NEWS Recommendations Using Generative adversarial neural network.

Information Retrieval and Extraction
(Dr. vasudeva varma)

-Major Project

DATE : 12-November-2017

IIIT-HYDERABAD

Mentor

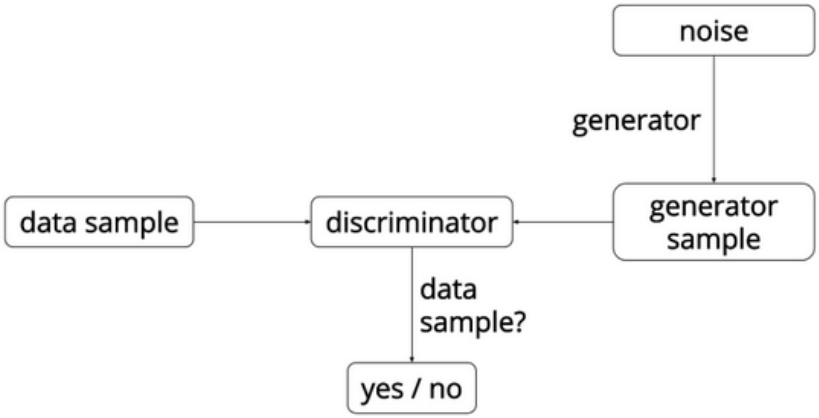
Vaibhav kumar

Contributors

Yashwanth Reddy	201525036
Varun Mundale	20162011
Hemang Akbari	20162112
Rachit Garg	201401082

ABSTRACT:

GAN have two competing neural network models, One takes noise as input and generates samples and so is called the generator. The other model, called the discriminator receives samples from both the generator and the training data, and has to be able to distinguish between the two sources. These two networks play a continuous game, where the generator is learning to produce more and more realistic samples, and the discriminator is learning to get better and better at distinguishing generated data from real data. These two networks are trained simultaneously.



As we are using IRGAN for NEWS Recommendation, the generative retrieval focusing on predicting relevant documents given a query. while the discriminative retrieval focusing on predicting relevancy given a query-document pair. On one hand, the discriminative model, aiming to mine signals from labelled data, provides guidance to train the generative model towards fitting the underlying relevance distribution over documents given the query. On the other hand, the generative model, acting as an attacker to the current discriminative model, generates diffcult examples for the discriminative model in an adversarial way by minimising its discrimination objective.

With the competition between these two models:(I) generative model learns to fitthe relevance distribution over documents via the signals from the discriminative model, and (ii) discriminative model is able to exploit the unlabelled data selected by the generative model to achieve a better estimation for document ranking.

Performance : 8.877% for additional layer on element wise product of scoring function. And 8.827% for dot product.

INTRODUCTION

We have a set of queries {q1,...,qN} and a set of documents{d1,...,dM}, where queries here is number of readers and documents are set of articles read by those readers.

Overall Objective:

Thus, inspired by the idea of GAN, we aim to unify these two different types of IR models by letting them play a minimax game: the generative retrieval model would try to generate (or select) relevant documents that look like the ground-truth relevant documents and therefore could fool the discriminative retrieval model, whereas the discriminative retrieval model would try to draw a clear distinction between the ground-truth relevant documents and the generated ones made by its opponent generative retrieval model. Formally, we have:

$$J^{G^*,D^*} = \min_{\theta}\max_{\phi}\sum_{n=1}^{N}\Big(\mathbb{E}_{d\sim p_{\text{true}}(d|q_n,r)}\left[\log D(d|q_n)\right] + \quad ($$
$$\mathbb{E}_{d\sim p_{\theta}(d|q_n,r)}\left[\log(1-D(d|q_n))\right]\Big),$$

where the generative retrieval model G is written as Pθ(d|qn,r), directly and the discriminative retrieval D estimates the probability of document d being relevant to query q, which is given by the sigmoid function of the discriminator score:

$$D(d|q) = \sigma(f_{\phi}(d,q)) = \frac{\exp(f_{\phi}(d,q))}{1+\exp(f_{\phi}(d,q))}.$$

Generative retrieval model:

Pθ(d|q,r), which tries to generate(or select) relevant documents, from the candidate pool for the given query q goal is to approximate the true relevance distribution over documents Ptrue(d|q,r) as much as possible.

$$\theta^* = \arg\min_{\theta}\sum_{n=1}^{N}\Big(\mathbb{E}_{d\sim p_{\text{true}}(d|q_n,r)}\left[\log\sigma(f_{\phi}(d,q_n))\right] +$$
$$\mathbb{E}_{d\sim p_{\theta}(d|q_n,r)}\left[\log(1-\sigma(f_{\phi}(d,q_n)))\right]\Big)$$
$$= \arg\max_{\theta}\sum_{n=1}^{N}\underbrace{\mathbb{E}_{d\sim p_{\theta}(d|q_n,r)}\left[\log(1+\exp(f_{\phi}(d,q_n)))\right]}_{\text{denoted as } J^{G}(q_n)}$$

