Orders Are Unwanted: Dynamic Deep Graph Convolutional Network for Personality Detection

Tao Yang[1*], **Jinghao Deng**[1*], **Xiaojun Quan**[1†], **Qifan Wang**[2]

[1] School of Computer Science and Engineering, Sun Yat-sen University
[2] Meta AI
{yangt225, dengjh27}@mail2.sysu.edu.cn, quanxj3@mail.sysu.edu.cn, wqfcr@fb.com

Abstract

Predicting personality traits based on online posts has emerged as an important task in many fields such as social network analysis. One of the challenges of this task is assembling information from various posts into an overall profile for each user. While many previous solutions simply concatenate the posts into a long document and then encode the document by sequential or hierarchical models, they introduce unwarranted orders for the posts, which may mislead the models. In this paper, we propose a dynamic deep graph convolutional network (D-DGCN) to overcome the above limitation. Specifically, we design a learn-to-connect approach that adopts a dynamic multi-hop structure instead of a deterministic structure, and combine it with a DGCN module to automatically learn the connections between posts. The modules of post encoder, learn-to-connect, and DGCN are jointly trained in an end-to-end manner. Experimental results on the Kaggle and Pandora datasets show the superior performance of D-DGCN to state-of-the-art baselines. Our code is available at https://github.com/djz233/D-DGCN.

Introduction

Text-based personality detection is an emerging task in computational psycho-linguistics and affective computing (Jiang, Zhang, and Choi 2019). The objective is to identify one's personality traits based on the texts she/he creates. Personality detection contributes meaningful cues to explaining an individual's behavior, emotion, and motivation (Mehta et al. 2019; Zhang, Peng, and Winkler 2019). The availability of a tremendous amount of social media posts containing users' digital traces provides rich data sources for this task and has aroused the interest of NLP and psychology researchers (Cui and Qi 2017; Xue et al. 2018; Keh, Cheng et al. 2019; Amirhosseini and Kazemian 2020; Lynn, Balasubramanian, and Schwartz 2020).

The taxonomy of personality is generally defined along different dimensions. Figure 1 shows an example of personality detection in the Myers-Briggs Type Indicator (MBTI) taxonomy (Myers-Briggs 1991). Although it can be naturally regarded as a multi-label classification task, personality detection has new challenges. First, the input of this task is not a single document but a set of posts, which are usually topic-agnostic short documents. Second, not every post necessarily contains personality clues, so the key is how to piece together useful information in different posts into a representative user profile. These posts can be arbitrarily merged into a long document and encoded sequentially (Jiang, Zhang, and Choi 2019; Zhou et al. 2019), or they can be encoded separately and then aggregated into a user representation by hierarchical networks (Lynn, Balasubramanian, and Schwartz 2020; Xue et al. 2018). In either case, however, orders are introduced among the posts. Intuitively, these posts should work complementarily to build the user personality profile rather than treated sequentially or hierarchically.

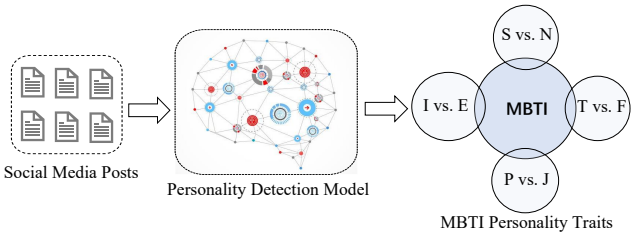


Figure 1: An example of personality detection for MBTI. The input are multiple social media posts from a single user and the output are her/his personality traits along dimensions of ***I**ntroversion* vs. ***E**xtroversion*, ***S**ensing* vs. *i**N**tuition*, ***T**hink* vs. ***F**eeling*, and ***P**erception* vs. ***J**udging*.

The graph is a natural structure to represent the posts in an unordered way, whereas it tends to be non-trivial to predefine the implicit personality-aware connections between posts as previous work of graph neural networks. In this paper, we propose a novel graph-based post fusion model, namely dynamic deep graph convolutional network (D-DGCN), to address the above issues. D-DGCN builds a graph to represent the posts of a user, in which each post is represented as a node and initialized by the embedding from a pre-trained language model such as BERT (Devlin et al. 2018). A special node denoting the user is also added to facilitate personality classification. Moreover, considering that the connection between two nodes is not established deterministically, and inspired by GraphMask (Schlichtkrull, De Cao, and Titov

[*]Equal contribution.
[†]Corresponding author.

2020), we propose a novel multi-hop-based learn-to-connect (L2C) module with a differentiable threshold function to automatically build the essential edges layer by layer. L2C enables the model to be more adaptive and scalable to different samples. Then, in light of the problem that conventional graph convolutional networks (GCNs) cannot stack too many layers because of the over-smoothing issue, a deep graph convolutional network (DGCN) (Liu, Gao, and Ji 2020) is applied to gather useful information from the posts from larger receptive fields. DGCN remedies over-smoothness by decoupling the transformation and propagation operations in GCNs, allowing the network to stack deeper. Extensive experiments are conducted on the Kaggle and Pandora datasets, and the results show that D-DGCN outperforms the baseline methods. Besides, extensive ablation studies and analysis further demonstrate that the L2C and DGCN modules play an indispensable role in the performance boosts.

The contributions are summarized as follows.

- We propose a novel D-DGCN to tackle the post order issue, which is critical in personality detection.
- The proposed D-DGCN contains two interactive modules, L2C and DGCN, that interact intimately to learn how to establish connections between nodes layer by layer, enabling an unordered fusion of post information.
- We conduct extensive experiments on two benchmarks and demonstrate that the proposed D-DGCN outperforms all baselines and establishes a new state of the art.

