Content Recommendation by Noise Contrastive Transfer Learning of Feature Representation

Yiyang Li[†], Guanyu Tao[‡], Weinan Zhang[†], Yong Yu[†], Jun Wang[♯*]

[†]Shanghai Jiao Tong University, [‡]ULU Technologies Inc., [♯]University College London
{henryly94,wnzhang,yyu}@sjtu.edu.cn, guanyu.tao@ulu.ai, jun.wang@cs.ucl.ac.uk

ABSTRACT

Personalized recommendation has been proved effective as a content discovery tool for many online news publishers. As fresh news articles are frequently coming to the system while the old ones are fading away quickly, building a consistent and coherent feature representation over the ever-changing articles pool is fundamental to the performance of the recommendation. However, learning a good feature representation is challenging, especially for some small publishers that have normally fewer than 10,000 articles each year. In this paper, we consider to transfer knowledge from a larger text corpus. In our proposed solution, an effective article recommendation engine can be established with a small number of target publisher articles by transferring knowledge from a large corpus of text with a different distribution. Specifically, we leverage noise contrastive estimation techniques to learn the word conditional distribution given the context words, where the noise conditional distribution is pre-trained from the large corpus. Our solution has been deployed in a commercial recommendation service. The large-scale online A/B testing on two commercial publishers demonstrates up to 9.97% relative overall performance gain of our proposed model on the recommendation click-though rate metric over the non-transfer learning baselines.

KEYWORDS

Noise Contrastive Estimation, Transfer Learning, Word2Vec, Text Representation, Article Recommendation

1 INTRODUCTION

Recommendation has been proved effective in many online news publishers for promoting relevant content to the end users. Typically, for each article that a user is currently browsing, the recommended articles are listed beside the article for the user's further read. A typical example is given in Figure 1. Relevant article recommendation would lift the users' volume on the online news publisher, for example Google News improved its traffic by 38% via its personalized recommender system [6].

With the recent development of platform as a service (PaaS) business, the recommendation technologies are being used by smaller publishers in the form of application program interfaces (APIs) by third-party recommendation services, including companies like Taboola[1], Outbrain[2] and ULU Technologies[3]. These services normally receive the publisher recommendation requests with the user ID (cookies for the web and the hashed device ID for the mobile) and the context article, and return a list of recommended article IDs to the publisher. Then the publisher loads the corresponding titles, abstract and thumbnail of each recommended article along with the organic content. Such recommendation service enables the long tail publishers to deploy high-quality recommendation service with little engineering cost. Thanks to the inter-publisher volume exchange or ads display via the recommendation panel, PaaS recommendations has developed quickly during the recent two years and have served thousands of long tail publishers. It has been reported that Outbrain serviced over 35,000 websites with over 250 billion recommendations and 15 billion page views per month, while Taboola reached over 1 billion unique users worldwide, including 250 million mobile users. ULU Technologies, a startup PaaS content recommender system on which we deploy our algorithms, has reached more than 320 million unique users and tracks 1.5 billion page views per month over tens of publishers.

In in-house content recommendation systems that only for a single large publisher, e.g., Google News [6], Yahoo! News [20], collaborative filtering (CF) is more often employed to leverage rich user-item interaction data [13, 15]. By contrast, for article recommendation over multiple long tail publishers, the content-based techniques are more widely used. In the situation where user overlaps across different publishers are hardly observed, collaborative filtering is likely to suffer from a cold-start problem [21, 43]. Moreover, most news articles stay relevant and attractive only within one or two days after their publications. The content-based techniques would help exploit the rich information within the article text by learning the representation cross articles, making it less dependent on user behavior data.

However, because a typical PaaS recommender system serves lots of publishers, many of which are long tail with fewer than 10,000 articles published each year, the text representation learning on each of them may not be an easy task. Due to the small corpus of the articles, it is much difficult to train satisfactory article text

*The corresponding author: Weinan Zhang.

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132855

[1]https://www.taboola.com/
[2]http://www.outbrain.com
[3]http://www.ulu.ai

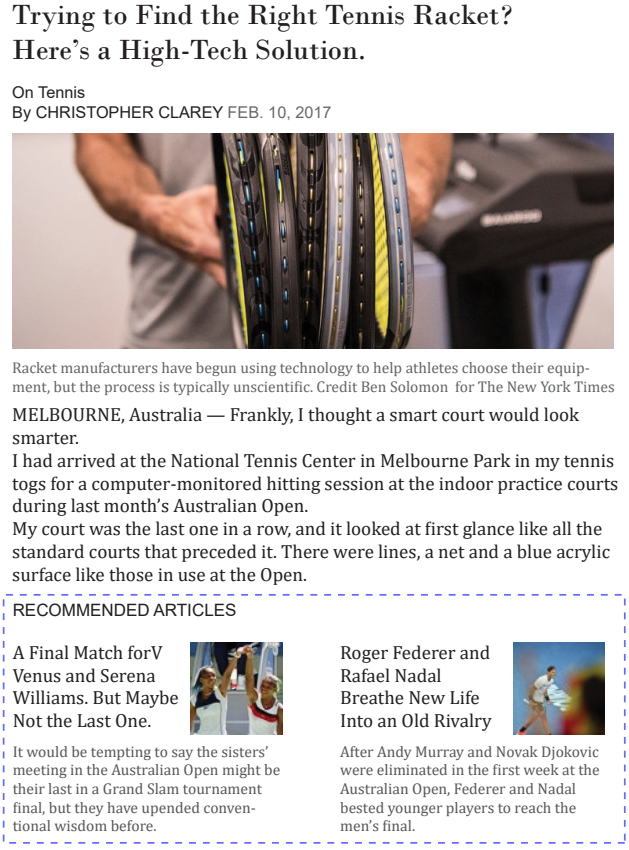
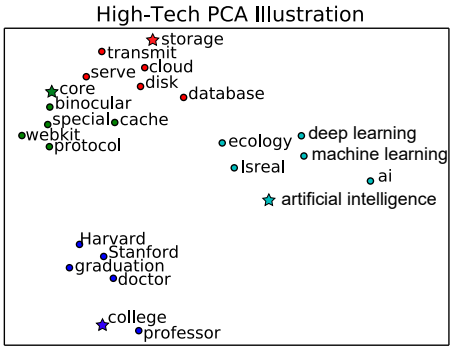
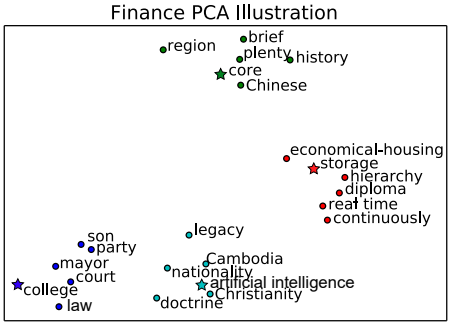


Figure 1: An illustration of related article recommendation scenario we work with.



