

Question 1

1. What range of values for C did you try? Explain, why this range is reasonable. Also explain what search technique you used and why it is appropriate here.

Range: [0.1, 0.15, 0.2, 0.5, 1, 2, 5, 10], because C indicates the inverse of regularization strength, that is, $C = 1/\lambda$. Therefore, the loss function with L2-penalty will be $loss = (pred - true)^2 + \frac{1}{C} \sum \theta_i^2$, then C must be a positive float, that's why my range is positive. From the equation we can see smaller C value specify stronger regularization, much penalty to the model's weights which contribute to overfitting. Since these 35000 data may not be fully representative of the real-world data, we don't want overfitting, so a small range of C is my choice.

I used grid search, which trains and tests every combination of parameters to find the best one. And because this task only asks for finding one parameter "C", there are no combinations, therefore using grid search won't take too much time and will be more accurate.

2. What was the best performing C value?

0.1

3. What was your final accuracy?

85.71%

Question 2

PS: the code of my best solution is in "genre_classifier_1.py" about using pretrained transformer, I also hand in the one I tried to build my own model and implement my own token masking preprocessing in "genre_classifier_2.py" since I spent most of time on it.

1. How your final solution works

I use BERT transformer, I tokenize the train_data with the pretrained BERT tokenizer and zeros padding, and then just use the pretrained BERT transformer for sequence classification, because "sentence to genre" is just sequence classification. Furthermore, answered in sub question 4.

2. How you trained and tested your model (e.g. validation split(s), hyperparameter search ...)

The training set is too small compares to the test set, moreover the test set labels are uneven without lots of label-1(science fiction), so I decide not using validation split for any early-stopping or learning_rate_decreasing purpose, though I already tried few times with almost futile efforts. Therefore, I prefer to not waste any training data, use the whole training set to

train the model. Since I use pre-trained transformer from the library transformers, so I do not have too much hyperparameter to test, I only try to set the batch_size to 32 (relatedly ok compared to the size of the small training set) because the default batch_size is 8. I do so because I want the gradient descent to happen in the better direction smoothly.

3.What models you tested, which worked, and which didn't

I knew that transformer might be the best model for the purpose of this task, however I still try to build many models myself, one of them is a model of concatenation of MLP and Bi-LSTM. It uses TF-IDF vector input for MLP and embedding input for LSTM, then use these two outputs as input to the final MLP. I also mask the training set in case of predicting unknown tokens. So my training set contain the original one and the masked one. The result I got without anything pretrained is not that high, mostly around 0.59~0.6.

Using normal RNN model without pretrained embedding or using only TF-IDF method doesn't work well, mostly get score below 0.5.

I stop trying anything outside transformer due to it cost lot of time on training, then I start using transformer with pretrained materials.

4.Why you think these other models didn't work

For example, TF-IDF. The vocabulary is large, which results in each vectors has lots of zeros. Because there are only less than 5% of vocabulary appear in each document. As a high dimension input to ML model, it is not good to have so many empty features, it will decrease the performance of the model.

Models with RNN cannot beat transformers in this task, I think its because RNN training is serial, that is, the current word has to be passed into the RNN cell before the later words being passed to RNN cell. And RNN has problem about relations between tokens, for example in this sentence "I am a computer science student", relationship between "I" and "am" is very close, but between "I" and "student" is relatively far. Transformer uses Attention mechanism, the training is parallel, that is, all words are trained at the same time, so it will not have the problem about long sentence as RNN has, and it has also increase calculation efficiency. Because parallel training ignores the order relationship between the tokens, transformer also uses positional encoding to understand the order of the tokens. Transformer use the combination of QKV(query, key, value) instead of normal embedding, it bring the words from the spatial space to another much efficient space.

Question 3

Epoch: [1/10], loss: 2.8788, val_loss: 2.9343.

Epoch: [2/10], loss: 2.6151, val_loss: 2.6948.

Epoch: [3/10], loss: 2.4799, val_loss: 2.5827.

Epoch: [4/10], loss: 2.3996, val_loss: 2.5244.

Epoch: [5/10], loss: 2.3469, val_loss: 2.4894.

Epoch: [6/10], loss: 2.3155, val_loss: 2.4660.

Epoch: [7/10], loss: 2.2909, val_loss: 2.4471.
Epoch: [8/10], loss: 2.2707, val_loss: 2.4327.
Epoch: [9/10], loss: 2.2521, val_loss: 2.4203.
Epoch: [10/10], loss: 2.2350, val_loss: 2.4088.
Argmax: Jon Brich
Random:
Adars Maskelnom
Bocex Burle
Rilices Crannun
Bran Alices
Delk We Dobgino
Conthuiy Mecborid
Ere Rims
Asrard Horrgt
Macet McAgich
Deran Cuwehn

My additional discover is that, the GRU hidden layer size is very important for the linear separability of the RNNLM. Because the vocabulary size is 66, 1 layer GRU hidden layer size is better be much higher than vocabulary size, or it will be hard to get good performance or slow to converge for the RNNLM.

