The Automated Acquisition of Suggestions from Tweets

Li Dong[*]
SKLSDE, Beihang University
donglixp@gmail.com

Furu Wei
Microsoft Research Asia
fuwei@microsoft.com

Yajuan Duan[*]
University of Science
and Technology of China
dyj@mail.ustc.edu.cn

Xiaohua Liu
Microsoft Research Asia
xiaoliu@microsoft.com

Ming Zhou
Microsoft Research Asia
mingzhou@microsoft.com

Ke Xu
SKLSDE, Beihang University
kexu@nlsde.buaa.edu.cn

Abstract

This paper targets at automatically detecting and classifying user's suggestions from tweets. The short and informal nature of tweets, along with the imbalanced characteristics of suggestion tweets, makes the task extremely challenging. To this end, we develop a classification framework on Factorization Machines, which is effective and efficient especially in classification tasks with feature sparsity settings. Moreover, we tackle the imbalance problem by introducing cost-sensitive learning techniques in Factorization Machines. Extensively experimental studies on a manually annotated real-life data set show that the proposed approach significantly improves the baseline approach, and yields the precision of 71.06% and recall of 67.86%. We also investigate the reason why Factorization Machines perform better. Finally, we introduce the first manually annotated dataset for suggestion classification.

Introduction

Twitter becomes one of the most popular social networking sites, which allow users to read and post messages (i.e. tweets) up to 140 characters. The service rapidly gained worldwide popularity, with over 140 million active users, generating over 340 million tweets daily in March 2012 [1]. Among the great varieties of topics, such as breaking news, opinions towards products or celebrities, and even status updates, people in Twitter might express their suggestions and proposals for brands, products and public events. For example, in a tweet - "*#microsoft #WindowsPhone7 I'd like multitasking please*" , the user directly expresses the suggestion of functionality (i.e. *multitasking*) for the product "#WindowsPhone7" (called the *target*). In another tweet, "*I think microsoft needs to a lot to make the WP7 marketplace better. ie giftcards, promo codes, like the iTunes store.*", the user even gives some concrete proposals for a company "@microsoft" (the target) for marketing. These kind of tweets are invaluable resources for companies to improve or enrich the quality and functionality of products and services, as well as for organizations to develop political and economic strategies. For example, we can get the real-time and the up-to-date feedbacks from Twitter, which can save time and avoid costly market survey. Moreover, it is invaluable for companies to respond quickly to customer demands. Suggestion analysis is also a meaningful form of utilizing the wisdom from crowds owing to the huge number of users and the diversity among them in Twitter. The users may propose some novel ideas even beyond the scope of experts.

In this paper, we target at automatically acquiring of user's suggestions from tweets. In particular, we propose a classification framework to efficiently and effectively detect suggestion tweets from huge amount of tweets available in Twitter. This task is extremely challenging owning to the following reasons. First, tweets are short and full of informal expressions and abbreviations, which make the word-level features (e.g. unigrams) not only cannot capture the informative context from tweets but also cannot deliver reliable features for classification algorithms. We are thus motivated to develop other representation of tweets for classification. Second, the characteristics of tweets make the feature space very sparse. Moreover, only limited words and phrases can indicate the suggestions, which make the sparsity problem more serious. This leads to unsatisfied performance of the traditional classification methods (e.g. Support Vector Machine (Cortes and Vapnik 1995)). To overcome the huge sparsity, we employ the feature grouping method and more robust model. Third, the imbalanced characteristics of suggestion tweets harm the performances of classifier on minority class (i.e. suggestion tweets). Our observation on a human annotated data set shows that only 7.93% tweets are labeled as suggestion tweets, which indicates that the classifier should be adaptive in imbalance classification.

To this end, we develop the classification framework on Factorization Machines (Rendle 2012), which is effective and efficient especially in classification tasks with feature sparsity settings. We propose to extract suggestion templates automatically, which captures the common expression templates for suggestions, from external resources (e.g. the message board of official sites for a product, where the company would like to collect comments and suggestions from customers) to develop more informative representations of feature for suggestion classification. However, such kind of feature is very effective but might be highly sparse, which in turn leads to ineffective performance in traditional clas-

[*]Contribution during internship at Microsoft Research Asia.

[1]http://blog.twitter.com/2012/03/twitter-turns-six.html

sification models. Factorization Machines based classifier, as shown in our experiments, works effectively in such settings. Furthermore, in order to overcome the data imbalance problem, we introduce cost-sensitive learning techniques in Factorization Machines. We have conducted extensive experiments on a manually annotated real-life data set. Experimental results show that the proposed approach significantly improves the baseline approach, and yields the precision of 71.06% and recall of 67.86%.

The major contributions of this paper are three-fold.

- We propose the task of suggestion analysis, which is invaluable for business intelligence and supporting decision making, but not well studied previously;
- We provide a comprehensive investigation of the challenges of the task in the context of Twitter, and propose to build the classification framework on Factorization Machines with automatically extracted template features as well as strategies to overcome data imbalance and feature sparsity;
- We present empirical studies on a real-life data set to evaluate the effectiveness of different approaches, and we further have a brief discussion to explain why the Factorization Machines perform better.