(a) high-tech publisher



(b) finance publisher

Figure 2: The 2D PCA of the representation word vectors learned from the content corpus of two different publishers. There are 4 words, i.e., 'core', 'storage', 'artificial intelligence' and 'college', with their nearest words shown in both subfigures. It is obvious to see that the nearest neighbor words are highly different, which means the same word's representations across different publishers are indeed different.

representation based on either bag-of-words [32] or Word2Vec [24]. Unfortunately, directly applying text representation learned from other publishers fails to work, because the data distribution of each publisher, especially the small vertical publisher, is different from others. Figure 2 shows a standard principal component analysis (PCA) on article representations from two publishers on high-tech and finance respectively. The vector representation of same word (e.g., 'core') learned from the corpus of the two publishers are quite different (in terms of the nearest words). This reveals the fact that even the same word can have different meaning across publishers, which is a practical problem for many PaaS recommender systems.

In this paper, we propose a novel text representation method by knowledge transfer from another large corpus with sufficient text data [28]. Unlike previous transfer learning solutions [29, 42], we utilize noise contrastive estimation (NCE) [8] to make the transfer. NCE is typically used in Word2Vec language model training [24, 25] for accelerating the learning of softmax-based word conditional distribution. NCE would help the model learn to figure out the difference between the data distribution and the noise distribution [7, 8]. Such a learning scheme can be naturally borrowed to our case of training Word2Vec-based language model: the noise word conditional distribution is implemented by the one trained on the source corpus, based on which we also initialize the word conditional distribution on the target domain. Then the NCE will drive the target domain model learn to distinguish between the target next-word conditional distribution and the source one, which makes the transfer learning. In our Word2Vec experiment, the NCE transfer learning method yields a 73.8% performance improvement on next-word probability prediction over the non-transfer baselines.

Moreover, based on the NCE transfer learning, the Word2Vec in the target domain is leveraged to build the article text representation and then used for the content-based article recommender system. The entire system has been deployed on the commercial PaaS article recommender system of ULU Technologies. In our 8-day online A/B testing on two well-known Chinese publishers, we observed 9.97% and 8.27% overall performance improvement on user click-through rate of the recommended articles against the non-transferred version, respectively.

To sum up, our technical contributions and novelty are twofold:

- We propose a novel content representation transfer learning scheme based on noise contrastive estimation, which could adaptively learn the difference of the word conditional distributions

between the target and source domains so as to better learn the target text representation based on the source one. Note that although NCE is a widely adopted technique in machine learning, to our knowledge, no one has previously leveraged NCE to perform knowledge transfer from the source domain (i.e. noise distribution) to the target domain (i.e. the data distribution to learn).
- Based on the transfer learned content representation, the experimental results of online A/B testing over two well-known Chinese news publishers demonstrate significant and consistent improvement of user click-through rate on the recommended articles than the non-transfer baselines.

In the rest of the paper, we will discuss related work in Section 2. After providing the preliminaries in Section 3, we present our transfer learning solution in Section 4. We will then introduce the deployed recommender system in Section 5. The experiments and the corresponding results will be discussed in Section 6. Finally, we conclude this paper and discuss some future work in Section 7.

2 RELATED WORK

Content-based recommendation is one of the major item recommendation techniques [22, 30]. It refers to recommending an item to the target user based on the match between the description of the item and the user's profile. Generally, the description of an item is about any of its attributes. For example, the description of a restaurant could be its name, cuisine type, location, service type, price level, and a piece of text description etc [30]. The content-based are unlikely to have a cold-start problem for new items or users, which is a common shortcoming for collaborative filtering techniques [13, 15].

Recommendation of textual items, e.g., news articles, research papers or social posts, is a typical scenario for content-based recommendation, where the item could be described mainly by a piece of free text [30]. To deal with such unrestricted text, many personalization systems use bag-of-words text representation, which regards each word (after stemming) as an independent dimension and calculates some statistics like TFIDF [2] to formulate each piece of text into a vector. As each piece of text usually involves a small portion of the whole vocabulary, the vector is highly sparse.

A typical user profile consists of demographic attributes of the user and the items she has consumed or are consuming (implicit feedback [14]). For the news article recommendation studied in this work, the user profile would be the users' recently read articles and the current article.

The text recommendation modeling focuses on how to match the user and text profiles, which depends on the form of text representation. For the bag-of-words text representation, the matching algorithm would be the cosine similarity between the TFIDF vectors of the user and text profiles [30]. In advance, learning some latent structures of both profiles helps make such match more flexible and learnable, extending to semantic and topic match [20, 26, 41]. The authors in [26] proposed to leverage naive Bayes classifier to categorize the book content and match it to each candidate book recommendation slot. Similar to content taxonomy based match, building tagging systems for articles and users and then matching them based on tags is flexible and explainable [16, 35]. Furthermore, latent factor based models such as latent Dirichlet allocation (LDA) [4] and singular value decomposition (SVD) are introduced to more adaptively learn the text representation and its interaction with the user behavior [20, 33].

As an alternative approach to the discrete text representation of bag-of-words models and the latent factor models built based on them, neural network models are leveraged to learn distributed representation [10] of the text, where there is a real valued vector learned for each word (Word2Vec) [23]. Such word vectors can be learned via a neural network architecture with a specific loss function, such as the likelihood of word generation in a word sequence window, namely continuous bag-of-words (CBOW) [23] or skip-gram [24]. Based on word distributed representation or Word2Vec, the neural language model are proposed [3] and the distributed representation of sentences or articles are further proposed [19].

