Recommendation System For HCD Connect

Kesinee Ninsuwan, Mike Phulsuksombati, Umnouy Ponsukcharoen

Abstract— **HCD Connect is a social platform by IDEO.org that aims to connect social entrepreneurs around the world. The users can post the stories about their past experiences in service work as well as their prospective project ideas on the website. When other users see the project ideas of their interest, they can interact on the platform and get connected with other users. With each story or project, a user is allowed to click like, follow a story, post comments, or share it via other social networks. Moreover, the website provides a space for the users to ask and answer questions which are not limited to specific stories or projects.**

This project aims to build a recommendation system on the stories to the users. There are four main states for this project including processing data, implementing unsupervised text classification algorithms, implementing supervised text classification algorithms, and building a score system for the stories. The unsupervised classifications that we uses includes Hierarchical Agglomerative Clustering (HAC), *k*-means clustering, and topic modeling. The results indicate that they do not discover significant topics underlying the stories. The supervised classifications used in the project includes *k*-nearest neighbors (*k*NN), support vector machine (SVM), and multi-label rank-SVM. *k*NN performs best among all algorithms. However, the precision are still low. This suggests that more data needs to be collected. Therefore, we have to wait for more stories to be posted on the website in order to improve the performance of the recommendation system.

I. INTRODUCTION

Since HCD was launched in 2011, the system is still in the state of initial development. Our project aims to use machine learning to build a recommendation system to feed users with the projects and stories that potentially interest them. Also, we expect to build a system that connects a user to another user who share the same interest which can create huge impact to social sector around the world.

From the initial data sets provided by the IDEO.org, HCD now has 15,000 users and 310 stories. Each user provides personal information including sex, age, location, interest, and organization. For each story, the data includes title, description, author, topic, location, and design methods used.

Due to the lack of information on browsing history and user-story interaction, we cannot build the personalized recommendation system as we wish before. Therefore, this project will focus on the text classification of the stories. On each page of story, we aim to build a recommendation section on the relevant stories of the same class based on machine learning algorithm. This will improve the user's experience on the website.

This project consists of four main states. The first state is processing data. In the second state, we use unsupervised text classification algorithms in order to discover topics or clusters underlying the stories. In the third state, we build "auto-labeling" system from supervised learning algorithms. When a user posts a story on the website, he is asked to choose appropriate labels for the story. There are nine labels to choose from including agriculture, community development, education, energy, environment, financial services, gender equity, health and water . A story can have multiple labels. Given the stories with labels, we can use supervised text classification algorithms to automatically recommend the labels for each story. The last state does not involve machine learning. However, it creates a complete recommendation system that can be used on the website. In this state, we give a score to each story based on page view, post date, number of comments and likes. On each page of story, we can then recommend three stories with highest scores which are in the same class as the story on that page.

II. PROCESSING DATA : VECTOR SPACE MODEL AND METRIC

In this project, the training set is a collection of m documents with n unique tokens. Each story gives one training example. Here, a token is one single word, not an n-gram. We use a vector space model to represent the training set as an $m \times n$ term-document matrix. Each document is a vector of m dimensions. The number in row i, column j of the matrix represents the number of times word i appears in document j. Note that this representation does not keep the information of the order in which each word appears in the document. The size of our training set is $m = 310$, and the number of words throughout documents after preprocessing is $n = 2037$. In order to construct a term-document matrix, we first have to preprocess the document. Preprocessing takes an input as a plain text document from the name, title, and description of a story and provides an output as a set of tokens. Preprocessing are done in steps by filtering, tokenization, stemming, stopword removal, and pruning. We use TMG matlab toolbox[8] to preprocess the documents. Once we obtain the term-document matrix, a weight is assigned to each token indicating the importance of that token. This is done by tf-idf weighing scheme. The document vectors are now composed of weights reflecting the frequency of terms in the document multiplied by inverse of their frequency in the entire collection.

In order to measure the distance or similarity between two documents, we use the Euclidean norm. Since SVM is developed under Euclidean norm, this gives the consistency in testing different algorithms that we use in this project. Then the document vectors are normalized to unit vectors.

III. UNSUPERVISED TEXT CLUSTERING

After we process the text document to get a matrix of training set, we apply unsupervised text clustering algorithms to the training set in order to discover the structure of the data and possibly new topics in the cluster.

