Data Augmentation Based on Adversarial Autoencoder Handling Imbalance for Learning to Rank*

Qian Yu, Wai Lam
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
{yuqian, wlam}@se.cuhk.edu.hk

Abstract

Data imbalance is a key limiting factor for Learning to Rank (LTR) models in information retrieval. Resampling methods and ensemble methods cannot handle the imbalance problem well since none of them incorporate more informative data into the training procedure of LTR models. We propose a data generation model based on Adversarial Autoencoder (AAE) for tackling the data imbalance in LTR via informative data augmentation. This model can be utilized for handling two types of data imbalance, namely, imbalance regarding relevance levels for a particular query and imbalance regarding the amount of relevance judgements in different queries. In the proposed model, relevance information is disentangled from the latent representations in this AAE-based model in order to reconstruct data with specific relevance levels. The semantic information of queries, derived from word embeddings, is incorporated in the adversarial training stage for regularizing the distribution of the latent representation. Two informative data augmentation strategies suitable for LTR are designed utilizing the proposed data generation model. Experiments on benchmark LTR datasets demonstrate that our proposed framework can significantly improve the performance of LTR models.

Introduction

In information retrieval, the task of Learning to Rank (LTR) (Liu and others 2009) refers to a category of methods where machine learning models are trained to rank documents based on the degree of relevance to the query. One challenge for LTR is that the available training data is usually imbalanced (Verberne et al. 2011; Macdonald, Santos, and Ounis 2013; Agarwal et al. 2012). The imbalance problem for LTR comes from two sources:

1. **Imbalance regarding Relevance (R-imbalance)**
   Given a query, the documents judged with high relevance level are much fewer than judged irrelevant documents. The reason is that the relevant documents are typically scarce in the whole collection and it is hard for an explicit or implicit relevance judgement annotation procedure to pick out all the documents with high relevance level.
2. **Imbalance regarding Query (Q-imbalance)**
   Different queries typically have different amounts of relevance judgements. Since relevance is always query-based, the imbalance of available information associated with different queries also influences the model training in LTR. Specifically, if some queries have much larger amount of judged documents than other queries, then these queries tend to dominate the learning of LTR models.

The performances of LTR models are restricted by imbalanced training data, especially the pointwise LTR models. The pairwise LTR models regard pairs of document with different relevance levels as training samples, and pairwise learning objectives are designed according to the relative relevance level for eliminating R-imbalance problem. However, the pairwise objective function and training samples still suffer from the scarcity of documents with high relevance level. The pairwise training set lacks of diversity since the "highly relevance" part of the pairwise training samples is still derived from the same small set of documents with high relevance level. Therefore, if we can address the issue of imbalanced data in LTR, the overall performance of pairwise methods can be improved.

Both resampling methods (Batuwita and Palade 2010; Chawla et al. 2002) and ensemble methods (Dietterich 2000) attempt to avoid learning a biased model with imbalanced data, but none of them provide informative augmentation for the training data. Instead of modeling complex decision boundaries with alternative models as in ensemble methods, we argue that a more proper solution for data imbalance in LTR should incorporate informative data into the training set. The data generation procedure should comply with two criteria. The first criterion is that it should generate informative data that are not merely duplications of existing data. The second criterion is that the generation procedure should be controllable regarding relevance levels and query characteristics. For data generation purpose, Generative Adversarial Networks (GAN) (Goodfellow et al. 2014) and its variants (Zheng, Zheng, and Yang 2017; Shrivastava et al. 2017; Mariani et al. 2018) have been proposed. These models try

*The work described in this paper is substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Codes: 14203414) and the Direct Grant of the Faculty of Engineering, CUHK (Project Code: 4055093).

to train a data generator and a discriminator to distinguish the generated data and real data. However, GAN-based models capture the input data distribution, and will intrinsically inherit the imbalance in the training set. GAN-based methods cannot address data imbalance since they cannot comply with the two data generation criteria mentioned before.

We propose a data generation model based on Adversarial Autoencoder (AAE) (Makhzani et al. 2015) to facilitate informative data augmentation for LTR. Basically, we attempt to generate latent representations which can be decoded to informative synthetic data. The proposed data augmentation has several characteristics for handling R-imbalance and Q-imbalance. First, relevance information is disentangled from the latent representation. Thus, the data reconstruction stage facilitates the data generation for a given target relevance levels for handling R-imbalance. We refer this data generation as informative since the generated data contains new information via combining the target relevance level and the relevance-independent latent representation. Second, with the aid of word embeddings, query types containing queries that are semantically close are discovered and incorporated in the adversarial training stage for regularizing the distribution of latent representations. For each query, sufficient training data derived from the same query type can be obtained via manipulating the generation of latent representation. This can eliminate the dominance of queries with large amount of relevance judgements, therefore handling Q-imbalance. Two informative data augmentation strategies for LTR are designed exploiting the above mentioned model. To the best of our knowledge, this is the first work for handling both types of imbalance in LTR setting.

