Content-Based Cross-Domain Recommendations Using Segmented Models

Shaghayegh Sahebi
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA
shs106@pitt.edu

Trevor Walker
LinkedIn
Mountain View, CA
twalker@linkedin.com

ABSTRACT

Cross-Domain Recommendation is a new field of study in the area of recommender systems. The goal of this type of recommender systems is to use information from other source domains to provide recommendations in target domains. In this work, we provide a generic framework for content-based cross-domain recommendations that can be used with various classifiers. In this framework, we propose an efficient method of feature augmentation to implement adaptation of domains. Instead of defining the notion of domain based on item descriptions, we introduce user-based domains. We define *meta-data features* as a set of features to characterize the fields that domains come from and introduce *indicator features* to segment users into different domains based on values of the *meta-data features*. We study an implementation of our framework based on logistic regression and perform experiments on a dataset from LinkedIn to perform job recommendations. Our results show promising performance in certain domains of the data.

1. INTRODUCTION

Recommender systems can help users to address the information overload problem by providing related items considering user's interests. So far, most of the recommender systems were focused on specific domains, recommending one type of item (such as books) to all categories of users. Recent research introduced cross-domain recommender systems that aim to take advantage of shared information among various domains [10]. In cross-domain recommendation, the goal is to use various source domain information to recommend items in target domains. Previous studies on collaborative filtering cross-domain recommender systems has shown an improvement of accuracy of recommendations, especially in the cold-start case [14]. Most of the work on cross-domain recommender systems and the definition of domains in them has been based on cross-domain collaborative filtering methods and ignored the domains that can be defined based on user specifications. Li [10] has categorized the domains in cross-domain recommendation into system, data, and temporal domains. These domains are related to, respectively, different datasets that a recommender system is built upon, various representation of user preferences (explicit or implicit), and various time points in which the data is gathered. Although this is a good classification of possible domains in recommender systems, it focuses on the type or domains that are defined based on items. In other words, usually the notion of domain is selected as a constant characteristic of items, systems, etc. For example, type of items (e.g. books, movies, etc.)[14], genre of items (e.g. for movies) [1], or indicators of various systems that the data is gathered from [6] are some of features that have been used as domain indicators. Joshi et. al. [5] have chosen domains based on meta-data features. These meta-data features can be selected from all unique subsets of features by experimenting (e.g. selecting the best performing features on a validation set) which is a time-consuming task. Choosing the proper domain indicator among features is still an open field of research. In this paper, we propose a framework for performing content-based cross-domain recommendation. We propose that in content-based and hybrid recommender systems, the domain notion can be extended to the type or domain of users. Here, the definition of domain can be determined based on the recommendation task and users' information, such as users' demographic data. For example, in a movie recommendation task, age of user can be an effective factor in deciding which movies match the best for her. As another example, in job recommendation, we expect the model parameters to be different for different job functions of users. For a designer, it is important to have matching skills with the job description, while for a network engineer, his certificates might have more importance.

We choose job recommender application in our experiment in this paper, although it is applicable to other recommender system domains. Having many different jobs listed online in various industries with different job descriptions, it is essential for people to find the job that best matches their abilities and specifications. Searching for the right job is a time-consuming task for a user. It needs spending a lot of effort on defining the criterion the user is looking for. Job recommender systems can address this problem by actively finding good job matches for the user, utilizing her profile information, search keywords, etc. Based on previous results in the job recommendation literature [8, 9] and our field experience, we believe that job recommendations can benefit from cross-domain information.

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

In this paper, our proposed framework can be utilized by various algorithms defined on any notion of domain from data attributes. Our work also differs from the existing literature in defining domains on the user profile side instead of item side. It can, of course, be used for domains defined on item-set features. We experiment with LinkedIn data for job recommendations. Our experiments lead to promising results for content-based cross-domain recommendations based on user job functions.

In Section 2, we briefly discuss related literature. In Section 3, we introduce our approach to content-based cross-domain recommendation. We present our dataset and experiment setup in Section 4 and we discuss the results in section 5.

2. RELATED WORK

2.1 Segmented Regression Model

Segmented regression or piece-wise regression [13] can be used as a classification method in which data features are partitioned into intervals using some breakpoints. In the final model, a separate model will fit each of the segments. This model is useful for approximating higher-degree models with multiple lower-degree models in smaller ranges. In this paper, we adopt this method to our problem of cross-domain recommendation.

