Sentiment Analysis on Amazon Fine Food Reviews

Hanyi Zhang
Yuchen Zeng
Yarong Liu
Stevens Institute of Technology
Hoboken, NJ 07030

Abstract

Nowadays more and more people choose to buy their food online. When they make a decision between similar items, reviews are much more important to them because these text could help them to learn much more information about this item. In our project, we try to use CNN, RNN, KNN and Seq2seq to sentiment analysis on these food reivews.

1 Introduction

In this project, we find a food reviews dataset from Kaggle. This dataset consists of reviews of fine foods from Amazon. The data span a period of more that 10 years, including all 500000 reviews up to October 2012. Reviews include product ID and user information, ratings of product, time of the review, a plain text reviews and the summary of every text reviews. There are about 568454 reviews, 256059 users, 74258 products and 260 users with 50 reviews. The purpose of this project is to predict the emotion of each reviews.

At the beginning, we analyzed the ratings of 35 products in the dataset over the last 10 years. Averaging the ratings of every products in every three months, using the linear regression of 3rd order polynomial and 4th order polynomial to fit the changes of ratings of products.These are the example pictures of two products.

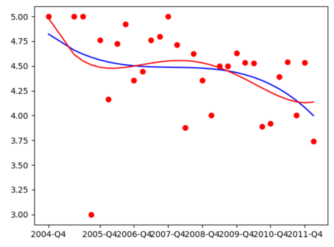


Figure 1: Product-01

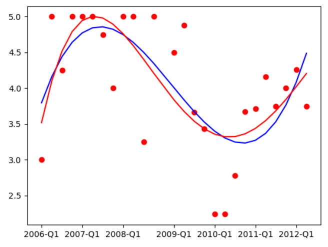


Figure 2: Product-02

2 Related Work

2.1 Stop Words

In computer science, stop words are words that are ignored by NLP(Nature language processing). Generally, stopwords are the most common words in natural language. And there are no universal stop words in NLP tools. These are some of the most common stop words, such as the, is, at, which, and on. On removing stopwords, dataset size decreases and the time to train the model also decreases. Removing stopwords can potentially help improve the performance as there are fewer and only meaningful tokens left. Thus, it could increase classification accuracy. But on the other hand, removing stopwords will also losing some information.

In our project, we are trying to find out the best stopwords list. We have tested the stopwords list of NLTK, two customized stopwords list, and None stopwords.

2.2 Length of input

During NLP, we need to Pre-processing our data, which are lines of words, and convert it to vectors with the same length. By reducing the length, we can reduce the training time. On the other hand, reducing the length will also result in losing information.

In our project, we are trying to find out the best length of vector of each review. And we have chosen the Mean, Quartile, Medium as vector length.

2.3 KNN

KNN(k-nearest neighbors algorithm) is a non-parametric method used for classification and regression. It was developed from the need to perform discriminant analysis when reliable parametric estimates of probability densities are unknown or difficult to determine. In an unpublished US Air Force School of Aviation Medicine report in 1951, Fix and Hodges introduced a non-parametric method for pattern classification that has since become known as the k-nearest neighbor rule (Fix & Hodges, 1951).

KNN can be used for classification — the output is a class membership (predicts a class — a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also be used for regression — output is the value for the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbors.

The performance of KNN is good in some cases even if it is less complex than CNN or RNN. In our project, We have implemented a KNN Classifier.

2.4 CNN and RNN

The most common use for CNN is image classification, for example identifying satellite images that contain roads or classifying hand written letters and digits. There are other quite mainstream tasks such as image segmentation ,signal processing and NLP. In our project, we will use CNN (Convolutional Neural Networks) built by keras to generate sentiment from Amazon reviews.

Recurrent Neural Network is a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. So, another model we want to use is RNN (Recurrent Neural Network). It is also built by keras.

2.5 seq2seq

In order to have better performance, we want to use a seq2seq model. Which means we want to get abstract from different reviews, because some of them are too long, so no matter CNN and RNN models will receive some redundant information and make a wrong wrong prediction.

In our project, we build a Deep Neural Networks which base on the sequence to sequence model which has two RNN parts in it, encoder model and decode model. The goal of encoder model is learn the input text and transfer them into different vectors which can represent the input and can be calculated by computer. Similarly, decoder model's task is translate these vectors into sentences which can be read by human.