For example, as shown in below result, the model learns that a 'space' has meaningful meaning, however, its performance after such a token is so bad,

```
Argmax based on first two random chars after 10 epoches:
```

```
Quint Martin  
Ken Martin  
Tom Martin  
John Martin  
Vic Martin  
Nick Martin  
Harry Martin  
Chris Martin  
Ian Martin  
Oliver Martin
```

And if I keep training after 100 epoches, I get below result, "Stenner, Sten, Stennerg, Sterberg" shows that the linear separability is starting to be better. This dataset is a bit large to train, using a small training dataset to train 1000+ epoches shows that it will eventually be random on the names after first 'space'.

Argmax based on first two random chars after 100 epoches:

Random:

Lars Stenner
Randa Stenner
Tommy Stenner
Dan Stenner
Ola Hardson
Elina Sten
Karina Stennerg
Stefan Sterberg
Carl Stennerg
Hans Sten

Result for GRU hidden layer size = 1024

Argmax based on first two random chars after 1 epoch with gru_size=1024

Gary Patterson
Yanny Castillo
Brian Hansen
Olan Harris
Larry McCarthy
Dave Harris
Zach Marting
Francisco Castro
Paul Harrison
Rick Jackson

Q2 part a

AB

Return value: True

Dependencies: [(0, 1), (1, 2)]

CBA

Return value: True

Dependencies: [(2, 1), (3, 2), (0, 3)]

ABC

Return value: True

Dependencies: [(0, 1), (1, 2), (2, 3)]

ABCBC

Return value: True

Dependencies: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)]

BAC

Return value: False

Dependencies: [(2, 1), (0, 2)]

Q2 part b

Action: ['right', 'right']

Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'b_buffer']]

Success case 1

Sentence: [ROOT] A man dressed in a tux holds a violin .

Actions: ['shift', 'left', 'shift', 'right', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'reduce', 'left', 'right', 'shift', 'left', 'right', 'reduce', 'right']

Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'man_buffer'], [['[root]_stack', 'man_buffer']]

Success case 2

Sentence: [ROOT] A small cheese pizza is hot on the pan .

Actions: ['shift', 'shift', 'shift', 'left', 'left', 'left', 'shift', 'left', 'right', 'right', 'reduce', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'right']

Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'small_buffer'], ['small_stack', 'cheese_buffer']]

Success case 3

Sentence: [ROOT] A woman at a restaurant sits in front of a finished plate .

Actions: ['shift', 'left', 'shift', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'left', 'right', 'right', 'right', 'right', 'shift', 'shift', 'left', 'left', 'right', 'reduce', 'reduce', 'reduce', 'reduce', 'right']

Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'woman_buffer'], [['[root]_stack', 'woman_buffer']]

Fail case 1

['[ROOT]', 'A', 'doughnut', 'that', 'a', 'person', 'is', 'using', 'to', 'see', 'out', 'of', '.']

Actions: ['shift', 'left', 'right', 'shift', 'shift', 'left', 'shift', 'shift', 'left', 'left', 'shift', 'shift', 'left', 'right', 'right', 'right', 'reduce', 'reduce', 'reduce', 'shift']

Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'doughnut_buffer'], [['root]_stack', 'doughnut_buffer']]

Parser Dependencies:(['[ROOT]', 'doughnut')('doughnut', 'A')('person', 'a')('using', 'person')('using', 'is')('using', 'see')('see', 'to')('see', 'out')('out', 'of')

Ground Truth Dependencies:(['[ROOT]', 'doughnut')('doughnut', 'A')('doughnut', 'using')('doughnut', '.')('person', 'a')('using', 'person')('using', 'is')('using', 'see')('see', 'that')('see', 'to')('see', 'out')('out', 'of')

Fail case 2

['[ROOT]', 'The', 'waters', 'are', 'very', 'choppy', 'today', 'which', 'is', 'making', 'it', 'hard', 'to', 'ride', '.']