2.2 Feature Augmentation in Domain Adaptation

Feature augmentation was introduced by Daumé [3] in domain-adaptation literature. In his paper, Daumé considers a source domain and a target domain separately. He augments each of these domains individually by copying the feature space three times: once copying all features for the general version, and once for each of the source and target versions. Eventually, the augmented source data will contain only general and source-specific versions and the augmented target data contains general and target-specific versions. After Daumé's paper, this method have been used in the domain-adaptation field, especially in Natural Language Processing (NLP)[2, 4, 5].

Our approach improves Daumé's model in the possibility of having multiple meta-data features for defining the domains (instead of having one dimension of source and target domains). In addition, each of the meta-data features can have multiple values and define multi-dimensional domains. Moreover, we can have separate sets of common (overlapping) and uncommon features in the main-effect and domain-specific models. Our model is extensible to incorporating cross-products of domain indicator features.

2.3 Job Recommendation

Despite of the importance of job recommender systems, there have not been many research on this subject. Rafter et al. [12] introduced CASPER, an intelligent online recruitment service. Keim [7] provided a multilayer framework to support the matching of individuals for recruitment processes. In [11] Hutterer used hybrid user profiling to enhance the job recommendation results. He incorporated explicit and implicit feedback of user in the user profile. Lee and Brusilovsky [8, 9] implemented and experimented with Proactive, which has multiple interfaces for various types of users. They showed that different users use various information resources to look for the perfect job.

3. OUR FRAMEWORK: SEGMENTED MODEL FOR CROSS-DOMAIN RECOMMENDATION

In cross-domain recommendation, we aim to build a model that can be general and flexible enough, to transfer the information in multiple related domains, and specific enough, to capture particular aspects of each individual domain. This means that we expect a trade off between the bias and the variance in our model. We want all models to be close to each other in particular dimensions (having less variance) and we want them to be biased towards each domain's specific distribution. For example, if we think of various job functions as different domains in job recommendation, we expect the user profile to have a good match to the job description in all domains (common feature of the domains). Also, we expect the skills feature to be more important for an artist than a university professor (domain-specific feature). If we consider one main model for all of the data present in various domains, we are going to have no variance in the model, but we will lose the bias we are looking for. On the other hand, if we treat each domain with a separate model, we will achieve the bias each domain is introducing, but we will have too much variance in the achieved models. In other words, we will lose the ability to transfer common information among different models. Our framework consists of two parts: the *main-effect model*, and *domain-specific models*. The main-effect model is to model the shared statistics among all domains. We have one domain-specific model per domain to capture the domain-specific characteristics. A general formulation of model can be seen in Equation 1.

$$\text{Final-Model} = \text{Main-Effect Model} + \sum_{i \in Domains} \text{Domain-Specific Model}_i \quad (1)$$

3.1 Cross-Domain Augmentation and Segmentation

As said before, we characterize the fields that the domains come from by some features called meta-data features. Each dataset has a set of features, such as user-related features, item-related features, and features that represent similarities between users and items, that we call them "base features". Meta-data features are a subset of base features, which specify aspects that we want to define the domains based on. Each domain is constructed based on the values of these meta-data features and their combinations. For example, if we want to recommend movies to users, base features are user features, such as age, education, language, etc, item features, such as movie genre, actors, director, etc, or the relationship between users and items, such as the similarity between each movie genre and genres that a user likes. We can choose some of these base features as meta-data features to define the notion of domain based on them. For example, we can choose the genre base feature as the meta-data feature. In this case, the defined domains will be action movies, drama movies, action-drama movies, etc. If we choose two meta-data features, the domains can be a combination of val-

ues for those meta-data movies. For example, if we choose genre of movie and age of user as meta-data features, the domains will be like: action-middle-age, drama-young, etc. In the case of user-based domains for job recommendation, we can choose some base features of users, such as job function, or job seniority of users, as meta-data features. For example, if we want to define the domains based on job function, people who have IT job function form one domain and people who have medical job function form another domain.

3.1.1 Augmentation

Each of the domain-specific models in Equation 1 works on one domain's data. As a result, we should split the dataset for each domain and provide each domain-specific model with the section of the dataset related to that domain. To address this splitting, we expand on the idea of segmented regression model. We propose to augment the feature space based on the domain definitions and copy each datapoint into the related domain's sub-space. The main space is then used for the *main-effect model* and each copy of the space is used for the related *domain-specific model*.