Related Work

Personality Detection

Traditional work on personality detection relies heavily on feature engineerings (Yarkoni 2010; Schwartz et al. 2013), such as extracting psycholinguistic features from Linguistic Inquiry and Word Count (LIWC) (Pennebaker, Francis, and Booth 2001). These features are then fed into such machine-learning models as support vector machines (Cui and Qi 2017) and XGBoost (Amirhosseini and Kazemian 2020). Recently, deep learning methods have dominated the research of personality detection and various post encoding methods have appeared. Jiang, Zhang, and Choi (2019) simply concatenated all the posts from a single user into a document and fed it into a pre-trained language model. Xue et al. (2018) and Gjurković et al. (2020) used CNN to aggregate the post representations. Lynn, Balasubramanian, and Schwartz (2020) and Wang et al. (2021) adopted a hierarchical attention network to generate the user representation from posts. However, these approaches introduce unnecessary orders into the posts, which are likely to be captured by the models and affect their generalization ability. Although Yang et al. (2021a) tried to store posts in the memory of Transformer-XL (Dai et al. 2019) to fix the order issue, the interactions between posts entirely rely on the self-attention mechanism (Vaswani et al. 2017), which is prone to introduce irrelevant information.

Graph Convolution Network

Graph Convolution Network (GCN) (Kipf and Welling 2016), which is a special kind of graph neural networks (GNNs), learns node feature representation by iteratively aggregating features from neighbors in a convolutional way. Although it has achieved notable success in many applications, GCN faces the problem of over-smoothness, resulting in rapid performance drops as the layers stack. Encouraging efforts have been made to address this issue. Wu et al. (2019) proposed a simple SGC model by reducing unnecessary complexity in GCN. Chen et al. (2020) designed two metrics to quantize the smoothness and over-smoothness in node representations. Liu, Gao, and Ji (2020) provided a theoretical analysis of this problem and came up with a deep GNN, namely DAGNN, by decoupling the propagation and transformation processes. Inspired by their work, our model employs a similar strategy to overcome the over-smoothing problem.

Graph Construction

The graph can be constructed statically or dynamically in NLP tasks (Wu et al. 2021). The static approach constructs the graph during preprocessing, which leverages linguistic knowledge and manually defined rules such as dependency parse trees (Wang et al. 2020), knowledge graphs (Xie et al. 2020; Yang et al. 2021c), and co-occurrence and document-word relations (Yao, Mao, and Luo 2019). A graph constructed by the static approach is usually fixed and the structure cannot be optimized during graph representation learning, which could be sub-optimal. The dynamic approach learns the graph during training and the structure can be optimized end-to-end. Prior works construct dynamic graphs via similarity metrics (Chen, Wu, and Zaki 2019a, 2020) or attention mechanisms (Chen, Wu, and Zaki 2019b), and have shown the effectiveness of this approach in different tasks.

Model Architecture

We first formally define the personality detection task. Given a set $P = \{p_1, p_2, \ldots, p_N\}$ of posts by a user, where $p_i = [w_{i1}, w_{i2}, \ldots, w_{iM}]$ is the i-th post with M tokens, the objective is to predict the personality traits $Y = \left\{y^{(1)}, y^{(2)}, ..., y^{(T)}\right\}$ of this user along T different dimensions. As the overall architecture shows in Figure 2, our D-DGCN mainly includes a pre-trained language model as post encoder and two interactive modules, namely learn-to-connect (L2C) and deep graph convolutional network (DGCN). The prediction of each personality dimension can be regarded as an individual classification task. Except for the shared post encoder, each classification task has respective L2C and DGCN modules so as to model trait-specific features. In the following, we take one of the classification tasks as an example to introduce D-DGCN.

Post Encoder

We employ BERT (Devlin et al. 2018) to encode each post separately. Note that applying BERT directly to personality detection with out-of-domain data may harm its performance. Previous work finds that incorporating domain knowledge into BERT is helpful to address the domain adaptation challenge (Gururangan et al. 2020). Therefore, we first post-train BERT via masked language model (MLM) on the training sets of Kaggle and Pandora.

