Internship Report

❑ **Abhinav Jha** ❑ **Summer Intern** ❑ **jhaabhinav1998@gmail.com**

Human Personality Detection

In this task, the main aim was to design a deep learning architecture to classify the text data based on the traits present. So, I designed a binary classifier which classified as 1 or 0 ,i.e, whether the trait is present or not. The basic idea of the deep learning architecture was taken from the **SenticNet paper** on Human Personality Detection from texts. The dataset used for training and testing the model consist of texts and their labels (n,y) of five traits namely, Extroversion (EXT), Neuroticism (NEU), Agreeableness (AGR), Conscientiousness (CON) and Openness (OPN).

The results produced were comparable with the baseline results.

CNN Classifier (Self)	Personality Traits

Metrics	EXT	NEU	AGR	CON	OPN
Accuracy	0.54	0.55	0.52	0.56	0.57
Precision (class-wise)	[0.50,0.57]	[0.56,0.52]	[0.45,0.56]	[0.53,0.57]	[0.54,0.60]
Recall (class-wise)	[0.44,0.63]	[0.65,0.42]	[0.35,0.66]	[0.34,0.75]	[0.63,0.50]
F1-score (class-wise)	[0.47,0.60]	[0.60,0.46]	[0.39,0.60]	[0.42,0.65]	[0.58,0.55]

Table 1 : Results obtained on SenticNet Dataset for Human Personality Detection.

DL Architecture

- In our deep learning model, we first preprocessed each text article. We tokenized each sentence into words and represented each word as a 300 dimension glove vector embedding.
- The vocabulary size is 30000 words, and each documented is represented as a vector of max length 1000.

- Then we applied multiple sized filters to capture the n-gram dependency structure in the text, filter size used - 3,5,7 .
- Glove embedding of 300 dimension is used to represent each word as a integer vector.
- Adam optimizer is used for the learning process.
- Few combinations of optimizer and learning rate were tried, the best result was optained using adam optimizer and learning rate of 0.001.
- After applying the convolution layer, the document level feature representation was obtained using global max pooling layer forming 128 dimension document level representation.
- And then two dense layers were applied to do the binary classification task.
- Same architecture is used to do the binary classification of all the personality traits.