However, this augmentation has a problem: if we have k different domains (e.g. k different job functions), we need to partition the data space into 2^k separate segments (copies) to capture all of different settings of the dataset which takes too much space. Each one of these copies is for each subset of the possible combination of meta-data feature values (domains). For example, if we want to partition users based on the job function represented in their resume, and we have three different possible job functions (e.g. operations, education, and sales), we will have to partition the data into $2^3 = 8$ segments: people for whom the job function is in operations, the ones who have sales function, the ones who have education job function, the ones who have sales and operation functions, and etc. In addition to the space problem, segmenting the data into combinations of domains may lead to very sparse copies of the original dataset. Additionally, looking at all various combinations of the domains and their interactions might not be necessary for our purpose.

3.1.2 Indicator Features for Segmentation

To alleviate the problem indicated in Section 3.1.1, we expand on the idea of Segmented Regression Model and introduce *indicator features* for each domain. These features allow us to augment the feature space in a polynomial order, while being able to keep the main effect model and control the granularity of combinations among different domains. Suppose that each meta-data feature can take k different values and segment our data into k domains and suppose that we are choosing only one meta-data feature. For each of the k possible values in each of the domains, we define a binary *indicator feature*, representing if a data point falls into that specific domain or not. As a result, we will end up with k binary indicator features for the selected meta-data feature. Eventually, we will augment the feature space based on the indicator features in the following way: We keep the original feature space for the common features used in the main effect model. For each of the features falling into the domain-specific models, we augment the feature space by copying it k times for each of the meta-data feature values. Consequently, we have a polynomial expansion of space. Note that we do not consider combinations of meta-data feature values yet.

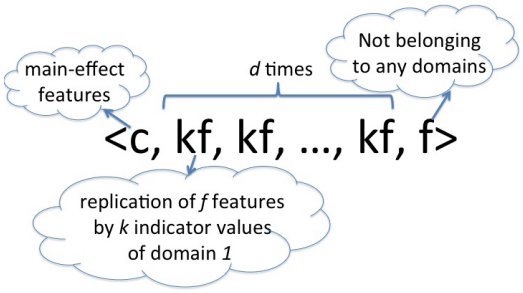

Figure 1: Augmented Feature Space with c Common Features in the Main Effect Model, f Features in Each of the d Domains, that Are Represented by Indicator Features with k Possible Values

Now, if we have d meta-data features to choose the domains from, each of which can take k values, and in each of the domain-specific models, f of the base-features exist, we should replicate this f dimensional space for $dk+1$ times. This number is polynomial in d (number of meta-data features) and k (number of values for each meta-data feature). While if we have not used the indicator features in segmented model, the f-dimensional feature space should have been replicated for d^k times. Considering having c common features for the *Main Effect Model*, we can represent the new feature space by Figure 1.

3.1.3 Challenges and Advantages

An advantage of this framework is its extensibility to higher order cross-products of values between and within domains. For example, if we want to consider the effect of interaction between two domains, we can extend the model to consider an indicator feature, representing cross-products of feature values in the domains. E.g. if we want to take into account the combination of every two feature values within each of the domains, we will end up with a space that has $O(c+f(dk+1)+f(dk^2+1))$ dimensions or is expanded for $dk+dk^2+2$ times. This gives us the ability to control the dimensionality of feature space while avoiding the sparsity in each of the segments.

Another challenge is choosing the features that should be in the main-effect model and the ones that should remain as domain-specific features. In other words, which features should be responsible for transferring the information among domains (controlling the variance) and which ones should provide the domain-specific bias? In our approach, the model can learn which features to use in the main effect model and which features to use in each of the domains using regularization. Since regularization imposes coefficient values to be as close to zero as possible, the less important coefficients of the model will have very small values and are removed from the model. While the model can choose between these sets of features, we can also initialize the main effect and

domain-specific features by the expert's domain knowledge.

Besides, in some of the cross-domain recommender problems, each domain has its own subset of a general feature set, which might differ in the size or type with other domains' feature sets. We expect our cross-domain solution to consider this problem and be extensible to domains with heterogeneous number and types of features.

3.2 Implementation Using Logistic Regression

Although the approach we presented here can be used in various classification algorithms, we used a straightforward classification algorithm to implement the model.

