Personality Detection From Text

Abstract

With the development of Internet and social media, personality detection based on text has more and more important applications nowadays which even includes election campaign. The most popular theory of personality is Big five, also known as OCEAN. It has five traits to measure personality which are Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. In this project, we developed deep-learning based methods for personality detection from text. We implemented four kinds of neural networks for feature extraction and document modeling, namely a hierarchical Convolutional Neural Network (CNN) model, a Long Short-Term Memory (LSTM) model, a hierarchical RNN model with LSTM and max pooling layers, and a hierarchical RNN model with two layers of LSTMs. We merged the features extracted with these four kinds of neural networks with Mairesse baseline feature set to obtain a document vector and then classify it with multilayer perceptron and logistic regression. Using stream-of-consciousness essay dataset, we trained the neural networks with mini-batch gradient descent and have a successful accuracy result.

Keywords: deep-learning, CNN, LSTM, personality detection, nature language processing

1 Introduction

Personality is the combination of characteristics or qualities that form an individual's distinctive character. Everyone have their own special personality traits. Personality is so important because it would affect our whole life, every choice you made, every friend you make, what kind of book you read or movie you watch and so on, also make a role in how other people see you, and how they make every choice related to you. However, personality has been considered as the most difficult human attribute to understand since everyone is special and unique.

Personality detection from text means to extract the behavior characteristics of authors who have written a text. Personality detection is significant because we can make a better decision in many aspects if we know more about ourselves, like career choosing, making friends, avoiding or controlling the shortcoming of some personalities. Also, having the data of customers' personality, accurate advertising is possible, which can save tons of money from inefficient businesses and decrease the cost of commodity.

Nowadays, personality detection already has many applications such as recommender systems, social network analysis, deception detection, authorship attribution, sentiment analysis, predicting job satisfaction, e-learning, information filtering, collaboration and e-commerce by a user interface that adapts the interaction according to user's personality.

1.1 Big Five Model

There are many different theories about personality, but the most popular theory in psychology

academia is called Big Five, also known as OCEAN. Many kinds of research have proved Big Five is the current definitive model of personality. Big Five traits are characterized by the following:

Openness to Experience (inventive/curious vs. consistent/cautious): curious, intelligent, imaginative, creative, artistic. With diverse views, ideas, and experiences. Also known as lack of focus, risky, more likely to take drug.

Conscientiousness (efficient/organized vs. easy-going/careless): responsible, organized, persevering, dependable, self-discipline, like to plan, strong aim. Also known as stubbornness and obsession.

Extraversion (outgoing/energetic vs. solitary/reserved): energetic, outgoing, positive, talkative, assertive, sociable. Also known as attention-seeking and domineering.

Agreeableness (friendly/compassionate vs. challenging/detached): cooperative, helpful, well-temper, easy-going, easy to trust others. Also known as naive or submissive.

Neuroticism (sensitive/nervous vs. secure/confident): anxious, insecure, sensitive, moody, tense, depressive, vulnerable. Also known as reactive, excitable, dynamic person, but they can be perceived as unstable or insecure.

2 Overview of Our Approach

In this project, we have developed deep learning based methods for detecting the personality of an author based on Big Five model. Our work is based on the work of Majumder et al [1] in which the authors propose a hierarchical CNN method for document modeling. Nevertheless, we have developed three new Recurrent Neural Networks based methods for personality detection which are our own work.

Our approach consists of input data preprocessing, sentence filtering, feature extraction and classification. For feature extraction and document modeling we have developed four different methods including a hierarchical CNN, an LSTM, a hierarchical model with LSTM and max pooling layers, and a hierarchical model with two layers of LSTMs. We experiment with each model one at a time for feature extraction.

During feature extraction, words are converted to word vectors using a word embedding matrix. At this stage, sentences are variable-length arrays of word vectors and documents are variable-length arrays of sentences. This variable-length representation is then fed into aforementioned models to obtain sentence vectors and then finally a document vector. This document vector is then concatenated with a document-level stylistic feature set called Mairesse baseline feature set to obtain a final document-level vector used for final classification.

