# A Transfer-Learning Approach to Exploit Noisy Information for Classification and Its Application on Sentiment Detection

Wei-Shih Lin[*], Tsung-Ting Kuo[*], Yu-Yang Huang[*], Wan-Chen Lu[+], Shou-De Lin[*]

[*]Department of Computer Science & Information Engineering, National Taiwan University
[+]Telecommunication Laboratories, Chunghwa Telecom Co., Ltd
[*]{r00922013, d97944007, r02922050, sdlin}@csie.ntu.edu.tw
[+]janelu@cht.com.tw

**Abstract.** This research proposes a novel transfer learning algorithm, Noise-Label Transfer Learning (NLTL), aiming at exploiting noisy (in terms of labels and features) training data to improve the learning quality. We exploit the information from both accurate and noisy data by transferring the features into common domain and adjust the weights of instances for learning. We experiment on three University of California Irvine (UCI) datasets and one real-world dataset (Plurk) to evaluate the effectiveness of the model.

**Keywords:** Transfer Learning, Sentiment Diffusion Prediction, Novel Topics

## 1 Introduction

This paper tries to handle the situation where there is no sufficient expert-labelled, high quality data for training by exploiting low-quality data with imprecise features and noisy labels. We generalize the task as a classification with noisy data problem, which assumes both features and labels of some training data are noisy, similar to [1]. More specifically, we have two different domains of labeled training data. The first we call it the high-quality data domain, which contains data of high quality labels and fine-grained features. We assume it is costly to obtain such data, therefore only a small amount of it can be obtained. The other is called the low-quality data domain, which contains noisy data and coarse-grained features. Unlike high quality data, the volume of this data can be large.

The example we use throughout this paper to describe our idea is the compulsive buyer prediction problem given transaction data from different online stores (e.g. Amazon, eBay, etc.). Let us assume the users' transaction records from different online websites are obtained as our training data to train a model for compulsive buyer classification. As shown in Fig. 1, there are some common features for users across these stores, such as gender and month or birth. However, there are also features that are common across different stores but have different granularity due to different registration processes. For instance, age can be exact (e.g. 25 years old) or in a range (e.g. 20~30), and same situation applies to locale and job categories.

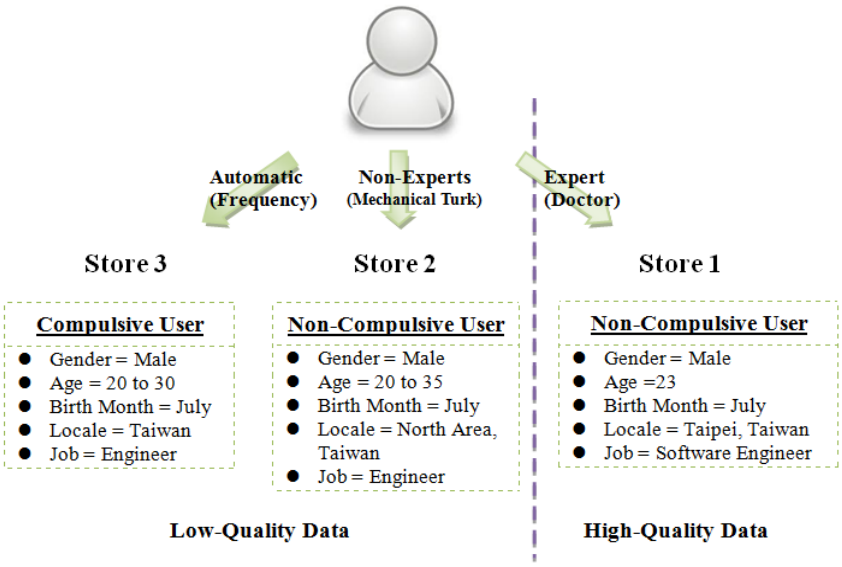


**Fig. 1.** An Example of Compulsive User Prediction.

Assume we ask experts to annotate whether a person is a compulsive buyer based on the transaction and content information of a more fine-grained dataset from a particular store. This dataset is considered the high-quality data. For the data with coarse-grained features, we can hire non-experts (e.g., through Mechanical Turk) or exploit some indicators such as shopping frequency and quantity to label data. Such data (we call it low-quality data) might not be as accurate and precise as the high-quality data, but can potentially boost the learning performance under the assumption that there is only few high-quality data available. Training a classifier using such data is non-trivial because (1) the features/labels in the low-quality domain might not be precise/correct, and (2) the data distribution in the low-quality training domain and the testing domain might not be identical.

