# Feature Analysis And Text Generation Of User Language For Different Chinese Social Network Platform

Meng Zhou
*Automation Department*
*Shanghai Jiao Tong University*
Shanghai, China
zhoumeng9904@sjtu.edu.cn

Yangqian Wu
*Automation Department*
*Shanghai Jiao Tong University*
Shanghai, China
wyq19981114@sjtu.edu.cn

Guanlin Zhang
*Automation Department*
*Shanghai Jiao Tong University*
Shanghai, China
zhanggl123@outlook.com

***Abstract*—We mainly talked about the users' language style in different social network platforms such as Zhihu, Tieba and Hupu, etc. Firstly, we focus on the topic of 'NBA' and 'CBA' to crawl user's posts on Zhihu, Tieba and Hupu, and got the dataset up to 100000 posts. We then tried several NLP models to accomplish the text generation, which includes GPT-2 and SeqGAN, and compare their performance. Futhermore, we used doc2vec method to get a 300 dimensions vector for each sentence and used T-SNE methods to analyse its principal component. Finally, we demonstrated it on the graph to give a further analysis and made some other efforts related to it.**

***Index Terms*—language feature, text generation, social platform**

## I. Introduction

According to 'Statistical Report on Internet Development in China', by June 2019, the number of Internet users in China has reached 854 million. There is no doubt that the network has become a wide information exchange platform, through which we can understand the world, communicate with others and express our own opinions. We can assume that different social network platforms, such as tieba, Zhihu, Weibo and so on, has their own user groups and language styles. It will be helpful to identify the corresponding users by analyzing the user's comments, posts and dialogues. And it colud be prepared to the further applications.

The existing research has focused on the user's language style in a single social network platform or simply comparing language features in the perspective of statistics. We have found that using machine learning techniques to analyse the user's language festures among different platforms and accomplishing corresponding text generation will be quite meaningful to reveal the hidden characteristics of online social network. Therefore, our work will be helpful to social networking platform companies for their better understanding of their users and also be auxiliary to monitor bad comments online.

In our work of project, we have done the following works and we will also talk about it for the following parts:

- Crawling user's posts on different platform, which includes Zhihu, Hupu and Tieba, and preprocess the data we have got.
- Using doc2vec method to generate the embedding vector of a sentence wihch involves the content and style of it. By applying TSNE, visualize the characteristic of the posts from different social network platform.
- Using GPT-2 and SeqGAN to generate the corresponding posts according to the data we have crawled and futhermore compare the performance of them.
- Analyzing the difference between posts from different platform and the texts we generate. And then come up with the information hidden behind it.

## II. Data Crawling and Preprocessing

Because our target is on the comparison between the user's language style among different platforms, in order to highlight the difference between each other, we decided to choose comments related to 'basketball', such as 'NBA' and 'CBA' as our topic. Around this topic, we target on three social platforms which have obvious differences in their user's identity, and this consequently results in the difference around language style: Hupu, Zhihu and Baidu.

To prepare for the user's language style analysis and corresponding text generation, we managed to scrape users' posts from Hupu and Baidu, and got more than 100 thousand posts on TieBa and 60 thousand posts on HuPu. For Zhihu, considering its language characteristics, we collect over 5 million words in several columns talking about basketball.

The obtaining of the posts is quite challenging, because we want to get the posts all in format of small sentence and filter the useless information after collecting it. Besides, we have to crawl data from three platforms and there are quite huge diversity among them.

For simplicity and indentity, we filter all the emojis in our data and we delete those reviews that are too short. In addition, some reviews include the share of other links. They are also

filtered. Moreover, we find on TieBa, there are some topics with quite a lot of posts with meaningless contents. They post reviews for some rewards from TieBa. So we only choose those topics with moderate number of posts.

In table 1, there are some examples from Tieba, Zhihu and Hupu respectively.

TABLE I
SOME EXAMPLES IN THE POSTS

| Source | Sentence |
|---|---|
| Hupu | '好球员不少，好教练就是浪里淘沙了' |
| Hupu | '不痛恨三分球了?' |
| Hupu | '墙头草 马后炮说的就是你这种人吧 这才几场啊' |
| Zhihu | '该是莱纳德vs勒布朗你凤翅鎏金镗，我五钩神飞枪' |
| Zhihu | '2011年总决赛MVP和2006年总决赛MVP，已经退役了' |
| Zhihu | '只能靠游刃有余的老辣走位，以及精纯的中投得分了' |
| Tieba | '纯杠精 还杠不好 概念性质都分不清就杠' |
| Tieba | '为什么要洗 球迷说什么你们也要管' |
| Tieba | '有对抗的上篮你去试试' |

## III. VISUALIZATION OF THE DIFFERENCE AMONG POSTS

### A. Overall Description

After acquiring the user's posts around the topic of 'NBA' and 'CBA' from three typical social network platforms: Hupu, Zhihu and Tieba, we visualize it in the dimension of 2 and find out that there are actually difference among them.