The advantage of the distributed text representation is that it can be trained with a deep neural network and thus largely explores the underlying patterns of the inter-word interaction via the high capacity of the neural network fed with large amount of data [23]. Typically, noise contrastive estimation (NCE) [8] is adopted for efficient training of Word2Vec or neural probabilistic language model with a softmax conditional distribution over the large word vocabulary. NCE tries to distinguish the data distribution from a predefined noise distribution, thus bypasses the computational complexity of directly fitting the data distribution. As studied in [7, 8], the noise distribution acts as the basis of NCE and highly influences the quality of the model predicted data distribution.

A significant drawback of the existing work on learning distributed text representation is that it heavily relies on large amount of data in order to train a useful model. In this paper, we address the issue by learning a text representation for a small target domain (i.e., the small publisher) by knowledge transfer from a source domain (i.e., the large universal text corpus). In this case, the feature spaces stay the same for both domains, but the data distributions and predictive functions are different. There are different transfer learning methods, such as instance selection or reweighting [5], feature selection [11] or feature mapping [27], and model parameter transfer [31]. In this paper, we adopt a feature transfer learning approach as it focuses on transferring the data representation knowledge between the two domains considered. Our training scheme is related to a previous work on user ad click behavior prediction via transfer learning [38], where the authors used a prior distribution to learn a logistic regression or factorization machine model, whereas we, for the first time, use negative contrastive estimation to establish the link between the target and source domains. Our work is also closely related to a previous work on Word2Vec domain adaption, called context vector concatenation (CVC) [37], where the authors proposed to concatenate the target domain vector of each word with its fixed vector pre-trained in source domain to encourage knowledge transfer. Such a straightforward solution might not be flexible as it directly fixes the "common knowledge" of two domain data. Also CVC introduces two times of feature vector dimensions for the word representation, which could be redundant or of high complexity. By contrast, our NCE transfer learning method learns the difference between two domain data distributions in a more

flexible way without introducing any higher complexity of feature representations. We will compare our method with [37] in the experiment.

For the recommendation problem via transfer learning, to our knowledge, there is only previous work on transfer learning solutions for collaborative filtering-based recommendation [29, 40, 42] but none for content-based article recommendation via transferring the content representations, which is the position of this work.

3 PRELIMINARIES

To make our paper self-contained, in this section we briefly discuss the skip-gram Word2Vec model and the noise contrastive estimation training scheme before presenting our technical contribution in Section 4.

3.1 Skip-Gram Word2Vec

In a Word2Vec model, a distributed representation of a word is given as a d-dimensional embedding vector $\mathbf{v}_w \in \mathbb{R}^d$ for each word w, which is learned by training a neural network with task-specific data and loss function. We define the parameter set $\theta = \{\mathbf{v}\}$ of a Word2Vec model. A widely used Word2Vec model is skip-gram [24], where given word sequences from a training corpus, the model maximizes the following log-likelihood estimation (MLE):

$$\max_{\theta} \frac{1}{T} \sum_{i=1}^{T} \sum_{o=i-c}^{i+c} \log p_\theta(w_o|w_i), \tag{1}$$

where c is the context word windows size, and the iterator i goes over all T possible center words with c-size context word window in the text corpus of the target publisher. The θ-parameterized word conditional probability $p_\theta(w_o|w_i)$ is defined by a softmax distribution over the whole vocabulary W:

$$p_\theta(w_o|w_i) = \frac{\exp(f_\theta(w_o, w_i))}{\sum_{w \in W} \exp(f_\theta(w, w_i))}, \tag{2}$$

where the scoring function $f_\theta(w_o, w_i)$ is normally implemented with vector inner product

$$f_\theta(w_o, w_i) = \mathbf{v}_{w_o} \cdot \mathbf{v}_{w_i}, \tag{3}$$

or a multi-layer neural network.

The output is the embedding vector for each word $\mathbf{v}_w$, and the conditional probability $p_\theta(w_o|w_i)$, which indicates how likely the word w_o will occur in the $\pm c$ context window of the word w_i.

3.2 Noise Contrastive Estimation

The training of skip-gram Word2Vec involves the gradient calculation of Eq. (1). For each (w_o, w_i) pair, the gradient of a parameter θ (could be $\mathbf{v}_{w_o}$ or $\mathbf{v}_{w_i}$) is given as:

$$\frac{\partial \log p_\theta(w_o|w_i)}{\partial \theta} = \frac{\partial f_\theta(w_o, w_i)}{\partial \theta} - \mathbb{E}_{w \sim p_\theta(w|w_i)}\Big[\frac{\partial f_\theta(w, w_i)}{\partial \theta}\Big], \tag{4}$$

where the calculation of the second term is expensive due to the large vocabulary size.

Noise contrastive estimation (NCE) [8] is proposed to accelerate such a training process by taking an alternative training objective to approximate the parameter gradient. For each word w_i, we observe (w_o, w_i) pairs sampled from the data distribution $p_d(w_o|w_i)$. For each observed pair (w_o, w_i), in NCE we sample K noise pairs $(w_k, w_i)_{k=1\dots K}$ from a known noise distribution $p_n(w_k|w_i)$. The NCE training objective is to maximize the log-likelihood of correctly distinguishing the data pair from the K noise pairs

$$\begin{aligned} J_\theta(w_o, w_i) = &\log \frac{p_\theta(w_o|w_i)}{p_\theta(w_o|w_i) + Kp_n(w_o|w_i)} \\ &+ \sum_{k=1}^{K} \log \frac{Kp_n(w_k|w_i)}{p_\theta(w_k|w_i) + Kp_n(w_k|w_i)}. \end{aligned} \tag{5}$$

As such, the NCE objective function $J_\theta(w_i)$ based on word w_i is

$$\begin{aligned} J_\theta(w_i) =&\mathbb{E}_{p_d(w_o|w_i)}[J_\theta(w_o, w_i)] \\ =&\mathbb{E}_{p_d(w_o|w_i)}\Big[\log \frac{p_\theta(w_o|w_i)}{p_\theta(w_o|w_i) + Kp_n(w_o|w_i)}\Big] \\ &+ K\mathbb{E}_{p_n(w_n|w_i)}\Big[\log \frac{Kp_n(w_n|w_i)}{p_\theta(w_n|w_i) + Kp_n(w_n|w_i)}\Big]. \end{aligned} \tag{6}$$

Taking the derivative of $J_\theta(w_i)$ w.r.t. θ, we have

$$\begin{aligned} \frac{\partial J_\theta(w_i)}{\partial \theta} = &\sum_{w_o \in W} \frac{Kp_n(w_o|w_i)}{p_\theta(w_o|w_i) + Kp_n(w_o|w_i)} \\ &\cdot \Big(p_d(w_o|w_i) - p_\theta(w_o|w_i)\Big) \frac{\partial \log p_\theta(w_o|w_i)}{\partial \theta}. \end{aligned} \tag{7}$$

It is proved that in [8] when $K \to \infty$, the gradient $\frac{\partial}{\partial \theta} J_\theta(w_i) \to \mathbb{E}_{p_d(w_o|w_i)}[\frac{\partial}{\partial \theta} \log p_\theta(w_o|w_i)]$, which is the MLE gradient as shown in Eq. (4) with the data observations. With such a nice property, NCE has been widely adopted in Word2Vec training [19, 23].