In this paper, we propose a novel transfer learning algorithm, Noisy-Label Transfer Learning (NLTL). First, we identify the mapping function between features from different domains. Next, we learn the importance of instance based on the labels from the different domain of data. Finally, we exploit the learnt importance of instances to improve the prediction accuracy. To summarize, the main contributions of this paper are as follows:

- We introduce a novel and practical classification task given noisy data. In this problem, only small amount of correctly labeled data along with large amount of roughly labeled data are available for training.
- We propose a transfer learning approach to solve the above-mentioned problem, and provide a practical application scenario on sentiment diffusion prediction.

- We experiment with three University of California Irvine (UCI) datasets and one real-world dataset (Plurk) and show that our algorithm significantly outperforms the state-of-the-art transfer learning and multi-label classification methods.

## 2 Related work

The concept of transfer learning lies in leveraging common knowledge from different tasks or different domains. In general, it can be divided into inductive and transductive transfer learning, based on the task and data [2].

TrAdaBoost [3] is an inductive instance transfer approach extended from AdaBoost. TrAdaBoost applies different weight-updating functions for instances in the target domain and in the source domain. Since the distribution in the target domain is more similar to that of the testing data, the incorrect predictions in the target domain generally are assigned higher weights, comparing to those in the source domain.

Structural Correspondence Learning (SCL) [4] is a transductive transfer learning with feature-representation transfer approach. It defines features with similar behavior in both domains as pivot features and the rest as non-pivot features. Then it tries to identify the correlation mapping functions between these features.

Our proposed algorithm belongs to transductive transfer learning, which applies both instance and feature-representation transfer. However, the most important difference is that we deal with items that have diverse labels in different domains. Those items are used to serve as a bridge to connect different domains.

## 3 Methodology

### 3.1 Problem Definition

We start by formulating the problem. Suppose a high-quality domain dataset $D_H$ and N different low-quality domain dataset $D_{L_j}$, where $1 \leq j \leq N$, are given. We define high-quality domain data as $D_H = \{(x_{H_1}, y_{H_1}), \dots, (x_{H_{n_H}}, y_{H_{n_H}})\}$, where $n_H$ is the number of instance in $D_H$, $x_{H_i} \in X_H$ represent the features of an instance, and $y_{H_i} \in Y_H$ is the corresponding label. Here we assume low-quality domain data can come from multiple sources, defined as $D_L = \{D_{L_1}, \dots, D_{L_N}\}$ and $|D_L| = n_L$. The low-quality domain data from each source can be presented as $D_{L\_j} = \{(x_{L\_j_1}, y_{L\_j_1}), \dots, (x_{L\_j_{n_{L\_j}}}, y_{L_{n_{L\_j}}})\}$, where $n_{L\_j}$ is the number of instance in $D_{L\_j}$, $x_{L\_j_i} \in X_{L\_j}$, and $y_{L\_j_i} \in Y_{L\_j}$. Moreover, we assume that instances in $D_H$ contain high quality labels and fine-grained features and those in $D_L$ have coarse-grained features and noisy labels. Note that in general we assume $n_H \ll n_L$, as obtaining high quality data is more expensive and time-consuming.