Suppose that f_{c_i} is the i^{th} common feature among the domains; $\mathcal{M}$ is the set of meta-data features; $\mathcal{V}_j$ is the set of values (domains) for the j^{th} meta-data feature; I_{ij} is the binary indicator feature for the domain i of meta-data feature j; f_{ijk} represents the k^{th} base feature specific to the i^{th} domain of meta-data feature j; and p is the probability of the model's outcome. Equation 2 shows the resulting segmented regression model with indicator features. As we can see, it has a simple representation that can be implemented easily for domain adaptation.

$$logit(p) = \sum_i w_{c_i} \times f_{c_i} + \sum_{j \in \mathcal{M}} \sum_{i \in \mathcal{V}_j} I_{ij} \sum_k w_{f_{ijk}} \times f_{ijk} \quad (2)$$

Here w represent the weight (importance) of each feature in the model. In case we want to extend it to having two-way interaction effects of values of each meta-data feature (belonging to two domains), we will have:

$$\begin{aligned} logit(p) = &\sum_i w_{c_i} \times f_{c_i} + \sum_{j \in \mathcal{M}} \sum_{i \in \mathcal{V}_j} I_{ij} \sum_k w_{f_{ijk}} \times f_{ijk} \\ &\sum_{j \in \mathcal{M}} \sum_{i \in \mathcal{V}_j} \sum_{l \in \mathcal{V}_j} I_{i,l,j} \sum_k w_{f_{i,l,j,k}} \times f_{i,l,j,k} \end{aligned} \quad (3)$$

In Equation 3 $I_{i,l,j}$ represents the binary indicator feature for the datapoint belonging to both i and l domains (or having both i and l values) of meta-data feature j. To decide which features should be in the common set of features and which should be in each of the domain-specific models, we use L_2 regularization.

4. EXPERIMENTAL SETUP

4.1 Data

The dataset we are using in this study is LinkedIn's job application data. It contains records of users and job features and a binary label indicating if the user has applied for the job or not. Some of base features are calculated similarities between the job and the user. For example, we use TF.IDF to calculate the similarity of job description with user's skills and store it as a feature in the user-job record. Some other features, from which we have chosen the meta-data features, are user-specific. For example, the past and current job functions of a user, past and current industries

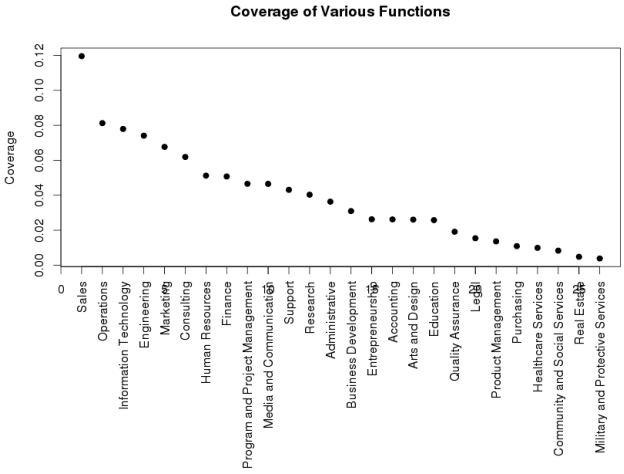

Figure 2: Coverage of User Current Job Functions in Offline Data

the user has worked at, or the geographical location of user. Meta-data features should have categorical values, so that we can extract binary indicator features from them. In case we want to use features with continuous values, we partition values into more than one category.

The offline dataset used in the following experiments consists of three million records of more than $150,000$ users. There is a one to ten ratio of positive job applications to negative job applications in the dataset. We use 100 user-job features as base features in the model.

We pick one meta-data feature (user's current job function) from user-specific features and split domains based on it. This feature is specifically related to what a user does in his/her job, e.g., a user can be an IT (job function) engineer in a bank. We choose this feature based on our experience that people working in various functions (e.g. arts and legal domain) have different requirements and definitions for a good job recommended to them.

Based on the LinkedIn data, current job functions of a user can have 26 different values. Each user can have multiple job functions at the same time. The distribution of user job functions is not uniform in the dataset: some functions are more covered in the dataset and some include less number of users. Figure 2 shows the coverage of current job functions in the data. As we can see in the picture, "Sales", "Operations", and "Information Technology" are among the most covered job functions in the data and "Community and Social Services", "Real Estate", and "Military and Protective Services" are the job functions with least coverage.

4.2 Implementation of Models for Job Recommendation

As we discussed in section 3, we can have two extremes of modeling users as our baselines: a) when there is only one main model for all of the users, and b) when there is a separate model for each segment of users and there is no shared

Figure 3: Three Experiment Settings with Different Granularities of Domain Definition

information among the models. In each of our studies, we compare our model to at least one of these two baselines.