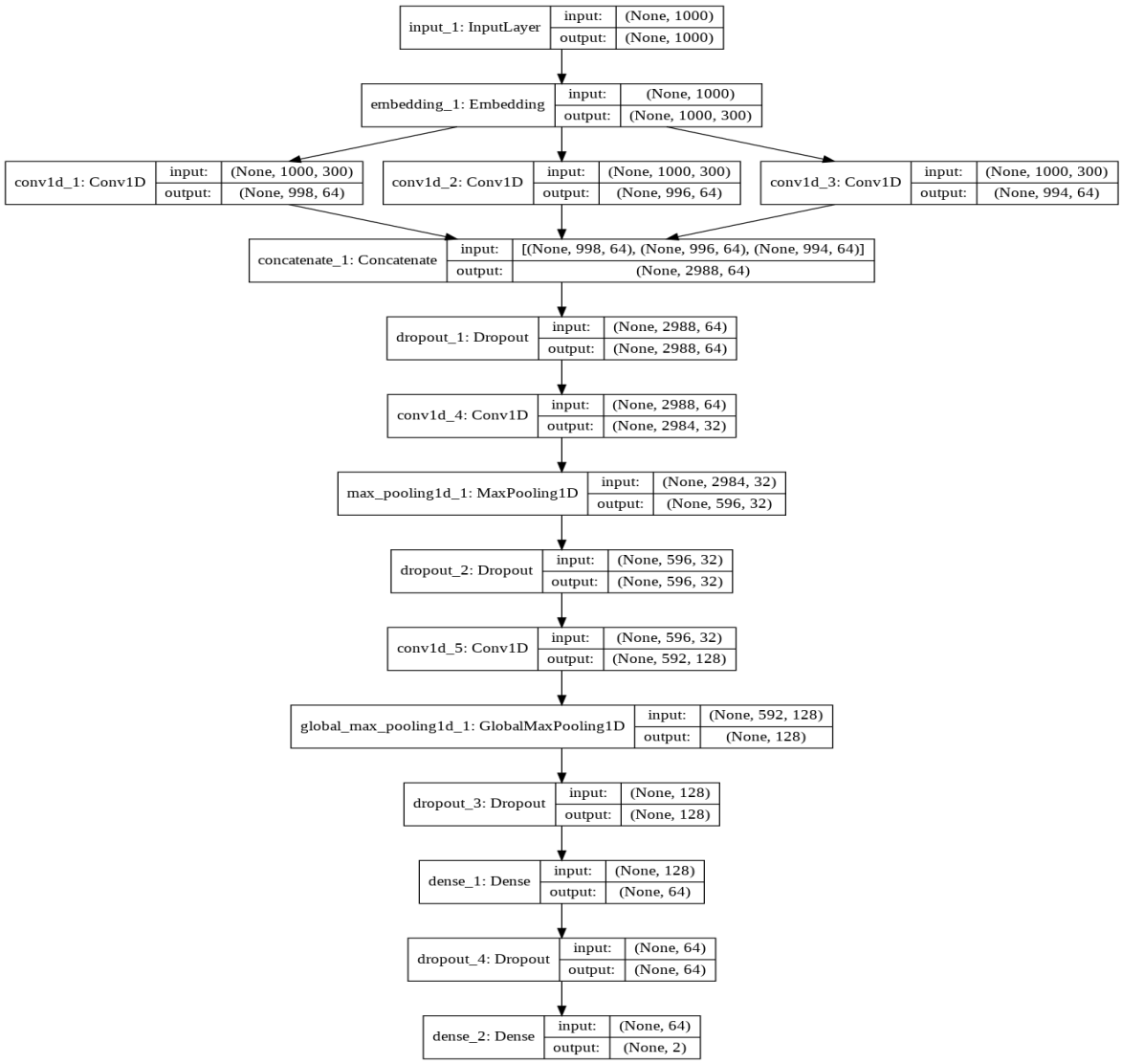

Fig 1: DL Architecture

Brand Personality Detection

In this task, the main aim was to detect the brand personality from texts. I worked on designing the deep learning model for the task of binary classification for the five brand personality traits (Sincerity, Excitement, Competence, Ruggedness, Sophistication) independently, using limited labeled data.

At first, I used the same previous model (Fig 1), and trained on the brand personality dataset, which was different from the earlier task as the earlier task was about detecting the human personality from texts. The training and validation results (90% training and 10% validation) were very promising but after testing on the heldout and the human annotated data, we realized that our model was overfitting to the training data. So, we tweaked our model.

CNN classifier (same)	**Personality Traits**

Metrics	Sincerity	Excitement	Competence	Sophistication
Validation Accuracy	0.86	0.84	0.80	0.82
Precision (class-wise)	[0.87,0.64]	[0.98,0.74]	[0.90,0.62]	[0.89,0.60]
Recall (class-wise)	[0.98,0.18]	[0.62,0.99]	[0.80,0.80]	[0.87,0.65]
F1-score (class-wise)	[0.92,0.28]	[0.76,0.84]	[0.85,0.70]	[0.88,0.62]

Table 2: Validation results on Brand Personality Dataset

We tweaked our classifier and the results that we got on the heldout data and the human annotated data(ht data) were comparable with the baseline results.

Preprocessing of dataset

- We denoised each of the text articles in the dataset, we removed the non-ascii characters, removed numbers, removed html tags, removed square brackets, fixed the contractions, lowercase each of the tokenized word
- Removed the stop words

- Lemmatized the verbs.

We tokenized the documents into words and represented each word as 300 dimension vector and represented each sentence as a 3000 dimension vector (fixed length because of padding each sequence).