Related Work

A basic task for information retrieval is to rank a set of documents given a query, based on the relevance between a document and a query (Cao et al. 2006). Learning to rank (LTR) methods (Liu and others 2009) solve this problem via training a ranking model with samples labeled with ranking information. There are three categories of LTR methods, namely pointwise, pairwise, and listwise methods (Cao et al. 2007). For pointwise methods, the inputs are feature vectors derived from the corresponding query and document. A input vector contains features associated with the document, such as document length, IDF of terms, etc., and features associated with both query and document, such as TF*IDF, BM25 score, etc. Each of the vector is labeled with the relevance score of the document to the query. The learned model is capable for relevance score prediction. In pairwise methods, each training sample is constructed by two feature vectors. The two feature vectors in one pairwise sample is associated with the same query and the relative relevance is the label for the training sample. Listwise methods use document lists as training samples. However, it is difficult to prepare the training set for listwise methods and the algorithm complexity tends to be high. Recently, features extracted from texts via deep learning models are also utilized in LTR methods (Wan et al. 2016; Severyn and Moschitti 2015; Pang et al. 2017), but we only consider the traditional features in LTR datasets in this paper.

Resampling (Good 2006) is an important technique for handling data imbalance (He and Garcia 2009) in machine learning. There are mainly two types of sampling methods, namely oversampling (Fernández-Navarro, Hervás-Martínez, and Gutiérrez 2011) and undersampling (Chawla 2003). In oversampling (Chawla 2003), data are sampled from the existing data set of minor classes and added into the training set. In this way, the sizes of minor classes are enlarged and the imbalance between major and minor classes is alleviated. However, the training data is augmented with duplicated data without new information, and these sampling methods make the learned machine learning model suffer from high risk of overfitting. One of the well-known oversampling methods is SMOTE (Chawla et al. 2002) and there are several variants (Han, Wang, and Mao 2005) of SMOTE for specific settings. One the other hand, undersampling methods (Liu, Wu, and Zhou 2009; Yen and Lee 2009) reduce the number of data in major classes to obtain similar amount of training data in different classes. The undersampling will result in information loss during the reduction of training data via sampling. Undersampling technique is usually utilized with ensemble methods (Galar et al. 2013; Liu, An, and Huang 2006; Wang, Minku, and Yao 2015), but the scarcity of data in minor classes is not directly handled in these methods. As we mentioned before, informative data generation is more helpful than resampling technique.

Informative Data Generation Model

Each training sample $\mathbf{v}_{q,d}$ for LTR is a feature vector derived from a pair of query q and document d. Features in $\mathbf{v}_{q,d}$ are assigned with nonnegative values. Also, a relevance judgement indicating the relevance degree is associated with a query-document pair. Let the relevance degree be denoted as $r_{q,d}$ which is an integer value typically in the range $[0, n_r - 1]$, where n_r is the number of relevance levels. The aim of the informative data generation model is to generate pseudo feature vector $\mathbf{v}^*$ given a target relevance degree r^*. Fig. 1 depicts our proposed informative data generation model. The decoder is equipped with a relevance level indicator in the reconstruction procedure to disentangle relevance information from the latent representation. Each data instance is reconstructed based on the target relevance level and relevance-independent latent representation. This method can incorporate new information into the training set via combining the desired relevance level and the relevance-independent latent representation. Query clusters containing queries that are semantically close are discovered with the aid of word embeddings. Each cluster can be regarded as a query type labeled with a type ID. The query type information is incorporated in the adversarial training stage for regularizing the distribution of latent representations. Thus, the prior distribution of the latent representation can be utilized for generating data via pseudo queries for a particular query type. Such strategy can tackle the Q-imbalance problem.

Background of Adversarial Autoencoder

The basic Adversarial AutoEncoder (AAE) (Makhzani et al. 2015) equips the autoencoder with an adversarial compo-

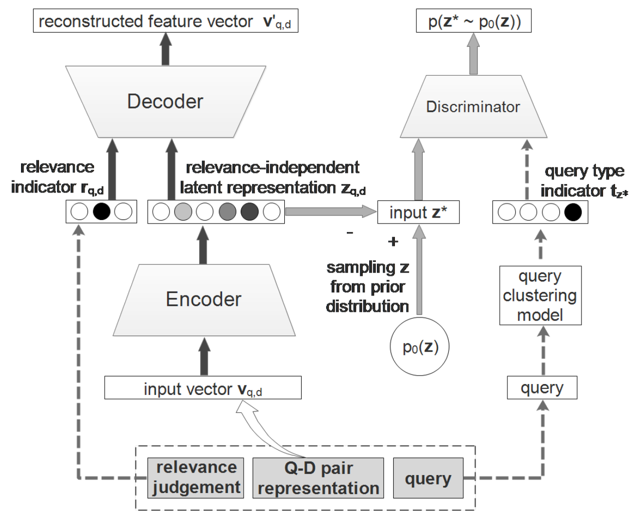
reconstructed feature vector $v'_{q,d}$
Decoder
relevance indicator $r_{q,d}$
relevance-independent latent representation $z_{q,d}$
Encoder
input vector $v_{q,d}$
$p(z^* \sim p_0(z))$
Discriminator
input z*
-
+
sampling z from prior distribution
$p_0(z)$
query type indicator t_{z^*}
query clustering model
query
relevance judgement
Q-D pair representation
query

Figure 1: Informative Data Generation Model.

nent. The adversarial learning is employed for imposing a prior distribution $p_0(\mathbf{z})$ to the latent representation $\mathbf{z}$. Sampling from $p_0(\mathbf{z})$ will result in latent representations that can be reconstructed to generate reasonable synthetic data. Compared with Variational Autoencoder (VAE) (Kingma and Welling 2013), AAE performs better when imposing complicated prior distribution

There are three basic components in AAE:

Encoder. A feed forward neural network consists of several layers with ReLU nonlinear activation function. The input is the real data, namely the normalized feature vector $\mathbf{v}$ and the output of encoder is the latent representation $\mathbf{z}$.