A. *Hierarchical Agglomerative Clustering (HAC)*

The first algorithm that we implement is Hierarchical Agglomerative Clustering (HAC). This algorithm is based on the idea of hierarchical clustering in which clusters build by the algorithm creates hierarchy or a rooted tree. There are two types of hierarchical clustering. The first type is “bottom-up”, where at the beginning each document is its own cluster. At each step, clusters merge based on some similarity measure. The second type is “bottom-down”, where initially all documents are in one cluster. Then at each step, a cluster is divided into two clusters. HAC uses the idea of “bottom-up” algorithm. The clusters will be merged and converged to the root of hierarchy. The merges are determined by similarity between two clusters called linkage. There are three common linkages: single, complete, and group average link. Here we use group average link only.

The algorithm can be summarized as follows:

1) Calculate linkage matrix whose entry (i, j) is the linkage between the i^{th} and j^{th} clusters.
2) Merge two clusters with highest linkage.
3) Update the linkage matrix to reflect pairwise linkage between the new cluster and the original clusters.
4) Repeat step 2 and 3 until there is only one cluster remains.

When implementing the algorithm, we can decide where to cut the hierarchical tree into clusters. There is a quality measure for HAC known as insufficiency measure cutoff. The results are the followings.

cutoff	number of clusters
1.0	79
1.1	60
1.15	30
1.155	1
1.16	1

TABLE I
THE RESULT FROM HAC

Here one can see that the number of clusters is either too large or too small. The results indicate that there is no significant topics or clusters discovered.

B. *K-means Clustering*

Another unsupervised algorithm is k-mean clustering. It is categorized as a non-hierarchical clustering. The data is assumed to have k clusters as priori without hierarchy. In our implementation, we measure the distance between clusters using the Euclidean norm. The algorithm can be summarized as follows:

1) Select k points as initial centroids.
2) Assign all points to the closest centroids using Euclidean norm.
3) Recompute the centroid of each cluster to be the centroid of those points assign to it.
4) Repeat step 2 and 3 until centroids no longer move.

First we attempt to find appropriate number of clusters k. Given k, we measure how well-separated the resulting clustering are using silhouette numbers. We then find the average of silhouette values of k clusters. Small silhouette value means the clusters are not well separated. The plot below shows the average silhouette numbers for each k.

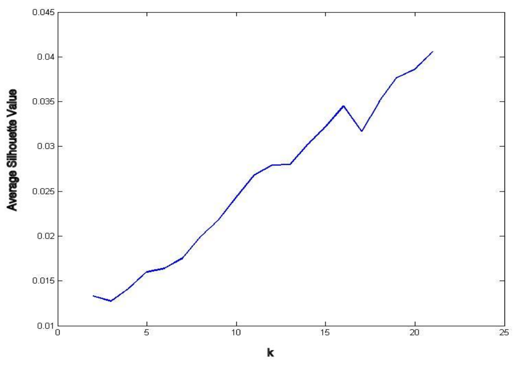

Fig. 1. The graph of average silhouette value vs. k

The average silhouette value is small (< 0.1) for all k. And for $k > 22$, the algorithm fails since an empty cluster is created. Therefore, there is no new significant topics or clusters discovered.

C. *Topic Modeling*

While HAC and k-means algorithms are known as hard-clustering, topic modeling is known as soft clustering because it clusters the training set by probabilistic model of topics. It specifies a simple probabilistic procedure by which documents can be generated. For each document, we assume that words are chosen in a two-stage process:

1) Randomly choose a distribution over topics.
2) For each word in the document
   - Randomly choose a topic from the distribution over topics in step 1).
   - Randomly choose a word from the corresponding distribution over the vocabulary.

The algorithm is designed to find the hidden structure, i.e. the per-document topic distributions, the per-document and per-word topic assignments. We use the observed documents to infer the hidden topic structure. This is known as Latent Dirichlet Allocation (LDA). After constructing the joint distribution of the hidden and observed variables, Bayes’s rule gives the conditional distribution of the topic structure given the observed documents. However, in general, the marginal probability of the observations, which is the sum of the joint distribution over every possible instantiation of the hidden topic structure, is exponentially large. Topic modeling algorithms need to form an approximation for such probability. Those algorithms fall into two categories - sampling based algorithms and variational algorithms. In this project, we only focus on a sampling-based algorithm in which Gibbs sampling is used. It is a statistical technique meant to quickly construct a sample distribution. Here we implement Gibbs sampling based LDA algorithm using 10 topics, which is comparable to the number of labels we had. The resulting topics are shown below.

topic	significant words
1	ideo learn organ org women earli tedx
2	commun creat model live work understand impact
3	project experi hear interview anim power base
4	team org busi fellow chang idea cookstov
5	improv health small process support opportun educ
6	prototyp field help research challeng india kenya
7	farm innov water urban program aquapon light
8	farmer food school local connect incom low
9	design center social rural human garden system
10	develop agricultur product sustain hcd servic technolog

TABLE II

THE SIGNIFICANT WORDS FOR EACH CLUSTER FROM TOPIC MODELING ALGORITHM

Notice that these clusters do not form according to the labels, for example, words related to agriculture appear in topic 7,8,9,10. Yet words in each topic do not share outstanding category. Hence, no new significant topics or clusters are discovered.