To capture the effect of domain granularity on recommendation results, we experiment with three different settings for domains: segmenting on two domain indicator features versus all other domains (*two-vs-all*), segmenting on all indicator features of a domain (*all-indicators*), and segmenting on clusters of indicator features of a domain (*domain-clusters*). Figure 3 shows a graphical demonstration of user segmentation in each case for job functions.

For the *two-vs-all* model, we choose the two most covered values of the selected meta-data feature and define three indicator features based on that: the indicator feature that selects users with the most covered value, the indicator feature that selects users with the second most covered value, and the indicator feature for the rest of users. For example, for user's job function meta-data feature, we will have the following indicator features: I_S for users with "Sales" job function, I_O for users with "Operations" job function, and I_{Other} for all other users. The final model is presented in Equation 4. Here, fc_i shows the i^{th} common features among domains, f_{S_i}, f_{O_i}, and f_{Other_i} are features used in the "Sales", "Operations", and "Other" domains respectively, w_j is the weight for feature j, and p is the probability that the target user applies for the target recommended job. Note that since we have chosen only one meta-data feature, we do not need to present it in the model (e.g. $j \in M$ in Eq. 2). Also, note that by including I_{Other} as an indicator feature, we are capturing the effect of the interaction or cross-product of "Sales" and "Operations" domains.

$$logit(p) = \sum_i w_{c_i} \times fc_i + I_S \sum_i w_{f_{S_i}} \times f_{S_i} + I_O \sum_i w_{f_{O_i}} \times f_{O_i} + I_{Other} \sum_i w_{f_{Other_i}} \times f_{Other_i} \quad (4)$$

For the *all-indicators* model, we segment based on all values of the selected meta-data feature. If the meta-data feature can take k values, we will end up with k indicator features to segment all users into k different partitions. For user's job function meta-data feature we end up with 26 different indicator features. Our final model is shown in Eq. 5.

$$logit(p) = \sum_i w_{c_i} \times fc_i + \sum_{i \in 1..26} I_i \sum_j w_{f_{i,j}} \times f_{i,j} \quad (5)$$

Here, I_i shows the indicator feature for domain i (different values of job function); and $f_{i,j}$ represents the j^{th} base feature of domain i.

Figure 4: Clusters of User Job Function

For the last set of experiments (*domain-clusters*), we cluster the values of meta-data features into groups. Each group represents a cluster of domains. We use one indicator feature for each group. We use spectral clustering to group 26 different job functions into 8 clusters. The clusters are based on the user transition between job functions in the data. Figure 4 shows a tag-cloud representation of these clusters. Each color indicates one cluster. As we can see in the picture, functions like "Sales" and "Marketing" are clustered together and "Research" and "Education" functions fall in one cluster. We run the segmented model having one indicator per cluster. Suppose that $\mathcal{C}$ is the set of cluster indicators for a meta-data feature. The final model is shown in Equation 6.

$$logit(p) = \sum_i w_{c_i} \times fc_i + \sum_{i \in \mathcal{C}} I_i \sum_j w_{f_{i,j}} \times f_{i,j} \quad (6)$$

The final model we have in *domain-clusters* is similar to the *all-indicators* model, but domain indicators are representative of each cluster of job functions.

5. PERFORMANCE ANALYSIS

We experiment in the *two-vs-all* and *domain-clusters* settings for current job function of users as meta-data feature. We divide the data into 70% train and 30% test subsets.

We measure accuracy of the algorithms on the test set. To find the performance of algorithm in each domains of the data, we partition the test set into domains in the same way that we partitioned the training set and calculate the accuracy in each domains of the dataset. Our model is compared to at least one of the two baseline models: "one-for-all" and "independent" models. The "one-for-all" model only contains one model for all of the datapoints, ignoring the domain-specific models. The "independent" model trains a separate model for each of the domains independently. This model ignores the common information among the domains and treats them as independent from each other.

To dig deeper into the offline results, we look at the coefficient values obtained by the algorithm in each of the models.

Table 1: Accuracy of "two-vs-all" vs. "one-for-all" and "independent" models for job functions

Domain	One-for-All	Two-vs-All (Segmented)	Independent
Sales	96.28%	96.33%	95.01%
Operations	96.43%	96.49%	94.93%
Sales and Operations	96.54%	96.58%	NA
Other	96.44%	96.45%	96.44%

5.1 Two vs. All Model

As explained in Section 4, in this two-vs-all setting two most covered domains are compared to the rest of the domains. We compare our model with the two base models: "one-for-all" and "independent" models. The most covered job functions in the data are "Sales" (about 12% coverage) and "Operations" (about 8% coverage). The accuracy results for the "job function" meta-data feature are shown in Table 1. The "Sales and Operations" row represent the domain with users in both "Sales" and "Operations" domains and the "Other" row shows the users who are not in any of "Sales" or "Operations" domains. The first two rows show the users who are only in "Sales" or only in "Operations" domains respectively.