Seemingly it's quite hard to extract user's language style precisely from the posts, we firstly use Doc2Vec method to extract sentence vector from each post, wihch not only includes the post's content, but also involves the language style of a certain post(emotion, ration, intensity, etc). Then we use T-SNE method to finish dimension reduction to visualize the embedding vectors we obtained above. Finally, we got a intuitive map which involves thousands of dots, each dot represents a post's embedding vector. From the distribution of dots pertaining to different platform, we give a general and overall analysis based on this.

### B. Doc2Vec & Sentence vector

Doc2Vec method is developed from Word2Vec method, and is widely used to extract an embedding vector from a sentence, a paragraph or a document. For our project, we use it and the posts data we crawled to train a model which could generate a 1500 dimension embedding vector when we pass in a sentence(post) according to its content.

Fig. 1. A framework for learning word vectors.

*1) Previous framework of word vectors:* A well known previous framework for learning word vectors is shown in Figure **??**. The task is to predict a word given the other words in a context.

In this framework, every word is represented by a unique vector. Then, the concatenation or sum of the vectors is used as features to predict next word in a sentence. More formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, the objective of the word vector model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t|w_{t-k}, ..., w_{t+k})$$

The prediction process is usually done through softmax. There, we have

$$p(w_t|w_{t-k}, ..., w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

Each of $y_i$ is un-normalized probability for each output word $i$, computed as

$$y = b + Uh(w_{t-k}, ..., w_{t+k}; W) \tag{1}$$

where $U, b$ are the softmax parameters. $h$ is constructed by a concatenation or average of word vectors extracted from $W$. [**?**]

*2) The principle of Doc2Vec:* Paragraph Vector is capable of constructing representations of input sequences of variable length. Unlike some of the previous approaches, it is general and applicable to texts of any length: sentences, paragraphs, and documents. It does not require task-specific tuning of the word weighting function nor does it rely on the parse trees.

In Doc2Vec method, every paragraph is also mapped to a unique vector, referred to a column in matrix $D$ and every word is also mapped to a unique vector, referred a column in matrix $W$. All vectors are averaged or concatenated to predict the next word.

Actually, the paragraph vector can be viewed as another word for a certain paragragh. It indicates what is missing from the current context or the paragragh topic.

All vectors(paragraph vector and word vector) are trained using stochastic gradient descent and backpropagation. At every step, we can extract a fixed-length context from a paragraph ramdomly, calculate the error gradient from the network in Figure **??** and use the gradient to update the parameters in the model.

For prediction, in order to get the vector for a new paragraph, we also use gradient descent. In this scenario, the parameters for the rest of the model, the word vectors $W$ and the softmax weights, are fixed.

The principle of Doc2Vec is shown as Figure **??**.

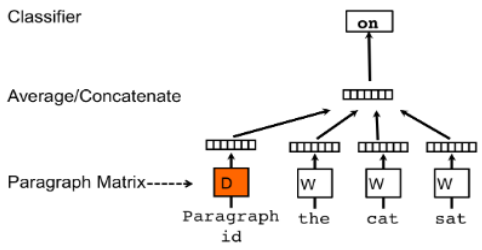


Fig. 2. The principle of Doc2Vec method

*3) Sentence Vector:* By using Doc2Vec method in our project, we treat every word in each sentence contains a same vector indicating its content.(Of course, the word in different sentence will have different sentence vetor.) Others are the same as the training process of Word2Vec, we pass in the sentence in our dataset and finally get a model which could infer a 1500 dimension embedding vector when we pass in a sentence.(After trying a lot of choices, we found 1500 dimension is suitable for our further analysis.)

### C. T-SNE & Visualization

From Doc2Vec method, we could train a model which could tell the embedding vector of each post, which involves the post's emotion, content and other language features. In order to visualize its distribution, we use T-SNE method to reduce the embedding vector's dimension and draw it on the 2-dimension map.

Representatively, we choose 3000 samples from each platform , infer their embedding vectors and after using T-SNE to draw their distribution in 2-dimension map.