2.6 technique in RNN

There are many technique which can be used in seq2seq models such as teacher forcing, attention and beam search.

Teacher forcing is a method for quickly and efficiently training recurrent neural network models that use the ground truth from a prior time step as input. It is a network training method critical to the development of deep learning language models used in machine translation, text summarization, and image captioning, among many other applications. In our project, we want to find a ratio of teacher forcing which means, some time steps use this technology, other time steps don't use it.

Attention is a mechanism that was developed to improve the performance of the Encoder-Decoder RNN on machine translation. And it is proposed as a solution to the limitation of the Encoder-Decoder model encoding the input sequence to one fixed length vector from which to decode each output time step. This issue is believed to be more of a problem when decoding long sequences. In out project, we will implement a global attention model.

3 Methods

3.1 Convolutional Neural Network

When we process the food reviews, the first thing we should do is using Word Embedding which could map words in the food reviews to vector of real number. Then we can train different kind of models to predict the emotion of each reviews.

In order to better introduce CNN, we will use an image as example. When using CNN for text, just change the image matrix to word matrix. In general, CNN includes two parts: a block of convolutional layers and a block of fully-connected layers, other than that, you can add Dropout layer if required.

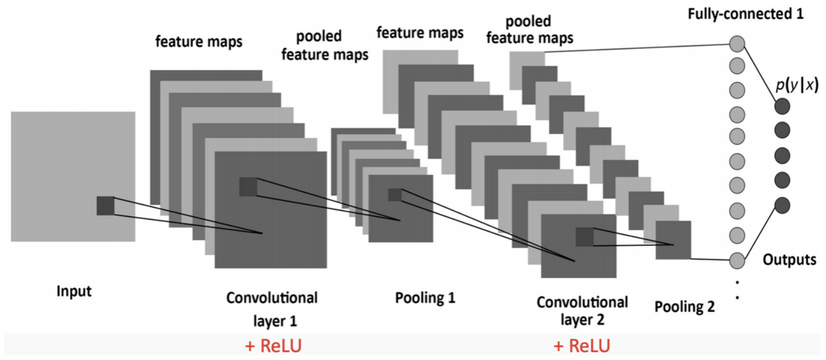


Figure 3: structure of CNN

Convolutional Layer

We analyze the influence of nearby pixels in an image by using something called a filter. A filter is exactly what you think it is, in our situation, we take a filter of a size specified by the user (a rule of thumb is 3x3 or 5x5) and we move this across the image from top left to bottom right. For each point on the image, a value is calculated based on the filter using a convolution operation. For example:

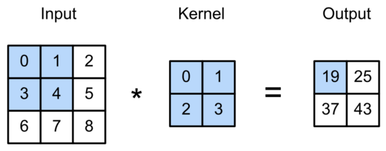


Figure 4: processing of convolutional

In this example, the input is 3*3 and the kernel window is 2*2. we begin with the convolution window positioned at the top-left corner of the input array and slide it across the input array, both from left to right and top to bottom. When the convolution window slides to a certain position, the input subarray

contained in that window and the kernel array are multiplied (element-wise) and the resulting array is summed up yielding a single scalar value. This result is precisely the value of the output array at the corresponding location.
We could formally express this idea as follows:

$$h[i,j] = u[i,j] + \sum_{k,l} W[i,j,k,l] * x[k,l] = u[i,j] + \sum_{a,b} V[i,j,a,b] * x[i+a,j+b]$$

After the filters have passed over the image, a convolutional operation is generated for each filter. These are then taken through an activation function, for example, ReLU and Sigmoid function, which decides whether a certain feature is present at a given location in the image.
We can then do a lot of things, such as adding more convolutional layers, which become more and more abstract as we create a deeper CNN.

