Behavior Matching between Different Domains based on Canonical Correlation Analysis

Lyu Yuan
lyu.yuan@ist.osaka-u.ac.jp
Osaka University
Suita-shi, Osaka, Japan

Daichi Amagata
amagata.daichi@ist.osaka-u.ac.jp
Osaka University
Suita-shi, Osaka, Japan

Takuya Maekawa
maekawa@ist.osaka-u.ac.jp
Osaka University
Suita-shi, Osaka, Japan

Takahiro Hara
hara@ist.osaka-u.ac.jp
Osaka University
Suita-shi, Osaka, Japan

Niu Hao
ha-niu@kddi-research.jp
KDDI Research, Inc.
Chiyoda-ku, Tokyo, Japan

Kei Yonekawa
ke-yonekawa@kddi-research.jp
KDDI Research, Inc.
Chiyoda-ku, Tokyo, Japan

Mori Kurokawa
mo-kurokawa@kddi-research.jp
KDDI Research, Inc.
Chiyoda-ku, Tokyo, Japan

ABSTRACT

With the recent proliferation of e-commerce services, online shopping has become more and more popular among customers. Because it is necessary to recommend proper items to customers, to improve the accuracy of recommendation, high-performance recommender systems are required. However, current recommender systems are mainly based on information of their own domain, resulting in low accurate recommendation for customers with limited purchasing histories. The accuracy may suffer due to a lack of information. In order to use information from other domains, it is necessary to associate behaviors in different domains of the behaviorally related users. This paper presents a preliminary analysis of matching behaviors of the behaviorally related users in different domains. The result shows that we got a better prediction rate than linear regression.

CCS CONCEPTS

• **Information systems → Association rules**.

KEYWORDS

behavior matching, canonical correlation analysis, multi-domains

ACM Reference Format:
Lyu Yuan, Daichi Amagata, Takuya Maekawa, Takahiro Hara, Niu Hao, Kei Yonekawa, and Mori Kurokawa. 2019. Behavior Matching between Different Domains based on Canonical Correlation Analysis. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3308560.3316595

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3316595

1 INTRODUCTION

Online shopping has nowadays become increasingly popular. Due to the abundance of items, however, customers may have difficulty in making selections among them. To avoid such situations, online shopping systems employ item recommendation. Normally, recommender systems utilize content-based recommendations [6]. Based on item information of user histories, user preference profiles are generated. First, some item features are used to represent the items. Next, some user history information is used to generate user preference profiles. Finally, by comparing the user preference profiles with the item features, the system recommends items most related to the users [10].

On the other hand, many tasks use collaborative filtering because users often get the best recommendations from someone with similar tastes and interests to themselves. Based on this phenomenon, collaborative filtering matches people with similar interests and helps make recommendations [15]. To be more specific, collaborative filtering is a method of filtering for information or patterns utilizing techniques involving collaboration among multiple agents, viewpoints, data sources, etc [17]. Tasks employing collaborative filtering typically have large data sets. Collaborative filtering methods have been used in many situations. Examples include sensing and monitoring of data for mineral exploration; environmental sensing over large areas with multiple sensors; financial data, such as that used in financial service institutions integrating numerous financial sources; and electronic commerce and web applications where the focus is on user data [19].

The above recommender systems are usually based on information from their own domains, meaning that the data for analysis is limited. For instance, assume that a customer has purchased diapers in a shopping site. Accordingly, the customer would have an interest in baby products, thus the recommender system of the shopping site would put baby products into his/her recommendation list. However, with the exception of baby products, no items can be added to the list. The reason is that the recommender system has no notion of his/her preferences in other situations, resulting

in a lack of information. For instance, if the customer has accessed web pages with titles of chocolates oranges in other domains, then we can know that the customer has a tendency to have interest in chocolates and fruits. However, the previous domain, the shopping site, would have no knowledge about that, thus its recommendation system cannot put chocolates and fruits in the recommendation list. Using information from other domains may solve this problem. A user usually has multiple accounts in different domains. In addition to an account on the shopping site, for example, we may assume that the above-mentioned user has various accounts in other domains. It is useful for recommender systems if they can obtain information from such varied domains. However, the shopping site would typically have no knowledge of individual users' account IDs in other domains. As a result, though there may be abundant information on behaviorally related users in other domains, it is impossible for recommender systems to utilize such information. Therefore, matching the behaviors of the behaviorally related users in different domains would offer a solution for this problem.