4 PROPOSED NCE TRANSFER LEARNING

A critical problem for NCE training of word embedding is that the number of noise samples K and noise distribution $p_n(w_o|w_i)$ highly influence NCE training performance [8, 25]. For publishers who have fewer articles, we propose to learn the word embedding based on knowledge transfer from another corpus with sufficient text data. Specifically, we take a large corpus as the source domain and employ it as the noise distribution in order to jointly learn the target distribution from the publisher's articles.

It is worthwhile noticing that the way we employ a source distribution to help the estimation of the target distribution is similar to the recent minimax game proposed in Generative Adversarial Nets (GAN) [7], with the difference that the source distribution (the generator) is fixed. A close look at Eq. (5) reveals that the MLE objective can be regarded as a classifier to distinguish the foreground word from the noise background (the source distribution), while the source distribution 'fools' the foreground model to improve the estimation. If the difference between the source and target is too large (thus the classification becomes too easy), it cannot produce any substantial gradient signal to train the model $p_\theta(w_o|w_i)$. Thus, a good choice of the noise distribution in NCE [8] (thus the source distribution) should be close to the data distribution (the target distribution).

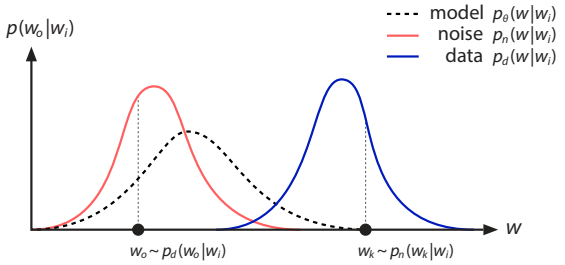


Figure 3: An example to show the missing of gradient $\frac{\partial}{\partial\theta}J_\theta(w_o, w_i)$ when $p_d(w|w_i)$ and $p_n(w|w_i)$ are much different (with little probability density overlap).

To see this further, let us take the derivative of $J_\theta(w_o, w_i)$ w.r.t. θ from Eq. (5):

$$\frac{\partial J_\theta(w_o, w_i)}{\partial\theta} = \frac{Kp_n(w_o|w_i)}{p_\theta(w_o|w_i) + Kp_n(w_o|w_i)} \frac{\partial \log p_\theta(w_o|w_i)}{\partial\theta} - \sum_{k=1}^{K} \frac{p_\theta(w_k|w_i)}{p_\theta(w_k|w_i) + Kp_n(w_k|w_i)} \frac{\partial \log p_\theta(w_k|w_i)}{\partial\theta}. \tag{8}$$

Figure 3 presents an example to explain such a "gradient vanishing" problem from Eq. (8) when the data and noise distributions are distant. In this example, as a positive word (the word from an article in the publisher website) is generated from the data distribution: $w_o \sim p_d(w_o|w_i)$, it typically satisfies $p_\theta(w_o|w_i) \gg p_n(w_o|w_i)$. As such, the first term in the right-hand side of Eq. (8) tends to be zero for a finite K; similarly, as a negative word (the word from a noise distribution): $w_n \sim p_n(w|w_i)$, it normally satisfies $p_n(w_n|w_i) \gg p_\theta(w_n|w_i)$; thus the second term in the right-hand side of Eq. (8) becomes close to zero too.

This is indeed the same assumption of transfer learning: the source and target tasks should be related [34]. Specifically, if we assume $p_s(w_o|w_i)$ in the source task is to-some-degree related to $p_t(w_o|w_i)$ in the target task, then NCE will produce helpful gradient signal to the learning of $p_t(w_o|w_i)$ given $p_s(w_o|w_i)$ as the noise distribution. The general idea of NCE transfer learning is illustrated in Figure 4. The noise distribution should be proposed as close to the data distribution. Although the true data distribution is unknown, we would leverage any domain knowledge to 'estimate' whether the noise and data distributions are to-some-degree close. Otherwise, NCE transfer learning will suffer the gradient vanishing problem as illustrated in Figure 3.

Specifically, given the well-trained word embedding vectors $\{\mathbf{v}^s\}$ on the source domain corpus and the resulted conditional word distribution

$$p_s(w_o|w_i) = \frac{\exp(\mathbf{v}^s_{w_o} \cdot \mathbf{v}^s_{w_i})}{\sum_{w\in W} \exp(\mathbf{v}^s_w \cdot \mathbf{v}^s_{w_i})}, \tag{9}$$

where our goal is to learn good word embedding vectors $\{\mathbf{v}^t\}$ for the words in the target publisher (and then the article representation, which will be discussed later).

