

Personality Detection on Persian Dataset

Phase 2

Mohammadjavad Mehditabar - 98522049

NLP Phase 2



July 12, 2023

Contents

1	Commands and Project Structure	2
2	Word2Vec	2
2.1	Implementation Approach	2
2.2	results	2
2.2.1	word vector differences	2
2.2.2	checking bias between two traits	3
3	tokenization	3
3.1	Implementation Approach	3
3.2	Results	3
4	Language Model	4
4.1	Implementation Approach	4
4.2	Results	4
5	Feature Engineering	6
5.1	Implementation Approach	6
5.2	Results	7
5.2.1	loss	7
5.2.2	accuracy	7
6	Model Architecture	8
6.1	Implementation Approach	8
6.1.1	Word2Vec + ParsBERT	8
6.1.2	BERT Truncated	8
6.1.3	BERT Into BERT	10
7	Data Augmentation	11
7.1	Implementation Approach	11
7.2	Results	12
8	Zero Shot Classification by GPT	12
8.1	Implementation Approach	12

1 Commands and Project Structure

Please refer to [GitHub repository](#) of this project, and follow guidelines on phase 2 to run this project. I have also explained how each sub module works.

2 Word2Vec

2.1 Implementation Approach

I developed a word2vec model that works based on skip-gram method and use negative sampling loss to update its weights. vectors are 10 dimensional and context window is 5. number of iteration was 60.

model has been trained on each 4 traits that each trait has 2 label individually, thus at the end of training we have 8 models. models are saved in *models* directory under name convention "*{trait}_{num_trait}.word2vec.npy*".

during training it was also needed to keep tokens because after that we need to query *token* dictionary and extract the idx of each token. tokens of this 8 available models are saved into "*experiments/word2vec/{trait}_{num_trait}_tokens.json*".

And in the end for further uses I generated a model that was trained on whole dataset. this model will be queried as feature extractor for a word. this model name is "*all.word2vec.npy*".

2.2 results

we will discuss about "E" and "I" trait and consider them representing our other models. we will dig through what biases our model has and how are the common words are differed from each other by their vector differences in these 2 traits. word vector differences are measured by *cosine similarity* metric.

2.2.1 word vector differences

0	
-0.25134	شما
0.07524	خونه
-0.04506	اتفاق
-0.07823	هيچان
-0.02406	تنهائي

The figure above shows cosine similarty between word vectors I , E for each selected word. as we see the word "خونه" which doesnt matter for Intorvert person or Extrovert person is being more similar to their context and they use this word in the same context. but the words "شما" is the most dissimialr word that means this word doesnt happens to be in the same context for Introvert and Extrovert person. which is reasonable because Introvert person would rather to use this word more and contact other person with this word but Extrovert person prefer to use the word "تو" more.

Other words such as "اتفاق ، هیجان" are clearly used in a different way for these 2 trait. Introvert would less use this word but Extrovert rather to use more.

2.2.2 checking bias between two traits

زن-ترس	مرد-ترس	
0.1274	-0.30583	I
0.21861	0.3789	E

The figure above shows *cosine similarity* between words "مرد ، ترس" and words "زن و ترس" in both traits *I*, *E*. As we can see there is high dissimilarity in *I* trait between the word "مرد" and the word "ترس". that this represent that as our common knowledge the word "fear" and the word "man" happens to be less relatable but women are more similar with the word "fear". But in the *E* trait surprisingly this fact doesn't hold that, in fact the contrary of this assumption is being shown. Therefore we can deduce that in *E* trait we don't have bias in word vectors or bias happens in an opposite way.

3 tokenization

3.1 Implementation Approach

tokenization process has been implemented by *SentencePiece* tokenizer which is improved version of *WordPiece* tokenizer. In fact this tokenizer achieved more accuracy since languages which don't use *space* in their sentences to separate words are hard to be tokenized. So *SentencePiece* uses "_" character to represent spaces and act with total context in the same way.

Training this tokenizer has implemented with their [official library](#). And for becoming more accurate we used *KFold Cross Validation* and we also considered different *vocab size* to make a total conclusion which model's performance is better. Model performances are assessed by computing ratio of "UNK" token on each fold and vocab size separately. models are saved into "models/fold_{num_fold}_v_{vocab_size}".

3.2 Results

results based on each fold and vocab size separately. I have also computed average on all folds for better decision making.

	vocab-size-1000	vocab-size-2000	vocab-size-3000	vocab-size-4000
fold-1	9.867	9.732	9.595	9.459
fold-2	9.867	9.732	9.595	9.46
fold-3	9.866	9.731	9.593	9.457
fold-4	9.87	9.739	9.606	9.473
fold-5	9.867	9.732	9.595	9.46
all-average	9.867	9.733	9.597	9.462

we see that as vocab size grows, small improvement on *UNK* tokens happens. and this is reasonable because if vocab size increases, it can accomodate more vocab inside it so less *UNK* tokens appear.

we can also take a look at the average on all fold in a scatter plot shown below:

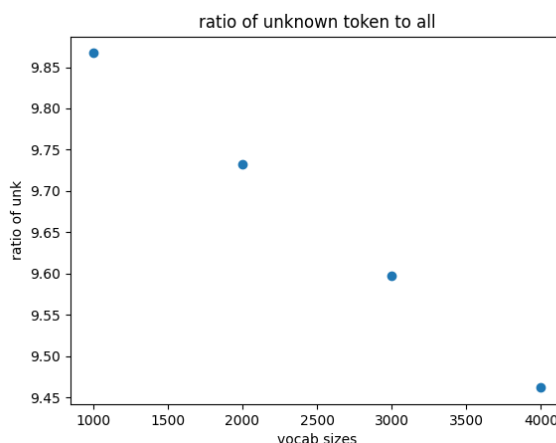


Figure 1: UNK token ratio

4 Language Model

4.1 Implementation Approach

This section tries to fine tune our dataset on "*GPT2-Medium*" and after fine tuning models will be saved. The saved model is then used to generate a text on each labels.

At first I was continuing with default *GPT2-Medium* but it didn't created a meaningful sentence then I tried to use some persian model that I came up with *GPT2-Persian* that would result in much better generation. models are saved into *models/[IE].language_model*. I will just cover the first trait *I,E* with more details.

4.2 Results

The examples of each labels are generated and then saved under the *stats/language_model* folder. So you can check their meaningfulness. I will get through the *I_example.txt* and *E_example.txt*. As we know recongonzing personality detection is such a hard task for even psychologist. for example with one or five sentences nobody can tell whether this person is Intorvert or Extrovert. But as much as I can, I'll check the generated sentences. we know that Extrovert people are more sociable and they tend to more interact with other people on the other hand Introvert people tend to be more isolated and don't like to communicate with other people and they are shy person.