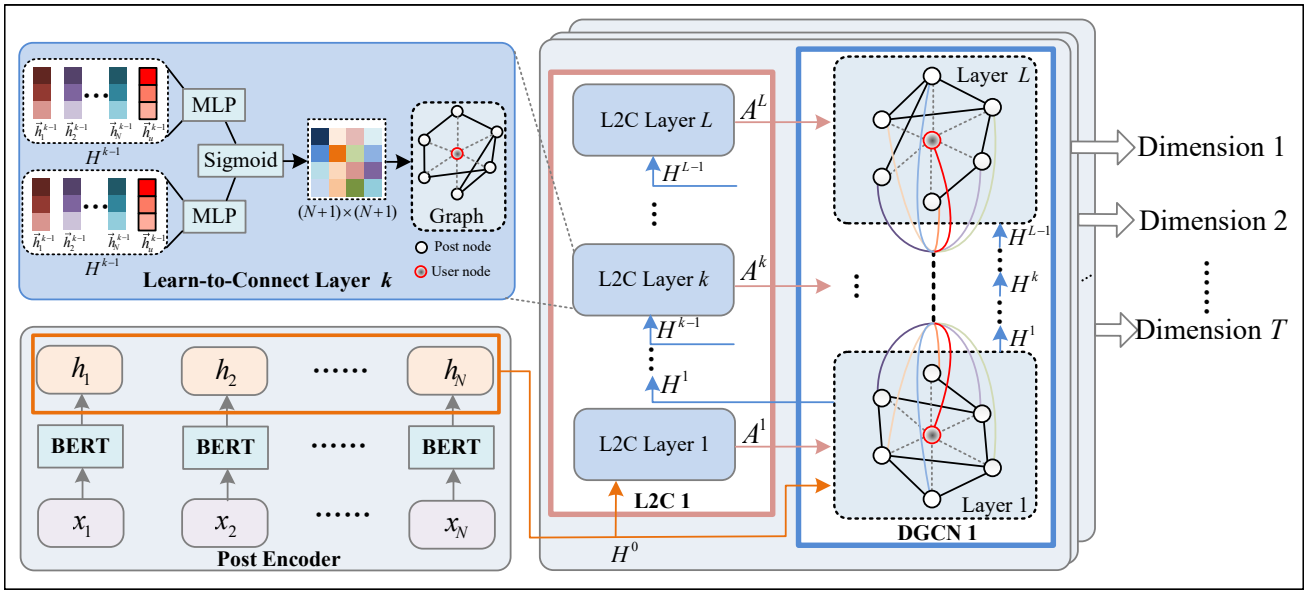


Figure 2: Architecture of our D-DGCN, which comprises a shared post encoder, T L2C modules and T DGCNs. The T L2C modules and the T DGCNs are parallelized to model T personality dimensions.

Formally, the context vector $\vec{h}_i$ for the i-th post p_i is obtained by:

$$\vec{h}_i = \text{BERT}\left(p_i\right) \in \mathbb{R}^{1\times d} \tag{1}$$

where $\text{BERT}\left(\cdot\right)$ denotes the final hidden state of "[CLS]" in post-trained BERT, and d is the dimension of the output. As a result, we obtain a set of contextual representations $\hat{H} = \{\vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N\}$ for the given N posts of a user.

Learn to Connect

After getting all the post representations $\hat{H}$ for a user, we then embark on how to fuse them into a user profile representation. Unlike previous studies that generally fuse them sequentially or hierarchically, we employ a more plausible method, namely deep graph convolutional network (DGCN), to fuse them in an unordered way. Specifically, we first represent each user by a graph and each of his posts by a node in the graph. To generate a user representation facilitating personality classification, we put a special user node u in the graph to aggregate information from other nodes. Therefore, the initial representations of all the nodes become $H = \{\vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N, \vec{h}_u\} \in \mathbb{R}^{(N+1)\times d}$, where $\vec{h}_u$ is the special node representation initialized by averaging the node representations in $\hat{H}$.

Next, we introduce how to construct the edges of the graph in a learning approach. Since it is unclear how to define the connections between posts in favor of personality detection, inspired by GraphMask (Schlichtkrull, De Cao, and Titov 2020), we adopt a learning approach to establish the connections based on proper node representations. Specifically, we propose a dynamic graph learning module L2C with L layers. As shown in Figure 2, each layer of L2C tries to adjust the learned graph dynamically based on the output of the previous layer of DGCN, and then passes the new graph back to DGCN to update node representations for the next layer.

Formally, the adjacency matrix A^k used to represent the graph in the k-th layer is calculated as:

$$A^k = \text{L2C}\left(H^{k-1}\right) \in \mathbb{R}^{(N+1)\times(N+1)} \tag{2}$$

where H^{k-1} denotes the node representations produced by the $(k-1)$-th layer of DGCN, and $\text{L2C}\left(\cdot\right)$ is the function to determine whether there is an edge between two nodes. As shown in Figure 2, two MLPs are introduced to implement this function. Letting H^{k-1} represent the query and key matrices, the L2C module computes an adjacency weight, r_{ij}^k, between nodes i and j as:

$$r_{ij}^k = \sigma\left(Relu\left(\vec{h}_i^{k-1}W_Q^k\right)\left(\vec{h}_j^{k-1}W_K^k\right)^{\mathrm{T}}\right) \tag{3}$$

where σ is the sigmoid function, $\vec{h}_i^{k-1}$ and $\vec{h}_j^{k-1}$ are the representations of nodes i and j in the $(k-1)$-th layer of DGCN, $W_Q^k \in \mathbb{R}^{d\times hid}$ and $W_K^k \in \mathbb{R}^{d\times hid}$ are layer-specific linear transformations, and d and hid are the dimensions of post representation and hidden state of MLP, respectively.

After obtaining the adjacency weight, a differentiable threshold function is utilized to determine whether there is an edge between two nodes, giving rise to the adjacency matrix A^k for this layer. We implement the differentiable threshold function via a programming trick. Specifically, the implementation details can be summarized as follows. First, the adjacency weight r_{ij}^k is normalized by a scaling factor that is almost equal to its own value and has no gradient:

$$\hat{a}_{ij}^k = \frac{r_{ij}^k}{\hat{r}_{ij}^k + \varepsilon} \tag{4}$$

where $\hat{r}_{ij}^k$ means detaching the gradient from r_{ij}^k during the training stage and $\varepsilon = 1e-6$ is used to prevent overflow.

Then, we use a mask matrix to produce the final adjacency matrix. The element a_{ij}^k in A^k is calculated by:

$$a_{ij}^k = \begin{cases} \hat{a}_{ij}^k & r_{ij}^k > \mu \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where μ is a threshold and is set to 0.5 naturally. To better help the L2C module remove unnecessary edges, we introduce ℓ_0 norm with Hard Concrete distribution (Louizos, Welling, and Kingma 2017) to minimize the number of graph connections:

$$\ell_0 = \sum_{k=1}^{L} \sum_{(i,j)\in A^k} a_{ij}^k. \tag{6}$$

Note that this implementation of the differentiable threshold function can regularize the gradients to some extent so that L2C can be fine-tuned layer by layer instead of being adjusted arbitrarily. In this way, L2C can automatically learn to establish connections between nodes effectively. The adjacency matrix A^k is then fed into DGCN to generate the node representations for the $(k+1)$-th layer.