We trained six neural networks with same architecture, one for each of the five personality traits plus the multi-label classification of all the five traits together. Each network was a binary classifier that predicted the corresponding trait to be positive or negative. In case of multi-label, all the five traits are considered together. We used five document modeling methods including the four aforementioned methods as well

as Mairesse baseline feature set as baseline for document modeling. Therefore, we trained 6*5=30 neural networks totally.

Specifically, the steps taken in our approach are as follows as reflected in Figure 1.

- *Preprocessing.* This includes data cleaning as well as sentence splitting and reduction to lower case.
- *Mairesse document-level feature extraction*. This step includes extracting global features such as word count and average sentence.
- *Sentence Filtering.* We remove the sentences that do not include any emotionally charged words. The sentences that do not carry personality clues can be removed for two reasons. Firstly, they represent noise which has a negative effect on the classification results and secondly removing them considerably reduces the input size and training time.
- *Word-level Feature Extraction*. We use GloVe pre trained embeddings to initialize our word embedding matrix and we let the embedding matrix be trained along with other parts of the model. Each word will be represented as a fixed-length word vector.
- *Document-level Feature Extraction.* We use four different document modeling methods as described below. Each method obtains a document feature vector for the whole document. We experiment with one method at a time for feature extraction and report and compare our results in the results section.
    - *LSTM:* This is our simplest model. We try to extract document features without splitting sentences first. After obtaining word vectors, they are all fed into a LSTM layer to generate a document feature vector. It is to be concatenated with Mairesse feature vector for further classification.
    - *Hierarchical RNN model with LSTM and max pooling layers.* In this part, word vectors are fed into an LSTM layer sentence by sentence, resulting in sentence feature vectors. Then, these sentence vectors are fed into a max pooling layer to obtain a document feature vector, which is to be concatenated with Mairesse feature vector for further classification.
    - *Hierarchical RNN model with two layers of LSTMs.* After applying word embedding on words and obtaining word vectors, these word vectors are fed into an LSTM layer one word at a time in each time step to generate a sentence vector for each sentence. At this point, a document is a variable-length representation of sentences. Then each sentence vector is fed into a higher LSTM layer to obtain a document vector. Finally, this vector is concatenated with Mairesse features to obtain the final document vector.
    - *Hierarchical CNN model.* We use a deep CNN in which the layers process the input data in a hierarchical manner. Each word is represented as a fix-length word vector and

sentences are represented as arrays of words. We apply three convolutional filters along with a max pooling layer to reduce a sentence to a fixed-length sentence vector which is a kind of sentence embedding in a continuous vector space. At this point, a document is a variable-length vector of sentence vectors. At a deeper level in the model, this variable-length representation is reduced to a fixed-length vector which represents the whole document. Finally, this vector is concatenated with Mairesse feature set to obtain the final document vector.

- *Classification.* For final classification of text using the obtained document vector, we use a fully connected network and logistic regression.