July 12, 2023

۴۰۴. جن البته اینا همه در راستای این هستن طرف میاد دو بار میاد یک پست بذاره بعد به دفعه با به حرکت دیگه میاد تو اتاق کاروبار باز می کنه دیگه باز نمیشه در تمام ۴۰۵. رزمینه داره نمیدونم چند واسطه خیلی خوشگله و این همه علفه تو این کاروبار فعال این فن اکانت مین کارو میکنه متاسفانه مواردی که گفتنم یلده مقصر تفحص ندادم وگرنه یکی از متاسفانه برای همه منال دختر قابل استفاده چه زن باشند چه مرد هر کسی در جایگاه خاص جوهرش خواهد داشت واقعا فکر نمیکردم شما حتی به کاربر دیگه باشید البته اگر فکر میکنید به راحتی میتونید با خودت بحث کنید پس وای این چه وضعه حد منظم هر سال یکی که فالو میکنم تا دو سه هفته و یکی بیشتر فالو پاک شده اما بعد چند روز که برمیگردم به حالت قبل دوباره فالوایم رو چک می کنن حالا دیگه برای من نفر هم حداقل باید دو بار اینکارو بشه پس واقعا باید این قضیه حل شه یا راهی هست یا اکانت اینستا داشته باشی برای لایو مکن بگیر به عکس از غذا سفارش بدیم عکاشو بده تا من براتون توپیت بزنم حد فکر کنم حق میدم کسی به سوت کامنتا بزاره این وسط حق داریم از بقیه کمک بگیریم حد در جواب اینا شما فرض ۴۰۶. کسب من بخوام بزم در معین لحظه توپیت کنم بچشم اون شخص چی توپیت من ریلای شه به توپیتش اگر ریلای ش کرد از کی به اکانت من ریلای شده در حالت خوش ۴۰۷. و منم دلم برای خودم پیر میکنم ۴۰۸ سال با معجزن حد بی اهمیت زنگی کردیم دیگه از هر اتفاق جدیدی که رخ میده ترسم می ترسم نمیدونم دلیلش چیه اما واقعا الان بگو ۴۰۹. میلیون دلاری شده بوده واقعا تو چه شرایطی میرسن با هزار تومن جنس ایرانیه بگیرن آخه به نگاه هم باید بهش می شد من خودم چند بار گفتم اگر به کار انجام میدن ۴۱۰. کمال اون انجام نداده باشه نمیشه بهش گفت تو باید باهاش معامله کنی اما چون هیچ اطلاعی درباره اید نداشت خودش شروع کرد فکر نکنم واقعا به ۴۱۱. ۴۱۲. ن ولی من تو خیلی چیزا مشکل داشتم با کسی که رزاد باهاش معاشرت کردی ولی چون رزاد بهش توجه میکنه راحت میشه رفت با تو صحبت میکنم بعد دوساره میام با تو تا با ۴۱۳. دوست درست در مقابل طرف خود کنار بیایم به نظرت از کی به خودمون ترسیدیم ما از مومن بقله اولیمون رو برمیگردونیم شاید به مدت کوتاهی ولی باز به یای خودت فقط برای یکی از به معجزن فردی وجود داره نمیدونم ولی اون همیشه خودش رو به اون اندازه محدود میکنه برای بعضی ها فرق ندازه ولی بقیه نباید با هر تغییریه در وضع ۴۱۴. ۴۱۵. ۴۱۶. ۴۱۷. ۴۱۸. ۴۱۹. ۴۲۰. ۴۲۱. ۴۲۲. ۴۲۳. ۴۲۴. ۴۲۵. ۴۲۶. ۴۲۷. ۴۲۸. ۴۲۹. ۴۳۰. ۴۳۱. ۴۳۲. ۴۳۳. ۴۳۴. ۴۳۵. ۴۳۶. ۴۳۷. ۴۳۸. ۴۳۹. ۴۴۰. ۴۴۱. ۴۴۲. ۴۴۳. ۴۴۴. ۴۴۵. ۴۴۶. ۴۴۷. ۴۴۸. ۴۴۹. ۴۵۰. ۴۵۱. ۴۵۲. ۴۵۳. ۴۵۴. ۴۵۵. ۴۵۶. ۴۵۷. ۴۵۸. ۴۵۹. ۴۶۰. ۴۶۱. ۴۶۲. ۴۶۳. ۴۶۴. ۴۶۵. ۴۶۶. ۴۶۷. ۴۶۸. ۴۶۹. ۴۷۰. ۴۷۱. ۴۷۲. ۴۷۳. ۴۷۴. ۴۷۵. ۴۷۶. ۴۷۷. ۴۷۸. ۴۷۹. ۴۸۰. ۴۸۱. ۴۸۲. ۴۸۳. ۴۸۴. ۴۸۵. ۴۸۶. ۴۸۷. ۴۸۸. ۴۸۹. ۴۹۰. ۴۹۱. ۴۹۲. ۴۹۳. ۴۹۴. ۴۹۵. ۴۹۶. ۴۹۷. ۴۹۸. ۴۹۹. ۵۰۰. ۵۰۱. ۵۰۲. ۵۰۳. ۵۰۴. ۵۰۵. ۵۰۶. ۵۰۷. ۵۰۸. ۵۰۹. ۵۱۰. ۵۱۱. ۵۱۲. ۵۱۳. ۵۱۴. ۵۱۵. ۵۱۶. ۵۱۷. ۵۱۸. ۵۱۹. ۵۲۰. ۵۲۱. ۵۲۲. ۵۲۳. ۵۲۴. ۵۲۵. ۵۲۶. ۵۲۷. ۵۲۸. ۵۲۹. ۵۳۰. ۵۳۱. ۵۳۲. ۵۳۳. ۵۳۴. ۵۳۵. ۵۳۶. ۵۳۷. ۵۳۸. ۵۳۹. ۵۴۰. ۵۴۱. ۵۴۲. ۵۴۳. ۵۴۴. ۵۴۵. ۵۴۶. ۵۴۷. ۵۴۸. ۵۴۹. ۵۵۰. ۵۵۱. ۵۵۲. ۵۵۳. ۵۵۴. ۵۵۵. ۵۵۶. ۵۵۷. ۵۵۸. ۵۵۹. ۵۶۰. ۵۶۱. ۵۶۲. ۵۶۳. ۵۶۴. ۵۶۵. ۵۶۶. ۵۶۷. ۵۶۸. ۵۶۹. ۵۷۰. ۵۷۱. ۵۷۲. ۵۷۳. ۵۷۴. ۵۷۵. ۵۷۶. ۵۷۷. ۵۷۸. ۵۷۹. ۵۸۰. ۵۸۱. ۵۸۲. ۵۸۳. ۵۸۴. ۵۸۵. ۵۸۶. ۵۸۷. ۵۸۸. ۵۸۹. ۵۹۰. ۵۹۱. ۵۹۲. ۵۹۳. ۵۹۴. ۵۹۵. ۵۹۶. ۵۹۷. ۵۹۸. ۵۹۹. ۶۰۰. ۶۰۱. ۶۰۲. ۶۰۳. ۶۰۴. ۶۰۵. ۶۰۶. ۶۰۷. ۶۰۸. ۶۰۹. ۶۱۰. ۶۱۱. ۶۱۲. ۶۱۳. ۶۱۴. ۶۱۵. ۶۱۶. ۶۱۷. ۶۱۸. ۶۱۹. ۶۲۰. ۶۲۱. ۶۲۲. ۶۲۳. ۶۲۴. ۶۲۵. ۶۲۶. ۶۲۷. ۶۲۸. ۶۲۹. ۶۳۰. ۶۳۱. ۶۳۲. ۶۳۳. ۶۳۴. ۶۳۵. ۶۳۶. ۶۳۷. ۶۳۸. ۶۳۹. ۶۴۰. ۶۴۱. ۶۴۲. ۶۴۳. ۶۴۴. ۶۴۵. ۶۴۶. ۶۴۷. ۶۴۸. ۶۴۹. ۶۵۰. ۶۵۱. ۶۵۲. ۶۵۳. ۶۵۴. ۶۵۵. ۶۵۶. ۶۵۷. ۶۵۸. ۶۵۹. ۶۶۰. ۶۶۱. ۶۶۲. ۶۶۳. ۶۶۴. ۶۶۵. ۶۶۶. ۶۶۷. ۶۶۸. ۶۶۹. ۶۷۰. ۶۷۱. ۶۷۲. ۶۷۳. ۶۷۴. ۶۷۵. ۶۷۶. ۶۷۷. ۶۷۸. ۶۷۹. ۶۸۰. ۶۸۱. ۶۸۲. ۶۸۳. ۶۸۴. ۶۸۵. ۶۸۶. ۶۸۷. ۶۸۸. ۶۸۹. ۶۹۰. ۶۹۱. ۶۹۲. ۶۹۳. ۶۹۴. ۶۹۵. ۶۹۶. ۶۹۷. ۶۹۸. ۶۹۹. ۷۰۰. ۷۰۱. ۷۰۲. ۷۰۳. ۷۰۴. ۷۰۵. ۷۰۶. ۷۰۷. ۷۰۸. ۷۰۹. ۷۱۰. ۷۱۱. ۷۱۲. ۷۱۳. ۷۱۴. ۷۱۵. ۷۱۶. ۷۱۷. ۷۱۸. ۷۱۹. ۷۲۰. ۷۲۱. ۷۲۲. ۷۲۳. ۷۲۴. ۷۲۵. ۷۲۶. ۷۲۷. ۷۲۸. ۷۲۹. ۷۳۰. ۷۳۱. ۷۳۲. ۷۳۳. ۷۳۴. ۷۳۵. ۷۳۶. ۷۳۷. ۷۳۸. ۷۳۹. ۷۴۰. ۷۴۱. ۷۴۲. ۷۴۳. ۷۴۴. ۷۴۵. ۷۴۶. ۷۴۷. ۷۴۸. ۷۴۹. ۷۵۰. ۷۵۱. ۷۵۲. ۷۵۳. ۷۵۴. ۷۵۵. ۷۵۶. ۷۵۷. ۷۵۸. ۷۵۹. ۷۶۰. ۷۶۱. ۷۶۲. ۷۶۳. ۷۶۴. ۷۶۵. ۷۶۶. ۷۶۷. ۷۶۸. ۷۶۹. ۷۷۰. ۷۷۱. ۷۷۲. ۷۷۳. ۷۷۴. ۷۷۵. ۷۷۶. ۷۷۷. ۷۷۸. ۷۷۹. ۷۸۰. ۷۸۱. ۷۸۲. ۷۸۳. ۷۸۴. ۷۸۵. ۷۸۶. ۷۸۷. ۷۸۸. ۷۸۹. ۷۹۰. ۷۹۱. ۷۹۲. ۷۹۳. ۷۹۴. ۷۹۵. ۷۹۶. ۷۹۷. ۷۹۸. ۷۹۹. ۸۰۰. ۸۰۱. ۸۰۲. ۸۰۳. ۸۰۴. ۸۰۵. ۸۰۶. ۸۰۷. ۸۰۸. ۸۰۹. ۸۱۰. ۸۱۱. ۸۱۲. ۸۱۳. ۸۱۴. ۸۱۵. ۸۱۶. ۸۱۷. ۸۱۸. ۸۱۹. ۸۲۰. ۸۲۱. ۸۲۲. ۸۲۳. ۸۲۴. ۸۲۵. ۸۲۶. ۸۲۷. ۸۲۸. ۸۲۹. ۸۳۰. ۸۳۱. ۸۳۲. ۸۳۳. ۸۳۴. ۸۳۵. ۸۳۶. ۸۳۷. ۸۳۸. ۸۳۹. ۸۴۰. ۸۴۱. ۸۴۲. ۸۴۳. ۸۴۴. ۸۴۵. ۸۴۶. ۸۴۷. ۸۴۸. ۸۴۹. ۸۵۰. ۸۵۱. ۸۵۲. ۸۵۳. ۸۵۴. ۸۵۵. ۸۵۶. ۸۵۷. ۸۵۸. ۸۵۹. ۸۶۰. ۸۶۱. ۸۶۲. ۸۶۳. ۸۶۴. ۸۶۵. ۸۶۶. ۸۶۷. ۸۶۸. ۸۶۹. ۸۷۰. ۸۷۱. ۸۷۲. ۸۷۳. ۸۷۴. ۸۷۵. ۸۷۶. ۸۷۷. ۸۷۸. ۸۷۹. ۸۸۰. ۸۸۱. ۸۸۲. ۸۸۳. ۸۸۴. ۸۸۵. ۸۸۶. ۸۸۷. ۸۸۸. ۸۸۹. ۸۹۰. ۸۹۱. ۸۹۲. ۸۹۳. ۸۹۴. ۸۹۵. ۸۹۶. ۸۹۷. ۸۹۸. ۸۹۹. ۹۰۰. ۹۰۱. ۹۰۲. ۹۰۳. ۹۰۴. ۹۰۵. ۹۰۶. ۹۰