Deep Graph Convolutional Network

After computing the adjacency matrix dynamically, post embeddings are fed into GCNs for encoding. As mentioned above, over-smoothness may limit the performance of our L2C module. To address this, we apply DGCN (Liu, Gao, and Ji 2020), which alleviates the issue by decoupling the transformation and propagation operations in Eq. (8), allowing the L2C module to stack deeper so that the learned graph can be fine-tuned better, and reducing the parameters of GCNs to prevent overfitting. Specifically, GCNs perform propagation and transformation as follows:

$$H^{k+1} = f\left(\hat{A}H^k W^k\right) \tag{7}$$

where $\hat{A} = D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix. For DGCN, it updates the node representations as follows:

$$H^{k+1} = \hat{A}^k H^k. \tag{8}$$

Compared to the original formula of GCNs in Eq. (7), Eq. (8) doesn't have learnable transformation matrix W^k and contains only the propagation operation. After L layers of iteration, we obtain the node representations for every layer:

$$\mathbf{H} = \left[H^0, H^1, \cdots, H^L\right] \in \mathbb{R}^{(N+1)\times(L+1)\times d} \tag{9}$$

where $H^0 = H$ are the initial node representations. Intuitively, $\mathbf{H}$ contains node information from both low and high layers. Then, a trainable projection vector $\vec{c} \in \mathbb{R}^{d\times 1}$ is employed to determine which layers are more useful:

$$S = \sigma\left(\mathbf{H} \cdot \vec{c}\right) \in \mathbb{R}^{(N+1)\times(L+1)\times 1} \tag{10}$$

$$\tilde{S} = \mathrm{Reshape}\left(S\right) \in \mathbb{R}^{(N+1)\times 1\times(L+1)} \tag{11}$$

where σ is the sigmoid function, and $\mathrm{Reshape}\left(\cdot\right)$ is used to reshape a matrix for further computation. The eventual node representations H^{out} are obtained by:

$$H^{out} = \tilde{S} \odot \mathbf{H} \in \mathbb{R}^{(N+1)\times d} \tag{12}$$

where $\odot$ is the multiplication of matrices of the last two dimensions.

Objective Function

As mentioned in Learn to Connect, the output H^{out} of DGCN contains a special node representation $\vec{h}_u^{out}$ for a user. Based on $\vec{h}_u^{out}$, we employ a linear transformation followed by a softmax function to predict each personality trait:

$$y = \mathrm{softmax}(\vec{h}_u^{out} W_u + b_u) \tag{13}$$

$$\ell_{ce} = \frac{1}{V}\sum_{i=1}^{V}\sum_{j=1}^{T}\left[-y_i^j \log p\left(y_i^j|\theta\right)\right] \tag{14}$$

where $W_u \in \mathbb{R}^{d\times 2}$ is a trainable weight matrix, b_u is a bias term, V is the number of training samples, y_i^j is the true label of the j-th personality dimension, and $p(y_i^j|\theta)$ is the predicted probability for this dimension under parameters θ. We use the cross-entropy loss function for all the T personality traits. The whole objective function is defined as:

$$\ell_{total} = \lambda\ell_{ce} + \sum_{i=1}^{V}\sum_{j=1}^{T}\ell_0 \tag{15}$$

where λ is an adaptive Lagrange multiplier. In this way, L2C and DGCN can be jointly optimized.

Experiments

In this section, we introduce the settings of our experiments and report the overall results.

Datasets

Following previous studies (Gjurković et al. 2020; Yang et al. 2021a,b,c), we choose the Kaggle[1] and Pandora[2] MBTI datasets for our evaluations. While the former is collected from PersonalityCafe[3], with 45-50 social media posts included for each of 8675 users, the latter is collected from Reddit[4], with dozens to hundreds of social media posts for each of 9067 users. As previous work (Yang et al. 2021a,b,c), we remove words that match any personality type from the posts. Since the two datasets are severely imbalanced, we employ the *Macro-F1* metric to measure the performance. Table 1 shows the distribution and split of personality and amount of used posts in the two datasets.

Baselines

To make comprehensive evaluations and comparisons, we adopt the following models as our baselines:

- **SVM** (Cui and Qi 2017) and **XGBoost** (Amirhosseini and Kazemian 2020): The posts of a user are firstly concatenated into a document. Then, a SVM or XGBoost is employed for personality classification based on features extracted using bag-of-words methods.

[1] https://www.kaggle.com/datasnaek/mbti-type
[2] https://psy.takelab.fer.hr/datasets/all
[3] http://personalitycafe.com/forum
[4] https://www.reddit.com

Dataset	Types	Train	Validation	Test
Kaggle	I / E	4011 / 1194	1326 / 409	1339 / 396
	S / N	610 / 4478	222 / 1513	248 / 1487
	T / F	2410 / 2795	791 / 944	780 / 955
	P / J	3096 / 2109	1063 / 672	1082 / 653
	Posts	246794	82642	82152
Pandora	I / E	4278 / 1162	1427 / 386	1437 / 377
	S / N	727 / 4830	208 / 1605	210 / 1604
	T / F	3549 / 1891	1120 / 693	1182 / 632
	P / J	3211 / 2229	1043 / 770	1056 / 758
	Posts	523534	173005	174080

Table 1: Statistics of the Kaggle and Pandora datasets.