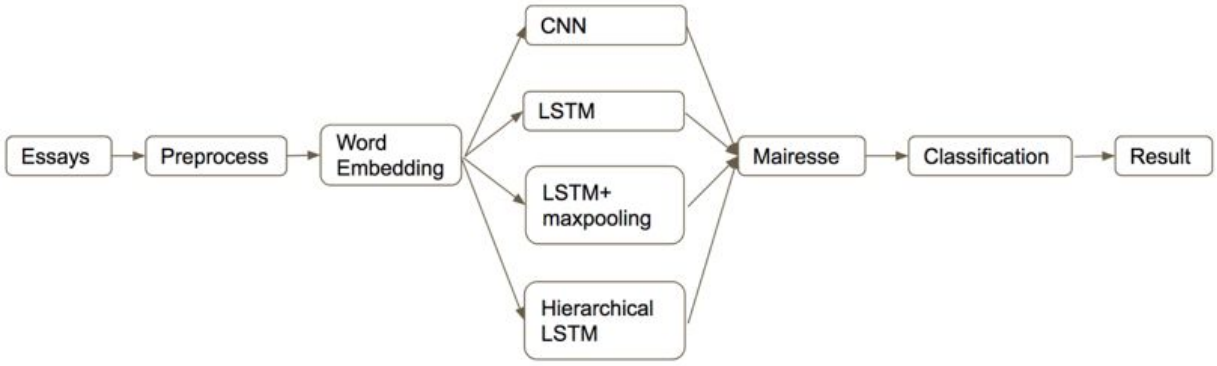

Figure 1: Steps taken in our approach

3 Network Architectures and Implementations

3.1 Dataset

A large number of researchers use the following the datasets for personality detection and currently they are considered the state-of-the-art datasets to test and develop personality models [2]. The most popular labeled datasets available for personality recognition are stream-of-consciousness essay dataset and myPersonality dataset. We used the first data set.

The stream-of-consciousness essay dataset [3] contain 2,468 essays one for each author tagged with the five personality traits: EXT, NEU, AGR, CON, and OPN. The essays were collected between 1997 and 2004 and labeled with personality classes. Text have been produced by students that took the Big5 test. The labels, that are self-assessments, are derived by z-scores computed by Mairesse et al. [4] and converted from scores to nominal classes by authors in [5] with a median split. Figure 2 illustrates a sample essay along with the five labels.

Well, here we go with the stream of consciousness essay. I used to do things like this in high school sometimes. They were pretty interesting, but I often find myself with a lack of things to say. I normally consider myself someone who gets straight to the point. I wonder if I should hit enter any time to send this back to the front. Maybe I'll fix it later. My friend is playing guitar in my room now. Sort of playing anyway. More like messing with it. He's still learning. There's a drawing on the wall next to me. Comic book characters I think, but I'm not sure who they are. It's been a while since I've kept up with comic's. I just heard a sound from ICQ. That's a chat program on the internet. I don't know too much about it so I can't really explain too well. Anyway, I hope I'm done with this by the time another friend comes over. It will be nice to talk to her again. She went home this weekend for Labor Day. So did my brother. I didn't go. I'm not sure why. No reason to go, I guess. Hmm. when did I start this. Wow, that was a long line. I guess I won't change it later. Okay, I'm running out of things to talk about. I've found that happens to me a lot in conversation. Not a very interesting person, I guess. Well, I don't know. It's something I'm working on. I'm in a class now that might help. The phone just rang. Should I get it? The guy playing the guitar answered it for me. It's for my roommate. My suitemate just came in and started reading this. I'm uncomfortable with that. He's in the bathroom now. You know, this is a really boring piece of literature. I never realized how dull most everyday thoughts are. Then again, when you keep your mind constantly moving like this, there isn't really time to stop and think deeply about things. I wonder how long this is going to be. I think it's been about ten minutes now. Only my second line. How sad. Well, not really considering how long these lines are. Anyway, I wonder what I'm going to do the rest of the night. I guess there's always homework to do. I guess we'll see. This seat is uncomfortable. My back sort of hurts. I think I'm going to have arthritis when I get older. I always thought that I wouldn't like to grow old. Not too old, I suppose. I've always been a very active person. I have a fear of growing old, I think. I guess it'll go away as I age gradually. I don't know how well I'd deal with paralysis from an accident though. As long as I have God and my friends around, I'll be okay though. I'm pretty thirsty right now. There isn't much to drink around my room. Ultimate Frisbee, I haven't played that all summer. Fun game, but tiring. I'm out of shape. I'd like to get in better shape, but I hate running. It's too dull for me. Hmmm. it's almost over now. Just a few more minutes. Let's see if I make it to the next line. Short reachable goals! Whatever. Anyway, what else do I have to do tonight. I guess I could read some. My shirt smells like dinner. It's pretty disgusting. I need to wake up for a 9:30 am class tomorrow. I remember when that wasn't early at all. Well, I made it to the next line. I'm so proud of myself. That's sarcasm, by the way. I wonder if I was suppose to right this thing as a narrative. Oh well too late now. Time for me to head out. Until next time, good bye and good luck. I don't know.

cEXT	cNEU	cAGR	cCON	cOPN
n	y	y	n	y

Figure 2: A sample essay from stream-of-consciousness essay dataset along with its five personality traits label

3.2 Preprocessing

Preprocessing included the following steps. We reduce the text to lower-case and remove all characters except ASCII letters, digits, question and exclamation marks, and single and double quotation marks. Documents are split into sentences at the period, question mark and exclamation mark characters and sentences are split into words at whitespace character.