۷. ۹۰۸. ۹۰۹. ۹۱۰. ۹۱۱. ۹۱۲. ۹۱۳. ۹۱۴. ۹۱۵. ۹۱۶. ۹۱۷. ۹۱۸. ۹۱۹. ۹۲۰. ۹۲۱. ۹۲۲. ۹۲۳. ۹۲۴. ۹۲۵. ۹۲۶. ۹۲۷. ۹۲۸. ۹۲۹. ۹۳۰. ۹۳۱. ۹۳۲. ۹۳۳. ۹۳۴. ۹۳۵. ۹۳۶. ۹۳۷. ۹۳۸. ۹۳۹. ۹۴۰. ۹۴۱. ۹۴۲. ۹۴۳. ۹۴۴. ۹۴۵. ۹۴۶. ۹۴۷. ۹۴۸. ۹۴۹. ۹۵۰. ۹۵۱. ۹۵۲. ۹۵۳. ۹۵۴. ۹۵۵. ۹۵۶. ۹۵۷. ۹۵۸. ۹۵۹. ۹۶۰. ۹۶۱. ۹۶۲. ۹۶۳. ۹۶۴. ۹۶۵. ۹۶۶. ۹۶۷. ۹۶۸. ۹۶۹. ۹۷۰. ۹۷۱. ۹۷۲. ۹۷۳. ۹۷۴. ۹۷۵. ۹۷۶. ۹۷۷. ۹۷۸. ۹۷۹. ۹۸۰. ۹۸۱. ۹۸۲. ۹۸۳. ۹۸۴. ۹۸۵. ۹۸۶. ۹۸۷. ۹۸۸. ۹۸۹. ۹۹۰. ۹۹۱. ۹۹۲. ۹۹۳. ۹۹۴. ۹۹۵. ۹۹۶. ۹۹۷. ۹۹۸. ۹۹۹. ۱۰۰۰. ۱۰۰۱. ۱۰۰۲. ۱۰۰۳. ۱۰۰۴. ۱۰۰۵. ۱۰۰۶. ۱۰۰۷. ۱۰۰۸. ۱۰۰۹. ۱۰۱۰. ۱۰۱۱. ۱۰۱۲. ۱۰۱۳. ۱۰۱۴. ۱۰۱۵. ۱۰۱۶. ۱۰۱۷. ۱۰۱۸. ۱۰۱۹. ۱۰۲۰. ۱۰۲۱. ۱۰۲۲. ۱۰۲۳. ۱۰۲۴. ۱۰۲۵. ۱۰۲۶. ۱۰۲۷. ۱۰۲۸. ۱۰۲۹. ۱۰۳۰. ۱۰۳۱. ۱۰۳۲. ۱۰۳۳. ۱۰۳۴. ۱۰۳۵. ۱۰۳۶. ۱۰۳۷. ۱۰۳۸. ۱۰۳۹. ۱۰۴۰. ۱۰۴۱. ۱۰۴۲. ۱۰۴۳. ۱۰۴۴. ۱۰۴۵. ۱۰۴۶. ۱۰۴۷. ۱۰۴۸. ۱۰۴۹. ۱۰۵۰. ۱۰۵۱. ۱۰۵۲. ۱۰۵۳. ۱۰۵۴. ۱۰۵۵. ۱۰۵۶. ۱۰۵۷. ۱۰۵۸. ۱۰۵۹. ۱۰۶۰. ۱۰۶۱. ۱۰۶۲. ۱۰۶۳. ۱۰۶۴. ۱۰۶۵. ۱۰۶۶. ۱۰۶۷. ۱۰۶۸. ۱۰۶۹. ۱۰۷۰. ۱۰۷۱. ۱۰۷۲. ۱۰۷۳. ۱۰۷۴. ۱۰۷۵. ۱۰۷۶. ۱۰۷۷. ۱۰۷۸. ۱۰۷۹. ۱۰۸۰. ۱۰۸۱. ۱۰۸۲. ۱۰۸۳. ۱۰۸۴. ۱۰۸۵. ۱۰۸۶. ۱۰۸۷. ۱۰۸۸. ۱۰۸۹. ۱۰۹۰. ۱۰۹۱. ۱۰۹۲. ۱۰۹۳. ۱۰۹۴. ۱۰۹۵. ۱۰۹۶. ۱۰۹۷. ۱۰۹۸. ۱۰۹۹. ۱۱۰۰. ۱۱۰۱. ۱۱۰۲. ۱۱۰۳. ۱۱۰۴. ۱۱۰۵. ۱۱۰۶. ۱۱۰۷. ۱۱۰۸. ۱۱۰۹. ۱۱۱۰. ۱۱۱۱. ۱۱۱۲. ۱۱۱۳. ۱۱۱۴. ۱۱۱۵. ۱۱۱۶. ۱۱۱۷. ۱۱۱۸. ۱۱۱۹. ۱۱۲۰. ۱۱۲۱. ۱۱۲۲. ۱۱۲۳. ۱۱۲۴. ۱۱۲۵. ۱۱۲۶. ۱۱۲۷. ۱۱۲۸. ۱۱۲۹. ۱۱۳۰. ۱۱۳۱. ۱۱۳۲. ۱۱۳۳. ۱۱۳۴. ۱۱۳۵. ۱۱۳۶. ۱۱۳۷. ۱۱۳۸. ۱۱۳۹. ۱۱۴۰. ۱۱۴۱. ۱۱۴۲. ۱۱۴۳. ۱۱۴۴. ۱۱۴۵. ۱۱۴۶. ۱۱۴۷. ۱۱۴۸. ۱۱۴۹. ۱۱۵۰. ۱۱۵۱. ۱۱۵۲. ۱۱۵۳. ۱۱۵۴. ۱۱۵۵. ۱۱۵۶. ۱۱۵۷. ۱۱۵۸. ۱۱۵۹. ۱۱۶۰. ۱۱۶۱. ۱۱۶۲. ۱۱۶۳. ۱۱۶۴. ۱۱۶۵. ۱۱۶۶. ۱۱۶۷. ۱۱۶۸. ۱۱۶۹. ۱۱۷۰. ۱۱۷۱. ۱۱۷۲. ۱۱۷۳. ۱۱۷۴. ۱۱۷۵. ۱۱۷۶. ۱۱۷۷. ۱۱۷۸. ۱۱۷۹. ۱۱۸۰. ۱۱۸۱. ۱۱۸۲. ۱۱۸۳. ۱۱۸۴. ۱۱۸۵. ۱۱۸۶. ۱۱۸۷. ۱۱۸۸. ۱۱۸۹. ۱۱۹۰. ۱۱۹۱. ۱۱۹۲. ۱۱۹۳. ۱۱۹۴. ۱۱۹۵. ۱۱۹۶. ۱۱۹۷. ۱۱۹۸. ۱۱۹۹. ۱۲۰۰. ۱۲۰۱. ۱۲۰۲. ۱۲۰۳. ۱۲۰۴. ۱۲۰۵. ۱۲۰۶. ۱۲۰۷. ۱۲۰۸. ۱۲۰۹. ۱۲۱۰. ۱۲۱۱. ۱۲۱۲. ۱۲۱۳. ۱۲۱۴. ۱۲۱۵. ۱۲۱۶. ۱۲۱۷. ۱۲۱۸. ۱۲۱۹. ۱۲۲۰. ۱۲۲۱. ۱۲۲۲. ۱۲۲۳. ۱۲۲۴. ۱۲۲۵. ۱۲۲۶. ۱۲۲۷. ۱۲۲۸. ۱۲۲۹. ۱۲۳۰. ۱۲۳۱. ۱۲۳۲. ۱۲۳۳. ۱۲۳۴. ۱۲۳۵. ۱۲۳۶. ۱۲۳۷. ۱۲۳۸. ۱۲۳۹. ۱۲۴۰. ۱۲۴۱. ۱۲۴۲. ۱۲۴۳. ۱۲۴۴. ۱۲۴۵. ۱۲۴۶. ۱۲۴۷. ۱۲۴۸. ۱۲۴۹. ۱۲۵۰. ۱۲۵۱. ۱۲۵۲. ۱۲۵۳. ۱۲۵۴. ۱۲۵۵. ۱۲۵۶. ۱۲۵۷. ۱۲۵۸. ۱۲۵۹. ۱۲۶۰. ۱۲۶۱. ۱۲۶۲. ۱۲۶۳. ۱۲۶۴. ۱۲۶۵. ۱۲۶۶. ۱۲۶۷. ۱۲۶۸. ۱۲۶۹. ۱۲۷۰. ۱۲۷۱. ۱۲۷۲. ۱۲۷۳. ۱۲۷۴. ۱۲۷۵. ۱۲۷۶. ۱۲۷۷. ۱۲۷۸. ۱۲۷۹. ۱۲۸۰. ۱۲۸۱. ۱۲۸۲. ۱۲۸۳. ۱۲۸۴. ۱۲۸۵. ۱۲۸۶. ۱۲۸۷. ۱۲۸۸. ۱۲۸۹. ۱۲۹۰. ۱۲۹۱. ۱۲۹۲. ۱۲۹۳. ۱۲۹۴. ۱۲۹۵. ۱۲۹۶. ۱۲۹۷. ۱۲۹۸. ۱۲۹۹. ۱۳۰۰. ۱۳