In this paper, we consider the following situation. A user has accounts in two domains. One of the domains is an e-commerce site. In this domain, there are descriptions (mainly text) of products and purchase histories of users. The other domain is an Ad-Network service. The Ad-Network service puts advertisement on various webpages, and thus can record the URLs that users have accessed. Our task is to match the behaviors of the behaviorally related users in these two domains, by only using the above information.

The proposed method comprises the following steps. First, we create vectors for the accounts, based on the text information of the users. Second, since vectors from different domains are difficult to compare, we map these vectors into a shared space. After the vectors are mapped, we match the accounts based on similarity.

In our experiments, we evaluate canonical correlation analysis compared with linear regression. The results show that the former achieves better precision than the latter. Further experiments show that cosine similarity is more proper than Euclidean distance.

The rest of the paper is organized as follows. In Section 2, we introduce some related work. In Section 3, we describe our problem and situation. In Sections 4 and 5, we described our proposed method. In Section 6, we summarize our main experiment and results.

2 RELATED WORK

Since cross-domain matching uses information from other domains, it can be regarded as a problem of domain adaptation [4]. Domain adaptation is a problem utilizing machine learning and transfer learning. This problem occurs when we seek to learn from a source data distribution and a model with good performance on a different but related target data distribution. A representative task is picture recognition. In this task, the source distribution might be a model of person recognition, which must be adapted into a target distribution of animal recognition.

There are several types of domain adaptation. They differ in the information given in the target distribution. In unsupervised domain adaptation, the learning sample contains a set of labeled source examples, while a set of unlabeled source examples and an unlabeled set of target examples are given in learning and prediction [18]. In supervised domain adaptation, all the samples are labeled [8]. Since labeled examples are difficult to obtain, it is not realistic in most situations. To solve this problem, semi-supervised domain adaptation is utilized. In this situation, a small set of labeled examples in the target domain are used [7].

In this work, we use canonical correlation analysis to map vectors into a shared space, while the above works use Hierarchical Bayesian Model, and do not consider vector similarity.

3 PROBLEM STATEMENT

In this paper, we consider the following problem. There are two domains. One is an e-commerce site, referred to as Domain A in this paper. The other is an Ad-Network service, called as Domain B in this paper.

For Domain A, many items belonging to several categories are for sale on a shopping site. For every item, several kinds of information is given, such as price, text descriptions, category, selling duration, etc. Meanwhile, user information is also given, including the user's profile, such as gender, email address, username, etc. The user purchase histories are also available, such as purchased item, time, amount, price, etc. For Domain B, when opening a URL including an advertisement provided by this Ad-Network service, the advertisement will be displayed on its web page. When users access this advertisement, information on them is recorded, such as their IP address, access time, accessed URLs, etc.

We assume that users have accounts in both two domains. Each user has only one account in Domain A, but may have more than one account in Domain B. This is because the accounts in Domain B automatically update at a certain period of time. That is, the user automatically gets an updated account at the certain period of time. The problem in this paper is to match behaviors belonging to the behaviorally related users, by only using the information listed above. Additionally, to solve the problem, we only utilize the purchase histories in Domain A and the URL access log in Domain B. The structure of the domains is shown in Figure 1.