Actions: ['shift', 'left', 'shift', 'left', 'right', 'shift', 'left', 'right', 'reduce', 'right', 'shift', 'shift', 'left', 'left', 'shift', 'shift', 'left', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'shift']

Features: [['[root]_stack', 'the_buffer'], ['the_stack', 'waters_buffer'], [['root]_stack', 'waters_buffer']]

Parser Dependencies:(['[ROOT]', 'are')('waters', 'The')('are', 'waters')('are', 'choppy')('are', 'today')('choppy', 'very')('making', 'which')('making', 'is')('making', 'hard')('hard', 'it')('hard', 'ride')('ride', 'to')

Ground Truth Dependencies:(['[ROOT]', 'are')('waters', 'The')('waters', 'making')('are', 'waters')('are', 'choppy')('are', 'today')('are', '.')('choppy', 'very')('making', 'which')('making', 'is')('making', 'hard')('hard', 'it')('hard', 'ride')('ride', 'to')

Fail case 3

['[ROOT]', 'A', 'woman', 'holding', 'three', 'bags', 'stands', 'by', 'the', 'open', 'doors', 'of', 'a', 'subway', 'car', ',', 'through', 'which', 'other', 'people', 'can', 'be', 'seen', '.']

Actions: ['shift', 'left', 'shift', 'right', 'shift', 'left', 'right', 'reduce', 'reduce', 'left', 'right', 'right', 'shift', 'shift', 'left', 'left', 'right', 'right', 'shift', 'shift', 'left', 'left', 'right', 'reduce', 'reduce', 'reduce', 'reduce', 'right', 'shift', 'right', 'shift', 'left', 'shift', 'shift', 'shift', 'left', 'left', 'left', 'reduce', 'left', 'shift', 'shift']

Features: [['[root]_stack', 'a_buffer'], ['a_stack', 'woman_buffer'], [['root]_stack', 'woman_buffer']]

Parser Dependencies:(['[ROOT]', 'stands')('woman', 'A')('woman', 'holding')('holding', 'bags')('bags', 'three')('stands', 'woman')('stands', 'by')('stands', ',')('by', 'doors')('doors', 'the')('doors', 'open')('doors', 'of')('of', 'car')('car', 'a')('car', 'subway')('through', 'which')('people', 'other')('seen', 'through')('seen', 'people')('seen', 'can')('seen', 'be')

Ground Truth Dependencies:(['[ROOT]', 'stands')('woman', 'A')('woman', 'holding')('woman', 'seen')('holding', 'bags')('bags', 'three')('stands', 'woman')('stands', 'by')('stands', ',')('stands', '.')('by', 'doors')('doors', 'the')('doors', 'open')('doors', 'of')('of', 'car')('car', 'a')('car', 'subway')('through', 'which')('people', 'other')('seen', 'through')('seen', 'people')('seen', 'can')('seen', 'be')

Total fails: 9, Total success: 19991

The underlying reason for fail cases is: whenever there is a Shift-reduce ambiguity, the oracle prediction will always be shift.

Q3

1. What pre-processing did you do?

Filter out the entities who are not Person and GPE.

Filter out outlier, e.g. `train_data[3655]` has 885 PERSON entities and 187 GPE entities, almost 3000 entities in total.

Split the dataset into training and validation set. Create dictionaries for pos-tag, dependency, and vocabulary (lemma version). Vocabulary dictionary has a frequency threshold of 10 occurrences.

Create a vocabulary of entity names, because I notice in Piazza that this dataset is particular, I think entity names should be useful as feature, I don't want to remove names from dictionary only because of they occur less than 10 times. However, my computer doesn't have enough memory for this purpose on a large dataset, so I only make the codes become comments.

After creating dataset, the first index of a single input is for indicating <UNK> of entity name, which is designed for the purpose of "(Julie Bishop, Australia) case" in A3 instructions.pdf

2. How did you split the data into training and validation sets?

I notice that `len(train_data)` is 20334, `len(test_data)` is 1340, so split 10% to validation set is ok. I split it using `train_test_split()` from `sklearn.model_selection` with `shuffle=True`.

3. How did you choose hyperparameters?

There aren't much hyperparameters for this dataset and this task to make big difference, so I only choose to set the "C" to 100.

4. What features did you choose and why did you choose these features?

Lemma: normalize the tokens and reduce the vocabulary size to some extent.

I use bag of words between entity, but the input shape will be too large, so I choose only the lemmas of dependency path from person entity to ROOT, and the lemmas of dependency path from GPE entity to ROOT. I do not use one-hot for each word, I use counting instead, which mean it is the element-wise sum of the one-hot vectors in a sentence.

`dep_head`: for search the dependency path.

Pos-tag, dependency: sometimes it's hard to know how a word is being used within the scope of a sentence, both features provide linguistic signal on this problem.

5. What post-processing did you do

Filter out the data whose relations are not `"/people/person/nationality"`.

Remove some false relation samples, since this dataset has only one relation for each data sample even if there were 100 person and 100 GPE. (`train_data[3655]` contributes 40,000+

true relation sample but only one false relation sample) Remove them in order to handle the balance of false and true relation sample, else the model won't be stable or more reply on the false relation samples.