As we see in table 1, the segmented model has slightly higher accuracy than the base models. The difference is bigger for the "independent" base model, specially in the two most-covered domains. To understand how the models work differently, we look at the coefficients assigned to base variables of "two-vs-all" segmented model and "one-for-all" base model in Figure 5. In this picture, we can compare the coefficient values for these two models. The "two-vs-all" segmented model can have more than one coefficient for each variable: the variable might repeat in the main-effect part of the model or in each of the domain-specific parts of the model. To be able to compare the coefficient values, we use the average coefficient value of the main-effect and domain-specific parts of the "two-vs-all" segmented model. The red dots represent this average value and the bars around them represent the variance of these coefficients in the model. The blue dots are coefficient values in the "one-for-all" base model. As we can see in the picture, some of coefficients have a different value in the two models. For example, the similarity of user skills with job description has more importance in the "two-vs-all" segmented model. In addition, we can see that some of base variables existing in the "two-vs-all" segmented model, do not exist in the "one-for-all" base model. The reason is that these variables were removed automatically during the regularization process. For example, the similarity between user location and the location of the job is only present in the "two-vs-all" segmented model. Based on Figure 5, the "two-vs-all" model's coefficients have different variance in the main-effect and domain-specific models. Some of the coefficients vary more than the others. This can indicate that this model is able to capture the difference between different domains.

To understand if the "two-vs-all" segmented model is capturing the difference between each of the domains, we look at coefficients of variables in all four job function domains (Sales, Operations, Sales and Operations, and other) in Figure 6. The coefficient values are shown as stacked over each other in the picture. Since there are many base variables in the model and their names are not easily readable, we removed the names in figures of this section. As we can see, coefficient values for some of the variables are different for various domains. With a closer look we can find the differences in coefficient values. For example, the similarity of past positions of user to the job description is more important for the Sales domain than the Operations domain. The similarity of user skills to the job's required skills are more important for users in the Operations domain than Sales domain.

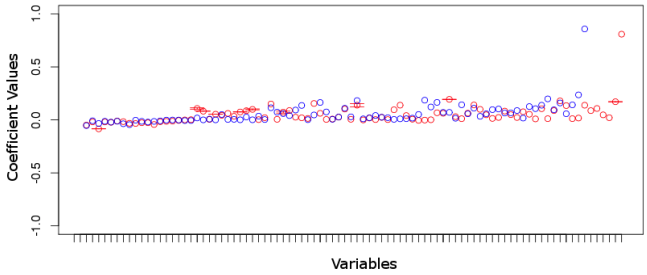

Figure 5: Coefficient Values for Two-vs-All Segmented Model (Red) Compared to the "One-for-All" Base Model (Blue)

Table 2: Accuracy of domain-clusters segmented model vs. "one-for-all" model for job functions

Domain	One-for-All	Domain Clusters (Segmented)
Cluster 1	96.57%	96.52%
Cluster 2	96.18%	96.26%
Cluster 3	96.62%	96.75%
Cluster 4	96.98%	97.09%
Cluster 5	97.58%	97.61%
Cluster 6	96.68%	96.85%
Cluster 7	96.56%	96.61%
Cluster 8	96.34%	96.36%

5.2 Domain Clusters Model

As explained in section 4, user job function meta-data features are grouped into 8 clusters. The accuracy results for the clusters in the "job function" meta-data feature is shown in Table 2. As we can see in this table, the accuracy of the models are very close to each other, for some clusters the baseline models work better than the domain-clusters segmented model and for others it is the reverse.

We compare coefficient values of the one-for-all baseline and domain-clusters model to have a more detailed insight of the results. Looking at the differences of coefficient values for each of the domains in the segmented model, we can understand how different features are more important for each of the domains. Figure 7 shows the coefficient values of domain-clusters model for the job function meta-data feature. As we can see in the picture, some of the coefficients are more important in some of the domains and less in others. For example, The similarity of user's previous searches to the job description is more important to users in cluster 2 (including sales, marketing, and similar job functions).