Some sentences do not include the terminating periods and this would cause an absurdly long sentence. If a sentence is longer than 150 words, we split it into shorter sentences of 20 words (except the last sentence which its size might be less than 20).

3.3 Sentence Filtering

We remove the sentences that do not include any emotionally charged words to improve the classification performance and reduce the training time. To obtain the list of emotionally charged words we use NRC Emotion Lexicon. It contains 14,182 words tagged with 10 attributes: anger, anticipation, disgust, fear, joy, negative, positive, sadness, surprise, and trust. We consider a word to be emotionally-charged if it at least contains one these attributes. There are 6,468 such words in the lexicon. Therefore, we remove the sentences that do not have such words before applying the word embedding.

3.4 Word Embedding

3.4.1 Word to Index

To implement vectorization, we first make a one-hot encode on all the words in our dataset. Totally, there are 25461 different words, and each word is assigned a unique index. In this way, one sentence is converted to a vector with variable length, depending on the number of words the sentence has. It is

obvious that different sentence vectors' lengths may not be equal, which is not good for later use. To fix this problem, we find out the largest sentence length among the whole dataset. For those sentences with less words, we add dummy words to them, so all sentences are as long as the largest one. In document level, we also add dummy sentences to short essays to make sure each essay has equal number of sentences.

As mentioned in the preprocessing part, one essay consists of sentences, while one sentence consists of words. After word-to-index implementation, one essay is now converted to a matrix. Its row number is the length of the longest essay, and column number is the length of longest sentences. In our case, the longest essay has 129 sentences, while the longest sentence has 149 words. Thus, one essay is now a 2D matrix with the shape of (129,149).

3.4.2 Word to Vector

From the last step, we have each word converted to a unique index. The next work is to convert word to vector. Here, we use the word-embedding layer in Keras to do the job. For convivence of training, we initialize the weight matrix with pre-trained GloVe.

The GloVe assigns each token word a 300-length vector to represent its feature. For those words not included in tokens, they are treated equally as the token "UNK". Then we build an embedding matrix with GloVe and the word indexes. The vector corresponding to a word with index i would be the ith row in the embedding matrix. We initialize the weight matrix in the embedding layer with this embedding matrix. Each timestep we feed a (129,149) matrix into the embedding layer and get a 3D matrix with the shape of (129,149,300) as result.

3.5 Document-level Feature Extraction

3.5.1 LSTM

Firstly, we try the simplest model to extract the document feature. Instead of splitting essay to sentences, we treat the essay as an extremely long sentence. To normalize, we also add dummy words to short essays. In this case, the longest essay has 2722 words. Therefore, after word vectorization, each essay is converted to a matrix with the shape of (2722,300).

Upon gaining the 2D essay matrix, we feed it into a LSTM layer and extract features directly. For the output feature dimension, we tried with 300/600, and finally decide to use 300. Though 600 features describe the sample better, it takes too long time to train. Compared the performance with 300 features, there is no significant difference. As a tradeoff between training time and good performance, we extract 300 features from each document.

The single LSTM layer extracts a 300-length feature vector for each essay. Later classification would base on this essay-feature vector.