۰۱. ۱۳۰۲. ۱۳۰۳. ۱۳۰۴. ۱۳۰۵. ۱۳۰۶. ۱۳۰۷. ۱۳۰۸. ۱۳۰۹. ۱۳۱۰. ۱۳۱۱. ۱۳۱۲. ۱۳۱۳. ۱۳۱۴. ۱۳۱۵. ۱۳۱۶. ۱۳۱۷. ۱۳۱۸. ۱۳۱۹. ۱۳۲۰. ۱۳۲۱. ۱۳۲۲. ۱۳۲۳. ۱۳۲۴. ۱۳۲۵. ۱۳۲۶. ۱۳۲۷. ۱۳۲۸. ۱۳۲۹. ۱۳۳۰. ۱۳۳۱. ۱۳۳۲. ۱۳۳۳. ۱۳۳۴. ۱۳۳۵. ۱۳۳۶. ۱۳۳۷. ۱۳۳۸. ۱۳۳۹. ۱۳۴۰. ۱۳۴۱. ۱۳۴۲. ۱۳۴۳. ۱۳۴۴. ۱۳۴۵. ۱۳۴۶. ۱۳۴۷. ۱۳۴۸. ۱۳۴۹. ۱۳۵۰. ۱۳۵۱. ۱۳۵۲. ۱۳۵۳. ۱۳۵۴. ۱۳۵۵. ۱۳۵۶. ۱۳۵۷. ۱۳۵۸. ۱۳۵۹. ۱۳۶۰. ۱۳۶۱. ۱۳۶۲. ۱۳۶۳. ۱۳۶۴. ۱۳۶۵. ۱۳۶۶. ۱۳۶۷. ۱۳۶۸. ۱۳۶۹. ۱۳۷۰. ۱۳۷۱. ۱۳۷۲. ۱۳۷۳. ۱۳۷۴. ۱۳۷۵. ۱۳۷۶. ۱۳۷۷. ۱۳۷۸. ۱۳۷۹. ۱۳۸۰. ۱۳۸۱. ۱۳۸۲. ۱۳۸۳. ۱۳۸۴. ۱۳۸۵. ۱۳۸۶. ۱۳۸۷. ۱۳۸۸. ۱۳۸۹. ۱۳۹۰. ۱۳۹۱. ۱۳۹۲. ۱۳۹۳. ۱۳۹۴. ۱۳۹۵. ۱۳۹۶. ۱۳۹۷. ۱۳۹۸. ۱۳۹۹. ۱۴۰۰. ۱۴۰۱. ۱۴۰۲. ۱۴۰۳. ۱۴۰۴. ۱۴۰۵. ۱۴۰۶. ۱۴۰۷. ۱۴۰۸. ۱۴۰۹. ۱۴۱۰. ۱۴۱۱. ۱۴۱۲. ۱۴۱۳. ۱۴۱۴. ۱۴۱۵. ۱۴۱۶. ۱۴۱۷. ۱۴۱۸. ۱۴۱۹. ۱۴۲۰. ۱۴۲۱. ۱۴۲۲. ۱۴۲۳. ۱۴۲۴. ۱۴۲۵. ۱۴۲۶. ۱۴۲۷. ۱۴۲۸. ۱۴۲۹. ۱۴۳۰. ۱۴۳۱. ۱۴۳۲. ۱۴۳۳. ۱۴۳۴. ۱۴۳۵. ۱۴۳۶. ۱۴۳۷. ۱۴۳۸. ۱۴۳۹. ۱۴۴۰. ۱۴۴۱. ۱۴۴۲. ۱۴۴۳. ۱۴۴۴. ۱۴۴۵. ۱۴۴۶. ۱۴۴۷. ۱۴۴۸. ۱۴۴۹. ۱۴۵۰. ۱۴۵۱. ۱۴۵۲. ۱۴۵۳. ۱۴۵۴. ۱۴۵۵. ۱۴۵۶. ۱۴۵۷. ۱۴۵۸. ۱۴۵۹. ۱۴۶۰. ۱۴۶۱. ۱۴۶۲. ۱۴۶۳. ۱۴۶۴. ۱۴۶۵. ۱۴۶۶. ۱۴۶۷. ۱۴۶۸. ۱۴۶۹. ۱۴۷۰. ۱۴۷۱. ۱۴۷۲. ۱۴۷۳. ۱۴۷۴. ۱۴۷۵. ۱۴۷۶. ۱۴۷۷. ۱۴۷۸. ۱۴۷۹. ۱۴۸۰. ۱۴۸۱. ۱۴۸۲. ۱۴۸۳. ۱۴۸۴. ۱۴۸۵. ۱۴۸۶. ۱۴۸۷. ۱۴۸۸. ۱۴۸۹. ۱۴۹۰. ۱۴۹۱. ۱۴۹۲. ۱۴۹۳. ۱۴۹۴. ۱۴۹۵. ۱۴۹۶. ۱۴۹۷. ۱۴۹۸. ۱۴۹۹. ۱۵۰۰. ۱۵۰۱. ۱۵۰۲. ۱۵۰۳. ۱۵۰۴. ۱۵۰۵. ۱۵۰۶. ۱۵۰۷. ۱۵۰۸. ۱۵۰۹. ۱۵۱۰. ۱۵۱۱. ۱۵۱۲. ۱۵۱۳. ۱۵۱۴. ۱۵۱۵. ۱۵۱۶. ۱۵۱۷. ۱۵۱۸. ۱۵۱۹. ۱۵۲۰. ۱۵۲۱. ۱۵۲۲. ۱۵۲۳. ۱۵۲۴. ۱۵۲۵. ۱۵۲۶. ۱۵۲۷. ۱۵۲۸. ۱۵۲۹. ۱۵۳۰. ۱۵۳۱. ۱۵۳۲. ۱۵۳۳. ۱۵۳۴. ۱۵۳۵. ۱۵۳۶. ۱۵۳۷. ۱۵۳۸. ۱۵۳۹. ۱۵۴۰. ۱۵۴۱. ۱۵۴۲. ۱۵۴۳. ۱۵۴۴. ۱۵۴۵. ۱۵۴۶. ۱۵۴۷. ۱۵۴۸. ۱۵۴۹. ۱۵۵۰. ۱۵۵۱. ۱۵۵۲. ۱۵۵۳. ۱۵۵۴. ۱۵۵۵. ۱۵۵۶. ۱۵۵۷. ۱۵۵۸. ۱۵۵۹. ۱۵۶۰. ۱۵۶۱. ۱۵۶۲. ۱۵۶۳. ۱۵۶۴. ۱۵۶۵. ۱۵۶۶. ۱۵۶۷. ۱۵۶۸. ۱۵۶۹. ۱۵۷۰. ۱۵۷۱. ۱۵۷۲. ۱۵۷۳. ۱۵۷۴. ۱۵۷۵. ۱۵۷۶. ۱۵۷۷. ۱۵۷۸. ۱۵۷۹. ۱۵۸۰. ۱۵۸۱. ۱۵۸۲. ۱۵۸۳. ۱۵۸۴. ۱۵۸۵. ۱۵۸۶. ۱۵۸۷. ۱۵۸۸. ۱۵۸۹. ۱۵۹۰. ۱۵۹۱. ۱۵۹۲. ۱۵۹۳. ۱۵۹۴. ۱۵۹۵. ۱۵۹۶. ۱۵۹۷. ۱۵۹۸. ۱۵۹۹. ۱۶۰۰. ۱۶۰۱. ۱۶۰۲. ۱۶۰۳. ۱۶۰۴. ۱۶۰۵. ۱۶۰۶. ۱۶۰۷. ۱۶۰۸. ۱۶۰۹. ۱۶۱۰. ۱۶۱۱. ۱۶۱۲. ۱۶۱۳. ۱۶۱۴. ۱۶۱۵. ۱۶۱۶. ۱۶۱۷. ۱۶۱۸. ۱۶۱۹. ۱۶۲۰. ۱۶۲۱. ۱۶۲۲. ۱۶۲۳. ۱۶۲۴. ۱۶۲۵. ۱۶۲۶. ۱۶۲۷. ۱۶۲۸. ۱۶۲۹. ۱۶۳۰. ۱۶۳۱. ۱۶۳۲. ۱۶۳۳. ۱۶۳۴. ۱۶۳۵. ۱۶۳۶. ۱۶۳۷. ۱۶۳۸. ۱۶۳۹. ۱۶۴۰. ۱۶۴۱. ۱۶۴۲. ۱۶۴۳. ۱۶۴۴. ۱۶۴۵. ۱۶۴۶. ۱۶۴۷. ۱۶۴۸. ۱۶۴۹. ۱۶۵۰. ۱۶۵۱. ۱۶۵۲. ۱۶۵۳. ۱۶۵۴. ۱۶۵۵. ۱۶۵۶. ۱۶۵۷. ۱۶۵۸. ۱۶۵۹. ۱۶۶۰. ۱۶۶۱. ۱۶۶۲. ۱۶۶۳. ۱۶۶۴. ۱۶۶۵. ۱۶۶۶. ۱۶۶۷. ۱۶۶۸. ۱۶۶۹. ۱۶۷۰. ۱۶۷۱. ۱۶۷۲. ۱۶۷۳. ۱۶۷۴. ۱۶۷۵. ۱۶۷۶. ۱۶۷۷. ۱۶۷۸. ۱۶