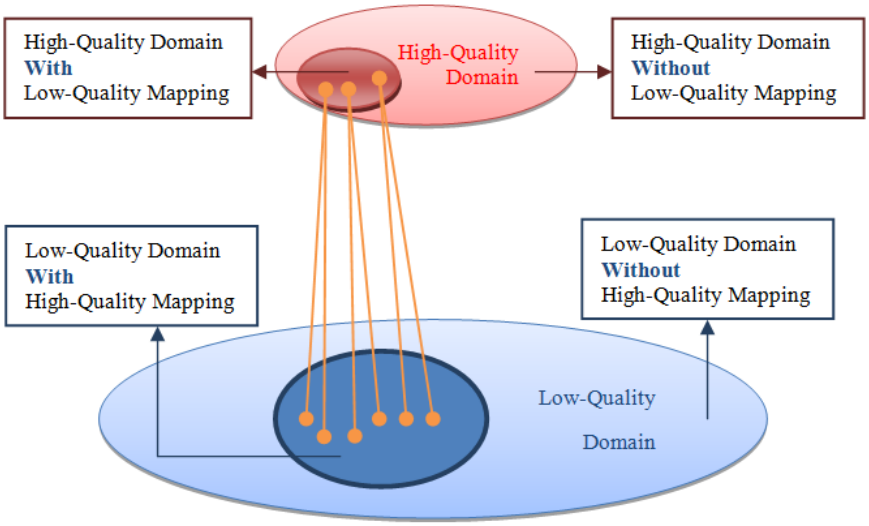


**Fig. 2.** Sketch Illustration for Instances

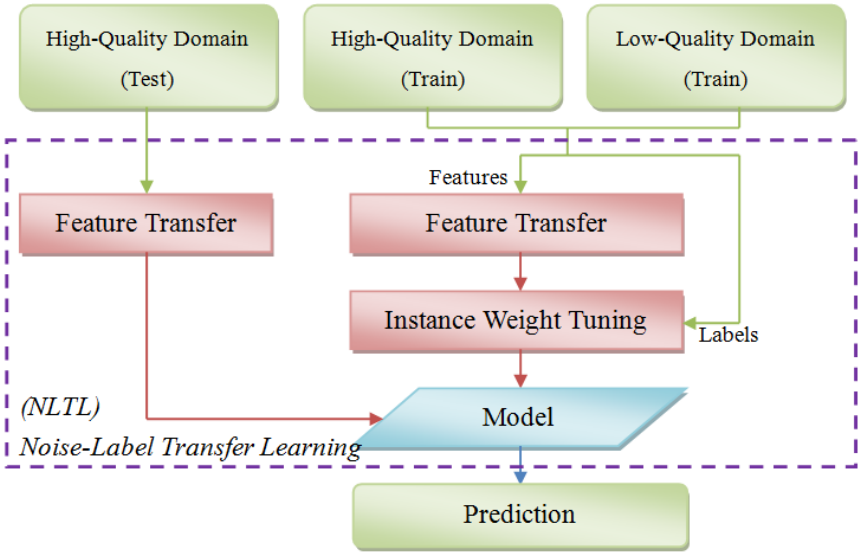


**Fig. 3.** Algorithm Architecture

**Input:** High-quality domain data $D_H$.
Low-quality domain data $D_L$.
Maximum iterations of AdaBoost $k$.

**Output:** The hypothesis $h$.

1. $X_{SL} = (D_H \cap D_L | D_L)$
2. $X_{SH} = (D_H \cap D_L | D_H)$
3. $\theta = (X_{SL}^T \cdot X_{SL})^{-1} \cdot X_{SL}^T \cdot X_{SH}$
4. Update the new feature space $X' = \{X'_H, X'_L\}$ for each instance:

$$x'_i = \begin{cases} [x_i, x_i], & x_i \in X_H \\ [x_i, x_{i*}], & x_i \in X_{SL} \wedge x_i \; maps \; to \; x_{i*} \\ [x_i, x_i\theta], & others \end{cases}$$

5. Initial weight vector by comparing the label for different domain: $\boldsymbol{w^1} = (w_1^1, \ldots, w_{n_H+n_L}^1)$.
6. **For** $t = 1$ to $k$:
7. Train a classifier $h_t: X' \rightarrow [0, 1]$.
8. Calculate the error rate $\epsilon_t$ of $h_t$ on $X'_H$:

$$\epsilon_t = \frac{\sum_{i=1}^{n_H} w_i^t \cdot |h_t(x'_i) - y_i|}{\sum_{i=1}^{n_H} w_i^t}$$

9. $$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}$$
10. $$\beta = \frac{1}{1 + \sqrt{2 \ln n_L / k}}$$
11. Update weight vector $\boldsymbol{w^{t+1}}$:

$$w_i^{t+1} = \begin{cases} w_i^t \beta_t^{-|h_t(x'_i) - y_i|}, & 1 \le i \le n_H \\ w_i^t \beta^{|h_t(x'_i) - y_i|}, & n_H \le i \le n_H + n_L \end{cases}$$

12. **End**
13. **Return** hypothesis $h_k$.