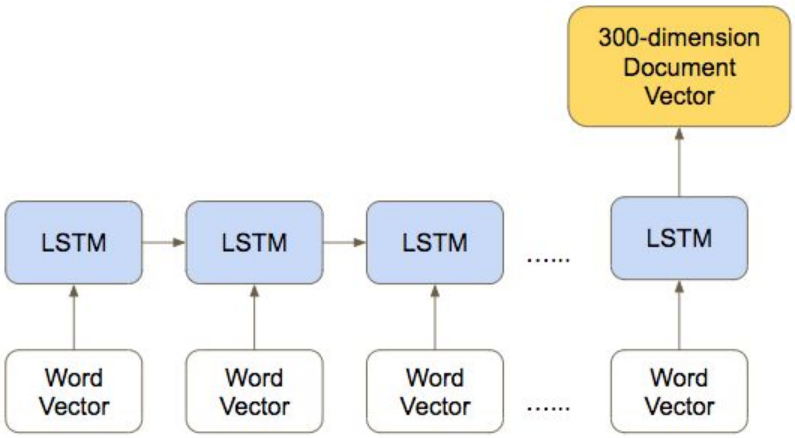

Figure 3: The architecture for RNN model with one single LSTM layer

3.5.2 Hierarchical RNN model with LSTM and max pooling layers

Another method to extract document feature is to split an essay into sentences, then get some features for each sentence. The following step is to extract document feature based on the sentence features. The result is more reliable than the only extract from words, as the split of sentences does tell something about the author's personality.

In the word vectorization part, we have got a 3D essay matrix with the shape of (129,149,300). Since normal LSTM layer can only receive 2D matrix as input, we implement a TimeDisturbed layer here. It splits the 3D matrix into 129 2D matrices, which is able to feed the LSTM layer directly. It is easy to figure out that each 2D result matrix has the shape of (149,300), from which LSTM layer extracts 300-length sentence feature vector. Since there are 129 matrixes, the result can be merged as a 2D matrix with the shape of (129,300).

There are various ways to extract document feature from sentence feature matrix. One method we choose is max pooling. This choice is based on an assumption: a document has some features if at least one of its sentences has this feature. To obtain the document feature, for each of these 300 features, we take the maximum across all the sentences vectors. As a result, we get a 300-length feature vector for the document.

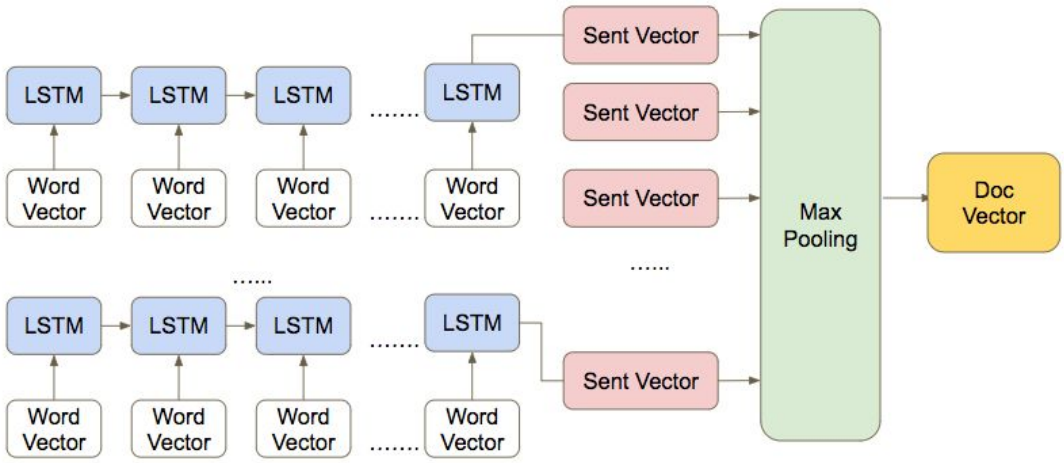

Figure 4: The architecture for hierarchical RNN model with LSTM and max pooling layer

3.5.3 Hierarchical RNN model with two layers of LSTMs

The third model is a more sophisticated hierarchical model with two layers of LSTMs where each layer is responsible for a vectorization task. The first LSTM layer is responsible for sentence vectorization and the second layer is responsible for document vectorization.