Pooling Layer

Pooling layers could mitigating the sensitivity of convolutional layers to location and spatially downsampling representations.In theory, any type of operation can be done in pooling layers, but in practice, max pooling perform better. Here is an example of max pooling:

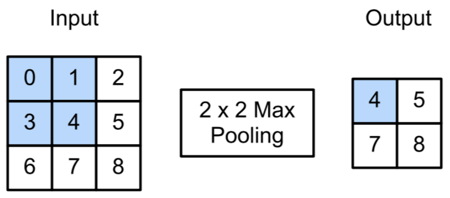


Figure 5: Max pooling

Maximum pooling with a pooling window shape of $2*2$. The shaded portions represent the first output element and the input element used for its computation: $\max(0,1,3,4)=4$

Dropout

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. In other words, dropout with drop probability p is applied as follows:

$$h' = \begin{cases} 0 & \text{with probability } p \\ \frac{h}{1-p} & \text{otherwise} \end{cases}$$

This prevents units from co-adapting too much.

3.2 KNN

Altman, N. S point out that "Nonparametric regression is a collection of techniques for fitting a curve when there is little a priori knowledge about its shape." in the paper "An introduction to kernel and nearest-neighbor nonparametric regression". It shows that KNN might be a simple and useful model in our condition.
In our project, we implement this model by sklearn. The k-nearest-neighbor classifier is

commonly based on the Euclidean distance between a test sample and the specified training samples. Let xi be an input sample with p features (x_{i1},x_{i2},...,x_{ip}) , n be the total number of input samples (i=1,2,...,n) and p the total number of features (j=1,2,...,p) . The Euclidean distance between sample xi and x_l (l=1,2,...,n) is defined as d(x_i,x_l) = $\sqrt{(x_{i1}-x_{l1})^2+(x_{i2}-x_{l2})^2+(x_{i3}-x_{l3})^2+...+(x_{ip}-x_{lp})^2}$ In our scenario, donate K as 2. We also test with different Stop-words.

3.3 RNN

LSTM is an important part of RNN.Long short-term memory (LSTM) networks were discovered by Hochreiter and Schmidhuber in 1997 and set accuracy records in multiple applications domains. (Long Short-Term Memory, 1997 volume 9).

Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data, such as reivew text. So, In our project, we implement LSTM with keras. We use keras. Tokenizer to transfer reviews into token. Then we embed it with Kearas embedding layer. We aslo add a layer of dropout to avoid over-fit.

3.4 Seq2seq

In our project, we implement this model by pytorch, and the base structure of encoder and decoder are shown as figure 6 figure 7. The first four time steps are encoder part and the other time steps are decoder part. No matter encoder or decoder, all of them are GRU model. GRUs are improved version of standard recurrent neural network. GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem which comes with a standard recurrent neural network. It can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results.

To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction. The structure of GRU is shown in figure 8.

Update gate: help the model to determine how much of the past information (from previous time steps) needs to be passed along to the future.

$$Z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{(t-1)})$$

Reset gate: decide how much of the past information to forget.

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{(t-1)})$$

Current memory content:

$$h_t^{'} = tanh(Wx_t + r_t \odot Uh_{(t-1)})$$

Final memory at current time step: The network needs to calculate h_t which holds information for the current unit and passes it down to the network. In order to do that the update gate is needed. It determines what to collect from the current memory content $h_t^{'}$ and what from the previous steps $h(t-1)$.

$$h_t = z_t \odot h_{(t-1)} + (1-z_t) \odot h_t^{'}$$

3.5 Attention

Attention was presented by Dzmitry Bahdanau, et al. in their paper "Neural Machine Translation by Jointly Learning to Align and Translate" that reads as a natural extension of their previous work on the Encoder-Decoder model. Instead of encoding the input sequence into a single fixed context vector, the attention model develops a context vector that is filtered specifically for each output time

