise multiplication for combining encodings. The best validation accuracy for logistic regression classifier is **65.4%** for vocabulary size 10k, embedding dimension 50 and element-wise multiplication for combination. Figure 1 shows that NN is consistently better than LR on the same set of parameters.

2.1 Combining encodings of premise and hypothesis

Encodings of premise and hypothesis sentences are concatenated, subtracted and element wise multiplied. Concatenation takes embedding of both premise and hypothesis, so the representation of each sentence is a feature for the model. Subtraction takes into account the difference between features of the two sentences. Element-wise multiplication captures the similarities between sentences or the contrasts. The best method of combination method for Logistic Regression is element wise multiplication. Concatenation works best for neural networks when the vocabulary and embedding dimension is small. As we increase the embedding dimension, multiplication begins to perform better.

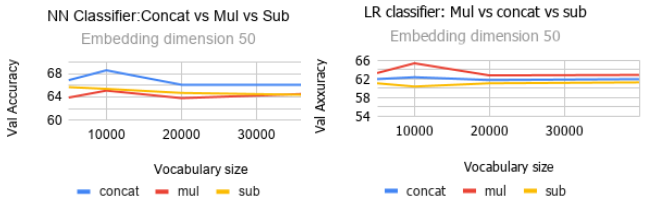

Figure 2: Comparison of different combination methods

2.2 Vocabulary size and embedding dimension

Vocabulary size

Using the full or a very large vocabulary is neither economical nor necessary, as many words may contribute little to the end task and could have been safely removed from the vocabulary [3]. We start with a small vocabulary size and gradually increase it to see it's effect on performance. As shown in figure 3, some models begin to perform better than others as vocabulary size increases (e.g. SUB for embedding dimension 200). We use the following vocabulary sizes in our experiments - 5000, 10000, 20000 and 40000.

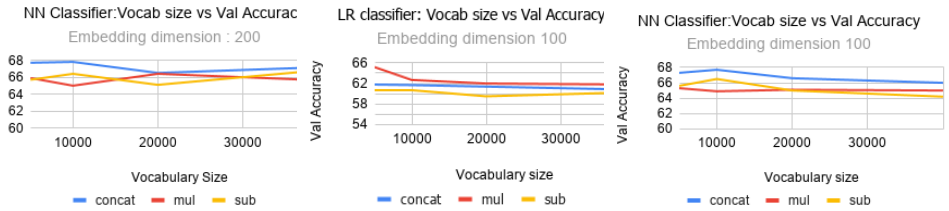

Figure 3: Performance of Classifiers for different vocabularies

Vocabulary size	Validation accuracy	No of trainable params	Training loss	Validation loss
5000	65.2	1019795	0.5661907196	0.5196363926
10000	62.7	2032411	0.5599887371	0.407695055
20000	62	3951459	0.5351073742	0.831408143
40000	61.8	6294139	1.009298325	1.470616102

Table 1: Tuning vocabulary (LR classifier, embedding dim = 100, element-wise multiplication)

Vocabulary size	Validation accuracy	No of trainable params	Training loss	Validation loss
5000	66.9	1515001	0.5473144054	0.8822601438
10000	68.6	3002180	0.5780542493	0.5924090743
20000	66.1	5720267	0.6549695134	0.3824564815
40000	66.1	7547312	0.5834906697	0.5888049006

Table 2: Tuning vocabulary (NN classifier, embedding dimension = 50, combination method - concatenation)

Embedding dimension

The quality of embeddings learnt is dependent on its dimension. If the dimension is small then the embedding may not be able to capture all possible word relations and if it is large then it might overfit.[4] The number of trainable parameters is usually linearly or quadratically proportional to the embedding dimension. So increasing the embedding size, drastically increases training time and computational cost. Figure 4 on the left shows a model with embedding dimension 500 which overfits the data. We can counter this problem by adding dropout regularization or weight decay. Figure 4 on the middle and left shows the result on different embedding dimensions. Embedding dimensions used in experiments are 50, 100, 200, 300.