Similar to the previous method, we first feed the word vectors obtained from the word embedding matrix into the first LSTM layer which would give out a sentence vector for each of the sentences. In our case, the input vector shape is (129,149,300) and the first layer returns a 300-dimensional vector for each sentence. Hence, a document would be a sequence of size 129 of sentence vectors of size 300. Then in the higher layer, this sequence is fed into the LSTM in the second layer where in each timestep one sentence vector is fed into the LSTM. The output of the second layer is a 300-dimensional vector representing the whole document. Figure 5 demonstrates this process.

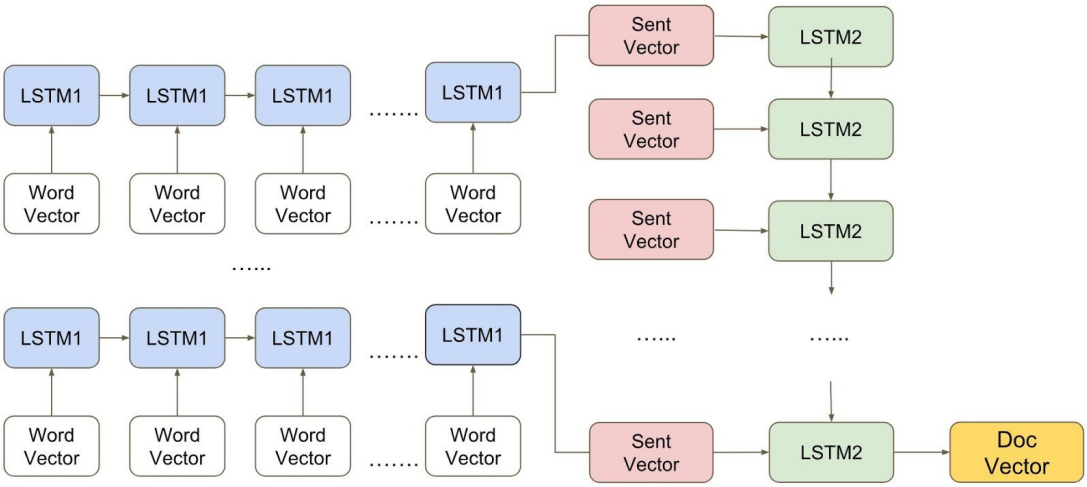

Figure 5: The architecture for the hierarchical RNN model with two layers of LSTMs

3.5.4 Hierarchical CNN model

Our CNN approach is the implementation of the method presented in [1]. After word vectors are obtained using word embedding, three convolutional filters are applied on sentences to obtain unigram, bigram, and trigram features for each sentence. To extract such n-gram feature, a convolutional filter of size nxE is applied to sentences where E denotes the dimension of embedding matrix (300). We use feature maps of size 200 for each of the n-grams where n = 1, 2, 3. Therefore, the convolutional filter applied is of size 200xnxE. We add a bias of size 200 to the output of the filter. Hence, we obtain three vectors of size 200x(W-n+1)x1, n = 1, 2, 3 for each sentence where W denotes the maximum number of words in a sentence (149). Also, in order to introduce nonlinearity, we apply the Rectified Linear Unit (ReLU) function to the vectors as well.

Next, we apply max pooling to these vectors to down-sample them to 200x1x1 vectors and then we flatten them to vectors of size 200. Finally, we concatenate the three n-gram vectors to obtain a vector of size 600 representing a sentence.

In order to obtain a document vector out of the array of sentence vectors, we assume that the document has some feature if at least one of its sentences has this feature. Each sentence is a 600-dimensional vector which has 600 features. We get the maximum of each of these features across all the sentences of a document. Therefore, we will obtain a 600-dimensional vector representing the whole document. Figure 6 illustrates the network architecture for this approach.

We could also apply convolutional filters to obtain n-grams for each sentence and then apply max pooling and concatenate them. However, the paper on which this CNN method is based reported that that such approach did not converge after 75 epochs and used the aforementioned 1-max pooling approach instead. Therefore, we followed the same approach.