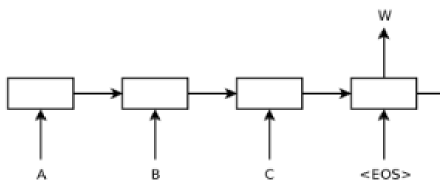


Figure 6: structure of encoder

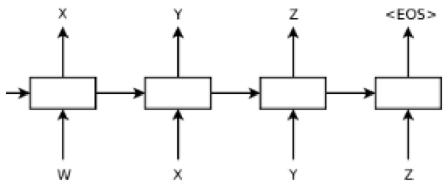


Figure 7: structure of decoder

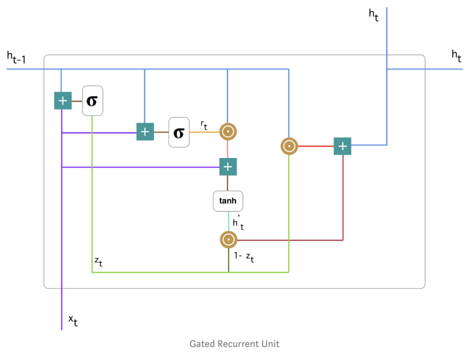


Figure 8: structure of GRU

step. There are many main attention mechanism, and we will implement one of them.
The structure of it is shown in figure 9. The ith Target word context vector c-i will be based on the hidden vector of each source word h-j weighted summation yields:

$$C_i = \sum_{j=1}^{Tx} \alpha_{ij} h_j$$

For each h-j of a-ij calculated as follows:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{Tx}(exp(e_{ik}))}$$

among them:

$$e_{ij} = a(s_{i-1}, h_j)$$

e_{ij} is the alignment model, representing the position j input and location i. The output matches the score of the degree based on the implied state of the i-1 position of the RNN s_{i-1} dnd j position h_j calculated.

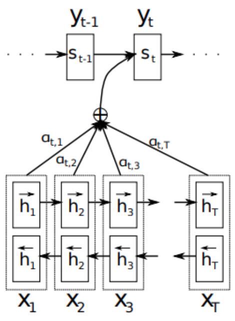


Figure 9: Structure of attention

3.6 Teacher Forcing

Teacher forcing is the technique where the target word is passed as the next input to the decoder. Figure 10 and figure 11 show the model with teacher forcing and without teacher forcing. Training with Teacher Forcing converges faster. At the early stages of training, the predictions of the model are very bad. If we do not use Teacher Forcing, the hidden states of the model will be updated by a sequence of wrong predictions, errors will accumulate, and it is difficult for the model to learn from that.

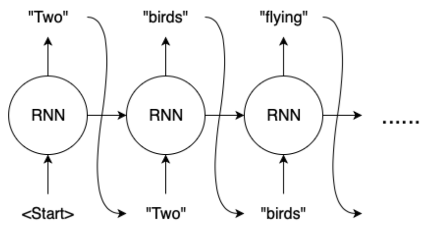


Figure 10: without teacher forcing

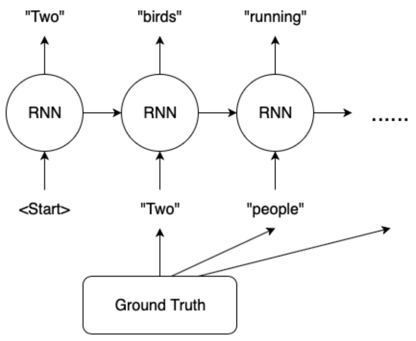


Figure 11: with teacher forcing

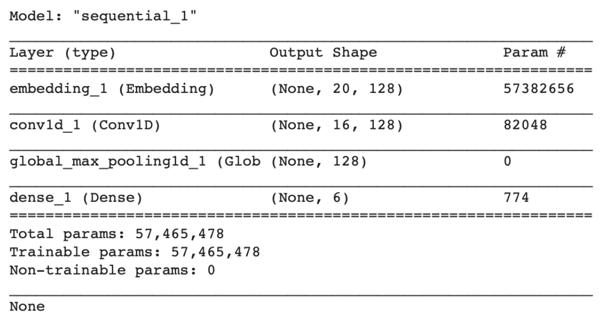
3.7 Evaluation methods

BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translations. Although developed for translation, it can be used to evaluate text generated for a suite of natural language processing tasks. In our project, we use BLEU score to evaluate our summarization model.

