A Survey on Self-Attention Generative Adversarial Networks Based Tag Recommendations

Chen Yuan*, Xiao Zhiting, Yu Changren

Department of information and communication, National Defense University of Science and Technology, Wuhan, China

Keywords: Tag recommendation, sagan

Abstract: Based on the analysis of previous research, this article deeply discusses the current tag-based recommendation methods from the above two aspects, briefly explains the current common tag recommendation methods and existing problems, and proposes a self-attention-based GAN (called SAGAN) tag recommendation method. In this paper, the performance index of the comparison experiment of the label recommendation approach is designed, our method is to implement self-attention mechanism to generative adversarial networks. The method of generating negative samples by the generator and identifying the samples by the discriminator enables the SAGAN to learn the pros and cons of the samples, and then use the method of generating an adversarial network to find and extract user and resource label data from the existing tags of the recommended object Characteristics, so as to better implement the recommendation that the user matches the resource tags.. Experimental results show that for different types of resources, SAGAN tag recommendation approach have achieved certain performance improvements.

1. Introduction

As data tagging is “the process or act of associating a data object with characterizing metadata for some purpose”, it has both an organizational (people, governance, and process) aspect and a functional (specification, implementation) aspect. A “tag” is an assertion describing some aspect of a resource, pairing a semantic label (or “tag name”) with a corresponding tag value.[1] Meanwhile, tags are metadata of user-defined data, with rich feature information, which is very helpful for obtaining user preference information. Common tag recommendation approach include collaborative filtering methods, content-based methods, hybrid methods of content-based and collaborative filtering, session-based methods, and neural network methods, but these methods also have "tags disorder", "tags Redundancy". The generative adversarial network (GAN) can perform well even with less training data, even if the training labels contain noise or even errors.

With the rapid development of social networks, the research of tag-based recommendation algorithms has become one of the research hotspots in this field. The label [2] can intuitively express the attributes of the user or the project, and can also indirectly represent the user's understanding and opinion of the item. Many Web2.0 social sites like Last.fm and MovieLens use the tag recommendation function, that is, users can freely create some tags to mark the content they are interested in, and social networking sites recommend some similar content for users through their tags Content. In order to better label and manage these contents, in general, social networking sites will provide some personalized tags that users may be interested in, but most users are not accustomed to tagging themselves. Sparse data. Therefore, in tag recommendation, a personalized recommendation system is required to recommend the tag content of interest. Lappas T [3] and Chen et al. [4] use user social relationships and user tags to mine user interest preferences and recommend users. Ma et al. [5] proposed a joint framework combining tags and user social relationships for Weibo recommendation, and achieved significant recommendation results. Xu et al. [6] proposed a novel Weibo personalized tag recommendation method based on the probabilistic generation model and using the user's viewpoint of tag generation on Weibo text. Gong et al. [7] proposed a multi-modal hashtag model for Weibo recommendation combining text and visual information. Zhao et al. [8] combined Hashtag and LDA to propose a Hashtag-LDA topic model to

  DOI: 10.25236/icbdem.2020.037

mine potential information in Weibo for recommendation. This method has significantly improved the accuracy of recommendation.

In addition to being used in the fields of speech and language in the fields of image and vision, GAN can also be used in the field of text. For example, the Google brain team invented a text generation model trained on in-filling (MaskGAN) [9], Which can be used to fill in the blanks. The specific task is to complete the missing part of the sentence. The researchers also use CycleGAN to perform symbolic music genre transfer.[10], using multi-track sequential generative adversarial networks (MuseGAN) [11] to generate piano music;Wu et al. [12] used 3D convolutional generative adversarial networks to synthesize novel objects, including tables, chairs, sofas, and cars. Yong et al. [13] applied deep neural networks and GAN networks to asynchronous motor fault detection to improve Accuracy and fault classification; Schlegl et al. [14] also used GAN for anomaly detection in the field of medical images.

The current research on tag recommendation based on generative adversarial network is mainly in two aspects. First, how to use the generation adversarial network to recommend tags for valuable objects such as text, audio, and images; second, how to use the generation adversarial network to discover and extract users and tags from the existing tags of the recommended objects data characteristics of resource tags, so as to better implement recommendations that match users' resource tags.