The samples' distribution is shown as Figure **??**.

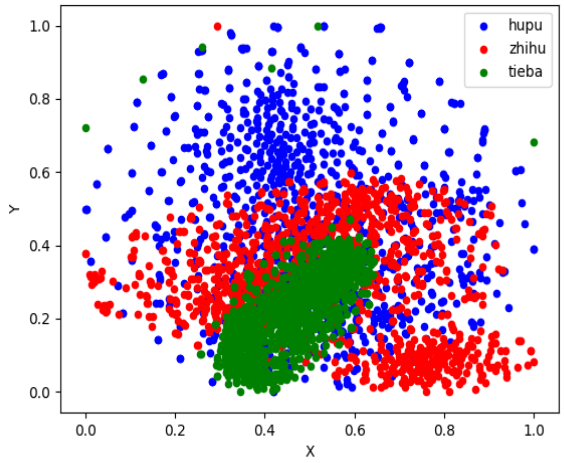


Fig. 3. The distribution of posts from different platform

### D. Difference Analysis

From the distribution shown above, it's easily recognized that there are huge distributional difference between the posts from Hupu, Tieba and Zhihu.

We should firstly point out that we don't know and don't to know the exact meaning of x-axis and y-axis, for we can't extract the explicit language features from its embedding vector. It turns out that after some easy linear transformation, the validity of Word2Vec is not changed. [**?**] But its distribution also denotes the real post's distribution on their content, emotion and so on.

From the distribution, we can find that the posts in Tieba is the most concentrated one, which means languages used on TieBa are simpler. Just like our intuition, a lot of posts in Tieba are intended to be emotional and irrational, and perhaps have the tendency of bullying.

In contrast, Zhihu has the middle scaled distributed posts and has a dense distribution in the right-bottom plain. Also in our previous understanding, the posts in Zhihu are more likely to be rational and sometimes copiously quote authoritative works, which enrich the abundance of themselves.

Moreover, Hupu's posts have the most wide distribution, and almost center in the middle of the map. In the process of crawling, we also realize that the posts in Hupu has plentiful style. Some could be very rational, and others perhaps already abuse each other.

The visualization of different posts based on Doc2Vec and T-SNE has given a solid support that there actually are huge difference between the user's language features in different social network platform. The following work we do is to seek out what the difference really is and what the corresponding post generation will be.

The examples in Table **??** could support the idea above.

## IV. Post Generation Through GPT-2

### A. Introduction

Recently, the world of natural language processing(NLP) world has been dominated by pretrained transformers. [**?**] Bert [**?**] from Google has achieved many State-Of-The-Art results in many NLP tasks. GPT-2 [**?**] is also a transformer model from OpenAI, which shows great power in natural language generation. However, the complete version is not open source for some reason, which has more than 1.5 billion parameters.

These huge models are designed for general language tasks. Besides showing language style in our work, doing generation task can also give us some insights about what the model has learned after all.

### B. Model Architectures

GPT-2 only contains decoders(note that the original transformer architecture has encoders and decoders). Each decoder

contains a self-attention layer and feed-forward network. The self-attention mechanism can be expressed as below:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

The inputs will be expressed as queries and keys of dimension $d_k$, and values of dimension $d_v$. Then We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values. To express this in the matrix form, we pack a set of queries into a matrix $Q$. The keys and values are also packed together into matrices $K$ and $V$. Then we will take a weighted sum of attentions.

The input will first be fed into an embedding layer and then go through several decoder blocks. Finally, the output vector will be fed into another embedding layer and output the probability of the next word in the vocabulary. After applying softmax funtion, because GPT-2 mask the attention of the latter words, it's an probability based on previous words.

### C. Language Models and Sampling Strategies

Language models are trying to estimate the conditional probability:

$$x_i \sim P(x|x_{1:i-1}) \tag{3}$$

where $x_{1:i-1}$ stands for all the words before and $x_i$ stands for the next word. One very intuitive way to generate natural language is to use the greedy policy:

$$x_i = \underset{x_i}{\text{argmax}}\ P(x_i|x_{1:i-1}). \tag{4}$$

But this can be problematic because always keeping maximizing the probability of the next word does not mean you maximize the probability of the whole sequence. Additionally, it turns out to be susceptible to language repetition, resulting in a meaningless sequence.

One classical approach is called beam search. We will trace words with top-k probability. Another [**?**] sampling strategy is called top-p sampling. given a distribution $P(x|x_{1:i-1})$, its top-$p$ vocabulary $V^{(p)} \subset V$ is the smallest set such that

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) >= p. \tag{5}$$

our final choice will come from this set. In our experiment, we have tried both of these two strategies and they both work well in our case.