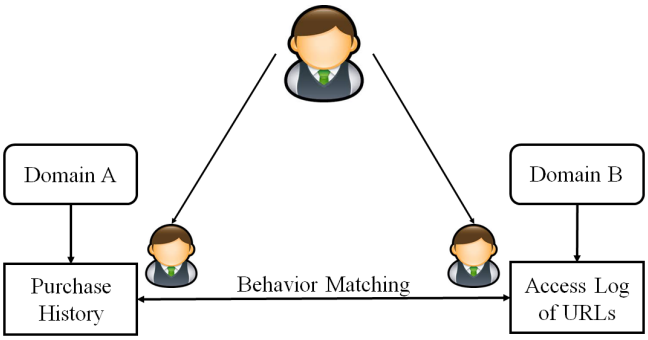

Figure 1: Domain structure

4 VECTOR GENERATION

For behavior matching, comparing the similarity of vectors of users between different domains may be the most practical method. The most meaningful information respecting the purchase histories and

URL access log is the articles involved. For example, the articles in the purchase histories may be item descriptions, while those in the URLs may be news articles. After we obtain articles, we use the obtained articles to generate user features.

4.1 Obtainment of articles

The URL access logs and purchase histories include articles, which are mainly obtained by crawling with special tags and in specific parts.

In Domain A, we have information on the purchase histories of users. We assume that items purchased by a user can represent the features of the user. Therefore, we use description articles of each item that the user has purchased, to obtain the users' features. For items in Domain A, the item title, promotion slogan and detailed description are combined as their articles.

In Domain B, we have the user access logs, in which the accessed URLs are recorded. For most URLs, web pages include articles, which could represent user preferences, and thus can be utilized to obtain their features. In HTML source files, since articles are usually included in tag *a* and/or tag *p* [16], we do crawling based on tag *a*, tag *p*, and the page title. In Web pages, tag *a* and tag *p* usually constitute the body, and thus include the main articles. The article of a Web page combines these three components.

4.2 Converting articles to vectors

After we have obtained articles from these two domains, we convert the articles into vectors. We utilize MeCab [1] for text segmentation, and create vectors using Doc2Vec [13]. The two procedures are shown below.

MeCab is an open-source text segmentation library for use with text in the Japanese language. It can analyze and segment a sentence into its parts of speech. Doc2Vec requires a list of words to make vectors, for which a long article is therefore unsuitable. Thus, it is necessary to divide articles into several words using MeCab.

The goal of Doc2Vec is to create a numerical representation of a document, regardless of its length. It learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. Based on Word2Vec, it represents each document by a dense vector which is trained to predict words in the document [14].

4.3 Vectors of user accounts

After we have converted the articles into vectors, we create vectors for each user's account based on these article vectors. In Domain A, many users have purchased more than one item. In Domain B, all the users have accessed more than one URL. Thus, every account, both Domain A and Domain B, can have more than one article vector. Therefore, it is necessary to generate one account vector from several article vectors.

We assume that account r has a set of s articles, $\boldsymbol{A}_r = \{a_{r1}, a_{r2}, a_{r3}, ..., a_{rs}\}$, where a_{ri} is the i-th article of account r. Additionally, by converting the articles into vectors, the set of articles $\boldsymbol{A}_r$ has a corresponding set of converted vectors: $\boldsymbol{T}_r = \{\boldsymbol{t}_{r1}, \boldsymbol{t}_{r2}, \boldsymbol{t}_{r3}, ..., \boldsymbol{t}_{rs}\}$, where $\boldsymbol{t}_{ri}$ is the vector of the i-th article of account r. Therefore, $\boldsymbol{t}_r$, the vector of account r, is calculated as:

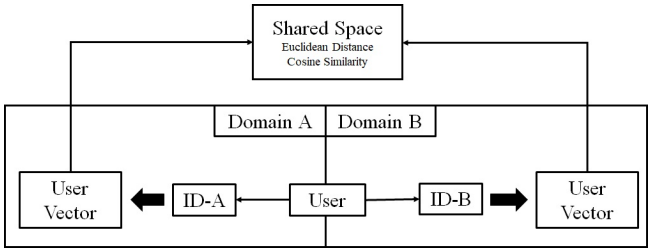

Figure 2: Matching System: the user has an account in both Domain A and Domain B. After obtaining vectors of the account, we map these into a shared space for matching.

$$\boldsymbol{t}_r = \frac{1}{s}\sum_{k=1}^{s} \boldsymbol{t}_{rk}. \tag{1}$$

That is, for a certain account, we use the average of all article vectors as its account vector.