5 Feature Engineering

5.1 Implementation Approach

In this part our aim was to compare different features and to conclude if we feed other features besides our text data to our model, would model generalize and make decision better or not?

So I defined a Neural Network model which can accept all of the features that I mention in next part. my model features was:

- model architecture :
 - a input size of 768 neurons(for sake of generalizing like BERT).
 - hidden layer of size 1000 with relu activation and dropout layer.
 - hidden layer of size 200 with relu activation and dropout layer.
 - output layer with one neuron and sigmoid activation
- model settings:
 - loss : Binary Cross Entropy
 - optimizer : Adam with learning rate = 0.001
 - epochs : 60
 - batch size : 32

inputs were fed in "cuda" format for utilization from gpu and training faster.

the features used in this experiments are explained below and I will get through the details of each input to be feedable to our model:

- sentence length : an array that first number in array is sentence length and the array was padded with zero to be in length 768.
- word length : an array that fills word length through 768 elements of it. if tweets has more word than 768 it will be truncated and if tweets has less length than 768 will be padded with zero to be in length 768
- words : assigning unique number to each word. I first created a *token dict* that each word has its unique number. And then tweets are tokenized each tokenized word are then mapped to their indexes. matching the dimension criterion was applied like previous approach.
- words-bigrams : just like previous feature but every consecutive two tokens are concatenated to represent one element of our array. array was filled until to get length 768. otherwise it will be padded by zero.
- word2vec : trained *all.word2vec.npy* model were used to extract each token vectors from it. and then vectors extracted are summed to be in same direction and then it was fed into model.

- word2vec-bigram : treated like before but for each two consecutive word two vectors are concatenated and then all these vectors are summed.
- ParsBERT : used [bert-fa-base-uncased](#) model to tokenize and then feed each tokenized input to extract vectors. like word2vec all these token vectors are summed to be in same dimension as model input.

models are saved into *models/feat_engineering_{feature_name}.pth*.