### D. Experiment

In our experiment, we follow suggestions from [**?**], using the smallest version of GPT-2, including over 1 million parameters, which is only around 1-1500th of the complete version. Even in this setup, the training still cost about 10 hours on GPU. Also, because the original version is in English, we do not have any pre-trained resource in Chinese. We train our model from scratch. The original Byte Pair Encoding is also not suitable for Chinese, so we use Bert tokenizer, which is character-level.

### E. Generation Results

Based on GPT-2, we generate the model based on posts from Zhihu and Tieba, and try to infer the posts by the starts we give.

The results are shown in Table **??**.

TABLE II
SOME EXAMPLE RESULTS

| Source | Starts | Generation |
|---|---|---|
| Tieba | '科比' | 科比连续三个赛季罚球10+乔丹场均37的赛季罚球1.9。真正看一眼就知道了，那时候哈登是独行侠，黑子还少? |
| Tieba | '科比' | 科比铁成那样的球员你们看下他投篮效率有多低吗? 是不在我之前听到一句话就给瞎几把黑 |
| Tieba | '科比' | 科比就可以，打铁榜前三。 |
| Zhihu | '湖人队' | 湖人队的百回合净胜分-0.8分，是除了魔术队，其他人的队友也很不错。湖人队是防守强队，百回合净胜分排在联盟第17，而球队的百回合净胜分排在联盟第8，他们的净胜分排在联盟第14。湖人队的防守第12，第11。 |
| Zhihu | '湖人队' | 湖人队还是西部第11，而他们现在有两个超级后场持球人，湖人距离发展是没有尽善尽的，好歹防守能力也不行。这还没完，但进入12月份之后，湖人的阵容搭配会变得更加明显 |
| Zhihu | '湖人队' | 湖人队已经有4年没有在这个夏天手握1.075万合同，可能增加到5年薪资空间的顶薪补强，空间未来的球队也会越来越难造成竞争力。 |

## V. Post Generation Through SeqGAN

### A. GAN Introduction

GAN(Generative Adversarial Net) is a popular deep learning model which can learn from complicated distributions. It is made up of two parts: a generative model and a discriminative model. The generator imitates data's real distribution, while the discriminator tries to distinguish between real samples and generated ones. GAN aims to train a model that fit the real data distribution so perfectly that the discriminator can hardly tell.

GAN achieved huge success in computer vision, but it had a hard time on NLP tasks such as text generation. One reason is that in such tasks, generator's outputs are discrete, which makes it hard for the discriminator to pass gradient update back. Another reason is that the discriminator can only assess a complete sequence, but in the middle of sequence generation, we would like to know the score of a partially generated sequence as well as the score of its possible full sequence generated in the future . [**?**]

### B. Basic Principles of SeqGAN

SeqGAN helps to deal with these two problems. Firstly, to accomplish gradient update, it treats text generation as

a reinforcement learning problem. The partially generated sequence is "current state". Next token is "action". Current sequence plus next token is "next state". And discriminator's assessment is the "reward". Generator's policy is to generate a sequence that maximizes expected end reward. Policy gradient is used to update generator. For the second problem, SeqGAN employs MCTS(Monte Carlo Tree Search) to generate all possible full sequences for an action so that discriminator can compute reward based on these full sequences.

### C. Algorithm Details

Train generator $G_\theta$ with $\theta$ as its parameters. Note that recursive neural network structures like RNN or LSTM are commonly used. In time T, $G_\theta$ generates a complete sequence

$$Y_{1:T} = (y_1, \cdots, y_t, \cdots, y_T), y_t \in V$$

where V is the vocabulary. At time step t, state $s$ is current generated sequence $(y_1, \cdots, y_{t-1})$. Action $a$ is next token $y_t$. Policy is generator $G_\theta\left(y_t|Y_{1:t-1}\right)$.

Train discriminator with $\phi$ as its parameters. The discriminator is just a text classifier. One common option is to use CNN. $D_\phi\left(Y_{1:T}\right)$ gives the probability that $Y_{1:T}$ comes from real data.