2. Related Works

Before our method, there are some common recommendation method in recommendation system which contain as follow:

2.1 Collaborative Filtering method

Collaborative filtering recommends to target users based on the preferences of other users. It first finds a set of neighbor users that are consistent with the preferences of the target user, then analyzes the neighbor users and recommends items that neighbor users like to the target user. Collaborative filtering recommended items that people with similar perferences in the past. Problems include 1) Nearest neighbor search is not accurate enough. It is often seen that there is no common scoring item between two users and items, resulting in similarity cannot be calculated. Even if some users and items have similarity between them, their reliability is it is difficult to guarantee. 2) The impact of tag categories is not considered. When the content of item categories in the website is completely different, the nearest neighbors searched by traditional collaborative filtering algorithms are often similar to the target user's preferences on individual item categories, resulting in insufficient recommendation results. Reasonable. 3) The number of users and tags in the website is huge and constantly increasing, which makes the user-tag scoring matrix a high-dimensional matrix, which results in the scalability problem of collaborative filtering, that is, as the number of users and tags increases, the computational complexity of the algorithm has increased dramatically, severely affecting the real-time nature of system recommendations.

2.2 Content-based method

The content-based recommendation system uses the matching phase of user interests and information content likeness to filter information. In a word, The content-based recommendation system recommended items similar to the ones the user preferred in the past.

2.3 Hybrid method

The idea of the hybrid algorithm based on content and collaborative filtering is as follows: content-based recommendation based on product feature information, collaborative filtering based on user rating recommendations, this chapter combines user feature information with user historical rating information to get user ratings for different features of the product Values to form a user-feature scoring matrix. The user-feature scoring matrix replaces the traditional user-item scoring matrix, and clusters users with the same feature preferences based on the user's favorite item features.

Users with the same characteristic preferences are added to the same cluster, become neighbors, and generate cluster centers. When recommending, first determine the feature type that the target user likes, and then calculate the similarity with all cluster centers to determine the cluster to which the nearest neighbor belongs. Finally, the nearest neighbor is used to complete the recommendation of the target user. In addition, for newly launched products, combined with the content-based hybrid algorithm, it can also use product characteristics to push it.

2.4 Session-base method

Session-base method recommendation system aim to predict the unknown part of a session or the future sessions based on modeling the complex relations in session(s), given partially known session information.

2.5 Neural Models based method

The research of recommendation algorithms based on deep learning is divided into 4 categories: deep learning recommendation algorithms using auxiliary information; model-based deep learning recommendation algorithms; dynamic deep learning recommendation algorithms; label-based deep learning recommendation algorithms. Among them, the deep learning recommendation algorithms using auxiliary information can be divided into 3 types: using auxiliary information to extract only the feature representations of users (or items); using auxiliary information to extract the feature representations of users and items respectively; using auxiliary information to extract users and Common characteristics of items.

3 Our Method

Our method is to implement self-attention mechanism to generative adversarial networks. The method of generating negative samples by the generator and identifying the samples by the discriminator enables the GAN to learn the pros and cons of the samples, and then use the method of generating an adversarial network to find and extract user and resource label data from the existing tags of the recommended object Characteristics, so as to better implement the recommendation that the user matches the resource tags.

3.1 Generative model

The generator takes the source text $x = \{w_1 , w_2 , ..., w_n \}$ as input and predicts the summary $\hat{y} = \{\hat{y}_1 , \hat{y}_2 , ..., \hat{y}_m \}$.Here, the n is the length of the source text x and m is the length of the predicted tag. We use a bi-directional LSTM encoder to convert the input text x into a sequence of hidden states $h = \{h_1 ,..., h_n \}$.On time step t, an attention-based LSTM decoder is then used to compute the hidden state s_t of the decoder and a context vector c_t. The parameters of the generator G are collectively represented by θ. The context vector c_t is concatenated with the decoder state s_t and fed through a fully connected layer and a softmax layer to produce the probability of predicting word from target vocabulary at each time step t.

$$P_{vocab}(\hat{y}_t) = softmax(V'(V[s_t, c_t] + b) + b') \quad (1)$$

where V', V, b, b' are learnable parameters. Similar to the work of [3], we incorporate a switching pointer-generator network to use either text generator from fixed vocabulary or pointer copying rare or unseen from the input sequence. Finally, we can get the final probability $P(\hat{y}_t)$ of each token $\hat{y}_t$ in the summary.

3.2 Discriminative model

The discriminator is a binary classifier and aims at distinguishing the input sequence as originally generated by humans or synthesized by machines. We encode the input sequence with a CNN as it shows great effectiveness in text classification [4]. We use multiple filters with varying window sizes to obtain different features and then apply a max-over-time pooling operation over the features. These pooled features are passed to a fully connected softmax layer whose output is the probability

of being original.