Embedding dimension	Validation accuracy	No of trainable params	Training loss	Validation loss
50	65.4	1036061	0.7017114162	0.738355279
100	62.7	2032411	0.5599887371	0.407695055
200	62.4	4025111	0.7023311257	0.4893234074
300	62.6	6017811	0.6586046219	0.5971964002
500	62.8	10003211	1.06279695	1.057050943

Table 3: Tuning embedding dimension (LR classifier, vocabulary 10k, element-wise multiplication)

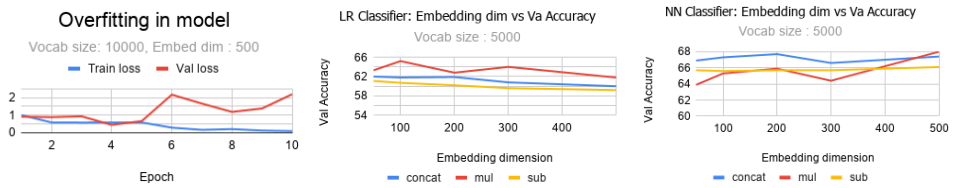

Figure 4: Left: Overfitting in Neural Network model as number of trainable parameters increases. Middle Right: Performance of Classifiers for different embedding dimensions

Embedding dimension	Validation accuracy	No of trainable params	Training loss	Validation loss
50	68.6	3002180	0.5780542493	0.5924090743
100	67.7	3998530	0.5985412002	1.355012655
200	67.8	5991230	0.9562202692	1.003657222
300	66.6	7983930	0.5517901182	0.7205377221
500	66.8	11969330	0.4274890423	0.6984376311

Table 4: Tuning embedding dimension (NN classifier, vocabulary 10k, concatenation)

2.3 Examples of Predictions

We look at specific correct and incorrect predictions,to understand the limitations of the model. Examples are shown in the table.

Correct Predictions	Incorrect Predictions
Premise: Man in overalls with two horses. ***Hypothesis***: a man in overalls with two horses ***Label***: entailment ***Prediction***: entailment	***Premise***: Man in white shirt and blue jeans looking to the side while walking down a busy sidewalk. ***Hypothesis***: Man has a blue shirt on . ***Label***: contradiction ***Prediction***: entailment
Premise: Man observes a wavelength given off by an electronic device. ***Hypothesis***: The man is examining what wavelength is given off by the device. ***Label***: entailment ***Prediction***: entailment	***Premise***: Three women on a stage, one wearing red shoes, black pants, and a gray shirt is sitting on a prop, another is sitting on the floor, and the third wearing a black shirt and pants is standing, as a gentleman in the back tunes an instrument. ***Hypothesis***: There are two women standing on the stage ***Label***: contradiction ***Prediction***: entailment
Premise: Three people and a white dog are sitting in the sand on a beach. ***Hypothesis***: Three dogs and a person are sitting in the snow. ***Label***: contradiction ***Prediction***: contradiction	***Premise***: Two people are in a green forest. ***Hypothesis***: The forest is not dead. ***Label***: entailment ***Prediction***: contradiction

Looking at the incorrect predictions, it is clear that one of the main limitations of the model is that it does not understand that certain words act upon other words specifically. In the first example, the model was unable to assign colors to specific articles of clothing. Similarly, in the second example, the model is unable to assign correct quantities to the correct nouns. In both cases, this leads the model to predict entailment. In the last example, it appears that the model is unable to understand that the phrase "not dead" actually means "alive".

This limitation is primarily due to the Bag-of-Words model architecture we used. These types of models discard word order and grammar, simply aggregating the meanings of each individual word. In our case, the model sums up the embedding of every word in a sentence. In doing so, the words are unable to give meaning in the context they appeared in. The word "not" will always have the same meaning regardless of what word follows it. It is likely that more advanced, recurrent models would not suffer the same limitation.

3 Evaluation and Finetuning on MultiNLI

3.1 Best Models Evaluated on MultiNLI

The best models, one for Logistic Regression and one for the 2-layer Neural Net based classifier, are evaluated using the validation set of the MultiNLI (MNLI) dataset for each genre. The evaluation results are shown in figure 5 on the left. Each bar on the plot represents the accuracy score evaluated on the given genre using models specified by colors.

The validation predictions are not as accurate on the MNLI dataset as they are on the SNLI dataset. Comparing these two models across genres, the Neural Net based model outperforms Logistic Regression in three out of the five genres.