Decoder. A neural network consists of similar structure as the encoder but the input is the latent representation $\mathbf{z}$. A latent representation can run through the decoder resulting in a reconstructed feature vector $\mathbf{v}'$ with the same size as the input for encoder.

Discriminator. A discriminative network makes use of a sigmoid output layer. The input $\mathbf{z}^*$ for the discriminator can be a latent vector sampled from the prior distribution $p_0(\mathbf{z})$ or a latent representation encoded from a real feature vector. The output of the discriminator is the probability that the input $\mathbf{z}^*$ is generated from $p_0(\mathbf{z})$.

The learning of AAE is guided by two objective functions, namely reconstruction loss function and adversarial loss function, as shown in Eqn 1 and Eqn 2 respectively given below.

$$\min_{Encode,Decode} \sum_{\mathbf{v}\sim p_\mathbf{v}} ||\mathbf{v} - Decode(Encode(\mathbf{v}))||^2 \quad (1)$$

$$\min_{Encode} \max_{D} E_{\mathbf{z}\sim p_0(\mathbf{z})} \log D(\mathbf{z}) - E_{\mathbf{v}\sim p_\mathbf{v}} \log D(Encode(\mathbf{v})) \quad (2)$$

where $Encode(\cdot)$ and $Decode(\cdot)$ stand for the data process function of the Encoder and Decoder respectively, $D(\cdot)$ stands for the discriminator, and $p_\mathbf{v}$ is the distribution of real data. The adversarial training (Lowd and Meek 2005) facilitates the AAE to embed the input in a space that is defined by the prior distribution $p_0(\mathbf{z})$. Thus, latent representations sampled from $p_0(\mathbf{z})$ can be reconstructed to reasonable synthetic data. Note that different from general GAN model, the latent representations generated from the prior distribution in AAE are regarded as positive samples for the discriminator, while the outputs of the encoder are regarded as negative samples. After the AAE is trained with a given prior distribution $p_0(\mathbf{z})$, we can sample the latent representation $\mathbf{z}^*$ from $p_0(\mathbf{z})$ and reconstruct the feature vector via the decoder.

Disentangling Relevance Information from Latent Representation

To facilitate the support of generating feature vector with a target relevance degree, the relevance information needs to be captured. A relevance indicator vector is added as the input for the decoder for disentangling the relevance information. This disentanglement enforces the encoder to generate relevance-independent latent representation. Then the reconstruction procedure can be controlled via assigning a specific target relevance degree to the latent representation for data reconstruction. In adversarial learning, the prior distribution is only imposed for the relevance-independent latent representation.

Given a real data $\mathbf{v}_{q,d}$ which passes through the encoder, the latent representation obtained is written as:

$$\mathbf{z}_{q,d} = Encode(\mathbf{v}_{q,d}) \quad (3)$$

where $Encode$ is the data embedding process in Encoder. We make use of a relevance degree indicator $\mathbf{r}_{q,d}$ as shown in Figure 1. $\mathbf{r}_{q,d}$ is a one-hot vector denoting the relevance degree of the current input vector. It participates in the reconstruction procedure together with the latent representation. Thus, the decoding process takes the concatenation of $\mathbf{r}_{q,d}$ and $\mathbf{z}_{q,d}$ as input for obtaining the reconstruction $\mathbf{v}'_{q,d}$:

$$\mathbf{v}'_{q,d} = Decode([\mathbf{r}_{q,d}, \mathbf{z}_{q,d}]) \quad (4)$$

where $Decode$ is the data reconstruction process in Decoder.

The learned latent representation is relevance-independent. In this way, the learned decoder becomes relevance-aware and the relevance degree of the generated data can be controlled.

Regularizing Latent Representation Distribution based on Query Type

Now if we sample a latent representation from the prior distribution $p_0(\mathbf{z})$, we can provide a target relevance degree to the decoder and reconstruct a feature vector. But the sampling may be influenced by dominant queries in the training data, namely queries with large amounts of document relevance judgements. To solve this Q-imbalance issue, we further regularize the latent representation with query content information and multi-mode prior distribution.

We discover some query clusters which contain queries that are semantically close. Each cluster can be regarded as a query type labeled with a type ID. The query type information is incorporated into the adversarial learning for the regularization of latent distribution. The particular query type discovery model will be presented below. Denoting the number of query types as n_t which is an assigned parameter, we design a prior distribution that is a mixture of n_t Gaussian distributions. $p_0(\mathbf{z}) = \sum_{k=1}^{n_t} p_0^k(\mathbf{z})$, where $p_0^k(\mathbf{z})$ is a normal distribution for latent representation. Different $p_0^k(\mathbf{z})$ is assigned with different means but sharing the same variance. In our implementation, we random select n_t points as the means and variance is set to 1.