3.6 Extracting Mairesse Document-level Features

Mairesse baseline feature set is a document-level feature set which was created by François Mairesse and colleagues [4]. It comprises the features of linguistic inquiry, word count, medical research council, utterance-type, and prosodic. The feature examples include word count, average number of words per sentence, the total number of pronouns, past tense verbs, present tense verbs, future tense verbs, letters, phonemes, syllables, questions, and assertions in whole essays. We concatenate them together as a vector with 84 features for each essay and then we merge it with 300 or 600 features from each neural network to a vector with 384 or 684 features which can present the whole document, respectively.

4 Experimental Results

We trained five networks, all with the same architecture, for each of the personality traits. Each network did binary classification for a corresponding personality trait. We also trained another network with the same architecture for multi-label classification of all the five traits together. We used four document

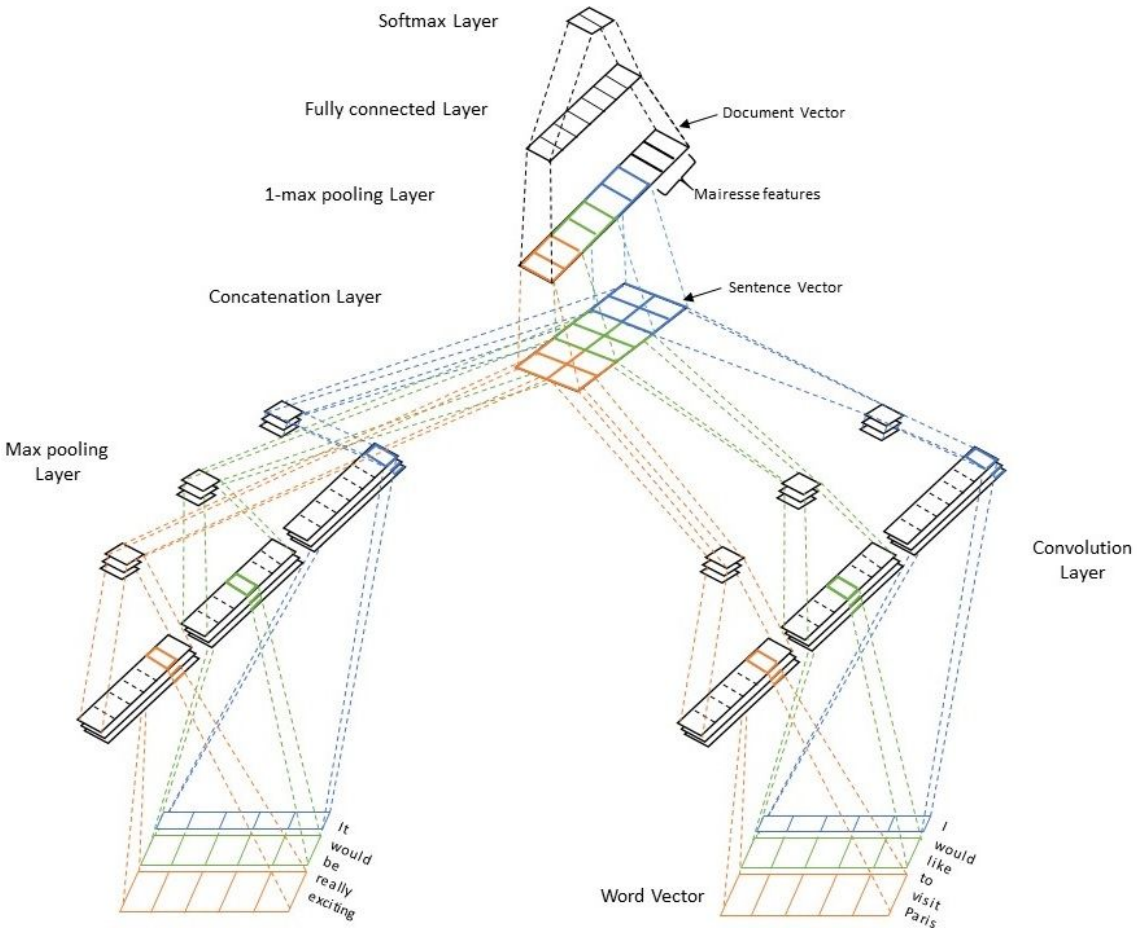

Figure 6: The architecture of the hierarchical CNN model from [1]

modeling methods explained in section 3.5 as well as Mairesse baseline feature set for document modeling. Therefore, we trained 6*5=30 neural networks totally.