3.2 Best Model Finetuned on MultiNLI

The NN model is then taken to fine-tuning on each of the MNLI genres. Each genre are trained for 10 epochs using 0.0005 as the learning rate. Figure 5 shows the validation accuracy of the fine-tuned Neural Net models in comparison with the accuracy before

	telephone	fiction	slate	government	travel
lr	38.5075	43.3166	39.7206	40.2559	40.1222
nn	39.7015	44.4221	37.525	39.4685	41.5479
nn finetuned	48.5572	45.6281	44.3114	50	50.611

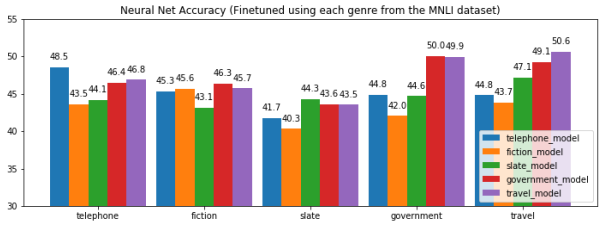

Figure 5: Left: Models validation accuracy on MNLI before and after finetuning. Right: Validation accuracy of all five fine-tuned models evaluated on each genre of the MNLI dataset (the "x" model is fine-tuned using the "x" training set).

fine-tuning. Each model is trained and evaluated using data from the corresponding genre. An increase of 1 to 11 percents are seen on each of the categories.

To further evaluate the effects on fine-tuning, we've applied the best models trained using each genre to all MNLI genres. Results are shown in figure 5 on the right. It is interesting to see that a genre's best validation accuracy does not necessarily come from the model fine-tuned using the training set of that genre. It seems that the "travel" model can carry it's advantage over to other genres, since all genres received top or second-to-the-top performance from the "travel" model. Similar, this results showed that the "government" model is doing the same for other datasets.

4 Pre-trained word embedding

We use the wiki-news-300d-1M.vec.zip which consists of 1 million word vectors trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset. The embedding dimension is 300.

4.1 Training on SNLI using pretrained word embeddings

We should set the requires_grad parameter of your embedding layer to False and don't update it during training. We get a validation accuracy of **69.1%** for our neural net classifier using frozen embeddings. The performance of our LR classifier deteriorates and we get an accuracy of **55.7%**

4.2 Models Trained with Pre-trained Word Embeddings Evaluated on MultiNLI

The LR and NN models trained using pre-trained embeddings are also evaluated using the MNLI dataset. The result is shown in figure 6.

	telephone	fiction	slate	government	travel
lr_pretrained	42.5871	39.2965	37.8244	38.878	38.391
nn_pretrained	42.3881	43.8191	41.4172	41.0433	40.0204

Figure 6: Evaluation of model using pre-trained embeddings on MNLI dataset.

5 Analyzing Learned Word Embedding

The most similar word pairs we learned from our best models of logistic regression and neural network are measure by the cosine similarity of the embedding vectors. The cosine similarity simply measures the angle between the embedding vectors of each token pairs, similar tokens suppose to have small angles between their embedding vectors. When use the same method to measure the similarities of the words in pre-trained word embedding we used in section 3.4, the results generated are fairly different. The most similar words measured by cosine similarities in for pre-trained embeddings are basically numbers. The biggest changes of the work embeddings to the fine-tuned models are: 1) government: 'required', 'ignored', 'executive', 'success', 'deal', 'senior', 'provides', 'just', 'did', 'largest' 2) fiction: 'people', 'mist', 'hit', 'paid', 'contents', 'reach', 'during', 'general', 'changing', 'hearing' 3) slate: 'full', 'operation', 'image', 'trying', 'era', 'seemingly', 'dreams', 'determine', 'players', '%' 4) telephone: 'peace', 'aim', 'expert', 'hated', 'got', 'price', 'never', 'meet', 'Donald', 'entire' 5) travel: 'nothing', 'never', 'most', 'tourists', 'their', 'wo', 'Many', 'any', 'anywhere', 'houses'.

References

[1] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[2] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426, 2017.
[3] Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan, and William Yang Wang. How large a vocabulary does text classification need? A variational approach to vocabulary selection. *CoRR*, abs/1902.10339, 2019.
[4] Zi Yin and Yuanyuan Shen. On the dimensionality of word embedding. *CoRR*, abs/1812.04224, 2018.