DL Architecture

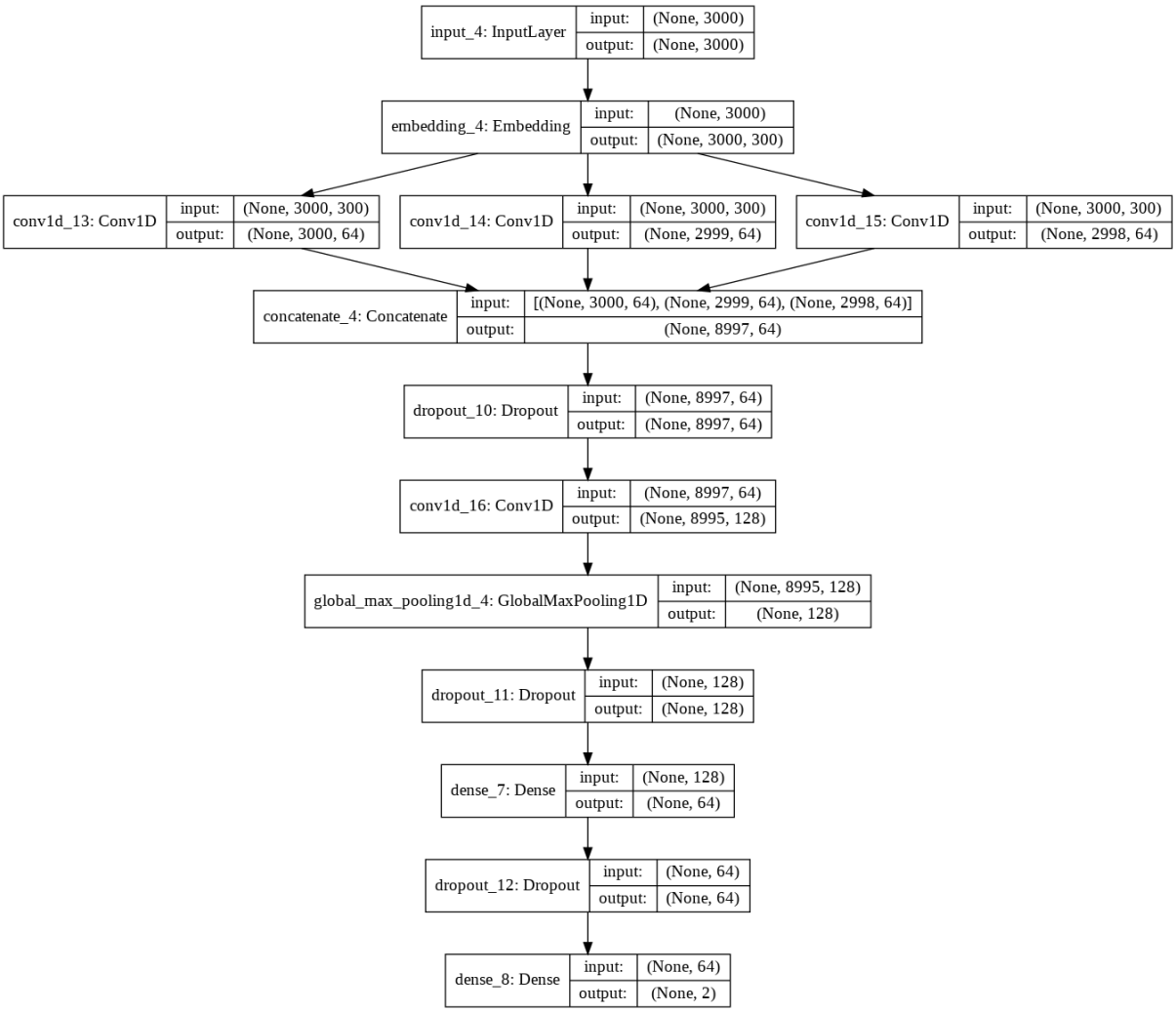

Fig 2: DL Architecture

- Represented each word as a 300 dimension vector(glove word2vec), then each sentence as 3000 length sequence.
- In the DL architecture, used multiple sized filters to capture the n-gram dependency. Filter size 1 to capture 1-gram,Filter size 2 to capture 2-gram and Filter size 3 to capture 3-gram dependency among the words.

- Then concatenated the output from these three convolution layers to form a vector of 8997*64.
- Then passed the vector through a CNN layer(filter size = 3, n_filters=128) to form the sentence vector.
- Then applied global max pooling to get the final document level vector representation.
- Then applied dense layers to classify the learnt document level representation into 2 classes namely yes or no(whether that trait is present or not).

Results Discussion

- In case of sincerity, excitement, competence and sophistication traits the amount of training data was balanced (2500 positive and 2500 negative), so the results we got on both the heldout data and the human annotated data(ht data) were comparable with the baseline results.
- In case of ruggedness the data was imbalanced around 250 positive and 2500 negative, so we did oversampling using SMOTE to have the balanced data (2500 positive and 2500 negative). It didn't improve the results much but the results were good enough, if we take the amount of data present for training into consideration.