5 BEHAVIOR MATCHING

After we have obtained account vectors of the two different domains, we match the accounts of the given user, based on these account vectors. A common solution is to match based on the similarity of vectors. However, since these vectors are generated from different domains, they would have different feature distributions. Therefore, it is impossible to employ vector similarity directly. Instead, it is necessary to map these vectors in a shared space. In this section, we first map account vectors into a shared space, and then calculate the similarity of the mapped vectors. Our matching system is shown in Figure 2.

A common solution is to calculate a conversion matrix for the two domains using linear regression. In this way, it is possible to use straight lines to fit the sample points [20]. We thus assume an n-dimensional vector X and an m-dimensional vector Y, where $\mathrm{X} = [x_1, x_2, x_3, ..., x_n]^\top$ and $\mathrm{Y} = [y_1, y_2, y_3, ..., y_m]^\top$. Both X and Y have several features, thus we want to analyze the relationship between X and Y. If $\mathrm{X} \in \mathbb{R}^n$ and $\mathrm{Y} \in \mathbb{R}^m$, we can establish the equation X = WY as:

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \dots & \dots & \dots & \dots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}, \tag{2}$$

where W is a conversion matrix for linear regression:

$$\mathrm{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \dots & \dots & \dots & \dots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}. \tag{3}$$

Since $y_i = w_i^\top x$, it is necessary to train m times to get w_i of amount m, where y_i is the i-th feature of y and w_i is the i-th row in W. However, there is a problem in that each feature of Y is correlated to every feature of X, but has no correlation to other features of Y itself. To solve this problem, we utilize canonical

correlation analysis (CCA), which enables us to calculate multiple objective variables simultaneously.

5.1 Canonical correlation analysis

Canonical correlation analysis can be used as to determine the correlation between two pairs of vectors. If we have two vectors, $X = (x_1, x_2, x_3, ..., x_n)$ and $Y = (y_1, y_2, y_3, ..., y_m)$, canonical correlation analysis will use linear calculations to determine the combinations of X and Y which have maximum correlation with each other, and give a solution for the maximum correlation [9].

We here assume that X and Y are account vectors in different domains of the same user, and canonical correlation analysis then seeks vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ for the following situation [11]:

$$\max \quad \rho = \mathrm{corr}(\boldsymbol{a}^\top X, \boldsymbol{b}^\top Y). \tag{4}$$

By maximizing the correlation between $\boldsymbol{a}^\top X$ and $\boldsymbol{b}^\top Y$, we can convert the paired vectors X and Y into similar vectors X' and Y', as a result of mapping them into a shared space, where $X' = \boldsymbol{a}^\top X$ and $Y' = \boldsymbol{b}^\top Y$ [5]. In this way, we can calculate the similarity of vectors in the two different domains.

In canonical correlation analysis, the converted vectors X' and Y' usually have the same number of dimensions, which can be selected according to our needs. The greater the number of dimensions, the more features the vectors will have. However, greater numbers of dimensions may require longer time for training models [3].

5.2 Similarity calculation

After obtaining converted vectors, we can match vectors based on similarity. Typically, when calculating the similarity of vectors, Euclidean distance and cosine similarity are used. In mathematics, Euclidean distance is the ordinary straight-line distance between two points in Euclidean space. The Euclidean distance between X and Y is calculated as:

$$\text{distance} = d(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{5}$$

Cosine similarity is a measure of the similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. For vectors X and Y, the cosine similarity is calculated as:

$$\text{similarity} = \cos(\theta) = \frac{X \cdot Y}{||X||||Y||}, \tag{6}$$

where ||X|| and ||Y|| are the magnitude of vectors X and Y.

6 EXPERIMENT AND RESULTS

Our experiment is based on real-world databases of two different domains. The user has accounts in both Domain A and Domain B. In Domain A, the user has only one account, and in Domain B, at least one account.