3.3 Updating model parameters

In the adversarial process, using the discriminator as a reward function can further improve the generator iteratively by dynamically updating the discriminator. Once we obtain more realistic and high-quality summaries generated by generator *G*, we re-train the discriminator as:

$$Min_{\phi} - E_{Y\sim p_{data}}[logD_{\phi}(Y)] - E_{Y\sim G_{\theta}}[log(1 - D_{\phi}(Y))] \quad (2)$$

When the discriminator *D* is obtained and fixed, we are ready to update the generator *G*. The loss function of our generator G consists two parts: the loss computed by policy gradient (denoted by J_{pg}) and the maximum-likelihood loss (denoted by J_{ml}). Formally, the objective function of *G* is $J = \beta J_{\text{pg}} + (1 - \beta)J_{\text{ml}}$, where β is the scaling factor to balance the magnitude difference between J_{pg} and J_{ml}. According to the policy gradient theorem [], we compute the gradient of J_{pg} w.r.t. the parameters θ:

$$\nabla_{\theta}J_{pg} = \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t} R_D^{G_\theta}\left((Y_{1:t-1}, X), y_t\right)\cdot\nabla_{\theta}(G_{\theta}\left(y_t|(Y_{1:t-1}, X)\right) = \frac{1}{T}\sum_{t=1}^{T} E_{y_t\in G_\theta}[R_D^{G_\theta}\left((Y_{1:t-1}, X), y_t\right)\nabla_{\theta} logp(y_t|Y_{1:t-1}, X)] \quad (3)$$

Where $R_D^{G_\theta}\left(\left(Y_{1:t-1}, X\right), y_t\right)$ is the action-value function, and we have $R_D^{G_\theta}\left(\left(Y_{1:t-1}, X\right), y_t\right) = D_{\phi}(Y_{1:T})$, *T* is the length of the text. We update the parameters using SGD(stochastic gradient descent), $Y_{1:t}$ is the generated summary up to time step *t*, *X* is the source text to be condensed.

3.4 Implement self-attention mechanism

Self-attention exhibits a better balance between the ability to model long-range dependencies and the computational and statistical efficiency. The self-attention mechanism calculates response at a position as a weighted sum of the features at all positions, where the weights or attention vectors are calculated with only a small computational cost.

We call our particular attention "Scaled Dot-Product Attention" (Figure 1). The input consists of queries and keys of dimension d_k, and values of dimension d_v. 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.

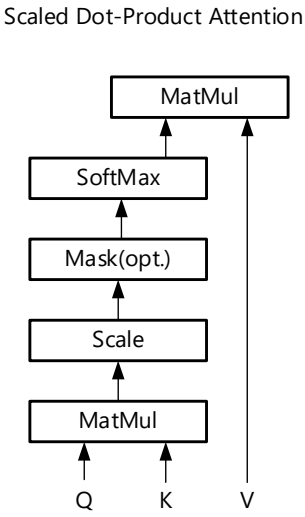

Figure 1 Scaled Dot-Product attention model.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^t}{\sqrt{d_k}}\right)V \quad (4)$$

The two most commonly used attention functions are additive attention [5], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [6]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

4 Experiments

We used web crawler technology to crawl about 6850 blog tag data released in 2009-2019 from Sina Blog. Stored in the database, containing 5,750 Chinese blog profiles and about 8,000 tags. Filter these tags to filter out labels for noun attributes. In the end, 2990 related independent tags are obtained. In the content similarity calculation, the parameter is set to the average of the Euclidean distances between all pairs of text. The size of the parameters will affect the results of the model. Too large values will cause the correction algorithm to have too little effect on the results, and the optimization effect will not be good. Too small a value will cause the original site's blog tags to be ignored. After experiments, the results can take into account the user's initial label settings, and reflect the superiority of the optimization algorithm.

4.1 Training data and batching

We trained on the standard words and sentences are encoded using byte-pair encoding, which has a shared source target vocabulary of about 3900 tokens. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 2500 source tokens and 2500 target tokens.

4.2 Hardware and schedules

We trained our models on one machine with NVIDIA GPUs. For our base models using the hyper-parameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 20,000 steps or 24 hours, step time was 1.0 seconds.