For each skip-gram word pair (w_o, w_i) in the target publisher corpus with w_i as the center word, we sample K noise words based on $p_s(w_k|w_i)$, i.e., the conditional probability distribution trained

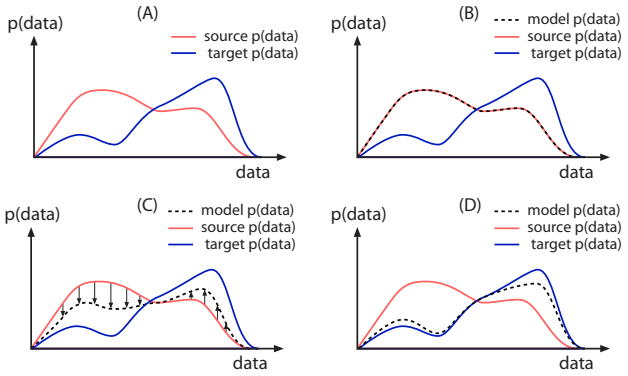


Figure 4: An illustration of NCE transfer learning: (A) the source and target (conditional) data distributions. (B) The model is initialized with the source data distributions. (C) The NCE gradient tries to distinguish the source and model data distributions by feeding the target data. The gradient is valid where both distributions have the same magnitude of probability density. (D) The final model data distribution may still has distance to the target one because of the model capacity and the target data insufficiency.

on the source domain with a large enough text corpus. Based on these $K+1$ word pairs, we apply the NCE training objective to maximize

$$J(w_o, w_i) = \log \frac{p_t(w_o|w_i)}{p_t(w_o|w_i) + Kp_s(w_o|w_i)} + \sum_{k=1}^{K} \log \frac{Kp_s(w_k|w_i)}{p_t(w_k|w_i) + Kp_s(w_k|w_i)}, \tag{10}$$

where the word conditional distribution in the target domain is similarly defined based on word vectors $\{\mathbf{v}^t\}$

$$p_t(w_o|w_i) = \frac{\exp(\mathbf{v}^t_{w_o} \cdot \mathbf{v}^t_{w_i})}{\sum_{w\in W} \exp(\mathbf{v}^t_w \cdot \mathbf{v}^t_{w_i})}. \tag{11}$$

The overall NCE transfer learning objective is

$$\max_{\{\mathbf{v}^t\}} \frac{1}{T} \sum_{i=1}^{T} \sum_{o=i-c}^{i+c} J(w_o, w_i), \tag{12}$$

where for each training round the iterator i goes over all the possible center words with c-size context word window in the text corpus of the target publisher.

Note that NCE transfer learning is practically very fast since the target domain text corpus is always small in our scenario. For example, each of the two NCE transfer learning tasks in our experiment takes about 1.5 hours on a server with one NVidia GeForce GTX 1080 GPU, one 12 Intel Core i7-6800K CPU @3.40GHz and 64GB RAM. Thus the training can be performed several times daily.

5 DEPLOYED RECOMMENDER SYSTEM

The NCE transfer learning discussed in Section 4 is expected to provide a better Word2Vec than the traditional one without knowledge transfer. Essentially, the new Word2Vec is of identical data

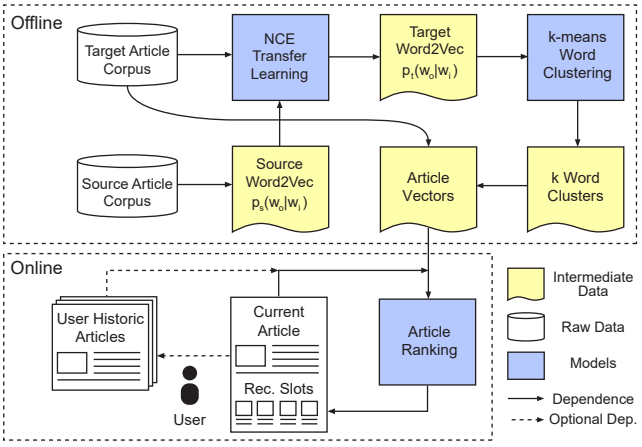


Figure 5: The architecture of the proposed word embedding via NCE transfer learning and how it is incorporated in the whole personalized recommender system.

structure with the old one, which is used in the model pipeline to build up the whole article recommender system. In this section, we briefly present the overall of the whole system.

The overall model pipeline of the system is shown in Figure 5, which can be separated into three parts.

- **NCE Word2Vec Transfer**: We first train the word vectors using the skip-gram model. And recall last part, we use a source article corpus to train the word vectors and $p_s(w_o|w_i)$ to provide noise samples in NCE transfer learning to output the word vectors for the target publisher.
- **Article Representation**: Then we perform a k-means to get the word clusters. Each cluster represents a certain set of words that appear similar to each other in the corpus. Then the article vector is built based on the word clusters.
- **Article Recommendation**: Based on the article vectors, for a user browsing the current article, we rank the candidate articles[4] according to the similarity function taking the user profile (the current article and optionally her historic articles) and the candidate article as input.

5.1 NCE Word2Vec Training

As discussed in Section 4, we propose to use NCE with the conditional word distribution trained on the source article corpus as the noise distribution to perform the knowledge transfer. Moreover, we perform an importance sampling to reduce the NCE sampling step [3]. Algorithm 1 shows how our algorithm works in a tensor form.

The functions used in Algorithm 1 work as the same way of the functions in TensorFlow (TF) [1]. We first sample some candidates from baseline vectors by uniform distribution. Then we calculate each candidate's softmax nominator value respectively. After that, we choose each candidate with the probability proportional to its softmax nominator value, thus produce the negative samples. Since importance sampling is unbiased, it can be proved that by this way

[4]The candidate articles are the ones marked as recommendable by the publisher editors.

Algorithm 1 Tensorized NCE with importance sampling

Input: $\mathbf{V}$: Word vector trained from source data
θ: Parameter in current model
K: Amount of required negative samples
Output: $\mathbf{V}_{\text{neg}}$
$\mathbf{V}_{\text{candidate}} \Leftarrow \text{Uniform\_Sample}(\mathbf{V})$
$\mathbf{z}_{\text{candidate}} \Leftarrow \text{Mat\_Mul}(\mathbf{V}_{\text{candidate}}, \theta)$
$\text{Index}_{\text{sampled}} \Leftarrow \text{Importance\_Sample}(\mathbf{z}_{\text{candidate}})$
$\mathbf{V}_{\text{neg}} \Leftarrow \text{Embedding\_Lookup}(\mathbf{V}, \text{Index}_{\text{sampled}})$

we will have a close distribution from the original one [3] (despite of the variance it may face [25]).

5.2 Word Clustering and Article Vector

For an article recommender system, it is practically important to model all articles in a unified data representation, i.e. article vectors, to tackle the problem of different-length articles.

With the NCE trained word vectors, we perform a k-means clusterings [9] for the whole set of words, with $k = 200$. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest centroid (i.e., the mean of the data vectors belonging to the cluster), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells [12]. The number of clusters was determined in advance according to the vocabulary size of each corpus and the preliminary recommendation performance.