6.1 Data set

We utilized real data[1] collected from May 11$^{\text{th}}$ 2017 to March 31$^{\text{th}}$ 2018. The dataset includes 82 thousand users. Therefore, in Domain A, there are 82 thousand accounts, for which there is at least one purchase record. Meanwhile, in Domain B, since one user could have more than one account, there are 92 thousand accounts, for each of which there are several URL access records.

For each of the URLs accessed in Domain B, we calculate the access time. Since there are more than 11 million accessed URLs, it is almost impossible to do crawling for all the articles. Therefore, we only crawled articles of URLs with more than 1,000 accesses.

After we obtained all the needed articles from the two domains, we created article vectors using MeCab and Doc2Vec, as outlined above. Vectors were created in 256-dimension.

[1]Due to the policy of the data providers, we cannot reveal the data sources.

6.2 Evaluation methodology

For the evaluation metric of our experiment, we used Precision@R, which involves matching performance based on using the precision with which the true target instance is included in a set of R proportion of candidate instances, $S(R)$, found by each method. More formally, the precision is given by:

$$\text{Precision@}R = \frac{1}{N_{te}} \sum_{i=1}^{N_{te}} \delta(t_i \in S_i(R)), \tag{7}$$

where N_{te} is the number of test instances in the target domain, t_i is the i-th true target instance, $S_i(R)$ is the R candidate instances of the i-th source instance and $\delta(\cdot)$ is the binary function that returns 1 if the argument is true, and 0 otherwise. Here, R means ratio in the dataset, but not an integer number.

In our experiments, we used 70% of the data to train our model and the remainder (30%) for testing. For each condition, we performed the experiment five times with different allocations of training and testing data. After all the experiments under the same condition were completed, we averaged the results of each experiment as the final result.

6.3 Similarity calculation

Since we have two options (cosine similarity and Euclidean distance) to calculate similarity, we need to determine which is suitable for our experiment. We tested under many conditions, and determined that cosine similarity is more suitable. An example is shown in Figure 3, where the dimension of mapped vectors is nine.

6.4 Dimension of mapped vectors

When training the model using canonical correlation analysis, we must set the parameter of the dimension of mapped vectors. A larger dimension means more user features, but also has negative effects, such as more calculation consumption and feature noise.

We tested using parameters of dimension of mapped vectors from 1 to 15. All the results of the canonical correlation analysis are shown in Tables 1 and 2, using both Euclidean distance and cosine similarity. The results in Tables 1 and 2 on dimension one are the same, since the calculation of cosine similarity and Euclidean distance is the same on dimension one. Bold values are the best performance given a specific dimension of mapped vectors. When the dimension of mapped vectors is near nine, the performance has a peak value. Since we selected cosine similarity, we determined the best dimension using cosine similarity.

Table 1: Result on Euclidean distance of number of dimension from 1 to 15

R	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.05	0.075	0.088	0.094	0.101	0.104	0.108	0.111	0.112	0.113	0.113	0.113	0.113	0.113	0.113	**0.114**
0.1	0.142	0.165	0.177	0.187	0.193	0.198	0.201	0.201	0.201	**0.202**	0.202	0.201	0.201	0.201	0.200
0.15	0.205	0.236	0.250	0.261	0.267	0.272	0.274	0.274	0.274	**0.274**	0.273	0.272	0.271	0.271	0.270
0.2	0.263	0.300	0.314	0.325	0.331	0.334	**0.336**	0.335	0.334	0.333	0.332	0.332	0.331	0.330	0.329
0.25	0.316	0.359	0.373	0.383	0.388	**0.391**	0.391	0.391	0.389	0.388	0.387	0.386	0.385	0.383	0.382
0.3	0.369	0.416	0.429	0.438	0.443	**0.444**	0.443	0.442	0.442	0.440	0.439	0.438	0.436	0.435	0.433
0.35	0.421	0.470	0.483	0.490	0.494	**0.494**	0.494	0.492	0.491	0.490	0.489	0.488	0.487	0.485	0.484
0.4	0.473	0.522	0.534	0.540	0.543	**0.543**	0.542	0.541	0.540	0.538	0.536	0.535	0.535	0.533	0.532
0.45	0.523	0.572	0.584	0.589	0.591	**0.591**	0.590	0.589	0.587	0.585	0.584	0.583	0.582	0.580	0.578
0.5	0.572	0.622	0.633	0.636	**0.637**	0.637	0.636	0.634	0.633	0.631	0.630	0.629	0.628	0.626	0.625