Multilayer perceptron (MLP) and logistic regression were used for classification. We splitted the train and test data with 0.7/0.3 ratio and we used Adam algorithm as optimizer with batch size = 200. We used accuracy for the metrics. Table 1 illustrates our accuracy results.

In the table, we also list other published results for comparison. Results from [1] are listed as “Majumder et al” in the table where we list their best result for each of the personality traits. We also list “Published state of the art per trait” [6] which was the best results before Majumder et al published their results. The rest of the table is our results.

Rows represent our accuracy results for different neural network architectures for each trait plus a multi-label classification of all the five traits. Bold indicates the best result for every trait or all five traits. As we can see in the table, our results are better. For EXT, LSTM with logistic regression and

Hierarchical LSTM are the best with 58.57% accuracy. NEU's best is Hierarchical LSTM with MLP classifier. The best for AGR is LSTM with logistic regression. Hierarchical CNN is the best neural network for CON and OPN. For multi-label, LSTM with MLP is the best but we don't have the results from [1] and [6] to be able to compare them since they did not train multi-label. We also can not compare the loss since they did not post that in their papers, but it is obvious our accuracy results are better.

Document Vectorization	Classifier	EXT	NEU	AGR	CON	OPN	Multi-label (All Five)
published state of the art per trait	N/A	56.45	58.33	56.03	56.73	60.68	N/A
Majumder et al	Different	58.09	59.38	56.71	57.30	62.68	N/A
Mairesse	MLP	57.89	54.66	48.58	55.87	57.62	56.87
LSTM	Logistic Regression	**58.57**	56.95	**56.95**	54.93	58.57	53.55
LSTM	MLP	54.12	58.57	54.25	57.62	62.21	**57.14**
LSTM with max pooling	MLP	54.52	59.38	56.01	51.55	56.14	54.06
Hierarchical LSTM	MLP	**58.57**	**60.05**	56.28	56.82	61.54	55.38
Hierarchical CNN	MLP	54.12	55.74	53.85	**57.89**	**63.56**	55.41

Table 1: Our accuracy results for each of the models and each of the personality traits/multi-label

5 Conclusion and Future Work

Texts often indicate various aspects of the author's personality. In this project, we implemented several document modeling methods for personality detection from text. We trained six neural networks with each of these models and published our successful results.

Our training is based on stream-of-consciousness essay dataset. Trying more datasets might even improve our results. Furthermore, we implemented the multi-label neural network with the same architecture as all traits and its design could be enhanced. Moreover, we plan to predict the personality of some authors based on their tweets.

References

1. N. Majumder, S. Poria, A. Gelbukh, and E. Cambria, ''Deep learning-based document modeling for personality detection from text,'' IEEE Intell. Syst., vol. 32, no. 2, pp. 74–79, Mar./Apr. 2017

2. B. Agarwal, “Personality detection from text: A review,” International Journal of Computer System 1, 1, Sep. 2014
3. J.W. Pennebaker and L.A. King, “Linguistic Styles: Language Use as an Individual Difference,” J. Personality and Social Psychology, vol. 77, no. 6, 1999, pp. 1296–1312.
4. Mairesse, F. and Walker, M. A. and Mehl, M. R., and Moore, R, K. “Using Linguistic Cues for the Automatic Recognition of Personality in Conversation and Text,” Journal of Artificial intelligence Research, 30(1), pp: 457–500, 2007.
5. Celli, F., Pianesi, F., Stillwell, D. S., and Kosinski, M. 2013. Workshop on Computational Personality Recognition (Shared Task). The Seventh International AAAI Conference on Weblogs and Social Media. Boston, MA, USA
6. S.M. Mohammad and S. Kiritchenko, “Using Hashtags to Capture Fine Emotion Categories from Tweets,” Computational Intelligence, vol. 31, no. 2, 2015, pp. 301–326.