**Algo. 1.** Noise-Label Transfer Learning (NLTL)

We show a simple sketch illustration for the relationship between training instances in Fig. 2, where the red and blue areas denote the instances in high-quality and low-quality domains respectively. The dark areas represent instances belonging to both domains. However, the features that represent these instances might have different granularity, and the labels in low-quality domain might be incorrect. Each instance can belong to one of the four groups (as shown in Fig. 2), high-quality domain with low-quality mapping, high-quality domain without low-quality mapping, low-quality domain with high-quality mapping, and low-quality domain without high-quality mapping. Finally, the task to be solved is defined as given $\mathrm{D}_{\mathrm{H}_{\mathrm{train}}}$ and $\mathrm{D}_{\mathrm{L}_{\mathrm{train}}}$, learn an accurate classifier to predict $\mathrm{D}_{\mathrm{H}_{\mathrm{test}}}$.

### 3.2 Noise-Label Transfer Learning (NLTL)

We propose NLTL, which is a transfer learning model to solve the above-mentioned problem. The overall architecture is shown in Fig. 3. The idea is to transfer information from low-quality domain data to improve the prediction in high-quality domain which has insufficient training instances. Note that for each object, we may integrate corresponding instances from multiple low-quality data sources. NLTL first uses instances existing in both high-quality and low-quality domains as a bridge to identify the correlation between coarse-grained and fine-grained features. Then it learns the weight of instances from each domain to train a binary classifier to predict testing data in the high-quality domain. It should be noted that we perform feature transfer on both training and testing data, however, only training data are used to learn the weight of instances since testing data are not labeled. We define NLTL in Algorithm 1. Feature transfer is performed using Structural Corresponding Learning (SCL) [4] (Step 1 to Step 4, see 3.3), and TrAdaBoost [3] is used to tune the weight of instances (Step 5 to Step 12, see 3.4).

### 3.3 Feature Transferring

We want to handle the problem that the quality of features in low-quality domain is not as good as that in high-quality domain in terms of granularity. The goal is to identify a mapping function to project the features in the low quality domain to the high quality domain, by changing their distributions.

We propose a method based on Structural Corresponding Learning (SCL) [4]. The intuition is to identify the correspondences among the features from different domains by modeling their correlation with features that have similar distribution in both domains. To transfer the low-quality data into high-quality domain, for each feature in the low-quality domain, it is necessary to find its mapping to the more fine-grained high-quality domain. Here we propose to create a prediction model to perform the mapping. That is, for each feature in the high-quality domain, we create a classification or regression model, for categorical and numerical features respectively, to predict its value given each corresponding instance in the low-quality domain. Assume an user $u$ appears in both high-quality domain (its feature vector, denoted as $u_{s1}$, is {“Male”, “22”, “May”, “Taipei”, “Software Engineer”} ) and low-quality domain (feature vector denoted as $u_{s2}$ , which is {“Male”, “20 to 30”, “May”, “Taiwan”, “Engineer”}). $u_{s1}$ will of course be used as the training example to learn a compulsive user model, but we want to use $u_{s2}$ as well to enlarge the training set. Therefore, for each feature in the high-quality domain, we create a classifier that maps $u_{s2}$ to a corresponding value. In our example, we will build 4 classifiers and 1 regressor (for ‘age’ feature), each of which takes an instance in $u_{s2}$ as input and output the possible assignment for the fine-grained feature.

We denote these models as mapping function $\theta$, and it models the correlation between the features from different domain. In the experiment we use linear regression to learn $\theta$.

$$\theta = (X_{SL}^T \cdot X_{SL})^{-1} \cdot X_{SL}^T \cdot X_{SH}$$

where $X_{SL}$ denotes features with instances in the low-quality domain that have high-quality mapping, and $X_{SH}$ denotes features with instances in the high-quality domain that have low-quality mapping.

Finally, we create a new feature space, which is twice in length comparing to the original feature space, for the processed instances. The instances are processed in three different ways. 1) For instances appear in both low-quality and high-quality domains, we concatenate the corresponding low-quality features with the original high-quality features. 2) For instances that only appear in high-quality domains, we simply copy the features and concatenate them to the end. 3) For instances that only appear in low-quality domain, we first generate the corresponding mapping to the high-quality domain, and then treat it like case 2.

### 3.4 Instance Weight Tuning

We are now ready to exploit the instances from both domains to train a classifier. However, it is apparent that the instances from high-quality and low-quality domains should not be treated equally during training. Here we propose a method to adjust the initial weights on each instance according to the following heuristics.