Table 2: Result on cosine similarity of number of dimension from 1 to 15

R	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.05	0.075	0.102	0.120	0.130	0.135	0.143	0.145	0.148	0.148	0.149	**0.150**	0.150	0.149	0.149	0.150
0.1	0.142	0.207	0.229	0.240	0.247	0.255	0.258	0.259	0.261	**0.261**	0.261	0.261	0.261	0.261	0.261
0.15	0.205	0.290	0.314	0.325	0.332	0.339	0.343	0.345	**0.347**	0.346	0.346	0.346	0.346	0.345	0.345
0.2	0.263	0.356	0.381	0.392	0.399	0.406	0.410	0.412	**0.413**	0.413	0.412	0.412	0.412	0.412	0.410
0.25	0.316	0.415	0.440	0.452	0.459	0.466	0.470	0.472	**0.472**	0.472	0.472	0.472	0.471	0.471	0.471
0.3	0.369	0.471	0.496	0.509	0.516	0.522	0.526	0.527	**0.528**	0.528	0.527	0.526	0.527	0.526	0.525
0.35	0.421	0.524	0.550	0.562	0.568	0.574	0.577	0.578	**0.580**	0.579	0.579	0.579	0.578	0.577	0.576
0.4	0.473	0.575	0.600	0.610	0.617	0.622	0.625	0.626	0.626	0.626	**0.626**	0.626	0.625	0.624	0.624
0.45	0.523	0.624	0.646	0.657	0.661	0.667	0.670	0.671	0.671	0.671	**0.671**	0.670	0.670	0.670	0.669
0.5	0.572	0.669	0.690	0.700	0.704	0.710	0.712	0.714	**0.714**	0.713	0.712	0.712	0.712	0.712	0.711

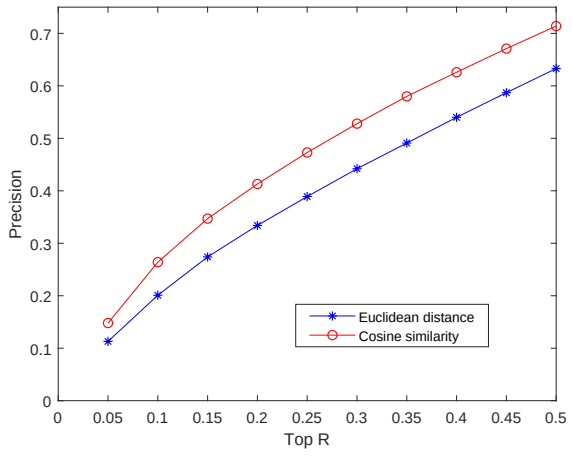

Figure 3: Similarity calculation: for each top R proportion of candidates, CCA's precision with Euclidean distance and cosine similarity.

6.5 Comparison with linear regression

Linear regression is a baseline method in our problem, and is also employed, for comparison with the proposed. Our method used nine dimensions for the account vectors. Both two experiments used cosine similarity. The results are shown in Figure 4. It is obvious that our method outperforms the baseline method.