Transfer Learning

- Trained the model on the MT data, then
    - The classification layers (dense_7 and dense_8 according to above architecture) were set to tainable while all other layers were kept as constant.
    - On HT data, we performed 7 fold validation. In which 6/7 of the data was used to train the classification layers while 1/7 was used for testing. The final result was calculated by averaging all the 7 folds.

- For all the traits except the ruggedness, the above approach worked, but for ruggedness we had to oversample the training data using SMOTE(Synthetic Minority Over-sampling Technique) to get the balanced data, then we trained our model on the oversampled data.
    - HT data also had data imbalance for ruggedness trait, so I oversampled the training data while performing the 7 fold validation and tested on the 1/7 data (without oversampling) and then the final results were calculated by averaging the results of all 7 folds.

- On heldout data, first trained the model on the MT data and then performed the transfer learning on HT data and then tested on the heldout data and the results were reported.

Transfer Learning Results Discussion

- By transfer learning the results improved in all the traits and results were even better than the baseline (FLCMT). For all traits except we performed the 7-fold validation(6/7 training and 1/7) testing on HT data, so the model got to learn the internal structure of features in HT data. So, the results improved.
- In case of heldout data, the model was trained on HT and keeping all layer constant except the two classification layers, the results were better because of higher recall value, because the model got more positive sample data so it was able to learn the features corresponding to positive label better.

With transfer learning, we were able to outperform the baseline results and got the best results for all the traits.