Each normal distribution $p_0^k(\mathbf{z})$ is responsible for generating queries from one query type. This is achieved by feeding a query type indicator into the discriminator together with the original input, namely the latent representation. When sampling a latent representation **z** from the prior distribution, we use only one normal distribution and record the query type with an indicator vector $\mathbf{t_z}$. If the latent representation $\mathbf{z}_{q,d}$ is encoded from a real data $\mathbf{v}_{q,d}$, then the query type for the corresponding query will be denoted with the indicator vector $\mathbf{t}_{\mathbf{z}_{q,d}}$. The discriminative process takes the concatenation of latent representation and the query type indicator vector as shown below:

$$p(\mathbf{z}* \sim p_0(\mathbf{z})) = D([\mathbf{z}^*, \mathbf{t}_{\mathbf{z}^*}]) \tag{5}$$

where $p(z \sim p_0(\mathbf{z}))$ is the predicted probability that the input **z** is generated from $p_0(\mathbf{z})$, and $\mathbf{z}^*$ can be a latent representation coming from the encoder or a latent representation sampled from $p_0(\mathbf{z})$. This query type indicator will influence the discriminator for using different decision boundaries for different query types. Positive samples for each query type are generated from the corresponding Gaussian distribution. In this way, the latent representation will be guided for fitting in the mixture prior distribution according to the query type.

One advantage of our design is that we can generate the latent representation for a particular query type and reconstruct synthetic data with a specific target relevance degree.

Query Type Discovery Given the query set, we discover the query types in an unsupervised manner. Since word embedding methods can capture semantic information for every individual term, we adopt word embedding and conduct query clustering. Specifically, we embed each query with the average of its term embeddings from GloVe (Pennington, Socher, and Manning 2014), and then the K-Means method with Euclidean distance is applied for discovering query types. The parameter k in the K-Means method is set to n_c. After such discovery process, given a query, we can determine the query type it belongs.

Training Procedure

The learning procedure for the proposed model consists of three stages described below. These training stages will be conducted iteratively until a stable model is obtained. The learned model is capable of reconstructing data with disentanglement of relevance information. The query type-based prior distribution will be used for regularizing the distribution of latent representation.

Update Encoder and Decoder for Reconstruction First of all, the encoder and decode should have the basic ability for data reconstruction. In this stage, the reconstruction loss is used for training. We assume that the values in input feature vector are normalized to be in the range $[0, 1]$. Then we adopt the cross-entropy loss for reconstruction:

$$L_{recon}(\mathbf{v}, \mathbf{v}') = -v_k \log(v_k') - (1 - v_k) \log(1 - v_k') \tag{6}$$

where $\mathbf{v}'$ is the reconstructed vector based on the real data **v**, and both of them are associated with a pair of query q and document d:

$$\mathbf{v}'_{q,d} = Decode(Encode(\mathbf{v}_{q,d}), \mathbf{r}_{q,d}) \tag{7}$$

For training data without normalization, the reconstruction loss can be replaced by other suitable functions, such as the squared error between input and output.

Update Discriminator Next, a good discriminator is needed for distinguishing the latent representations encoded from the real data and the latent representations sampled from the prior distribution. Recall that both kinds of latent representation are associated with a query type. The input for the discriminator are latent representation **z** and its corresponding query type indicator $\mathbf{t_z}$. The discriminator output $D(\mathbf{z})$ is the probability that the input is generated from the prior distribution, namely $p(\mathbf{z}^* \sim p_0(\mathbf{z}))$. Then the learning objective function for the discriminator is formulated as:

$$L_D = -\log D(\mathbf{z}^*, \mathbf{t}_{\mathbf{z}^*}) + \log(1 - D(\mathbf{z}_{q,d}, \mathbf{t}_{\mathbf{z}_{q,d}})) \tag{8}$$

where $\mathbf{z}^*$ is the latent representation sampled from the prior distribution $p_0(\mathbf{z})$, and $\mathbf{z}_{q,d}$ is the latent representation encoded from the real data. This loss function will guide the model to assign 1 to the latent representation that generated from the prior distribution, whereas 0 for latent representation encoded from the real data. The learned discriminator is used for distinguishing these two kinds of latent vectors.

Update Encoder for Generation After the discriminator is updated, it will perform better in distinguishing two kinds of latent representations. The next step is to further adjust the encoder to generate latent representations that can confuse the discriminator. The target is to train the encoder to output latent representations that can be predicted to be generated from the prior distribution by the discriminator. Thus, the prediction probability of discriminator is used as the loss function for training the encoder as a generator.

$$L_G = -\log D(\mathbf{z}_{q,d}, \mathbf{t}_{\mathbf{z}_{q,d}}) \tag{9}$$

where $\mathbf{z}_{q,d}$ is the latent representation generated from encoder based on real data.

Data Augmentation for Learning to Rank

After the above proposed model is trained, it is capable of generating data with desired characteristics. In this section, we first illustrate how this model can be used for data generation with a target relevance degree and query type. Then two strategies are proposed to augment the training set in LTR for alleviating R-imbalance and Q-imbalance.