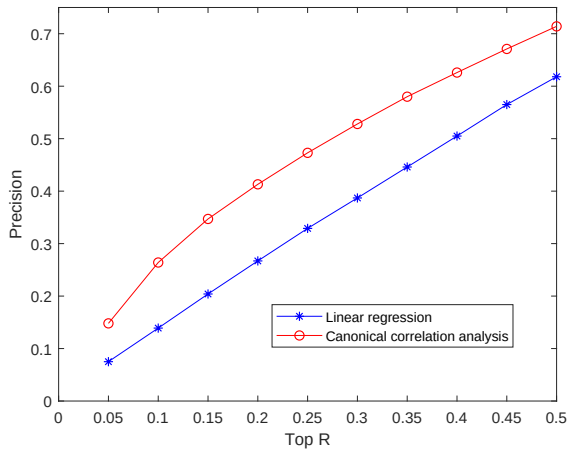

Figure 4: Comparison with linear regression: for each top R proportion of candidates, CCA's precision compared with linear regression.

7 CONCLUSION

In this study, we developed a system to match behaviors in different domains, for the same users. In order to use vector similarity to match the behaviors, we generate vectors based on articles related to users, by using MeCab and Doc2Vec. When matching behaviors, we have found that canonical correlation analysis and cosine similarity show better performance results than linear regression. Actually, our result is a preliminary and the precision is not high enough for practical applications. Therefore, as for future work, we plan to utilize some non-linear methods, such as deep canonical correlation analysis [2] and kernel canonical correlation analysis [12].

ACKNOWLEDGMENT

This research is partially supported by JST CREST Grant Number J181401085.

REFERENCES

[1] 2013. Mecab. http://taku910.github.io/mecab/.
[2] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *Proc. 2013 International Conference on Machine Learning*. 1247–1255.
[3] Francis R Bach and Michael I Jordan. Technical report, University of California, Berkeley, 2005. A probabilistic interpretation of canonical correlation analysis. (Technical report, University of California, Berkeley, 2005).
[4] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Vaughan. 2010. A theory of learning from different domains. *Machine Learning* 79, 151–175.
[5] Patricia Cohen, Stephen G West, and Leona S Aiken. 2014. *Applied multiple regression/correlation analysis for the behavioral sciences*.
[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proc. 10th ACM Conference on Recommender Systems*. 191–198.
[7] Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain adaptation. In *Proc. the 2010 Workshop on Domain Adaptation for Natural Language Processing*. 53–59.
[8] Daniel Garcia-Romero and Alan McCree. 2014. Supervised domain adaptation for i-vector based speaker recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4047–4051.
[9] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation* 16, 12 (2004), 2639–2664.
[10] Wenyi Huang, Zhaohui Wu, Chen Liang, Prasenjit Mitra, and C. Lee Giles. 2015. A Neural Probabilistic Model for Context Based Citation Recommendation. In *Proc. 29th AAAI Conference on Artificial Intelligence*. 2404–2410.
[11] Thomas R Knapp. 1978. Canonical correlation analysis: A general parametric significance-testing system. *Psychological Bulletin* 85, 2 (1978), 410.
[12] Pei Ling Lai and Colin Fyfe. 2000. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems* 10, 05 (2000), 365–377.
[13] Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368* (2016).
[14] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. 2014 International Conference on Machine Learning*. 1188–1196.
[15] Dongsheng Li, Chao Chen, Qin Lv, Hansu Gu, Tun Lu, Li Shang, Ning Gu, and Stephen M. Chu. 2018. AdaError: An Adaptive Learning Rate Method for Matrix Approximation-based Collaborative Filtering. In *Proc. 2018 World Wide Web Conference*. 741–751.
[16] Gengxin Miao, Junichi Tatemura, Wang-Pin Hsiung, Arsany Sawires, and Louise E Moser. 2009. Extracting data records from the web using tag path clustering. In *Proc. the 18th International Conference on World Wide Web*. 981–990.
[17] Shakiba Rahimiaghdam, Pinar Karagoz, and Alev Mutlu. 2016. Personalized Time-aware Outdoor Activity Recommendation System. In *Proc. 31st Annual ACM Symposium on Applied Computing*. 1121–1126.
[18] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric Tri-training for Unsupervised Domain Adaptation. In *Proc. 2017 International Conference on Machine Learning*.
[19] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proc. 10th International Conference on World Wide Web*. 285–295.
[20] George AF Seber and Alan J Lee. 2012. *Linear regression analysis*. Vol. 329.