With each word assigned to a certain cluster C_i and the cluster set denoted as C, we build up the vector representation $\mathbf{x}_d \in \mathbb{R}^k$ for the article d using these k clusters. Specifically, for each article, we map all its words into corresponding cluster IDs. Thus the article is represented by the 'bag-of-clusters'. The value of each cluster dimension i of the article vector is calculated as

$$\mathbf{x}_{di} = \sum_{w \in d} \text{TFIDF}(w)\delta(w \in C_i), \tag{13}$$

where the TFIDF [2] term weighting is adopted, $\delta(w \in C_i)$ is the delta function that equals to 1 if $w \in C_i$ and 0 otherwise.

Note that it is possible to adopt more advanced Paragraph2Vec or Doc2Vec models [19] to build the vector representation of articles. In this work we choose not to implement such models because we focus on scope on knowledge transfer to the target publisher, where the text corpus is too small to effectively train Word2Vec, let alone the Doc2Vec.

5.3 Article Recommendation

Given a user's current reading article vector d_0, suppose the system could access the user's recent reading history D, it is straightforward to score each candidate article d by a similarity function

$$g(d, d_0, D) = \alpha \cos(\mathbf{x}_d, \mathbf{x}_{d_0}) + (1 - \alpha)\frac{1}{|D|}\sum_{d_j \in D} \cos(\mathbf{x}_d, \mathbf{x}_{d_j}), \tag{14}$$

where α is the hyperparameter to balance the two cosine similarity scores. Specifically, we use Elasticsearch[5] to index the article vectors and retrieve the relevant candidate articles efficiently and

[5]https://www.elastic.co/

Table 1: Word2Vec source and target datasets.

Dataset	Training instances	Test instances	Total
Source	13,181,278	-	13,181,278
high-tech	82,387	35,308	117,695
finance	56,000	24,000	80,000

finally perform the candidate ranking according to $g(d, d_0, D)$ of each candidate d.

For specific product, other types of recommendation, e.g., popular new articles, CF based recommendations for warm users, and some editors' recommendations, would be deployed to blend the recommendation list but this is out of the scope of this paper.

6 EXPERIMENTS

The proposed NCE Word2Vec transfer learning program[6] has been deployed on ULU Technologies recommender system, serving tens of publishers in China and United States. In this section, we present the experimental results with both offline Word2Vec conditional log-likelihood performance and the article recommendation click performance during an 8-day A/B testing on two publishers.

Specifically, these two publishers are well known in China and United States, one on high-tech and one on finance. For business concerns, we have to anonymize their names.

6.1 Offline Word2Vec Evaluation

6.1.1 Experimental Setting. Prior to deploying the NCE transferred Word2Vec into the recommendation model pipeline, we perform an offline evaluation on the trained Word2Vec. Two publishers finance and high-tech are used respectively as the target data, and the large text corpus crawled from the web is used as the source data. Each publisher's dataset is separated into training and test sets with a 7:3 ratio. Each data instance is a (w_o, w_i) pair picked from the sentences of the publisher's text corpus. The window size $c = 5$. Consider the source text corpus could be biased to influence the target performance, we try to include all kinds of crawled web text to make the corpus of high diversity, leading to a 13.2 million article source corpus with 8.15 trillion word pairs. Table 1 shows the details of the datasets.

We first use traditional skip-gram Word2Vec to train vectors from source data denoted as W_b. Then we use W_b as both a baseline and the noise distribution. Then we use our proposed NCE-Transferred Word2Vec to perform the transfer learning. It generated the experimental model denoted as W_e. Then we use two sets of word vectors and test data set to conduct the offline evaluation, calculating the average negative log-likelihood (NLL) of the word conditional occurrence $-\log p(w_o|w_i)$ by each model.

The pre-defined model hyperparameter setting for the whole experiment are listed: stochastic gradient descent (SGD) learning rate 0.2, batch size 128, minimum word counting 10, window size $c = 5$ and subsampling limit 0.001 in Word2Vec, respectively. Those values were chosen by preliminary experiments so that we can focus on main parameter study such as word embedding size and the models performance. The model was built on Tensorflow (TF) [1] based on CUDA 7.5.

[6]We publish the NCE Word2Vec transfer learning experiment code at https://github.com/GuanyuTao/cross-media-word2vec

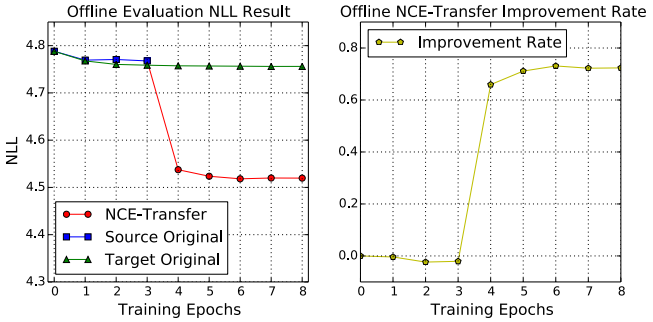


Figure 6: Offline evaluation results and the performance lift the NCE-transfer brings on high-tech publisher. Specifically, the NCE transfer learning is performed after the 3-rd training epoch of Source Original, which has already converged.

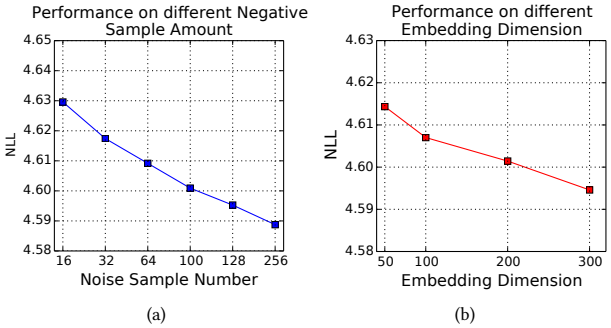


Figure 7: Hyperparameter study for the model: the NLL performance against the tuned noise sample number and word embedding dimension.

Table 2: Overall performance comparison.

Model	Performance (NLL)
NCE-Transfer	**4.537**
CVC [37]	4.661

6.1.2 Performance and Results. Our offline experiment is focused on the evaluation the NLL metric of the conditional word probabilistic distribution $p_\theta(w_o|w_i)$ based on the trained word vectors. Such a performance indicates the quality of the learned word representation, which is the fundamental of the subsequent recommendation pipeline.