- $\mathbf{LSTM_{mean}}$ (Cui and Qi 2017) and $\mathbf{BERT_{mean}}$ (Keh, Cheng et al. 2019): BiLSTM or BERT is firstly utilized to encode each post, and the averaged post representation is used to represent each user.
- $\mathbf{BERT_{concat}}$ (Jiang, Zhang, and Choi 2019): This method simply concatenates the posts of a user into a long document and feeds it into BERT.
- $\mathbf{BERT_{CNN}}$ (Gjurković et al. 2020) and $\mathbf{BERT_{LSTM}}$: The two methods are similar to $\mathrm{BERT_{mean}}$ but utilize CNN or LSTM to fuse the encoded posts.
- $\mathbf{BERT_{att}}$: This method is similar to $\mathrm{BERT_{mean}}$, except that the post representations are summarized by attention instead of the mean pooling strategy.
- **SN+Attn** (Lynn, Balasubramanian, and Schwartz 2020): This method employs a hierarchical attention network with both word-level and post-level attentions.
- **Transformer-MD** (Yang et al. 2021a): Transformer-MD encodes the posts sequentially by BERT and stores them in memory, which allows posts to access the information of former ones.
- **TrigNet** (Yang et al. 2021c): TrigNet applies a modified GAT to fuse the posts and constructs the graph with psycholinguistic knowledge in LIWC.

Implementation Details

All the deep learning models are implemented in PyTorch and trained with Adam (Kingma and Ba 2014). We set the maximum length of a post to 70 for both datasets and the maximum number of posts to 50 for Kaggle and 100 for Pandora. For non-pretrained baselines, we use 300-dimensional GloVe word embeddings (Pennington, Socher, and Manning 2014). For pre-trained models, BERT is first initialized by BERT-BASE-CASED (Devlin et al. 2018) and then post-trained via standard MLM on two million posts selected from the training sets of Kaggle and Pandora with learning rate 6e-5 and batch size 256.

For the L2C module, we set the learning rate of MLPs to 1e-5. We set the learning rate of λ in Eq. (15) to 1e-2, and initialize λ to 5.0 and limit its value between 0.0 to 100.0. After tuning on the validation dataset, we set the learning rate of BERT to 1e-5 and the dropout to 0.1, while the learning rate and dropout of other components are set to 1e-3 and 0.2, respectively. The hidden size of BiLSTM is set to 300. The depth L of DGCN firstly changes from 1 to 6 with step size 1 and then changes from 6 to 24 with step size 3. We train D-DGCN for 25 epochs with 3 random seeds.

Overall Results

Table 2 presents the overall results, in which the baselines are organized into four groups: shallow models (SVM and XGBoost), non-pretrained model ($\mathrm{LSTM_{mean}}$ and SN+Attn), pre-trained models with simple methods ($\mathrm{BERT_{concat}}$, $\mathrm{BERT_{CNN}}$, $\mathrm{BERT_{LSTM}}$, $\mathrm{BERT_{mean}}$, and $\mathrm{BERT_{att}}$) and with complex methods (Transformer-MD and TrigNet). We also present two types of our model D-DGCN and D-DGCN+ℓ_0, whose objective functions are Eq. (14) and Eq. (15), respectively. Our observations from the table are as follows. First, our models achieve the highest average F1 scores, verifying the effectiveness of our D-DGCN. Specifically, in the pre-trained setting, compared with traditional $\mathrm{BERT_{att}}$ and SN+Attn, D-DGCN achieves significant boosts in average F1 on the Kaggle and Pandora datasets. The boosted performance comes from two aspects: strong DGCN without introducing unnecessary order and a well-defined graph learning method. Compare with the two complex methods, D-DGCN and D-DGCN+ℓ_0 also get better performance and show their strong competitiveness. Second, the result of D-DGCN family models shows the difference between the two datasets. D-DGCN+ℓ_0 gets better performance in Kaggle, revealing that posts of Kaggle contain more noise. While D-DGCN performs better on Pandora, it seems that L2C should preserve more information about Pandora.

Surprisingly, the shallow models SVM and XGBoost achieve comparable performance with certain deep learning models such as $\mathrm{LSTM_{mean}}$, and even $\mathrm{BERT_{concat}}$ and $\mathrm{BERT_{CNN}}$, showing that deep models are not omnipotent in personality detection and should be combined with appropriate encoding methods. Furthermore, among the five pre-trained models with simple methods, $\mathrm{BERT_{att}}$ and $\mathrm{BERT_{mean}}$ don't introduce order information and achieve higher scores, verifying our proposition.

Analysis

In this section, we conduct extensive evaluations and provide thorough analysis and discussions.

Impact of Order

To investigate the impact of post order, we conduct experiments with two representative order-sensitive models, namely $\mathrm{BERT_{concat}}$ (sequential) and $\mathrm{BERT_{LSTM}}$ (hierarchical). We randomly disrupt the initial post orders (by publication time) and re-train the two models ($\mathrm{BERT_{concat/rd}}$ and $\mathrm{BERT_{LSTM/rd}}$) for five times. The results in Table 4 show that $\mathrm{BERT_{concat/rd}}$ and $\mathrm{BERT_{LSTM/rd}}$ do not become worse but perform better than $\mathrm{BERT_{concat}}$ and $\mathrm{BERT_{LSTM}}$, suggesting that the initial post orders are non-essential information for personality detection.