Basic Data Generation

Typically, there are much more irrelevant documents for a query than relevant ones. Our proposed model can generate feature vectors associated with high relevance degree. In general, a generated data instance with a specific target relevance degree can be regarded as representation for a pseudo-pair of query and document. Given a real data **v** and its latent representation **z** encoded by Encoder. According to the disentanglement ability of our model, **z** is relevance-independent and it can be used for generating data given a particular target relevance degree. First the target relevance degree is represented by the relevance indicator vector **r**∗, the data generation is formulated as:

$$\mathbf{v}^* = Decode([\mathbf{z}, \mathbf{r}*]) \tag{10}$$

Different queries have different amounts of relevance judgements. Some queries with large amounts of judgements tend to dominate the learned ranking model, while some queries have just a few available judgements. Since our model can generate latent representation for each query type, we can enlarge the training set for those query types with small amount of judgements. The prior distribution of latent representation is a mixture of normal distributions, and each of them is responsible for one query type. Thus, we select the specific normal distribution to sample latent representation for data generation, and the output will possess the query characteristic of the corresponding query type. Assume that a latent representation $\mathbf{z}^*$ is sampled from the normal distribution associated with the query type $\mathbf{t}_{\mathbf{z}^*}$, then $\mathbf{z}^*$ can be decoded to generate.

Data Augmentation Strategy I: Augment(R)

The first data augmentation strategy intends to generate pairwise training data with different relevance levels but sharing the same relevance-independent latent representation. This kind of pairwise training data will help the LTR model to focus only on the relevance-related features.

Let us consider a training data **v** with relevance degree r as an example. We obtain its latent representation **z** and use it to reconstruct the data with relevance degree $r - 1$ and $r + 1$, denoted as $\mathbf{v}_{r-1}$ and $\mathbf{v}_{r+1}$ respectively. Two pairwise training samples are constructed: $(\mathbf{v}, \mathbf{v}_{r-1})$ and $(\mathbf{v}_{r+1}, \mathbf{v})$.

Since the data, $\mathbf{v}_{r-1}$, $\mathbf{v}$ and $\mathbf{v}_{r+1}$ are all reconstructed from the same relevance-independent latent representation **z**, the main information that can be obtained from the ranking difference is the relevance information, without interference from the content information. In this way, this strategy Augment(R) utilizes the disentanglement ability of the data generation model to directly embed the relative relevance information in the training pairs. Since the training pairs share the same latent representation, it reduces the burden of the ranking model to distinguish relevance information and content information. Note that only R-imbalance is handled in this strategy. The augmented pairwise data set is added to the original pairwise training set.

Data Augmentation Strategy II: Augment(R+Q)

This data augmentation strategy aims at alleviating the two types of data imbalance. Given a training data, we generate new data associated with the next higher relevance level to incorporate more data with high relevance. For example, assuming that the range of the relevance degree is $[0, 2]$. For a query q, we sample training data from the training subset with relevance degree 1, denoted as V_1^q. Then a new data set with target relevance degree 2 is generated based on V_1^q via the above proposed data generation model, denoted as V'^q_2. Similarly, based on the training subset with relevance degree 0, denoted as V_0^q, we can generate a data set with relevance degree 1, denoted as V'^q_1. A ratio $r_R \in (0, 1)$ is introduced to provide the stopping threshold for the generation procedure:

$$r_R < S_2^q / S_1^q, \quad r_R < S_1^q / S_0^q \tag{11}$$

where $S_n^q = |V_n^q| + |V'^q_n|$ is the number of original data and generated data with relevance degree n.

Then we use the data generation to generate data for query types with small amount of relevance judgement via sampling from the prior distribution based on specific query types. We sum all the available training data for each query type to find out the query types with small amount of judgements. More training data is generated for these query types via a ratio r_Q used similar as r_R:

$$r_Q < S_{min}^q / S_i^q \tag{12}$$

where S_{min}^q is the number of the training data associated with the query type with fewest judgements, and S_i^q is the number of the training data for any other query type. We repeatedly find the query type with the minimum training set for data generation, until the stopping threshold expressed in Eqn 12 is reached.

Experiments

Datasets

Three benchmark LTR datasets are used in our experiments to test the proposed data augmentation methods. The datasets are MQ2007 and MQ2008 from the LETOR 4.0 package, and MSLR-WEB10K. The queries for MQ2007 and MQ2008 are from Million Query track of TREC 2007 (1692 queries) and TREC 2008 (784 queries) respectively. For each query in these two datasets, retrieved documents from the Gov2 web page collections (about 25 million pages) are labeled with relevance judgements ranging from 0 standing for irrelevance to 2 standing for high relevance.

Table 1: Statistics of Datasets.

Dataset	MQ2007	MQ2008	MSLR-WEB10K
query (Q)	1,692	784	10,000
document (D)	65,323	14,384	-
Q-D pair	69,623	15,211	1,200,192
Q-D pair with r = 4	0	0	8,881
Q-D pair with r = 3	0	0	21,317
Q-D pair with r = 2	3,863	931	159,451
Q-D pair with r = 1	14,628	2,001	386,280
Q-D pair with r = 0	51,632	12,279	624,263
max Q-D pair per query	147	121	908
min Q-D pair per query	6	6	1

Table 2: Performance (P@5/NDCG@5/MAP) of Learning to Rank Models. * indicates a statistically significant improvement over the SMOTE resampling method, at level 0.05.