4.3 Optimizer

We used the Adam optimizer with $\boldsymbol{\beta_1 = 0.9, \beta_2 = 0.98}$ and $\boldsymbol{\epsilon = 10^{-9}}$. We varied the learning rate over the course of training, according to the formula:

$$lrate = d_{model}^{-0.5} \cdot \min\left(step_{num}{}^{-0.5}, step_{num} \cdot warmup_{steps}{}^{-1.5}\right) \quad (5)$$

4.4 Regularization

We employ three types of regularization during training: to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the the content of the experiment includes comparing the effects of the method without the correction term, the benchmark method and the method.

4.5 Performance evaluation metrics

This section presents the three most widely used metrics for evaluating the performance. These standard information retrieval metrics include Precision, Recall, and F-measure [8].

$$\text{Precision} = \frac{|relevant\ tags| \cap |retrieved\ tags|}{retrieved\ tags} \quad (6)$$

$$\text{Recall} = \frac{|relevant\ tags| \cap |retrieved\ tags|}{relevant\ tags} \quad (7)$$

$$\mathbf{F \cdot measure} = \frac{2 * Presicion * Recall}{Presicion + Recall} \quad (8)$$

4.6 Results

As can be seen from Figure 2, in terms of accuracy, the method without the correction term does not refer to the tags given by users, and recommends tags directly through the blog profile, resulting in lower accuracy; in the benchmark method, users on the website gave the tag is more random and the accuracy is not high; this research method combines the blog profile and the tag given by the user, and the accuracy has been significantly improved. In terms of recall, the method without the correction term is recommended only through the blog profile, and it is almost impossible to recommend all the correct tags; in the benchmark method, the tags given by users can cover most of the correct tags; this research method combines the previous Both methods have also improved the recall rate.

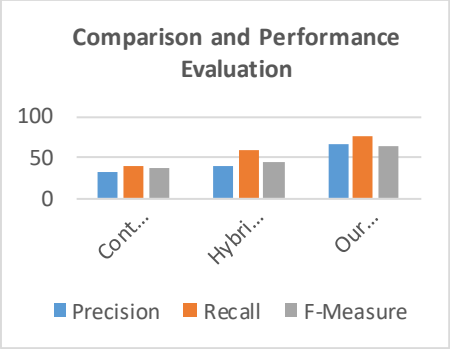

Figure 2 Comparison and performance evaluation

5 Conclusion

In this paper, we proposed an adversarial process for tag recommendation. Experimental results showed that our model could generate more abstractive, readable and diverse summaries. Our method is to implement self-attention mechanism to generative adversarial networks. The method of generating negative samples by the generator and identifying the samples by the discriminator enables the SAGAN to learn the pros and cons of the samples, and then use the method of generating an adversarial network to find and extract user and resource label data from the existing tags of the recommended object Characteristics, so as to better implement the recommendation that the user matches the resource tags. We regret that there is not enough time to compare all the recommended methods one by one which will be our next stage of work.

References

[1] Direct of national intelligence, PO3-Data-Tagging-Functional-Requirements, https://www.dni.gov/files/ISE/documents/DocumentLibrary/PO3-Data-Tagging-Functional-Requirements.pdf

[2] Chen X, Duan Y, Houthooft R, et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. Advances in neural information processing systems, 2016: 2172-2180.

[3] Zhang H, Goodfellow I, Metaxas D, et al. Self-attention generative adversarial networks, 2018.

[4] Vairavasundaram S, Varadharajan V, Vairavasundaram I, et al. Data mining - based tag recommendation system: an overview, 2015, 5(3): 87-112.

[5] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Advances in neural information processing systems, 2017: 5998-6008.

[6] Qian S, Peng F, Lu J J J O S U. Tag optimization based on semantic similarity, 2015(2): 7.

[7] Cao Z, Wang R, Wang X, et al. Improving Human Pose Estimation with Self-Attention Generative Adversarial Networks. 2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 2019: 567-572.

[8] Maimon O, Rokach L. Data mining and knowledge discovery handbook, 2005.

[9] Page L, Brin S, Motwani R, et al. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab, 1999.

[10] Chen P, Shao X. Audio auto-tagging based on generative adversarial networks, 2018, 10(6): 754.

[11] Chen X, Duan Y, Houthooft R, et al. SAGAN: Interpretable representation learning by information maximizing generative adversarial nets. Advances in neural information processing systems, 2016: 2172-2180.

[12] Bai J, Bu Y. An Improved Algorithm for Semantic Similarity Based on HowNet. 2018 2nd International Conference on Data Science and Business Analytics (ICDSBA), 2018: 65-70.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[14] Dosovitskiy A, Tobias Springenberg J, Brox T. Learning to generate chairs with convolutional neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 1538-1546