There has been some work done on applying domain adaptation on a recurrent neural network (RNN) language model [37]. In Table 2 we compared one of its networks, context vector concatenation (CVC), with our model. It is seen clearly that applying our NCE transfer directly on word vectors has better performance against the RNN language model.

Figure 6 presents the NLL performance and the NCE-transfer performance lift w.r.t. to the training epochs on high-tech while the results on finance are similar. The NCE transfer learning scheme is added after the third training epoch on the source data, where the

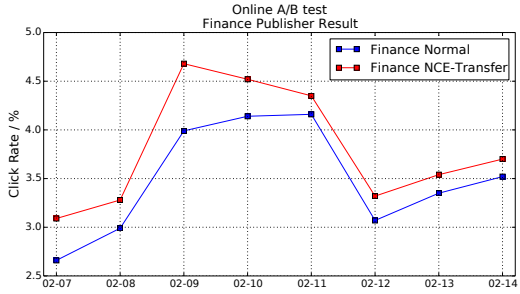
Table 3: Online A/B testing CTR performance.

finance publisher

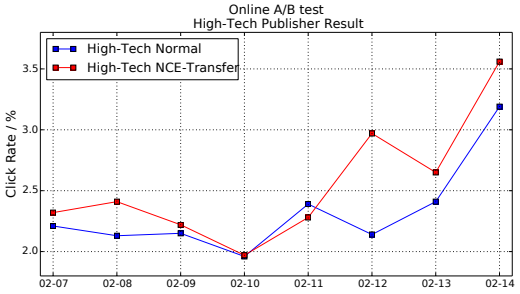
	Normal		NCE Transfer		CTR
Date	Vol. (k)	CTR	Vol. (k)	CTR	Impv.
02-07	198.9	2.66%	200.5	3.09%	16.17%
02-08	217.2	2.99%	217.5	3.28%	9.70%
02-09	175.0	3.99%	177.5	4.68%	17.29%
02-10	166.4	4.14%	167.5	4.52%	9.18%
02-11	86.9	4.16%	87.1	4.35%	4.57%
02-12	110.7	3.07%	113.3	3.32%	8.14%
02-13	168.9	3.35%	163.3	3.54%	5.67%
02-14	148.2	3.52%	151.5	3.70%	5.11%
Overall	1,272.1	3.42%	1,278.3	3.76%	9.97%

high-tech publisher

	Normal		NCE Transfer		CTR
Date	Vol. (k)	CTR	Vol. (k)	CTR	Impv.
02-07	25.52	2.21%	25.94	2.32%	4.98%
02-08	26.66	2.13%	27.92	2.41%	13.15%
02-09	30.37	2.15%	31.16	2.22%	3.26%
02-10	26.78	1.96%	26.82	1.97%	0.51%
02-11	9.67	2.39%	8.84	2.28%	-4.60%
02-12	9.58	2.14%	9.81	2.97%	38.79%
02-13	28.74	2.41%	27.65	2.64%	9.54%
02-14	16.44	3.19%	17.00	3.56%	11.61%
Overall	173.76	2.28%	175.16	2.47%	8.27%



(a) finance publisher



(b) high-tech Publisher

Figure 8: Online A/B test of CTR performance over 8 days.

noise conditional distribution $p_n(w_n|w_i)$ is set as the model p.d.f. $p_\theta(w_n|w_i)$ trained on the source data. We can see the results and the improvement the NCE-transfer learning brings once deployed, which indicates the effectiveness of leveraging NCE for Word2Vec transfer learning: the NLL drops from 4.76 (only target data trained) to 4.52 (NCE transfer trained), which suggests a 73.8% increased probability that the model would choose the correct next-word[7].

In addition, the hyperparameter study is shown in Figure 7, where we track the NLL performance by tuning noise sample number K in Eq. (10) and word embedding size d, respectively. We can observe that both higher noise sample number K and higher word embedding dimension d lead to better NLL performance, which is consistent with the NCE gradient approximation to MLE gradient when $K \to \infty$. However, picking large K and d will lower the efficiency of the NCE training. The empirical optimal K and d depends on the tradeoff between the performance requirement and the system configuration. We set $K = 100$ and $d = 200$ to train Word2Vec and its NCE transfer learning, and output the produced word vectors to the recommender system model pipeline for further steps.

6.2 Online A/B Testing

6.2.1 Experimental Setting. The NCE transferred Word2Vec has been deployed into ULU Technologies recommender system. Via offering recommendation services APIs, the system processes over 30 million recommendation requests daily. To compare the recommendation results of the model pipelines with and without the NCE transferred Word2Vec, we conducted an 8-day A/B testing over finance and high-tech publishers from Feb. 7 2017 to Feb. 14 2017. For each publisher, we randomly allocate each user's cookie into the tested recommendation bucket with NCE transferred Word2Vec with 50% probability and the control bucket with traditional Word2Vec[8]. For industrial platforms, by comparing the user click-through rate (CTR) by the tested and control systems, we can check whether the proposed NCE-transferred recommendation really works in practice. We choose not to use the learning-to-rank performance metrics (e.g. NDCG or MAP [39]) because the recommendation panel may not be in a list format.

The allocated ULU recommender service for A/B testing is deployed on a three-node clusters on Aliyun Elastic Compute Service. Each server is in CentOS 7.2 with 2 cores CPU and 16GB RAM. Tomcat is used for API and Nginx is used for load balance.

6.2.2 A/B Testing Results. Table 3 shows the detailed results of each day, including the recommendation request volume, the CTR performance of the normal recommendation and the NCE transferred recommendation. Figure 8 illustrates the CTR changes during the 8-day A/B test. It is observed that (i) NCE transferred recommendation offers an overall higher CTR performance on both publishers, with 9.97% and 8.27% improvement respectively. (ii) On finance publisher, NCE transferred recommendation consistently yields higher CTR than the traditional baseline over all the 8 days. (iii) On high-tech publisher, the CTR improvement is more fluctuant, which could be as high as 38.79% improvement (on Feb. 12) and as lower as -4.60% (on Feb. 11). This would be caused by the low volume of the publisher. (iv) The CTR on two publishers are in the range of 2-4%, which is relatively low compared with some

[7] Caclulated by $(10^{-4.52} - 10^{-4.76})/10^{-4.76} = 73.8\%$.