Model	RankSVM	RankNet	ListNet	AdaRank	LambdaMart
MQ2007					
Original	0.414 / 0.414 / 0.464	0.392 / 0.397 / 0.450	0.393 / 0.397 / 0.446	0.407 / 0.413 / 0.460	0.419 / 0.419 / 0.466
Undersampling	0.379 / 0.368 / 0.387	0.376 / 0.378 / 0.365	0.376 / 0.378 / 0.375	0.381 / 0.383 / 0.394	0.392 / 0.391 / 0.414
Oversampling	0.415 / 0.416 / 0.465	0.392 / 0.399 / 0.452	0.396 / 0.398 / 0.450	0.409 / 0.413 / 0.463	0.421 / 0.412 / 0.459
SMOTE	0.418 / 0.418 / 0.466	0.391 / 0.398 / 0.450	0.395 / 0.399 / 0.452	0.411 / 0.415 / 0.463	0.424 / 0.420 / 0.465
Augment(R)	0.429*/0.427*/**0.473***	0.429*/0.429*/**0.475***	0.433*/**0.435***/0.475*	0.438*/0.436*/**0.478***	0.439*/0.434*/0.476*
Augment(R+Q)	**0.431***/**0.430***/**0.473***	**0.430***/**0.431***/**0.475***	**0.434***/**0.435***/**0.477***	**0.439***/**0.437***/**0.478***	**0.441***/**0.439***/**0.481***
MQ2008					
Original	0.327 / 0.450 / 0.450	0.328 / 0.448 / 0.451	0.334 / 0.455 / 0.456	0.343 / 0.476 / 0.478	0.341 / 0.469 / 0.475
Undersampling	0.227 / 0.345 / 0.331	0.236 / 0.359 / 0.362	0.233 / 0.349 / 0.371	0.263 / 0.383 / 0.390	0.313 / 0.397 / 0.432
Oversampling	0.329 / 0.450 / 0.452	0.327 / 0.451 / 0.454	0.346 / 0.456 / 0.456	0.345 / 0.475 / 0.476	0.343 / 0.475 / 0.478
SMOTE	0.331 / 0.451 / 0.452	0.327 / 0.453 / 0.454	0.347 / 0.456 / 0.458	0.346 / 0.477 / 0.477	0.345 / 0.476 / 0.477
Augment(R)	0.342*/0.463*/0.466*	0.350*/0.478*/0.477*	0.352*/0.482*/0.484*	0.352*/0.482*/0.484*	0.353*/0.485*/**0.487***
Augment(R+Q)	**0.345***/**0.465***/**0.467***	**0.351***/**0.480***/**0.481***	**0.353***/**0.485***/**0.486***	**0.353***/**0.484***/**0.485***	**0.354***/**0.488***/**0.487***
MSLR-WEB10K					
Original	0.412 / 0.438 / 0.345	0.420 / 0.452 / 0.360	0.425 / 0.457 / 0.362	0.423 / 0.453 / 0.360	0.429 / 0.461 / 0.365
Undersampling	0.266 / 0.303 / 0.213	0.297 / 0.332 / 0.245	0.305 / 0.361 / 0.386	0.300 / 0.346 / 0.370	0.319 / 0.366 / 0.391
Oversampling	0.415 / 0.440 / 0.349	0.422 / 0.452 / 0.361	0.428 / 0.463 / 0.366	0.427 / 0.458 / 0.362	0.439 / 0.467 / 0.367
SMOTE	0.423 / 0.446 / 0.375	0.428 / 0.459 / 0.363	0.433 / 0.472 / 0.371	0.434 / 0.467 / 0.371	0.452 / 0.479 / 0.378
Augment(R)	0.445*/0.468*/**0.380***	0.450*/0.471*/0.382*	0.452*/0.474*/0.385*	0.450*/0.472*/0.381*	0.469*/0.489*/0.393*
Augment(R+Q)	**0.448***/**0.470***/**0.380***	**0.453***/**0.475***/**0.386***	**0.460***/**0.487***/**0.392***	**0.458***/**0.482***/**0.392***	**0.473***/**0.496***/**0.401***

For each query-document pair, a 46-dimentional feature vector is prepared in the datasets. A query level normalization is conducted on the feature vectors. The description for each feature and normalization can be found in (Qin and Liu 2013). For the dataset MSLR-WEB10K, 10000 queries are collected from a commercial search engine, and documents for each query are annotated with 5 different relevance level, namely from 0 for irrelevance to 4 for high relevance level. There are 136 features extracted for each pair of query and document.

The statistics of the three datasets are presented in Table 1. We can observe that relevant query-document pairs are relatively scarce in the training set, especially the ones with high relevance degree. In addition, the large range of the number of query-document pairs for a query indicates the data imbalance regarding queries.

Experiment Settings

We apply our proposed two data augmentation strategies to several state-of-the-art LTR models. Given a set of training data and corresponding query set, we first conduct unsupervised learning via the query type discovery model described above for labeling each query with a particular query type. As a result, each training data is associated with a query type ID and a relevance degree. Then the informative data generation model as described above can be trained. After that, the two data generation strategies are applied for augmenting the training set. Then the LTR models are learned based on the augmented training set. We apply the data augmentation methods on the following state-of-the-art LTR models. **RankSVM** (Joachims 2002) is a variant of SVM model for LTR task. **RankNet** (Burges et al. 2005) is a classic pairwise ranking algorithm making use of a neural network with a natural probabilistic loss function on pairs of training data. **ListNet** (Cao et al. 2007) uses lists of document as training instances employing listwise loss function. **AdaRank** (Xu and Li 2007) trains one ranking model at each iteration followed with re-weighting document pairs, and combines these weak rankers for the final ranking output. **LambdaMART** (Burges, Ragno, and Le 2007) incorporates tree-based boosting methods into LambdaRank for LTR, and directly optimizes the evaluation metric such as NDCG.