4 Result of models

4.1 CNN

In the project, we build an CNN model to predict the sentiment of these food reviews. Here is the structure of CNN model:



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 20, 128)           57382656
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 16, 128)           82048
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 57,465,478
Trainable params: 57,465,478
Non-trainable params: 0
_________________________________________________________________
None
```

At the beginning, we are confused about the standard length of food reviews. Should we choose longest length of texts or just choose the mean value of texts? Then we analyse the length of these texts. Here is the boxplot of the length of food reviews:

In the experiment, we choose two types of length of texts to see if there is the any difference in final accuracy between the mean of texts and the longest value of texts. But finally we find the accuracy of

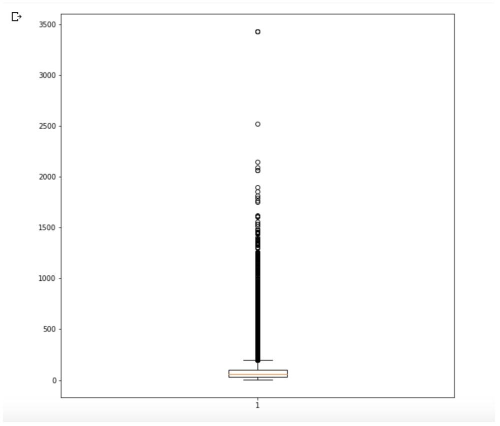
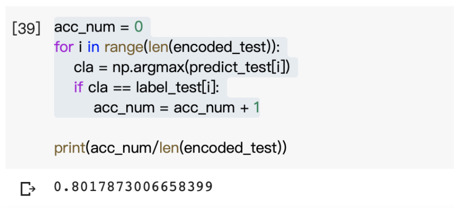


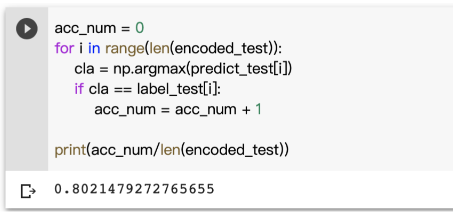
Figure 12: Length of texts

model does not improve a lot. So in the following models we just choose mean value of texts.



```
[39] acc_num = 0
     for i in range(len(encoded_test)):
         cla = np.argmax(predict_test[i])
         if cla == label_test[i]:
             acc_num = acc_num + 1

     print(acc_num/len(encoded_test))
```

```
0.8017873006658399
```



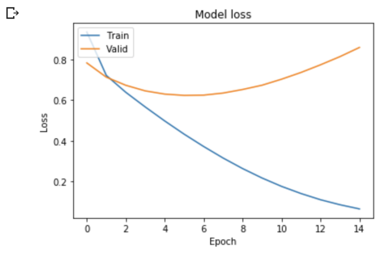
```
acc_num = 0
for i in range(len(encoded_test)):
    cla = np.argmax(predict_test[i])
    if cla == label_test[i]:
        acc_num = acc_num + 1

print(acc_num/len(encoded_test))
```

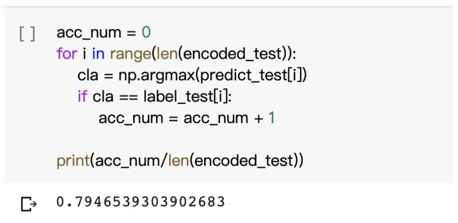
```
0.8021479272765655
```

Figure 13: mean value of texts

Figure 14: longest texts

We also use the "Stopword" to reduce some irrelevant words in texts in order to improve the accuracy of this model. Unfortunately, the result is not very well.





```
[ ] acc_num = 0
    for i in range(len(encoded_test)):
        cla = np.argmax(predict_test[i])
        if cla == label_test[i]:
            acc_num = acc_num + 1