IV. SUPERVISED TEXT CLUSTERING : AUTO-LABELING SYSTEM

In this section, we use the information on labels from the writers of the stories to implement the supervised learning algorithms. For each story, we will use clustering algorithm to identify the labels that are fit the story the most based on our models. This system will help the writer identify the correct labels to the story. Throughout this section, we use q to denote the number of all possible labels, and m to denote the number of data. For this project, $q = 9$ and $m = 310$. Because a story can have multiple labels, this system relies on multi-label classification and corresponding evaluation methods. The multi-label classification methods can be divided into two categories: problem transformation methods and algorithm adaption methods.

The problem transformation methods transform one multi-label classification problem into multiple single-label classification problems, or into one classification problem with multiple labels. There are two canonical ways to do this. The first way is to do q-binary classifications, where q is the number of all possible labels. For each label i, we perform a single-label classification on the data to identify if each story should have label i or not. The other way is to do one classification with 2^q classes. Each of 2^q classes represents a subset of the set of q labels. In this project, $q = 9$ gives $2^9 = 512$ classes, while there are 310 training examples. The training size is very small compared to the number of classes. This method will not work well with our problem. Therefore, we will only implement the first method of transforming the problem into multiple single-label classification problems. We will use the k-nearest neighbor (kNN), and support vector machine (SVM) for this project. For the algorithm adaption methods, we will use method based on kNN and SVM.

A. Evaluations

For the evaluations, we use three methods including persistent precision, persistent recall, and Hamming loss. Suppose there are m test documents and q labels. The persistent precision and recall break the data into $m \times q$ instance-label pairs, and evaluate by regular precision and recall. Notice that these two evaluations do not utilize the idea of multi-label data. Another evaluation is Hamming loss. It measures the cardinality of symmetric difference between the set of predicted labels and the set of actual test labels averaged over the number of test documents and number of all possible labels. Hamming loss equal to zero means the algorithm makes a perfect prediction. And the algorithm performs well when Hamming loss closes to 0, while the algorithm performs poorly when Hamming loss closes to 1. In this project, we use simple cross validation by splitting 310 data instances into 210 instances for training, and 100 instances for testing.

B. Problem Transformation Methods

***k*-Nearest Neighbors (*k*NN).** In kNN method, for each label and each example in testing set, we find its k nearest neighbors in the training set using Euclidean norm. Then that example is identify whether it should be tagged with that label or not based on the statistical model we construct from the training set. The results from implementing q single-label kNN methods are below.

Value of k	8	9	10	11	12
Precision	0.659	0.764	0.711	0.713	0.697
Recall	0.400	0.353	0.465	0.444	0.451
Hamming Loss	0.247	0.231	0.221	0.224	0.228

TABLE III

THE EVALUATION OF kNN METHOD

Support Vector Machine (SVM). We implement SVM using 5 different kernels including linear kernel $K(x^{(i)}, x^{(j)}) = (x^{(i)})^T x^{(j)}$, Gaussian kernel $K(x^{(i)}, x^{(j)}) = \exp(-\gamma \|x^{(i)} - x^j\|^2)$, and n-degree polynomial kernel $K(x^{(i)}, x^{(j)}) = \left(\gamma (x^{(i)})^T x^{(j)}\right)^n$. Here we use $\gamma = 1$ for both Gaussian kernel and n-degree polynomial kernel. The cost parameter C is chosen to be one by default. The results are shown below.

Kernel	Gaussian	Linear	Polynomial of degree		
			2	3	4
Precision	0.550	0.550	0.550	0.550	0.550
Recall	0.200	0.200	0.200	0.200	0.200
Hamming Loss	0.294	0.294	0.294	0.294	0.294

TABLE IV

THE EVALUATION OF SVM

Value of k	8	9	10	11	12
Precision	0.659	0.764	0.711	0.713	0.697
Recall	0.400	0.353	0.465	0.444	0.451
Hamming Loss	0.247	0.231	0.221	0.224	0.228

TABLE V
THE EVALUATION OF MUTI-LABEL kNN

For this application, it is more important that the users do not miss a chance to read the stories they are interested in than they are recommended the stories in other topics. Therefore, recall is more important than precision. In kNN algorithm, recall is highest when $k = 10$. In SVM algorithm, it is somewhat surprising that different kernels give the same performance through all evaluation measure. One possible reason is the examples $x^{(i)}$'s do not contribute significant changes. Thus, we conclude that among SVM algorithm, linear is sufficient to implement. Overall, kNN algorithm outperforms SVM algorithm.