- Instances in the high-quality domain should have higher weights. Furthermore, if the corresponding low-quality instances also contain identical label, the weight is even higher.
- For instances in the low-quality domain that can be mapped to high-quality domain with the same labels, their weights should be greater than the weights of the instances that cannot be mapped to high-quality domain.

We order the instances based on the above heuristics, and assign initial weight as $W_i = W_{i-1} \times \alpha$ where $\alpha<1$, $W_i$ and $W_{i-1}$ stands for instances of order *i* and *i-1*. $W_i$ represents the set of weights to the instances. After setting initial instance weights, we apply TrAdaBoost [3] to tune the weights iteratively. The intuition of TrAdaBoost is to use different weight-updating function for different domain data. More specifically, we increase the weight more if the instance is predicted incorrectly in high quality domain. The assumption of this setting is that the data in low-quality domain does not have as high confidence score as those in high-quality domain. The formulas of TrAdaBoost to update the instance weights are as follows:

$$w_i^{t+1} = \begin{cases} w_i^t \beta_t^{-|h_t(x_i)-y_i|}, \textit{ in high-quality domain} \\ w_i^t \beta^{|h_t(x_i)-y_i|}, \textit{ in low-quality domain} \end{cases}$$

where $\beta$ and $\beta_t$ are multiplier calculated by error rates and traditional AdaBoost.

| | CTG | Magic | Wine |
|---|---|---|---|
| **High-Quality** | 88.35% | 75.49% | 69.95% |
| **Low-Quality_1** | 85.58% | 76.91% | 73.89% |
| **Low-Quality_2** | 85.14% | 58.81% | 66.28% |
| **All Instance** | 89.20% | 78.22% | 75.37% |
| **TrAdaBoost** | 89.56% | 81.28% | 76.52% |
| **SCL** | 86.58% | 76.04% | 69.42% |
| **Label-Powerset** | 85.77% | 78.82% | 71.29% |
| **NLTL** | **91.83%** | **81.71%** | **76.63%** |

**Table 1.** Experiment Results in AUC

# 4 Experiments

## 4.1 Dataset and Settings

We test our model on three datasets (CTG, Magic, and Wine) collected from UCI Machine Learning Repository [5]. We preprocess the labels to binary classes in our experiment. The three datasets contain 2126, 19020, and 6497 instances and 21, 10, and 11 features, respectively. For each dataset, we use original features and labels as high-quality domain data. To generate noisy low-quality domain data, we randomly pick c% of instances, flip their labels, and modify their features to be coarser. For example, for a numerical feature, we quantize its values into K groups, and assign the medium value for each group as the new feature value. In our experiment, we generate two low-quality domain datasets with $(c, K) = (20, 5)$ and $(c, K) = (50, 10)$. To reflect the fact that correctly labeled data are rare, we randomly choose 10% of high-quality domain data for training and keep the remaining for testing. We use 4-fold cross validation for evaluation.

We choose area under ROC curve (AUC) as the evaluation metric because of data imbalance. We rank the testing instances base on the predicted positive probability, and then compare it to the ground truths to produce AUC. For weight tuning, we manually assign the largest weight to 10 and $\alpha$ to 0.7. That is, the second largest weight is 7, third is 4.9, and so on. We compare our model with three types of algorithms, traditional non-transfer learning (High-Quality, Low-Quality_1, Low-Quality_2 and All Instance), transfer learning (TrAdaBoost and SCL), and multi-label (Label-Powerset) algorithms.

## 4.2 Results

We show the results comparing other baselines to NLTL in Table 1. The best results are marked in bold.