$G_\theta\left(y_t|Y_{1:t-1}\right)$ aims to maximize expected reward:

$$J(\theta) = \mathbb{E}\left[R_T|s_0, \theta\right] = \sum_{y_1 \in V} G_\theta\left(y_1|s_0\right) \cdot Q_{D_\phi}^{G_\theta}\left(s_0, y_1\right) \quad (6)$$

In the equation, $R_T$ is the reward of fully generated sequences. $Q_{D_\phi}^{G_\theta}(s, a)$ is the action-value function of the whole sequence, representing the total accumulated expected reward starting from state $s$ and choosing action $a$ according to policy $G_\theta$ until the end.

In SeqGAN, the possibility that $D_\phi\left(Y_{1:T}^n\right)$ judges $Y_{1:T}^n$ as true is taken as the reward:

$$Q_{D_\phi}^{G_\theta}\left(a = y_T, s = Y_{1:T-1}\right) = D_\phi\left(Y_{1:T}\right) \quad (7)$$

As has been noted, the discriminator can only return a reward after a full sequence is generated. To estimate action-value of intermediate time steps, Monte-Carlo Tree Search and roll-out policy are used to sample the rest $T - t$ unknown tokens. This MCTS process repeats for N times and can be represented as below.

$$\left\{Y_{1:T}^1, \cdots, Y_{1:T}^N\right\} = \mathrm{MC}^{G_\beta}\left(Y_{1:t}; N\right) \quad (8)$$

After N MCTS processes, we get an output sample with its batch equals to N. We have:

$$Q_{D_\phi}^{G_\theta}\left(s = Y_{1:t-1}, a = y_t\right) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi\left(Y_{1:T}^n\right), Y_{1:T}^n \in \mathrm{MC}^{G_\beta}\left(Y_{1:t}; N\right) & t < T \\ D_\phi\left(Y_{1:t}\right) & t = T \end{cases} \quad (9)$$

Every time we generate some sequences, we need to retrain the discriminator. The discriminator wants to maximize its chance to judge a real sample as "real" and a generated sample as "fake", and this is what formula (10) means.

$$\min_\phi -\mathbb{E}_{Y \sim p_{\text{data}}}\left(\log D_\phi(Y)\right) - \mathbb{E}_{Y \sim G_\theta}\left[\log\left(1 - D_\phi(Y)\right)\right] \quad (10)$$

And every time we get a new discriminator, we can use policy gradient method to update the generator. The gradient $\nabla J(\theta)$ can be computed by differentiating (6). And then we can use gradient descent to update $\theta$:

$$\theta \leftarrow \theta + \alpha_h \nabla_\theta J(\theta) \quad (11)$$

This process is repeated until convergence.

### D. Experiment

We do text generation experiments on our Tieba dataset. Several aspects of the experiment process are worth noted.

First of all, as the SeqGAN paper's authors remarked in the appendix, there should be a pretraining process in the first place. Otherwise the training process would be extremely slow and ineffective. [**?**] Our experiment confirms this. When there is no pretraining, the results are bad even after a great many training epochs on a very large dataset. After sufficient pretraining is done, the results are much better.

Besides, like other GAN models, SeqGAN's biggest challenge is its training instability. Epochs of generator training and discriminator training can greatly affect final results and should be carefully chosen. We follow the advice of paper [**?**] and have done some trials. The outcome is satisfactory, but still there may be some space to improve.

### E. Generation Results

We used SeqGAN model to generate texts imitating Tieba's language style. We feeded 20000 Tieba comments into the model and trained for about 6 hours. Some good results are selected and shown below.

TABLE III
SEQGAN GENERATION RESULTS

| |
|---|
| 老狗粉一言不合就开咬? |
| 历史地位超越伦纳德，詹姆斯为什么个人会差在得分 |
| 过分了，看湖人那狗贼实力值联盟第一。 |
| 未来喀麦隆总统蕾哈娜指定炮友大帝恩比德 |
| 单手抓帽弹簧人香浓布朗。 |
| 楼主这贴炸出一大堆太监 |
| 蜀中无大将的，你鹈鹕怕起不来呀， |
| 反正今天吊打詹姆斯哈登么？哈哈。 |
| n吧低能儿属实不在少数 |

### F. Comparsion with GPT-2

Both two models are used to generate natural languages but their mechanisms are very different. GPT-2 can generate a sequence from a given start, while seqGAN can only generate a sequence that has a similar distribution with the training data. The input of seqGAN generator is a noise. But if we have a pretrained model for GPT-2, finetuning will be much faster than training from scratch.

## VI. Analysis of user's language features

By generating the posts by GPT-2 and seqGAN based on the posts selected from three platforms: Hupu, Zhihu and Tieba, it's more confirmed that there are huge difference among the posts in different social network platforms.