C. Algorithm Adaptation Methods

Algorithm adaptation methods consider all q labels together. Most of these methods are variations of single-label classification algorithms. In this project, we use methods based on kNN and SVM.

Multi-label kNN For this method, we used the algorithm based on the paper by Zhang and Zhao[9]. Proposed by Zhang and Zhao in 2007, multi-label kNN is a variation of kNN algorithm applied to multi-label classification. In regular kNN algorithm, an unlabeled vector is classified by assigning the label which has the greatest probability among the k nearest training examples query point based on the statistical model constructed from the training data. In multi-label kNN, we need an alternative method to utilize information from neighbors. After identifying all neighbors, the algorithm chooses labels on the query point to maximize a posteriori (MAP) principle. When training the data, for each label t, we construct the probability that a given example has label t and the probability that it does not have label t given that it has l neighbors with label t, where $t = 1,\ldots,q$ and $l = 0,\ldots,k$. Based on these probabilities, we can assign the labels to each test example, and evaluate the performance of the algorithm. The results are shown below.

Similar to regular kNN, Multi-label kNN performs best when $k = 10$. Notice that the value of all evaluation measure are same as regular kNN method. This is because in the paper by Zhang and Zhao[9], each label is treated separately in building the statistical model and in assigning the labels to the test examples. Therefore, this is equivalent to the regular kNN we present in section IV-B.

Rank-SVM Proposed by Eisseeff and Weston in 2002, Rank-SVM is a variation of SVM algorithm applied to multi-label classification[3]. In regular SVM algorithm, an unlabeled vector is classified by its location with respect to a separating hyperplane. The separating hyperplane is chosen to maximize the margin and minimize penalty from separating hyperplane violation. In multi-label kernel learner, we need an alternative method to draw hyperplanes and assign penalty function for multi-label data. In regular SVM, we find the hyperplanes by solving the optimization problem

$$\min_{\xi,w,b} \quad \tfrac{1}{2}||w||^2 + C\textstyle\sum_{i=1}^{m} \xi_i \tag{1}$$
$$\text{subject to} \quad y^{(i)}(< w, x^{(i)} > -b) \geq 1 - \xi_i, \tag{2}$$
$$\xi_i \geq 0, i = 1,\ldots,m \tag{3}$$

Ideally, we would prefer all data with positive label to be above the hyperplane $< w, x^{(i)} > -b = 0$. ξ represents the violation to such ideal. In Rank-SVM, we now have multiple hyperplanes $\{w_k, b_k\}$ where $k = 1,\ldots,q$. The optimization problem associated to Rank-SVM is the following.

$$\min_{\xi,w_j,b_j,j=1,\ldots,q} \frac{1}{2}\sum_{k=1}^{q} ||w_k||^2 + C\sum_{i=1}^{m} \frac{1}{|Y_i||\bar{Y}_i|} \sum_{(k,l)\in Y_i\times\bar{Y}_i} \xi_{ikl} \tag{4}$$

$$\text{subject to} \quad < w_k - w_l, x_i > -b_l + b_k \geq 1 - \xi_{ikl}, (k,l) \in Y_i \times \bar{Y}_i, \tag{5}$$
$$\xi_{ikl} \geq 0, \tag{6}$$

where Y_i is the set of labels for document i, and $\bar{Y}_i$ is the complement of Y_i. Here, we may view quantity $< w_k, x > +b_k$ as a k^{th} score. Ideally, we would prefer all data with label k to have k^{th} score higher than any scores associated to non-labels l. That is why this algorithm called Rank-SVM. ξ represents the violation to such ideal.

To implement rank-SVM algorithm, we need to specify parameters associated to each kernel as in regular SVM in Section IV-B. Here we use $\gamma = 1$ for Gaussian kernel and for n-degree polynomial kernel. The cost parameter C is chosen to be one by default. The results are shown below.