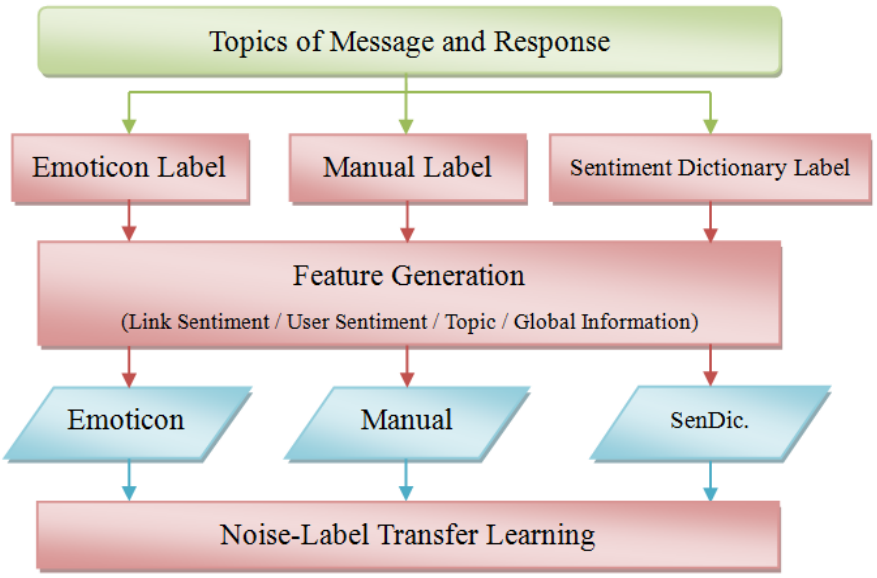


**Fig. 4.** Framework of Sentiment Diffusion Prediction with NLTL

The results show that NLTL outperforms the competitors for all dataset, especially for CTG. It also shows that by exploiting low-quality domain data, NLTL is useful and can improve the result using only high-quality domain data (denoted as High-Quality in Table 1) up to 6.7% in terms of AUC. On the other hand, NLTL combines the advantages of TrAdaBoost and SCL. It considers not only features but also labels to the same items together. The improvement of NLTL over baseline algorithms shows that both features and labels information from low-quality domain data are important and useful.

## 5 Sentiment diffusion prediction on novel topics

In this section, we use NLTL to handle a novel real-world sentiment diffusion prediction problem. Sentiment prediction aims at predicting whether an opinion is positive or negative [6]. However, in this application, we are interested in predicting the diffusion of sentiment through social networks. In other words, we emphasize on sentiment "diffused" rather than sentiment "expressed" by a user. Analyzing sentiment diffusion allows us to understand how people react to other people's comments on micro-blog platforms.

Traditional sentiment prediction uses a variety of textual or linguist information as features [6]. Such solution has a serious drawback as it is unable to handle new topics that appear rarely. On the other hand, Kuo et al. [7] propose a method to predict the

diffusion on novel topics utilizing latent and social features. Rather than predicting the existence of diffusion, we extend [7] to predict the diffusion of sentiments.

Our framework applies NLTL as shown in Fig. 4. We first provide high-quality and low-quality labels using three methods, and then the features are generated as described before. Finally, we learn a classifier using both high-quality and low-quality domain data, and show that low-quality domain data is useful in improving the performance.

### 5.1 Labeling

We provide high-quality labels (manual labeling) as well as low-quality labels (using emoticon and sentiment dictionary). The low-quality labeling methods are automatic and low-cost but the result may contain noises.

- **Emoticon Labeling.** We first manually classify the emoticons which are clearly positive or negative. Then, we use the emoticons to decide the label (positive or negative) of the diffusions.
- **Manual Labeling.** Human annotators are asked to label whether the content is positive, negative, or unknown.
- **Sentiment Dictionary Labeling.** We construct a sentiment dictionary and label the diffusions based on the voting of the words in the sentiment dictionary.

### 5.2 Dataset

We first identify 100 top discussion topics from Plurk micro-blog site [8]. We collect the messages and responses from users who discuss about those topics in the period from 01/2011 to 05/2011. A diffusion of sentiment is denoted as $(x, y, t, s)$, which means user $x$ posts a message of topic $t$, and user $y$ responses $x$ with sentiment $s$ (positive or negative, labeled by different methods introduced in 5.1). This dataset contains 699,985 objects, thus is not practical to label them all manually. We choose 17% of the objects to be labeled manually, while other objects are labeled using emoticon and sentiment dictionary. Finally, we obtain 82,277 diffusions from manual labeling, 117,876 diffusions from emoticon labeling, and 396,370 diffusions from sentiment dictionary labeling.

### 5.3 Feature Generation

To perform sentiment prediction, we design the following features. We divide the proposed features into four types as follows.