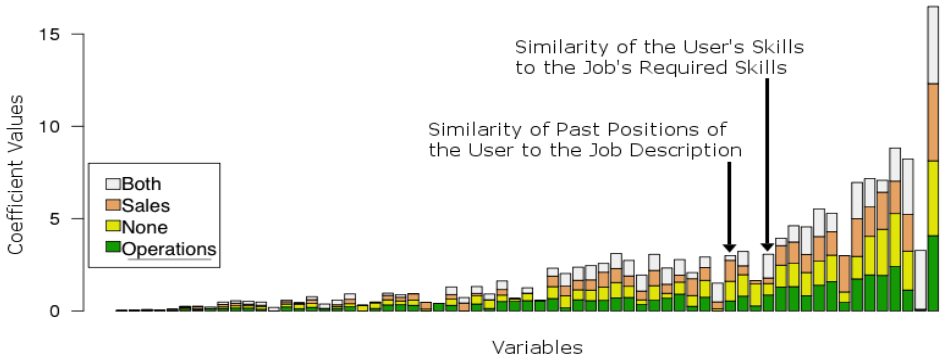

Figure 6: Coefficient Values for Different Domains in Two-vs-All Segmented Model

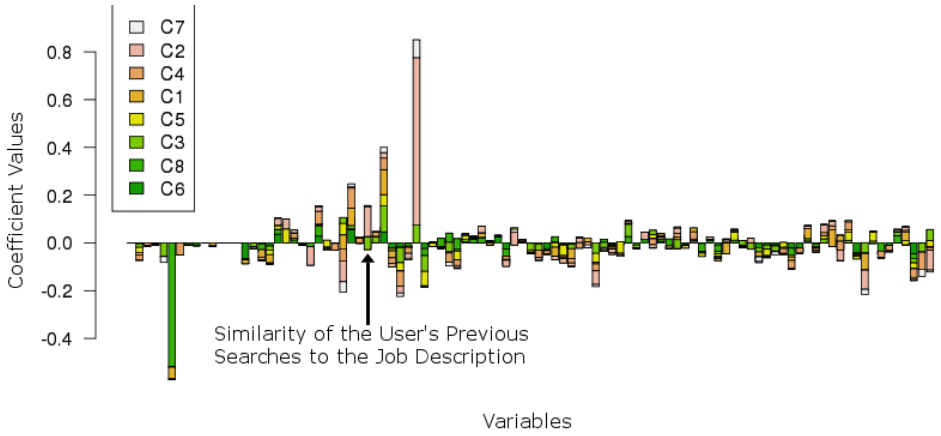

Figure 7: Coefficient Values for Domain Clusters Segmented Model for Job Functions

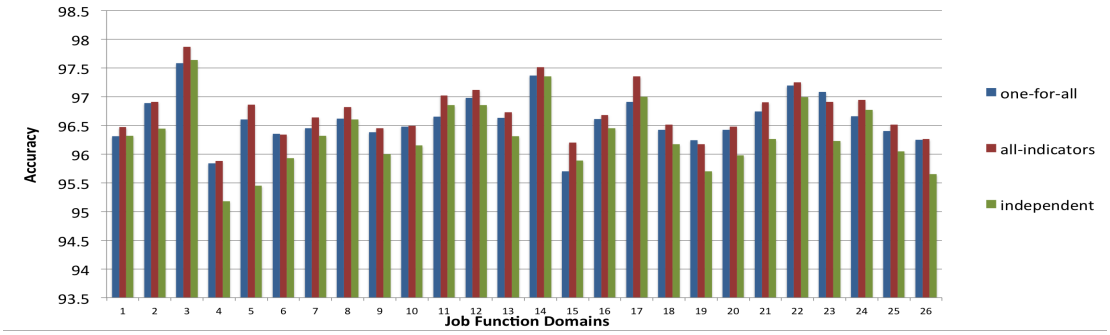

Figure 8: Accuracy of for All Indicators Segmented Model for Job Functions

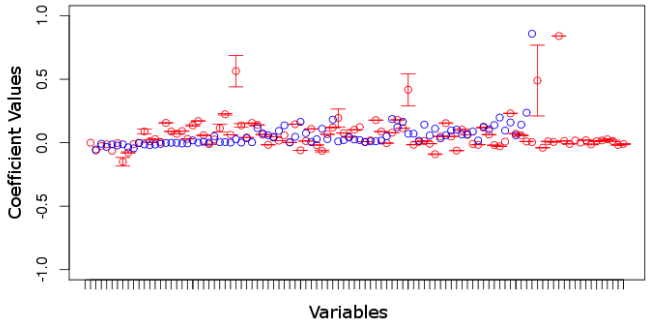

Figure 9: Coefficient Values for All-Indicators Segmented Model (Red) Compared to the "One-for-All" Base Model (Blue)