Kernel	Gaussian	Linear	Polynomial of degree		
			2	3	4
Precision	0.515	0.591	0.515	0.515	0.515
Recall	0.382	0.426	0.382	0.382	0.382
Hamming Loss	0.299	0.266	0.299	0.299	0.299

TABLE VI
THE EVALUATION OF RANK-SVM

Here one can see that Rank-SVM performs better than regular SVM throughout all evaluation measures. It shows effectiveness of algorithm adaptation methods. However, Rank-SVM gives relatively poor performance compared to even regular kNN. Overall, regardless of algorithms, the auto-labeling system performs poorly, recall rate is below 50%. There are two explanations for this. First, the quality of labels given in our data set may be poor. Consider the histogram below.

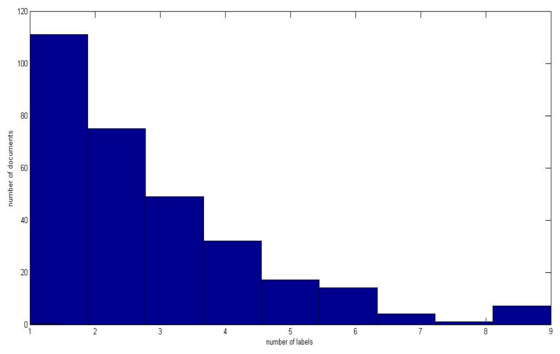

Fig. 2. Histogram of number of labels for 310 documents

There are 43 documents with at least 5 labels. These labels may not reflect important topics that we can capture by learning from the content of the document. It is possible that the writers know by themselves that the story related to many labels. However, the content of the documents do not reflect the relationship with the labels. If we remove those documents out, we might improve the performance of the auto-labeling system. Another way to improve the labeling system is to have a person other than the writer assigning the labels to the stories. Another reason explaining why the algorithms perform poorly is the dataset here may be too small. This might be the reason why kNN, a local-based algorithm, outperforms SVM, a global-based algorithm. We need to wait for more stories and labels from the users in order to fully implement the auto-labeling system.

V. SCORE OF RECOMMENDATION SYSTEM

In this part, we assign a score for each story. Once the user visits a story j, we will collect all stories with at least one similar label to the current story and compute the score as follows:

score (story j, current story i) = f_2(number of page views j) + f_3(number of comments j) + f_4(number of likes j) +g_1(current date - post date j) + g_2(distance(i,j)),

where f refers to an increasing function since parameters inside fs contribute to positive effects (more preferable to users), while parameters inside gs contribute to negative effects (less preferable to users). Simple functions for fs and gs are linear functions with positive and negative parameters, respectively. These parameters should be weighted depending on importance of each factor. For example, if it does not matter how long the stories has been published, we may set a factor of g_1 to be zero. Then, we can recommend three stories with highest scores on the page of the current story. This score system may extend to personalized recommendation system if we know a story-user interaction. For example, we may add $g_3(\sum_k(j,k))$ where k refers to the stories that the user has visited. To complete machine-learning based recommendation system even with the early state, we need to improve auto-labeling system which requires more data.

VI. CONCLUSION

The unsupervised classification algorithms do not give significant topics underlying the stories. From the supervised classification algorithms, the multi-label SVM performs better than the regular SVM. Overall, the k-nearest neighbors algorithm (kNN) outperforms all other algorithms. However, the precision of the algorithm is still very low. This is because the data set is too small. Therefore, looking at the local structure at each data point (kNN) gives the better results. Moreover, this indicates that we should wait for more stories to be posted by the writers in order to improve the performance of the recommendation system. However, in term of memory storage, SVM is the most efficient algorithm because the computer only have to store the relevant parameters instead of the whole data set. Therefore, we predict that once there is large enough data set, rank-SVM will perform best in term of both precision and efficiency.

REFERENCES

[1] A. Nicholas O., and F. Edward A., *Recent Developments in Document Clustering*, October 2007.
[2] B. David M., *Introduction to Probabilistic Topic Models*, Princeton University.
[3] E. Andre, and W. Jason, *A kernel method for multi-labelled classification.*
[4] G. Tom, and S. Mark, *Matlab Topic Modeling Toolbox 1.4*. Release Date: April 2011.
[5] G. Tom, and S. Mark, *Probabilistic Topic Models*. University of California, Irvine.
[6] "Hierarchical Clustering." *MathWorks.* The MathWorks, Inc.
[7] "K-Mean Clustering." *MathWorks.* The MathWorks, Inc.
[8] "TMG." *Text To Matrix Generator*. <http://scgroup20.ceid.upatras.gr:8000/tmg/>.
[9] Z. Min-Ling, Z. Shi-Hua, *ML-KNN: A Lazy Learning Approach to Multi-Label Learning*. Pattern Recognition, 2007, 40(7): 2038-2048.