- **Link Sentiment Information.** The link sentiment information describes the tendency of each link in the network to be positive or negative for a given topic. For a link, link sentiment score ($LS$) is calculated by comparing the number of times that a positive or negative content is diffused. That is, we increase $LS$ by one for each positive diffusion and decrease LS by one for each negative diffusion.

| | All Features | Best Features |
|---|---|---|
| **High-Quality** | 62.90% | 65.04% |
| **Low-Quality_1** | 61.73% | 65.36% |
| **Low-Quality_2** | 63.47% | 66.26% |
| **All Instance** | 62.13% | 64.25% |
| **SCL** | 61.65% | 62.33% |
| **TrAdaBoost** | 61.84% | 65.27% |
| **Label-Powerset** | 59.58% | 62.59% |
| **NLTL** | **64.21%** | **68.30%** |

**Table 2.** Sentiment Diffusion Prediction results

- **User Sentiment Information.** Similar to link sentiment information, user sentiment information models the tendency of each user to reply to positive/negative posts. For a user, we generate the user sentiment score according to sender aspect ($USS$), receiver aspect ($USR$), and sender-receiver aspect ($USSR$). More specifically, for $USS$ we only consider the number of positive and negative posts sent by user, and ignore those received by this user. On the other hand, $USR$ only considers the number of positive and negative posts received by user. $USSR$ considers both aspects.
- **Topic Information.** We follow the same approach described in [7] to extract latent topic signature ($TG$) features. Besides *TG*, we also extract topic similarity (*TS*) features weighted by link sentiment information and user sentiment information. There are four features generated based on topic similarity, topic similarity for link sentiment ($TSLS$), topic similarity for user sentiment with sender aspect ($TSUSS$), topic similarity for user sentiment with receiver aspect ($TSUSR$), and topic similarity for user sentiment with sender-receiver aspects ($TSUSSR$).
- **Global Information.** We extract global social features such as in-degree (*ID*), out-degree (*OD*), and total-degree (*TD*) from social network. Note that these three features remain the same for different labeling methods; thus, we utilize them as pivot features in SCL and NLTL algorithms.

### 5.4 Results

The experiment setting of sentiment diffusion prediction task is the same as that described in Section 4. We compare NLTL that utilizes three sources to the competitors as described in 4.1. We run the experiment on two set of feature combinations: using all features and the best feature combination chosen using wrapper-based forward selection method [9]. The result shows that NLTL is able to integrate the information of features and labels to outperform the competitors by a large margin.

## 6 Conclusion

In this paper, we propose a novel prediction problem together with a transfer learning algorithm to solve it. We serve the objects which have multiple labels as a bridge and transfer knowledge from different data domains. We update instance weights and transfer features by comparing labels and features in high-quality domain and low-quality domain simultaneously. The experiment result shows NLTL consistently outperforms the competitors. Furthermore, we propose a real-world task of sentiment diffusion prediction that can benefit from our framework. Our experiments demonstrate how such problem can be formulated into a noisy-label prediction task that can be solved using NLTL.

## Acknowledgement

This work is primarily supported by a grant from Telecommunication Laboratories, Chunghwa Telecom Co., Ltd under the contract No. TL-103-8201.

**References.**

1. J. A. Sáez, M. Galar, J. Luengo and F. Herrera.: Tackling the Problem of Classification with Noisy Data Using Multiple Classifier Systems: Analysis of the Performance and Robustness. in Information Science, 2013.
2. S. J. Pan and Q. Yang.: A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22:1345-1359, 2010.
3. W. Dai, Q. Yang, G. Xue and Y. Yu.: Boosting for Transfer Learning. In: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 193-200.
4. J. Blitzer, R. McDonald and F. Pereira.: Domain Adaptation with Structural Correspondence Learning. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 2006, pp. 120-128.
5. Bache, K. and Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science.
6. X. Meng, F. Wei, X. Liu, M. Zhou, S. Li and H. Wang.: Entity-Centric Topic-Oriented Opinion Summarization in Twitter. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 379-387, 2012.
7. T.-T. Kuo, S.-C. Hung, W.-S. Lin, N. Peng, S.-D. Lin and W.-F. Lin.: Exploiting Latent Information to Predict Diffusions of Novel Topics on Social Network. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pp. 344-348, 2012.
8. T.-T. Kuo, S.-C. Hung, W.-S. Lin, S.-D. Lin, T.-C. Peng and C.-C. Shih.: Assessing the Quality of Diffusion Models Using Real-World Social Network Data. In: Technologies and Applications of Artificial Intelligence (TAAI) 2011, pp. 200-205, 2011.
9. R. Kohavi and G. H. John.: Wrappers for feature subset selection. In: Artificial Intelligence, 97.1:273-324, 1997.