Discriminative retrieval model :

fφ(q, d), which, in contrary, tries to discriminate well-matched query-document tuples (q, d) from ill-matched ones, where the goodness of matching given by fφ(q, d) depends on the relevance of d to q; in other words,its goal is to distinguish between relevant documents and non-relevant documents for the query q as accurately as possible. It is in fact simply a binary classifier, and we could use 1 as the class label for the query-document tuples that truly match (positive examples) while 0 as the class label for those do not really match (negative examples).

$$\phi^* = \arg\max_{\phi} \sum_{n=1}^{N} \Big(\mathbb{E}_{d \sim p_{\text{true}}(d|q_n, r)} \left[\log(\sigma(f_\phi(d, q_n)) \right] + \mathbb{E}_{d \sim p_{\theta^*}(d|q_n, r)} \left[\log(1 - \sigma(f_\phi(d, q_n))) \right] \Big),$$

Algorithm of IRGAN for minimax game:

Algorithm 1 Minimax Game for IR (a.k.a IRGAN)

Input: generator $p_\theta(d|q, r)$; discriminator $f_\phi(\boldsymbol{x}_i^q)$;
training dataset $\mathcal{S} = \{\boldsymbol{x}\}$

Initialise $p_\theta(d|q, r)$, $f_\phi(q, d)$ with random weights θ, ϕ.
Pre-train $p_\theta(d|q, r)$, $f_\phi(q, d)$ using $\mathcal{S}$
repeat
  for g-steps **do**
    $p_\theta(d|q, r)$ generates K documents for each query q
    Update generator parameters via policy gradient Eq. (5)
  end for
  for d-steps **do**
    Use current $p_\theta(d|q, r)$ to generate negative examples and combine with given positive examples $\mathcal{S}$
    Train discriminator $f_\phi(q, d)$ by Eq. (3)
  end for
until IRGAN converges

Data Generation

The data dump consists of user_id and set of article_id which represent the documents that the user read.

Each sample is a tuple of the form
<user_id> <article_id> <label>

<label>=5 if user reads the article
<label>=0 if user does not read the articles.

Only those users were taken who read >=10 articles
The ratio for read article to unread articles was 10:3

User and article Representation

Every user and article is represented as a k dimensional vector.

Scoring Function

(I) Dot product

Now, for news recommendation, following which we define our scoring function for the preference of reader u (i.e. the query) to news-article i (i.e. the document) as,

$$s(u, i) = b_i + \boldsymbol{v}_u^\top \boldsymbol{v}_i,$$

where bi is the bias term for article i, vu and vi ∈ Rk are the latent vectors of user u and item i respectively, defined in the k-dimensional continuous space. There are 2 ways experimented for scoring function as,

(II) Element-wise product.

Our next **s**coring function for the preference of reader u (i.e. the query) to news-article i (i.e. the document) as,

s (u, i) = element_wise_multiply(u,i)

This gives us k x 1 dimensional vector.
A neural network layer is added on this and the activation used is **softmax.**

Experiment

when training the discriminative retrieval model,the generative retrieval model is leveraged to sample negative items with the sample number of positive items for each user.

when training the discriminative retrieval model,the generative retrieval model is leveraged to sample negative items with the sample number of positive items for each user.

Then train the discriminative retrieval model is via it's Equation, On the other hand, since the training of the generative retrieval model is performed by reinforcement Learning., which is normally implemented by the policy gradient on the sampled K items from Pθ(d |qn,r). As such the positive reward can be observed from REINFORCE and the generative retrieval model can be learned properly by it’s defined equation.

Result

For 833 Readers dataset,

	P@1	P@2	P@2	P@4	P@5	P@6	P@7	P@8	P@9	P@10
Dot	0.0616	0.0786	0.0696	0.06597	0.06752	0.0668	0.0668	0.0665	0.07839	0.06839
Element	0.064	0.0792	0.0704	0.0695	0.06776	0.077	0.0668	0.0665	0.06613	0.06596
	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@10
Dot	0.0816	0.07928	0.08045	0.08176	0.0216	0.0826	0.08416	0.08546	0.08677	0.08829
Element	0.084	0.08029	0.0817	0.0823	0.08237	0.08323	0.08454	0.08595	0.08723	0.08877

For 400 Readers dataset,

	P@1	P@2	P@2	P@4	P@5	P@6	P@7	P@8	P@9	P@10
Dot	0.08	0.0775	0.08833	0.07562	0.0765	0.07458	0.07428	0.07343	0.07416	0.07399
Element	0.08249	0.08249	0.07916	0.07812	0.0775	0.0775	0.07678	0.0775	0.07722	0.07774
	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@	NDGC@10
Dot	0.08	0.07806	0.07941	0.07887	0.08102	0.08084	0.08165	0.08185	0.08366	0.0845
Element	0.08249	0.08249	0.08109	0.07203	0.08276	0.08437	0.08508	0.08741	0.08855	0.0904

References

https://arxiv.org/pdf/1705.10513.pdf
https://en.wikipedia.org/wiki/Generative_adversarial_network
https://www.oreilly.com/learning/generative-adversarial-networks-for-beginners
https://blog.openai.com/generative-models/