To demonstrate the effectiveness of the proposed data augmentation methods for improving LTR performance, we compare with the different resampling methods including random **oversampling**, random **undersampling**, and one of the state-of-the-art oversampling method called **SMOTE** (Chawla et al. 2002). We denote our proposed two data augmentation methods as **Augment(R)** and **Augment(R+Q)**. Each dataset is partitioned to five subsets with roughly equal size, and 5-fold cross validation is adopted for all the LTR models. Specifically, in each fold, there are 3 subsets utilized as training set, one subset as validation set and one as test set. Data augmentation will be conducted on each training set and the improved model will be evaluated on the corresponding test set. The average performance on test sets of 5 folds is recorded for each method. We use the Wilcoxon significance test to examine the statistical significance of the improvements over the SMOTE resampling method.

Implementation Details In the data generation model for MQ2007 and MQ2008, the encoder is a fully connected neural network with the layer size as 46-50-50-10, whereas the structure for the decoder is 10-50-50-46. As for MSLR-WEB10K, the network structures are 136-100-50-20 and 20-50-100-136 for encoder and decoder, respectively. The discriminator is a softmax prediction model with two hidden layers, and each of which is 50-dimension. Since both datasets MQ2007 and MQ2008 contain only three levels of relevance degree, i.e., 0, 1, and 2, the relevance indicator vector in the model is 3-dimension. This dimension is 5 for MSLR-WEB10K. The query cluster number is set to 10 for MQ2007, 5 for MQ2008, and 10 for MSLR-WEB10K. The analysis for this hyper-parameter can be found in Section . r_R and r_Q is set to 1. The framework is implemented via Pytorch 0.4.0 and Python 3.6, and all AAE based models are run with a single NVIDIA Tesla K80 GPU.

Performance Improvement on LTR Models

The performance of LTR models with different data augmentation methods on MQ2007, MQ2008 and MSLR-WEB10k are presented in Table 2. We can observe that our proposed two data augmentation strategies can significantly improve the LTR performance of all the LTR models. LambdaMART method with **Augment(R+Q)** achieves the best performance regarding all evaluation metrics. Specifically, **Augment(R+Q)** can improve the NDCG@5 performance of LambdaMART by more than 3% on both MQ2007 and MQ2008. **Augment(R+Q)** generally improves LTR performance a little more than **Augment(R)**, which indicates that handling data imbalance regarding queries (Q-imbalance) can also help LTR models as well as imbalance regarding relevance levels (R-imbalance). According to the results, all the other tested resampling methods cannot influence the performance of LTR models significantly. **Undersampling** achieves worse performance than the baseline methods, which reveals that irrelevant query-document pairs are also valuable. The random **oversampling** and the **SMOTE** method obtain similar performance as the original LTR model since these resampling methods cannot add informative data into the training set for LTR. Our framework achieves significant improvement due to the ability of informative data generation considering both relevance degree and query characteristics.

Table 3: Number of queries with improved/reduced MAP via data augmentation on LambdaMART.

augment method	MQ2007	MQ2008	MSLR-WEB10k
Undersampling	327 / 842	143 / 467	1474 / 8035
Oversampling	641 / 684	329 / 352	4902 / 4864
SMOTE	653 / 674	346 / 331	4964 / 4795
Augment(R)	956 / 504	475 / 240	5625 / 3792
Augment(R+Q)	1035 / 495	493 / 213	5871 / 3675

Apart from the overall performance, we further look into the effect of data augmentation for individual queries. In Table 3, the number of queries whose MAP can be improved or reduced via a specific data augmentation method is recorded. The base model is LambdaMART and similar trends are observed for other LTR models. We can observe that our proposed two augmentation methods can improve the performance of most queries. Specifically, the MAP performance of 1035 out of 1692 queries can be improved by **Augment(R+Q)**. As for other resampling methods, such dominance of queries with improved MAP performance cannot be observed. This demonstrate the proposed informative data augmentation can benefit most individual queries.

Investigation of Hyper-parameters

We present some analysis on some important hyper-parameters in our framework. The P@1 evaluation metric is used for comparison of different parameter settings. LambdaMART is adopted as the base model for this analysis.

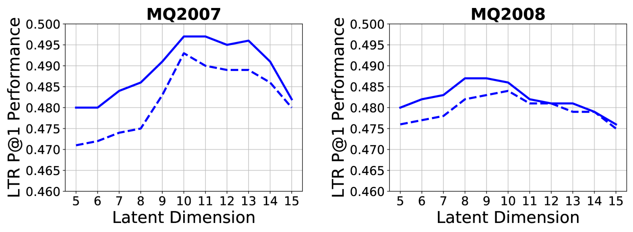

Figure 2: Investigation of Latent Representation Dimension. Solid lines represent the performance of **Augment(R+Q)** and dash lines for **Augment(R)**.

Figure 2 shows the investigation of the dimension of the relevance-independent latent representation **z**, with regard to the LTR performance of **Augment(R)** and **Augment(R+Q)** on MQ2007 and MQ2008. We can observe that the performances are stable as long as the dimension of latent representation is set in a specific range. Too small or too large dimension reversely affect the performance.