Ablation Studies

Module Comparison Our D-DGCN comprises two main modules: L2C and DGCN, which represent the graph learn-

Methods	Kaggle					Pandora				
	I / E	***S / N***	***T / F***	***P / J***	**Avg**	***I / E***	***S / N***	***T / F***	***P / J***	**Avg**
SVM	53.34	47.75	76.72	63.03	60.21	44.74	46.92	65.37	56.32	53.34
XGBoost	56.67	52.85	75.42	65.94	62.72	45.99	48.93	66.38	55.55	54.21
$\text{LSTM}_{\text{mean}}$	57.82	57.87	69.97	57.01	60.67	48.01	52.01	63.48	56.12	54.91
SN+Attn	62.34	57.08	69.26	63.09	62.94	54.60	49.19	61.82	53.64	54.81
$\text{BERT}_{\text{concat}}$	58.33	53.88	69.36	60.88	60.61	54.22	49.15	58.31	53.14	53.71
$\text{BERT}_{\text{LSTM}}$	58.12	51.44	70.02	55.92	58.88	52.70	47.92	62.27	49.97	53.22
BERT_{CNN}	58.17	53.87	75.66	54.05	60.44	50.08	51.34	61.72	51.33	53.62
$\text{BERT}_{\text{mean}}$	63.50	55.34	78.55	66.06	65.86	53.35	50.56	64.06	56.83	56.20
BERT_{att}	63.76	58.32	77.99	65.42	66.37	56.03	53.81	67.47	58.57	58.97
TrigNet*	**69.54**	67.17	79.06	67.69	70.86	56.69	55.57	66.38	57.27	58.98
Transformer-MD*	66.08	**69.10**	79.19	67.50	70.47	55.26	**58.77**	69.26	**60.90**	61.05
D-DGCN	68.41	65.66	79.56	67.22	70.21(70.33)	**61.55**	55.46	**71.07**	59.96	**62.01(62.49)**
D-DGCN+ℓ_0	69.52	67.19	**80.53**	**68.16**	**71.35(71.59)**	59.98	55.52	70.53	59.56	61.40(61.50)

Table 2: Overall results of our D-DGCN and baseline models in Macro-F1 (%) score. * means the results are cited from original papers, and both reported the highest results. We report the mean score of D-DGCN after running with three random seeds. We also report the highest average-F1 scores in the bracket for making fair comparisons with existing SOTA. Best results are highlighted in bold.

Methods	Kaggle					Pandora				
	I / E	***S / N***	***T / F***	***P / J***	**Avg**	***I / E***	***S / N***	***T / F***	***P / J***	**Avg**
D-DGCN/MTGNN	67.43	62.98	78.10	**67.96**	69.12	58.94	47.36	68.06	59.05	58.35
D-GCN	64.62	62.23	79.01	64.77	67.66	55.67	55.00	69.19	57.51	59.34
D-GAT	66.28	63.74	**80.93**	67.54	69.62	60.34	53.12	68.18	59.43	60.27
DGCN/fix	66.49	62.60	78.86	61.90	67.46	58.87	53.71	63.76	53.90	57.56
D-DGCN/single-hop	67.58	60.08	81.53	67.09	69.07	59.87	55.66	69.36	57.81	60.67
D-DGCN/undirected	**69.17**	59.76	78.15	67.68	68.69	60.41	**56.26**	69.60	58.39	61.16
D-DGCN/-u	66.85	64.14	79.14	64.76	68.72	58.28	54.51	68.83	56.44	59.51
D-DGCN/-DAPT	67.37	64.47	80.51	66.10	69.61	58.28	55.88	68.50	57.72	60.10
D-DGCN	67.22	**65.81**	80.57	66.81	**70.11**	**62.31**	55.45	**70.32**	**59.51**	**61.90**

Table 3: Results of ablation studies in Macro-F1 (%) score, where different fusion methods and different settings of D-DGCN are implemented for comparison.

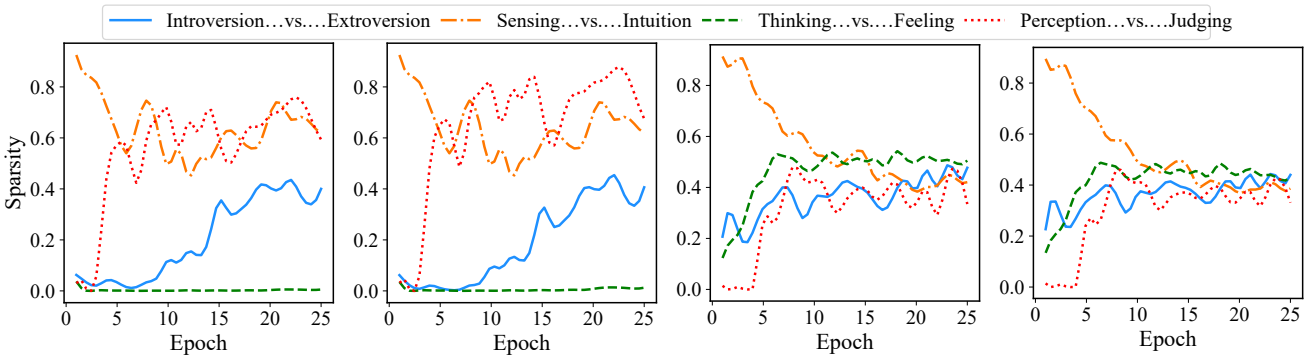
Methods	Kaggle	Pandora
$\text{BERT}_{\text{concat}}$	60.61	53.71
$\text{BERT}_{\text{concat/rd}}$	**61.30**	**54.92**
$\text{BERT}_{\text{LSTM}}$	58.88	53.22
$\text{BERT}_{\text{LSTM/rd}}$	**59.41**	**53.34**