5.3 All Indicators Model

In this model, we pick all of the possible values for a meta-data feature as domain indicators: each indicator feature is representative of one of the values the meta-data feature can take. Since we choose job function as our meta-data feature, we will end up with 26 domains related to job functions, such as IT, sales, engineering, real estates, and marketing. We can see accuracy results of the all-indicators segmented model and the two one-for-all and independent baseline models in Figure 8. As we can see here, the all-indicators model is usually slightly better than the two other models. In some of the domains (such as Product Management (number 19) and Real Estate (number 23) domains) the one-for-all model has more accuracy than all-indicators model.

Comparing coefficients of one-for-all and all-indicators models in Figure 9, we can see that some of the base features have a very different coefficients in the model. Also, some of base features have a large variance in different domains of the all-indicators model. There are some base features that are removed from the one-for-all model by regularization, while they still play a role in the all-indicators model.

6. CONCLUSION AND FUTURE WORK

In this paper, we propose a framework for content-based cross-domain recommender systems. This framework is flexible enough to be implemented with various classifiers. The model in this framework can transfer common information among different domains while keeping them distinct. We define user-based domains based on users' meta-data features and implement our framework using logistic regression. The regularization in the model allows us to pick important features of each of the domains automatically, while keeping it flexible to accept expert knowledge in choosing the features. We experiment on job recommendations for LinkedIn users. Our results indicate slight improvement in recommendation accuracy in the offline setting. Furthermore, the experimental results are promising: i) different features have different coefficient values in each of the domains; and ii) coefficients are different in the cross-domain model compared to the one-for-all base model. As a result, we are hopeful that this model can be a good fit to our problem in the online experiments (A/B testing). We expect several directions for future work: implementing the framework based on various classifier algorithms, expansion of experiments of the model using different meta-data features, and experimenting on interaction of various possible domains. Automatic selection of meta-data features is another interesting direction of research.

7. REFERENCES

[1] S. Berkovsky, T. Kuflik, and F. Ricci. Mediation of user models for enhanced personalization in recommender systems. *User Modeling and User-Adapted Interaction*, 18(3):245–286, 2008.

[2] J. H. Clark, A. Lavie, and C. Dyer. One system, many domains: Open-domain statistical machine translation via feature augmentation. In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas*, 2012.

[3] H. Daumé III. Frustratingly easy domain adaptation. In *ACL*, volume 1785, page 1787, 2007.

[4] L. Duan, D. Xu, and I. Tsang. Learning with augmented features for heterogeneous domain adaptation. *arXiv preprint arXiv:1206.4660*, 2012.

[5] M. Joshi, M. Dredze, W. W. Cohen, and C. P. Rosé. What's in a domain? multi-domain learning for multi-attribute data. In *Proceedings of NAACL-HLT*, pages 685–690, 2013.

[6] M. Kaminskas and F. Ricci. Location-adapted music recommendation using tags. In *User Modeling, Adaption and Personalization*, pages 183–194. Springer, 2011.

[7] T. Keim. Extending the applicability of recommender systems: A multilayer framework for matching human resources. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 169–169, 2007.

[8] D. Lee and P. Brusilovsky. Fighting information overflow with personalized comprehensive information access: A proactive job recommender. In *Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on*, pages 21–21, 2007.

[9] D. Lee and P. Brusilovsky. Proactive: Comprehensive access to job information. *Journal of Information Processing Systems*, 8(4):721–738, December 2012.

[10] B. Li. Cross-domain collaborative filtering: A brief survey. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 1085–1086. IEEE, 2011.

[11] H. M. *Enhancing a Job Recommender with Implicit User Feedback*. PhD thesis, FakultŁt f§r Informatik der Technischen UniversitŁt Wien, 2011.

[12] R. Rafter, K. Bradley, and B. Smyth. Personalized retrieval for online recruitment services. In *In Proceedings of the 22nd Annual Colloquium on Information Retrieval*, 2000.

[13] H. Ritzema. Frequency and regression analysis (chapter 6). *Drainage principles and applications*, 16:175–224, 1994.

[14] S. Sahebi and P. Brusilovsky. Cross-domain collaborative recommendation in a cold-start context: The impact of user profile size on the quality of recommendation. In *User Modeling, Adaptation, and Personalization*, pages 289–295. Springer, 2013.