5.2 Results

since the whole plot wouldn't be located in one figure it has been plotted in multi figure.

5.2.1 loss

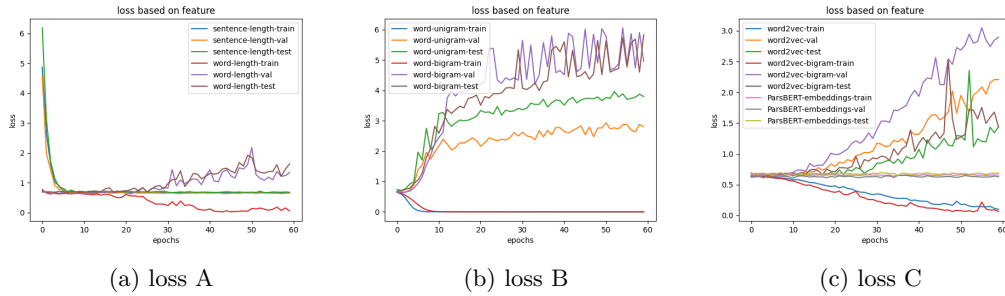


Figure 4: loss over 7 trained model on 3 parts train / val / test

Figure 1 : As we see sentence length model on each 3 parts have been covered well. but word length model just perform fantastic in train part but on test and val doesn't generalize well. so it shows there is over-fit word length model.

Figure 2 : former behaviour is also seen in word-unigram and word-bigram model, so these models also overfit to just train data.

Figure 3 : both word2vec unigram and bi-gram are overfit to train data and they perform well on training data. best model seen so far can be ParsBERT model with a low loss and without any effect of overfitting.

5.2.2 accuracy

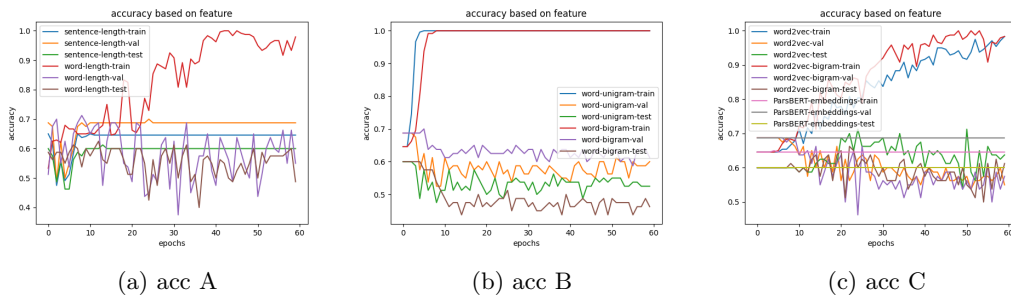


Figure 5: accuracy over 7 trained model on 3 parts train / val / test

If we look at three figures closely we'll see effect of overfitting on 5 models from our 7 models. the only two models that don't overfit are "sentence length" and "ParsBERT" that ParsBERT with higher accuracy will defeat sentence length.

6 Model Architecture

6.1 Implementation Approach

As I see the progress on *transformer* models in recent years. I chose my three models have some usage of transformer in it. I will dig through the 3 model architecture I have implemented.

6.1.1 Word2Vec + ParsBERT

This model involve both Word2Vec and ParsBERT together. such that my base model is Neural Network and the logics are just like as I explained in prevoius section but the input I fed into this model was vector of tokenized word on each model and then get average of these two different vecotrs and then feed it to the model. Since this model don't generlaize well(lack of sequence knowledge) I didnt try this model for other 3 traits.

Model is save into models are saved into *model_architecture_combined_parsbert_word2vec.pth* and the results of the models are convincing:

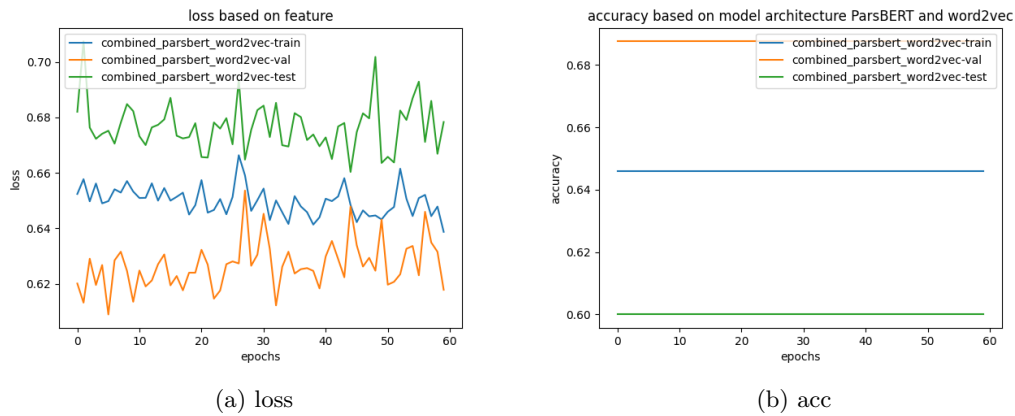


Figure 6: accuracy over 7 trained model on 3 parts train / val / test

6.1.2 BERT Truncated

As in the name of section says, this model are feed tweets with truncated length to be not more than 512 tokens. during training this model, I have ran into serious GPU out of memory issue and I have tried many different solutions to overcome this problem. for example I used the built in *Trainer* object of pytorch for training or I decreased length of sentence which didn't lead to be escape way. At the end I tried to lowered *batch size* so that trade-off between batch size and model preserves. because lower batch size results in much more training time and higher batch size will result in GPU shortage. So I chose

$batch\ size = 14$ that happens to be the best batch size. through this batch size GPU usage was about 14GB. According to *colab* free account users don't have more than 15GB GPU.

Model Explanation

1. first of all I concatenated all tweets of user with space and this makes our input to be in string format instead of list.
2. after that I make 4 individual trait for training 4 four different models. I meant that "I,E" are together and then binary classified, "S, N" are together and then binary classified and etc.
3. then I splitted my dataset into 3 parts train/validation/test.
4. I defined Custom model which uses BERT CLS embeddings which stands for classification. and with this embeddings I managed to build a fully connected layer which maps (768, 1). as we know our output neuron should be one because we have binary classification and use sigmoid as activation function. and *BCELoss* for loss function and AdamW for optimizer. And in the end model was trained on 8 epochs.
5. I make use of pretrained [ParsBERT model](#) as my pretrained BERT embeddings.
6. Fine-tuning phase is applied on all layer even BERT layer.
7. The model is saved into *models/BERT_TRUNCATED_MODEL*

let's take a look at models results:

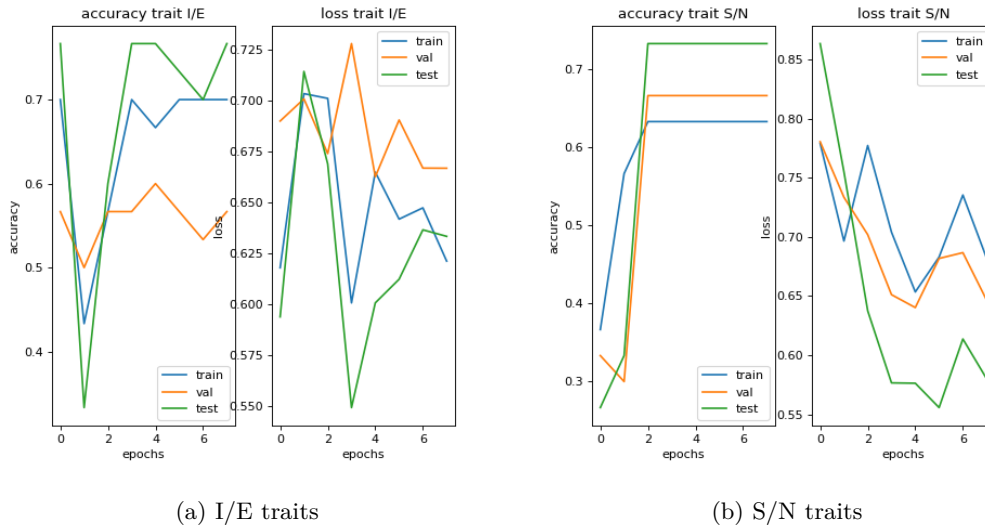
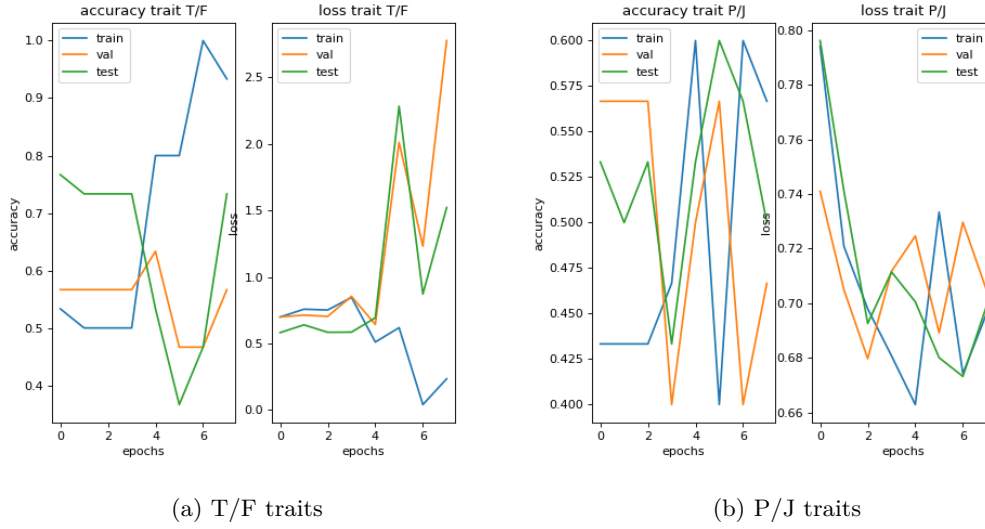


Figure 7: mmd



As we see in all of 4 traits test data has achieved acceptable accuracy and its more than 70 percentage in all of them. that means we have made a progress in that way. and we don't see any overfit or at least a little overfit on our data which is great.

6.1.3 BERT Into BERT

This my final model which after consulting with many senior in NLP field and my mentors I came up into it. In fact I have about 4000 users and collected all of their tweets which on average each user has 700 tweets. And if we count each tweet has at least 30 tokens, for each user I have 21000 tokens and for a model like bert which at most can be feed 512 tokens it's such a enormous challenge.

1. So the idea was to leverage each user as a batch(that cleary make our batch size length = 1).
2. and then feed all tweets into a embedding bert that would result in shape $(tweets_length, sequence_length, dimension)$
3. then we extract CLS embedding to have shape $(tweets_length, dimension)$
4. so in this step we have repsresenation for each tweet(embedding for each tweet)
5. now we feed the extracted CLS embeddings to another bert
6. on top of the second BERT we extracted CLS layer and then feed it into a fully connected layer for binary classification.
7. At first I tried to fine-tune on 2 BERTs but because lack of GPU memory I had to just fine-tune on the second BERT. Although fine-tuning on second BERT is a great approach.

the model is saved under `models/bert_in_bert` directory.

let's take a look at the model results:

	loss	acc
train	0.59	0.74
validation	0.71	0.58
test	0.58	0.74

I/E trait

	loss	acc
train	0.6	0.72
validation	0.6	0.72
test	0.59	0.72

S/N trait

	loss	acc
train	0.69	0.54
validation	0.69	0.54
test	0.64	0.72

T/F trait

	loss	acc
train	0.76	0.38
validation	0.72	0.44
test	0.7	0.54

P/J trait

as we see *test* part on first 3 trait are above 70 percentage that is great but for the 4th trait model hasn't performed well.

So all in all the "BERT Into BERT" model outperforms other models and it's picked as our idol model.

7 Data Augmentation

7.1 Implementation Approach

At first I got ChatGPT api key. and then I make truncated text from list of tweets since tweets length exceeds ChatGPT api free access. then I fine tuned chatgpt on my data(chosen model was *text-davinci-003*). After fine tuning I iterated over 16 possible labels and inside this labels I've randomly chosen some rows. within selected row I concatenated each tweets into string and I made a prompt like this:

"I want you to act as a Data Augmenter Specialist. I will provide you some users tweets and their related MBTI types. And then I want you to make a tweet that a person with that MBTI type would tweet. tweets examples are provided below: {tweets : some sample tweets concatenated 1, mbti_type:some label from 16 labels 1},{tweets : some sample tweets concatenated 2, mbti_type:some label from 16 labels 2}"

[illegible]

Figure 9: data generated for each trait by chatgpt



Figure 10: actual data in our dataset

As we see there is high resemblance in data created by GPT and our actual data. but GPT in someway lost the context in between its sentence but it can ends its sentence much well.

8.1 Implementation Approach

As I said earlier because GPT max token is 4097 and my tokens is much greater than this number I had to truncate my tweet until it can be requested to GPT. like previous part my chosen model was *text-davinci-003*. after that I fed the truncated tweets to model and asked model to generate their label from 16 possible labels. my prompt was in this format: "You are picking the labels for a zero shot classification model on user tweets, domain is MBTI personality types, Generate a label from 16 MBTI types. tweet is : '{some tweet from user}'."

Unfortunately I encountered *RateLimitError* request that I couldn't accomplish this task.