### A. WordCloud

In order to straightly view the difference among the posts from Hupu, Zhihu and Tieba, we have built the WordCloud for the posts from every platform. The WordCloud could be shwown below:


Fig. 4. The WordCloud generated from Hupu


Fig. 5. The WordCloud generated from Zhihu


Fig. 6. The WordCloud generated from Tieba

### B. Characteristic Analysis

From the WordCloud shown above, it's easy to conclude the following characteristics:

- The posts from Hupu: The posts like '哈哈', '有一说一', '手动狗头' are appearing in a high frequency, which indicates that Hupu's users are more intended to express their emotion in popular network words and hardly explain the rational reason behind it.
- The posts from Zhihu: Quite different from Hupu, the posts in Zhihu are filled with logical words like '然而', '实际上', '划重点'. This phenomenon indicates that Zhihu's users always express their thoughts and opinions based on the deep insight into a topic. They hardly give a post without any explanation.
- The posts from Tieba: The posts in Tieba shares some similarity with the posts from Hupu, both of them have a high appearence of popular network phrases like '滑稽', '哈哈', '有一说一'.

### C. Difference conclusion

To draw the conclusion, based on the WordCloud and Text Generation, we can summrize that Hupu users' posts are more likely to be short and always use some popular phrases. Quite

different from it, Zhihu users' posts are more intended to be long and logical, and furthermore, they always explain their thoughts related to the topic. Moreover, Tieba users' posts are similar to these in Hupu, but always appear some bullying words and negative phrases.

Though we analyse the result around the topic of 'basketball', we can conclude that the users' language around other topics shares the similar features because of our extra analysis on it and their same targeted user groups.

## VII. Conclusion and some thoughts

### A. Language style from different platforms

From previous language generation and some visualization techniques, we can conclude that some kind of "community culture" will shape in different social media platforms. The users of different platforms may consist of people of different ages. A person may also organize his languages in a different way when he is using a different platform. Despite the fact that we only choose a certain topic to do our research, it's reasonable to broadcast to other topics.

### B. Cutting-edge NLP

Unlike Computer Vision(CV), which can be transferred between countries, cutting-edge NLP techniques are always in English. In our project, when we are trying to use the latest transformer model(GPT-2), we find that although it has been one of the hottest models in English, there is little resource in Chinese. There is no pre-trained model even for the smallest version. In past days, people dealt with NLP problems by training a model from scratch. But recently, a very significant trend in NLP is multitask learning. Transfer learning from a huge model and fine-tune our downstream tasks is a common choice nowadays. But there are so many languages in our world, how to transfer SOTA model between languages is a problem.

## VIII. Group work and Acknowledgement

### A. Group work

We collect the data from three platforms respectively. Meng Zhou is responsible for ZhiHu, Yangqian Wu is responsible for HuPu and Guanlin Zhang is responsible for TieBa. Besides, Meng Zhou applys the dataset to GPT-2 language model and writes the corresponding code. Guanlin Zhang tries seqGAN and organizes this part. Yangqian Wu does the visualization, Doc2Vec and interaction part. We borrow the discussion room in the library every week to do discussion. The presentation is given by Yangqian Wu, and Meng Zhou organizes the overall structure of the project. Finally, we write the report together.

### B. Acknowledgement

Thanks to Prof.Tu's instruction in this semester, your course is well-organized and focuses more on the basic part of machine learning. It's amazing that by the end of this semester we find we are able to understand the cutting-edge NLP models and implement them on our own dataset.

Besides, thanks to TA. You pay attention to every single detail of our homework. Every time you analyze problems in our homework, you provide some other aspects that we do not fully pay attention when we were doing it.

Finally, best wishes to Prof.Tu and TA. Hope your research work will always receive good results! : )

## References

[1] Quoc Le, Tomas Mikolov, "Distributed Representations of Sentences and Documents" Google Inc, 1600 Amphitheatre Parkway, Mountain View, CA 94043

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "Attention Is all you need" arXiv:1706.03762

[3] Andrew Ng, Deep learning.ai, Sequence Models, Coursera

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" arXiv:1810.04805

[5] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, "Language Models are Unsupervised Multitask Learners" OpenAI

[6] Ari Holtzman, Jan Buys, Maxwell Forbes, Yejin Choi, "The Curious Case of Neural Text Degeneration" arXiv:1904.09751

[7] https://github.com/Morizeyao/GPT2-Chinese

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Networks" arXiv:1406.2661

[9] Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu, "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient" arXiv:1609.05473