    print(acc_num/len(encoded_test))
```

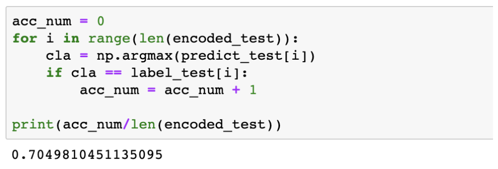
```
0.7946539303902683
```

Figure 15: Loss

Figure 16: Final result

4.2 RNN

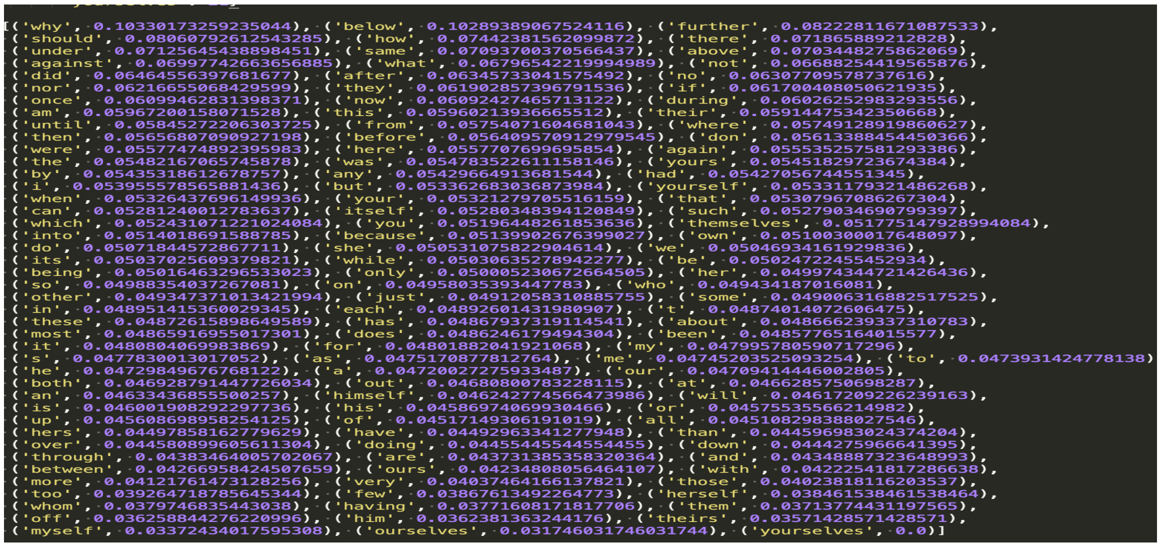
At first, the result is not so good. We use NLTK stopwords list as filter to eliminate stopwords. The Accuracy is not so good.



```
acc_num = 0
for i in range(len(encoded_test)):
    cla = np.argmax(predict_test[i])
    if cla == label_test[i]:
        acc_num = acc_num + 1

print(acc_num/len(encoded_test))
```

```
0.7049810451135095
```

At first, we think it was result from the fact that we did not implement dropout. But After we add dropout, the performance still not good.
Then we try to improve Accuracy by changing stopwords. So, we calculated the frequency of occurrence of each stopword in the review text of error prediction. We think the words with high frequency will reduce the accuracy of our prediction. So, we created a new stopword list based on the NLTK stopword list by removing stopwords that have a high frequency of occurrence in the review text of error prediction. Luckily, it works. The accuracy increased as we remove stopwords from the stopwords list. Finally, the accuracy reaches the peak after we use an empty list as stopword list.

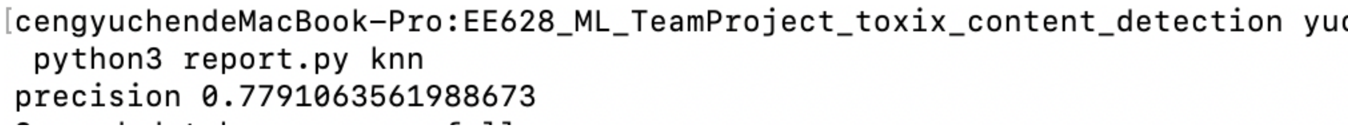