[8] For recommendation quality concern of the commercial platform, we choose not to compare some simple baselines, such as random recommendation.

previous reported figures on other platforms [6, 17]. The reason is that the recommendation slots are at the bottom of the webpage, which would be missed by users. The finance publisher conducted their first-hand investigation and demonstrated that based on the volume user scrolling down to the recommendation slots, our recommendation achieves a 16.2% CTR performance.

In sum, the online A/B testing results indicate that the proposed NCE-transfer training scheme for Word2Vec is promising for improving the performance of article recommendation on small publishers with insufficient text data.

7 CONCLUSION

In this paper we have presented a PaaS content recommender system based on word embedding representation learning. To address the problem of content data sparsity on long tail publishers, we propose a transfer learning scheme that initializes the Word2Vec of a target publisher with the one trained from a large source corpus, and then leverages noise contrastive estimation to learn to diverge from the initialized word embeddings by feeding the target publisher content data. In the offline word representation evaluation, we observed a significantly higher log-likelihood of next-word prediction based on the NCE transferred word embeddings than the traditional ones, which indicates a 73.8% increased probability that the model would choose the next word correctly. More importantly, in the 8-day online A/B testing stage, the recommender system based on the NCE transferred word embeddings shows 9.97% and 8.27% CTR improvement on two publishers respectively, which indicates the practical effectiveness of the proposed transfer learning scheme. From the engineering perspective, the proposed NCE transfer learning scheme is easy to be deployed and debug-friendly, which makes it potential to be adopted in various intelligent systems based on text representation, such as the named entity recognition [18] and entity relation classification [36], which is our planned future research work. Also we plan to consider the anchor information of the web pages to assign different importance to the text in difference web fields (e.g. higher importance for title words).

Acknowledgement The SJTU team is financially supported by NSFC (61632017) and Shanghai Sailing Program (17YF1428200).

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, et al. 2016. TensorFlow: A system for large-scale machine learning. In *OSDI*.
[2] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*. Vol. 463. ACM press New York.
[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JMLR* 3, Feb (2003), 1137–1155.
[4] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.
[5] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2007. Boosting for transfer learning. In *ICML*. ACM, 193–200.
[6] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 271–280.
[7] Ian J Goodfellow. 2014. On distinguishability criteria for estimating generative models. *arXiv preprint arXiv:1412.6515* (2014).
[8] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, Vol. 1.
[9] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28, 1 (1979), 100–108.
[10] Geoffrey E Hinton. 1984. Distributed representations. (1984).
[11] Jing Jiang and ChengXiang Zhai. 2007. A two-stage approach to domain adaptation for statistical classifiers. In *CIKM*. ACM, 401–410.
[12] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. 2002. An efficient k-means clustering algorithm: Analysis and implementation. *PAMI* 24, 7 (2002), 881–892.
[13] Noam Koenigstein, Gideon Dror, and Yehuda Koren. 2011. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys*. ACM, 165–172.
[14] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. ACM, 426–434.
[15] Yehuda Koren, Robert Bell, Chris Volinsky, et al. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[16] Ralf Krestel and Peter Fankhauser. 2012. Personalized topic-based tag recommendation. *Neurocomputing* 76, 1 (2012), 61–70.
[17] P. Lamere and S. Green. 2008. Project Aura: recommendation for the rest of us. *Presentation at Sun JavaOne Conference* (2008).
[18] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016).
[19] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*, Vol. 14. 1188–1196.
[20] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*.
[21] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. 2014. Facing the cold start problem in recommender systems. *Expert Systems with Applications* 41, 4 (2014), 2065–2073.
[22] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*. Springer, 73–105.
[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[25] Andriy Mnih and Yee W Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *ICML*.
[26] Raymond J Mooney and Loriene Roy. 2000. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*. ACM, 195–204.
[27] Sinno Jialin Pan, James T Kwok, and Qiang Yang. 2008. Transfer Learning via Dimensionality Reduction. In *AAAI*, Vol. 8. 677–682.
[28] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.
[29] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. 2010. Transfer Learning in Collaborative Filtering for Sparsity Reduction. In *AAAI*.
[30] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.
[31] Claudia Perlich, Brian Dalessandro, Troy Raeder, Ori Stitelman, and Foster Provost. 2014. Machine learning for targeted display advertising: Transfer learning in action. *Machine learning* 95, 1 (2014), 103–127.
[32] Magnus Sahlgren and Rickard Cöster. 2004. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *ACL*. 487.
[33] Xiance Si and Maosong Sun. 2009. Tag-LDA for scalable real-time tag recommendation. *Journal of Computational Information Systems* 6, 1 (2009), 23–31.
[34] Shiliang Sun. 2013. A survey of multi-view machine learning. *Neural Computing and Applications* 23, 7-8 (2013), 2031–2038.
[35] Jun Wang, Maarten Clements, Jie Yang, Arjen P de Vries, and Marcel JT Reinders. 2010. Personalization of tagging systems. *IPM* 46, 1 (2010), 58–70.
[36] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation Classification via Convolutional Deep Neural Network. In *COLING*. 2335–2344.
[37] Jian Zhang, Xiaofeng Wu, Andy Way, and Qun Liu. 2016. Fast Gated Neural Domain Adaptation: Language Model as a Case Study. In *ACL*.
[38] Weinan Zhang, Lingxi Chen, and Jun Wang. 2016. Implicit Look-Alike Modelling in Display Ads. In *ECIR*. Springer, 589–601.
[39] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *SIGIR*. ACM.
[40] Weinan Zhang, Ulrich Paquet, and Katja Hofmann. 2016. Collective Noise Contrastive Estimation for Policy Transfer Learning. In *AAAI*. 1408–1414.
[41] Weinan Zhang, Dingquan Wang, Gui-Rong Xue, and Hongyuan Zha. 2012. Advertising keywords recommendation for short-text web pages using Wikipedia. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 2 (2012), 36.
[42] Lili Zhao, Sinno Jialin Pan, Evan Wei Xiang, Erheng Zhong, Zhongqi Lu, and Qiang Yang. 2013. Active Transfer Learning for Cross-System Recommendation.. In *AAAI*. Citeseer.
[43] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. 2013. Interactive collaborative filtering. In *CIKM*. ACM, 1411–1420.