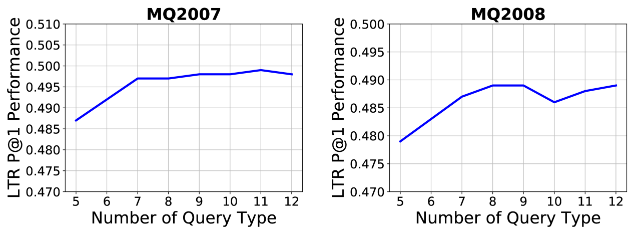

Figure 3: Investigation of Number of Query Types.

Figure 3 shows the investigation of the number of clusters in the query type discovery component of the framework, with regard to the performance of **Augment(R+Q)** on MQ2007 and MQ2008. It can be observed that a larger number of query clusters generally yield better performance. The performance improvement is negligible when the number of query clusters exceeds a certain value.

Conclusions

We propose to handle the data imbalance issue in learning to rank via a data augmentation framework. An informative data generation model is designed based on adversarial autoencoder. In this model, relevance information is disentangled from the latent representation and query information is exploited for regularizing the prior latent distribution. Two data augmentation strategies are proposed. Experiments on benchmark datasets with state-of-the-art models demonstrate that the proposed data augmentation is effective in improving the LTR performance. Data imbalance problem in LTR is alleviated via the proposed informative data augmentation.

References

Agarwal, A.; Raghavan, H.; Subbian, K.; Melville, P.; Lawrence, R. D.; Gondek, D. C.; and Fan, J. 2012. Learning to rank for robust question answering. In *CIKM*, 833–842.

Batuwita, R., and Palade, V. 2010. Efficient resampling methods for training support vector machines with imbalanced datasets. In *IJCNN*, 1–8. IEEE.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML*, 89–96. ACM.

Burges, C. J.; Ragno, R.; and Le, Q. V. 2007. Learning to rank with nonsmooth cost functions. In *NIPS*, 193–200.

Cao, Y.; Xu, J.; Liu, T.-Y.; Li, H.; Huang, Y.; and Hon, H.-W. 2006. Adapting ranking svm to document retrieval. In *SIGIR*, 186–193. ACM.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML*, 129–136. ACM.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16:321–357.

Chawla, N. V. 2003. C4. 5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *ICML*, volume 3, 66.

Dietterich, T. G. 2000. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, 1–15. Springer.

Fernández-Navarro, F.; Hervás-Martínez, C.; and Gutiérrez, P. A. 2011. A dynamic over-sampling procedure based on sensitivity for multi-class problems. *Pattern Recognition* 44(8):1821–1833.

Galar, M.; Fernández, A.; Barrenechea, E.; and Herrera, F. 2013. Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition* 46(12):3460–3471.

Good, P. I. 2006. *Resampling methods*. Springer.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*, 2672–2680.

Han, H.; Wang, W.-Y.; and Mao, B.-H. 2005. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing (ICIC)*, 878–887. Springer.

He, H., and Garcia, E. A. 2009. Learning from imbalanced data. *TKDE* 21(9):1263–1284.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *SIGKDD*, 133–142. ACM.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Liu, Y.; An, A.; and Huang, X. 2006. Boosting prediction accuracy on imbalanced datasets with svm ensembles. In *PAKDD*, 107–118. Springer.

Liu, T.-Y., et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3(3):225–331.

Liu, X.-Y.; Wu, J.; and Zhou, Z.-H. 2009. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39(2):539–550.

Lowd, D., and Meek, C. 2005. Adversarial learning. In *SIGKDD*, 641–647. ACM.

Macdonald, C.; Santos, R. L.; and Ounis, I. 2013. The whens and hows of learning to rank for web search. *Information Retrieval* 16(5):584–628.

Makhzani, A.; Shlens, J.; Jaitly, N.; Goodfellow, I.; and Frey, B. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Mariani, G.; Scheidegger, F.; Istrate, R.; Bekas, C.; and Malossi, C. 2018. Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*.

Pang, L.; Lan, Y.; Guo, J.; Xu, J.; Xu, J.; and Cheng, X. 2017. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *CIKM*, 257–266. ACM.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

Qin, T., and Liu, T.-Y. 2013. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*.

Severyn, A., and Moschitti, A. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, 373–382. ACM.

Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; and Webb, R. 2017. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2107–2116.

Verberne, S.; van Halteren, H.; Theijssen, D.; Raaijmakers, S.; and Boves, L. 2011. Learning to rank for why-question answering. *Information Retrieval* 14(2):107–132.

Wan, S.; Lan, Y.; Xu, J.; Guo, J.; Pang, L.; and Cheng, X. 2016. Match-srnn: modeling the recursive matching structure with spatial rnn. In *IJCAI*, 2922–2928.

Wang, S.; Minku, L. L.; and Yao, X. 2015. Resampling-based ensemble methods for online class imbalance learning. *TKDE* 27(5):1356–1368.

Xu, J., and Li, H. 2007. Adarank: a boosting algorithm for information retrieval. In *SIGIR*, 391–398. ACM.

Yen, S.-J., and Lee, Y.-S. 2009. Cluster-based undersampling approaches for imbalanced data distributions. *Expert Systems with Applications* 36(3):5718–5727.

Zheng, Z.; Zheng, L.; and Yang, Y. 2017. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In *CVPR*, 3754–3762.