```
[('why', 0.10330173259235044), ('below', 0.10289389067524116), ('further', 0.08222811671087533),
 ('should', 0.08060792612543285), ('how', 0.07442381562099872), ('there', 0.071865889212828),
 ('under', 0.07125645438898451), ('same', 0.07093700370566437), ('above', 0.0703448275862069),
 ('against', 0.06997742663656885), ('what', 0.06796542219994989), ('not', 0.06688254419565876),
 ('did', 0.06464556397681677), ('after', 0.06345733041575492), ('no', 0.06307709578737616),
 ('nor', 0.06216655068429599), ('they', 0.06190285739679l536), ('if', 0.06170040805062193 5),
 ('once', 0.06099462831398371), ('now', 0.06092427465713122), ('during', 0.06026252983293556),
 ('am', 0.05967200158071528), ('this', 0.05960213936665512), ('their', 0.05914475342350668),
 ('until', 0.05845272206303725), ('from', 0.05754071604681043), ('where', 0.05749128919860627),
 ('then', 0.05656807090927198), ('before', 0.05640957091297954 5), ('don', 0.05613388454450366),
 ('were', 0.05577474892395983), ('here', 0.0557707699695854), ('again', 0.05553525758129338 6),
 ('the', 0.05482167065745878), ('was', 0.05478352261115814 6), ('yours', 0.05451829723674384),
 ('by', 0.05435318612678757), ('any', 0.05429664913681544), ('had', 0.05427056744551345),
 ('i', 0.05395557856588143 6), ('but', 0.05336268303687398 4), ('yourself', 0.05331179321486268),
 ('when', 0.05326437696149936), ('your', 0.05321279705516159), ('that', 0.05307967086267304),
 ('can', 0.05281240012783637), ('itself', 0.05280348394120849), ('such', 0.05279034690799397),
 ('which', 0.05243107122102408 4), ('you', 0.05196448261853636), ('themselves', 0.05177514792899408 4),
 ('into', 0.0514018691588785), ('because', 0.05139902676399027), ('own', 0.05100300017648097),
 ('do', 0.05071844572867711), ('she', 0.05053107582290461 4), ('we', 0.05046934161929836),
 ('its', 0.05037025609379821), ('while', 0.05030635278942277), ('be', 0.05024722455452934),
 ('being', 0.05016463296533023), ('only', 0.05000523067266450 5), ('her', 0.04997434472142643 6),
 ('so', 0.04988354037267081), ('on', 0.04958035393447783), ('who', 0.04943418701608 1),
 ('other', 0.04934737101342199 4), ('just', 0.04912058310885755), ('some', 0.04900631688251752 5),
 ('in', 0.04895141536002934 5), ('each', 0.04892601431980907), ('t', 0.04874014072606475),
 ('these', 0.04872615898649589), ('has', 0.04867937319114541), ('about', 0.04866623933731078 3),
 ('most', 0.04865916955017301), ('does', 0.0486246179494304), ('been', 0.04857765164015577),
 ('it', 0.0480804069983869), ('for', 0.04801882041921068), ('my', 0.04799578059071729 6),
 ('s', 0.04778300130170 52), ('as', 0.04751708778127 64), ('me', 0.04745203525093254), ('to', 0.0473931424778138)
 ('he', 0.04729849676768122), ('a', 0.04720027275933487), ('our', 0.04709414446002805),
 ('both', 0.04692879144772603 4), ('out', 0.04680800783228115), ('at', 0.04662857506982 87),
 ('an', 0.04633436855500257), ('himself', 0.04624277456647398 6), ('will', 0.04617209226239163),
 ('is', 0.04600190829229773 6), ('his', 0.04586974069930466), ('or', 0.04575535566214982),
 ('up', 0.04560869895825412 5), ('of', 0.04517149306191019), ('all', 0.04510829838802754 6),
 ('hers', 0.04497858162779629), ('have', 0.04492963341277948), ('than', 0.04459698302437420 4),
 ('over', 0.04458089960561130 4), ('doing', 0.04455445544554455), ('down', 0.0444275966641395),
 ('through', 0.04383464005702067), ('are', 0.04373138535832036 4), ('and', 0.04348887323648993),
 ('between', 0.04266958424507659), ('ours', 0.04234808056464107), ('with', 0.04222541817286638),
 ('more', 0.04121761473128256), ('very', 0.04037464166137821), ('those', 0.04023818116203537),
 ('too', 0.03926471878564534 4), ('few', 0.03867613492264773), ('herself', 0.03846153846153846 4),
 ('whom', 0.0379746835443038), ('having', 0.03771608171817706), ('them', 0.03713774431197565),
 ('off', 0.03625884427622099 6), ('him', 0.0362381363244176), ('theirs', 0.03571428571428571),
 ('myself', 0.03372434017595308), ('ourselves', 0.031746031746031744), ('yourselves', 0.0)]
```

The Accuracy is 78.7080%

4.3 KNN

In our project, we have implement it with NLTK stopwords list, Customized stop words, and none stopwords. We find that the less the stopwords list, the higher the accuracy is.We figure out that when we donate that K = 2 and without stopwords the performance will reach the peak. The Accuracy reach peak 77.91%.