Table 4: Results of different post orders in averaged Macro-F1 (%) score. Merging posts with random orders (/rd) performs better than initial orders.

ing method and graph neural network. To demonstrate our model's superiority, we replace the two modules respectively. We replace L2C module with MTGNN (Wu et al. 2020) (D-DGCN/MTGNN) first. From Table 3, it is clear that the new model D-DGCN/MTGNN underperforms D-DGCN by a considerable margin, showing the power of L2C in modeling implicit personality connections. To make a comparison among different graph neural networks, we replace DGCN with GCN (D-GCN) and GAT (D-GAT) in order. As the result shows, D-GCN suffers from over-smoothing issues and gets poorer performance. GAT (Veličković et al. 2018) does not have over-smoothing issues and is able to adjust the graph connection dynamically, which is an excellent substitute for GCN. However, D-GAT is still inferior to D-DGCN.

Graph Construction Strategies The proposed L2C module relies on self-attention with a multi-hop structure and a differentiable threshold function to construct a directed and non-weighted graph dynamically. To investigate the effect of these factors, we make extensive comparisons from the following aspects:

Pre-defined Graph vs. Learned Graph The main difference between many previous graph-based models and our



(a) Graph level on Kaggle (b) Special node level on Kaggle (c) Graph level on Pandora (d) Special node level on Pandora

Figure 3: Curves of edge sparsity at the graph level and the special node (u) level on Kaggle and Pandora. The sparsity is calculated by the ratio of valid edges to total edges that can produce connections. Here, we set the layer of D-DGCN to 1 for the sake of simplicity.

D-DGCN is that the former requires us to build a graph in advance. To verify the effectiveness, we implement DGCN/fix on pre-defined and fixed graphs constructed based on cosine similarities between posts that are encoded by Sentence-BERT (Reimers and Gurevych 2019). We also implement a degenerating model of D-DGCN, D-DGCN/single-hop, which only learns the graph in the first layer. The two variants both share the graph in all layers. As shown in Table 3, DGCN/fix underperforms D-DGCN/single-hop apparently, implying that high-quality personality connections between posts are difficult to pre-define. Moreover, D-DGCN performs better than D-DGCN/single-hop, indicating that a single-hop structure is not enough to learn an effective graph for personality detection.

Undirected Graph vs. Directed Graph The graph learned by our L2C is directed since the connections established by Eq. (3) is asymmetric. To explore the necessity of a directed graph, we implement the undirected one, D-DGCN/undirected, by constraining the learned adjacency matrix to symmetry. As the result shows, D-DGCN/undirected shows poorer performance than the directed one (D-DGCN). The reason is probably that the directed graph contains richer personality information than the undirected graph.

Pooling vs. Special Node Our D-DGCN inserts a special node u in the graph to aggregate information and directly uses its final representation for classification. We experimentally compare it with the conventional pooling approach (D-DGCN/-u) which averages all the node representations in the last layer. As shown in Table 3, D-DGCN with the special user node outperforms D-DGCN/-u, confirming the effectiveness of inserting the special user node.

Domain Adaptivity Recall that we use the training sets of Kaggle and Pandora for domain adaptive post-training of our D-DGCN model. To verify the effectiveness and necessity of this post-training, we conduct an experiment by replacing the post encoder with the original BERT (D-DGCN/-DAPT). As expected, the results in Table 3 show that the performance declines to some extent on the two datasets. This implies that the original BERT does not fully learn transferable personality knowledge during pre-training, making it worthwhile to collect more unannotated social media texts and design personality-related pre-training tasks in future work.

Sparsity Analysis

To investigate the amount of required information and learning difficulty of different personality dimensions and datasets, we visualize the sparsity curves in terms of different personality dimensions during the training process. For each dataset, two set of sparsity curves, i.e., the percentage of valid edges at the graph level and the percentage of valid edges at the special node (u) level, are plotted in Figure 3. From Figure 3 (a) and (c), we note that the curves at the graph level converge to different percentages, illustrating that different personality dimensions need different amounts of information. In addition, Figure 3 (b) and (d) show that the curves at the special node level are close to the graph ones (Figure 3 (a) and (c)), demonstrating the consistency between the special node and the whole graph. Finally, Figure 3 (a) and (b) show that the dimension (T vs. F) remains sparse during training on Kaggle, and it gets the highest F1 score among the four dimensions in Table 2, implying that D-DGCN can easily distinguish whether a person is thinking or feeling.

Conclusion

In this paper, we presented an unordered post fusion model, D-DGCN, for personality detection. It uses a deep graph convolutional network to try to piece together information from multiple posts into an overall user profile. Unlike previous work that generally requires deterministic graphs to be pre-defined, D-DGCN employs a learn-to-connect approach that learns to build the graphs. Experimental results on two datasets show D-DGCN outperforms the baseline models. Moreover, we conducted extensive ablation studies and analysis to verify the effectiveness of the L2C and DGCN modules. To sum up, proper connection of posts is essential information in personality detection, but not order. Finally, developing personality-related pre-training tasks is a promising direction for future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62176270), the Guangdong Basic and Applied Basic Research Foundation (No. 2023A1515012832), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (No. 2017ZT07X355), and the Foundation of Key Laboratory of Machine Intelligence and Advanced Computing of the Ministry of Education.

References

Amirhosseini, M. H.; and Kazemian, H. 2020. Machine Learning Approach to Personality Type Prediction Based on the Myers–Briggs Type Indicator®. *Multimodal Technologies and Interaction*, 4(1): 9.

Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; and Sun, X. 2020. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. In *AAAI*, 3438–3445.

Chen, Y.; Wu, L.; and Zaki, M. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in Neural Information Processing Systems*, 33: 19314–19326.

Chen, Y.; Wu, L.; and Zaki, M. J. 2019a. Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension. *arXiv preprint arXiv:1908.00059*.

Chen, Y.; Wu, L.; and Zaki, M. J. 2019b. Reinforcement learning based graph-to-sequence model for natural question generation. *arXiv preprint arXiv:1908.04942*.

Cui, B.; and Qi, C. 2017. Survey analysis of machine learning methods for natural language processing for MBTI Personality Type Prediction. *Final Report Stanford University*.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J. G.; Le, Q.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2978–2988.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Gjurković, M.; Karan, M.; Vukojević, I.; Bošnjak, M.; and Šnajder, J. 2020. PANDORA Talks: Personality and Demographics on Reddit. *arXiv preprint arXiv:2004.04460*.

Gururangan, S.; Marasović, A.; Swayamdipta, S.; Lo, K.; Beltagy, I.; Downey, D.; and Smith, N. A. 2020. Don't stop pretraining: adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Jiang, H.; Zhang, X.; and Choi, J. D. 2019. Automatic Text-based Personality Recognition on Monologues and Multiparty Dialogues Using Attentive Networks and Contextual Embeddings. *arXiv preprint arXiv:1911.09304*.

Keh, S. S.; Cheng, I.; et al. 2019. Myers-Briggs Personality Classification and Personality-Specific Language Generation Using Pre-trained Language Models. *arXiv preprint arXiv:1907.06333*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Liu, M.; Gao, H.; and Ji, S. 2020. Towards Deeper Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 338–348.

Louizos, C.; Welling, M.; and Kingma, D. P. 2017. Learning sparse neural networks through L_0 regularization. *arXiv preprint arXiv:1712.01312*.

Lynn, V.; Balasubramanian, N.; and Schwartz, H. A. 2020. Hierarchical Modeling for User Personality Prediction: The Role of Message-Level Attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5306–5316.

Mehta, Y.; Majumder, N.; Gelbukh, A.; and Cambria, E. 2019. Recent trends in deep learning based personality detection. *Artificial Intelligence Review*, 1–27.

Myers-Briggs, I. 1991. Introduction to Type: A Description of the Theory and Applications of the Myers-Briggs Indicator. *Consulting Psychologists: Palo Alto*.

Pennebaker, J. W.; Francis, M. E.; and Booth, R. J. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001): 2001.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. EMNLP-IJCNLP 2019-2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing. In *Proceedings of the Conference*, 3982–3992.

Schlichtkrull, M. S.; De Cao, N.; and Titov, I. 2020. Interpreting graph neural networks for nlp with differentiable edge masking. *arXiv preprint arXiv:2010.00577*.

Schwartz, H. A.; Eichstaedt, J. C.; Kern, M. L.; Dziurzynski, L.; Ramones, S. M.; Agrawal, M.; Shah, A.; Kosinski, M.; Stillwell, D.; Seligman, M. E.; et al. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PlOS One*, 8(9): e73791.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, K.; Shen, W.; Yang, Y.; Quan, X.; and Wang, R. 2020. Relational Graph Attention Network for Aspect-based Sentiment Analysis. *arXiv preprint arXiv:2004.12362*.

Wang, X.; Sui, Y.; Zheng, K.; Shi, Y.; and Cao, S. 2021. Personality Classification of Social Users Based on Feature Fusion. *Sensors*, 21(20): 6758.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying Graph Convolutional Networks. In *International Conference on Machine Learning*, 6861–6871.

Wu, L.; Chen, Y.; Shen, K.; Guo, X.; Gao, H.; Li, S.; Pei, J.; and Long, B. 2021. Graph neural networks for natural language processing: A survey. *arXiv preprint arXiv:2106.06090*.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. *arXiv preprint arXiv:2005.11650*.

Xie, Z.; Zhou, G.; Liu, J.; and Huang, X. 2020. ReInceptionE: relation-aware inception network with joint local-global structural information for knowledge graph embedding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5929–5939.

Xue, D.; Wu, L.; Hong, Z.; Guo, S.; Gao, L.; Wu, Z.; Zhong, X.; and Sun, J. 2018. Deep learning-based personality recognition from text posts of online social networks. *Applied Intelligence*, 48(11): 4232–4246.

Yang, F.; Quan, X.; Yang, Y.; and Yu, J. 2021a. Multi-document transformer for personality detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 14221–14229.

Yang, F.; Yang, T.; Quan, X.; and Su, Q. 2021b. Learning to answer psychological questionnaire for personality detection. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 1131–1142.

Yang, T.; Yang, F.; Ouyang, H.; and Quan, X. 2021c. Psycholinguistic Tripartite Graph Network for Personality Detection. *arXiv preprint arXiv:2106.04963*.

Yao, L.; Mao, C.; and Luo, Y. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7370–7377.

Yarkoni, T. 2010. Personality in 100,000 words: A large-scale analysis of personality and word use among bloggers. *Journal of Research in Personality*, 44(3): 363–373.

Zhang, L.; Peng, S.; and Winkler, S. 2019. PersEmoN: a deep network for joint analysis of apparent personality, emotion and their relationship. *IEEE Transactions on Affective Computing*.

Zhou, J.; Han, X.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2019. GEAR: Graph-based evidence aggregating and reasoning for fact verification. *arXiv preprint arXiv:1908.01843*.