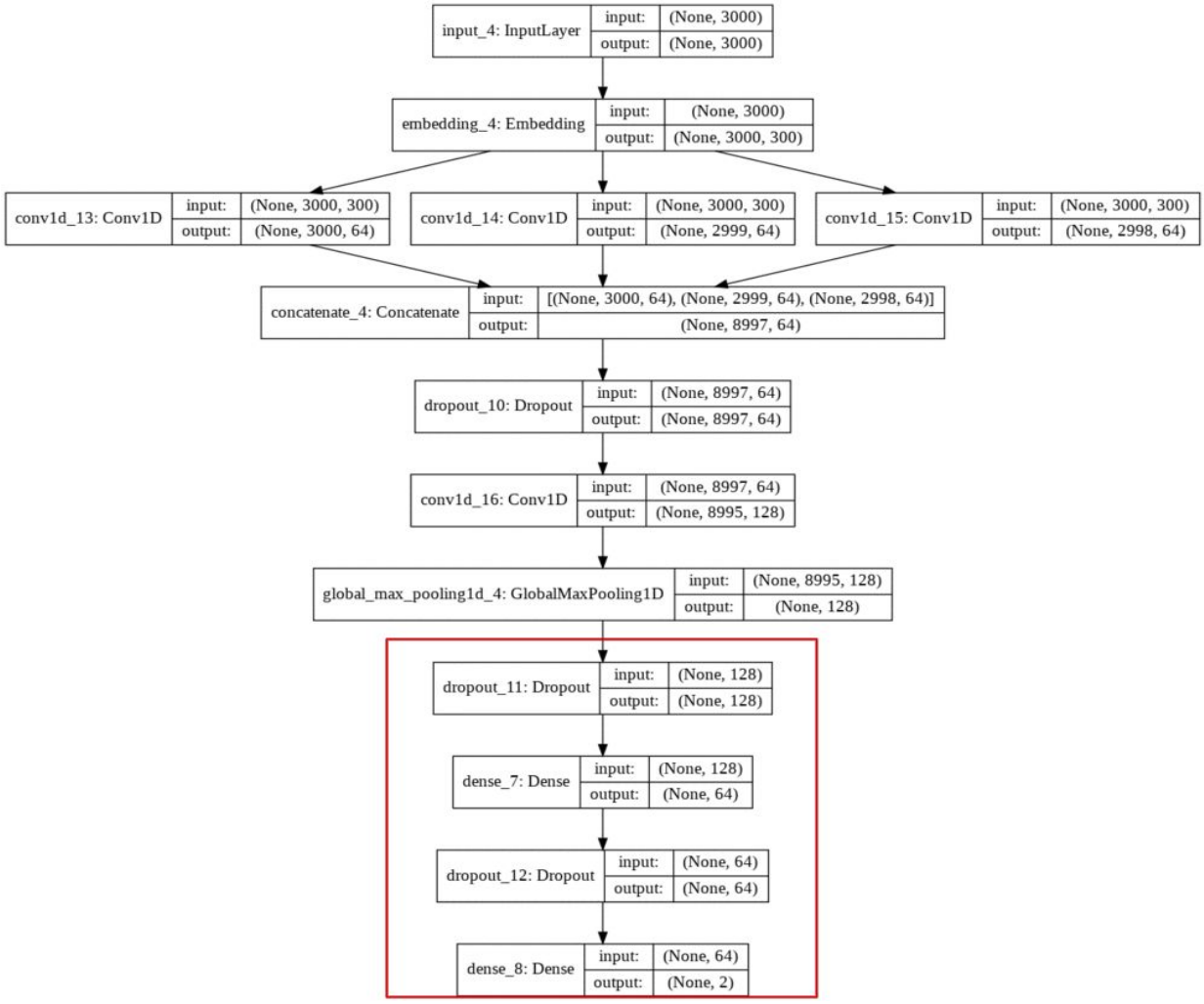

Fig 3: Layers trained during transfer learning process

Effect of domain features

We used LIWC (Linguistic Inquiry and Word Count) which are a document level features, for the classification task. These LIWC features are calculated based on internal default dictionary that defines which words should be counted in the target text file, these words generally are negative emotion words, positive words etc. We had these linguistic features for each text.

In the DL architecture (Fig. 2) after calculating the document level features, by applying the global max pooling layer, we added the 70 dimension LIWC features to create a 198 dimension document level feature map, then we passed these features to the classification layer(dense layers) and did the binary classification to predict that personality trait is present or not.

We thus developed the $DL_{lingset}$, which improves the F1 score across all the traits, except sophistication. Without using transfer learning, with the $DL_{lingset}$, we were able to match the baseline results.

Models	sincerity	excitement	competence	ruggedness	sophistication
FLCMT	0.82	0.61	0.6	0.38	0.65
FastText	0.84	0.74	0.96	0.31	0.47
DLinter	0.6	0.64	0.64	0.16	0.46
DLlingset	0.7	0.76	0.96	0.54	0.36
DLfinal	0.93	0.81	0.95	0.57	0.73

Table 3: Results on HT(Human Annotated) dataset

Models	sincerity	excitement	competence	ruggedness	sophistication
FLCMT	0.65	0.47	0.53	0.28	0.61
FastText	0.8	0.72	0.95	0.25	0.47
DLinter	0.53	0.57	0.61	0.16	0.48
DLlingset	0.7	0.78	0.95	0.55	0.43
DLfinal	**0.93**	**0.81**	**0.95**	**0.57**	**0.73**

Tabel 4: Results on heldout dataset

Model Description :
FLCMT, FastText - baseline models and the results corresponding to it are the baseline results.
DLinter - deep learning model trained on the MT data.
DLlingset - deep learning model in which the LIWC features are appended with the document level features and then used for classification.
DLfinal - final deep learning model, trained on the MT data and then transfer learning on the HT data.

Sentence Classification Task
The main aim of this task was to classify each sentence from the 202 articles using the DL_{final} model, but after getting the annotation from DL_{final} model, the output was biased and most of the sentences were tagged [1,1,1,0,1] and then we realized that since our model was trained on documents having max length of 3000 words and some of the sentences contain 5-6 words after preprocessing (removing stop words, removing punctuations, lemmatizing the verbs and removing the unnecessary characters like '()','[]' etc.) and after concatenating each sequence with unknowns to make it of length 3000 doesn't make sense, so we dropped the idea of annotating the sentences using the DL-final model.
So, we annotated each sentence using the feature summarization algorithm and the topic summarization algorithm.

Quantitative Analysis task on dataset
In this task, the main aim was to analyze the articles and the relationship among the words with the label assigned to them. So, following are my analysis -

- The articles which are assigned to be competent, in those articles generally there is the discussion of the other companies which manufacture the same products, and discussion about they being better than the others.
- In sincerity tagged articles, there is talk about the family and the family product.

Recommendation Tool
Using the previous methods of feature summarization algorithms, we analyzed the texts on the lexicon level and calculated the informative sentences based on the score. We need a method to analyze the texts on the semantic level which will certainly help in improving the recommendation tool. For that we can look into following approaches -

Reference resolution - Using this method we can find about the entities targeted in the texts; company, its employees, its products, products and personalities in similar domain, or miscellaneous, which will certainly help in improving the recommendation system. We have looked at some of the resources available for the task of reference resolution they seem promising to work ahead in the direction.