```
[cengyuchendeMacBook-Pro:EE628_ML_TeamProject_toxix_content_detection yuc
  python3 report.py knn
 precision 0.7791063561988673
```

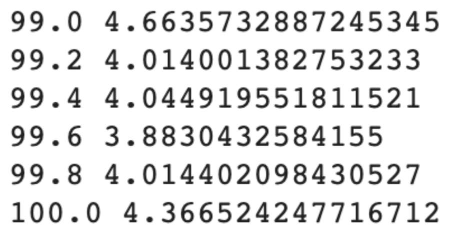
4.4 seq2seq

Loss is shown in figure 17 and the final loss is 4.36, the left column is process bar. Total BLEU score is shown in figure 18, it is 0.7, not a bad score. Finally, the total accuracy about sentiment is shown in figure 19, it is 0.83. Here are some examples about what the machine generate from original text. First two is right and the third one has different sentiment to label's sentiment which is negative.

(1). text: 13 dollars for 24 cups free shipping over 50 dollars is the going rate. These folks are pulling a swiftie. label: beware sum: beware<EOS>

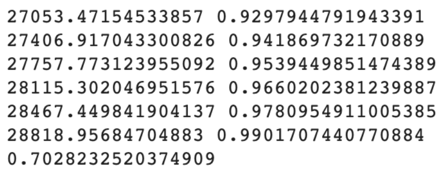
(2). text: If your dog likes Naked bars then this is the same consistency. It's hard and sometimes my dog will eat it sometimes he'll... label: Not happy about this sum: Not as good as what I find <EOS>

(3). text:I ordered the ranch flavor and when I received them they had a sell by date of September of 2008!!! label: Dont Buy!! sum: Great Taste<EOS>



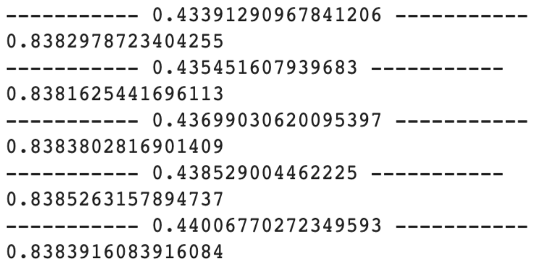
```
99.0 4.6635732887245345
99.2 4.014001382753233
99.4 4.044919551811521
99.6 3.8830432584155
99.8 4.014402098430527
100.0 4.366524247716712
```

Figure 17: loss



```
27053.47154533857 0.9297944791943391
27406.917043300826 0.941869732170889
27757.773123955092 0.9539449851474389
28115.302046951576 0.9660202381239887
28467.449841904137 0.9780954911005385
28818.95684704883 0.9901707440770884
0.7028232520374909
```

Figure 18: BLEU score



```
----------- 0.43391290967841206 -----------
0.8382978723404255
----------- 0.435451607939683 -----------
0.8381625441696113
----------- 0.43699030620095397 -----------
0.8383802816901409
----------- 0.438529004462225 -----------
0.8385263157894737
----------- 0.44006770272349593 -----------
0.8383916083916084
```

Figure 19: accuracy

5 Conclusion

After train and test models that we mentioned previously, CNN got the best performance. RNN and KNN are little bit lower than CNN. Although seq2seq method got a better result, but it only test on a part of whole dataset.

6 Future work

In our project, we only set teacher forcing parameter is 0.5 which means not all the time steps use ground truth. In other words, we can try to find a better ratio of it to get better performance.

Another thing is, we can implement beam search in NLG process which can generate better sentences.

References

[1] Minh-Thang Luong. & Hieu Pham & Christopher D. Manning(2015) Effective Approaches to Attention-based Neural Machine Translation. Computer Science Department, Stanford University, Stanford, CA 94305

[2] Dzmitry Bahdanau & KyungHyun Cho & Yoshua Bengio(2016) neural machine translation by jointly learning to align and translate.

[3] Kishore Papineni, Salim Roukos & Todd Ward (2002) BLEU: a Method for Automatic Evaluation of Machine Translation. IBM T. J. Watson Research Center Yorktown Heights, NY 10598, USA

[4] Hochreiter, Sepp; Schmidhuber, Jürgen (1997-11-01). "Long Short-Term Memory". Neural Computation.

[5] Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician. 46 (3): 175–185.