Related work

In this section, we briefly review the work which is close to this paper. We organize the related work from the perspectives of the tasks we studied as well as the algorithms we employed.

From sentiment to suggestion

Sentiment analysis (Pang and Lee 2008) has been extensively studied on user-generated-content (i.e. UGC), such as movie reviews, customer feedback and social media (e.g. tweets), in the community of natural language processing. In recent years, there is research work focusing on other interesting topics other than sentiment from UGC. (Kanayama and Nasukawa 2008; Goldberg et al. 2009) tackle textual demand analysis and “wish detector”, the task of capturing what people want or need, rather than identifying what they like or dislike, on which much conventional work has focused. Demand analysis complements traditional sentiment analysis and is valuable for collecting business intelligence and insights into the world’s wants and desires. Unlike previous tasks, we mainly focus on suggestions analysis which aims at automatically extracting suggestions or ideas of different users in Twitter. Furthermore, we do not make use of the Part-Of-Speech and syntactic parse tree information owing to that there are many noises in tweets and most of them are casually written, so it is difficult to parse them accurately and reliably.

Feature sparsity in classification

The dimension of feature space in text classification is high due to the huge number of words in natural language. The sparsity problem is more critical in the short text owing to the short length and diversity of language usage. The methods to overcome the sparsity problem can be mainly divided into three categories. The first one is enriching document feature space using background data. (Hu et al. 2009; Gabrilovich and Markovitch 2005; 2006) propose to reconstruct the corresponding feature space with the integration of multiple semantic knowledge bases (e.g., Wikipedia and WordNet), they enrich document representation through automatic use of a vast compendium of human knowledge. In (Phan, Nguyen, and Horiguchi 2008), the authors collect a large-scale external data collection called ”universal dataset”, and then build a classifier on both a small set of labeled training data and a rich set of hidden topics discovered from that data collection. The second one is feature reduction method which makes use of semantic information or high-level features such as topics to reduce the feature space. (Lacoste-Julien, Sha, and Jordan 2008) present DiscLDA which takes side information into account in finding a reduced dimensionality representation, it uses topic mixture proportions as a new representation of documents. (Caragea, Silvescu, and Mitra 2012) propose to combine two types of feature clustering namely hashing and abstraction based on hierarchical agglomerative clustering, it uses significantly smaller number of features and gives similar performance when compared with the bag-of-words and n-gram approaches. (Saif, He, and Alani 2012) use semantically hidden concepts, latent topics and the associated topic sentiment from tweets to improve short text sentiment classification accuracy. The third one is employing semi-supervised learning (Nigam et al. 2000), transfer learning (Jin et al. 2011) and other learning methods to better use unlabeled data instances and auxiliary long texts. In this work, we use the Factorization Machines which proposed in (Rendle 2012) as well as feature grouping method to relieve the feature sparsity problem. Factorization Machines are able to estimate parameters under huge sparsity since they bring the benefits of factorization models.

Imbalance classification

In many real-world applications, there may be a big gap between the sample numbers of different classes, and our interest is usually on the minor class which has less number of samples. One common way to solve the imbalance problem is using cost-sensitive learning. It can be classified into two categories. The first one is algorithm specific approaches, they modify different algorithms directly (Ling et al. 2004; Akbani, Kwek, and Japkowicz 2004). The second one is cost-sensitive meta-learning methods, and they can be further categorized into Sampling (Drummond and Holte 2003; Estabrooks, Jo, and Japkowicz 2004) and Thresholding (Elkan 2001; Zhou and Liu 2010). Cost-sensitive meta-learning methods convert existing cost-insensitive classifiers into cost-sensitive ones without modifying them, and this is the main difference between algorithm specific approaches and cost-sensitive meta-learning methods. Sampling adjusts the distribution of training data by Undersampling and Oversampling. Thresholding searches a threshold which can achieve minimal total misclassification cost for classification, and it is applicable to any existing classifiers which can produce probability predictions without modifying them.

Suggestion classification

We start this section by an introduction to the task of suggestion classification from tweets. We aim to classify a tweet into suggestion (positive class) or non-suggestion(negative class). A tweet is referred as a suggestion if the tweet is talking about suggestions and proposals towards a target (usually a company, product or a person), and put forward some ideas or plans for someone to think about. We show a suggestion tweet as follows.

> I have an idea for "*Microsoft*". Make an app on WP7 that can remote login into your desktop and u can do everything.

The author explicitly expresses the suggestion, namely "*Make an app on WP7 that can remote login into your desktop*", towards the target (i.e. *Microsoft*). The work presented in this paper is to automatically detect and classify such kind of tweets in Twitter.

We develop the suggestion classification framework based on Factorization Machines (Rendle 2012). In this section, we first present the Factorization Machines, and then present feature generation with the focus on automatically expanded syntactic templates. Finally, we present the cost-sensitive Factorization Machines to overcome the imbalance problem in suggestion classification.

Factorization Machines

Support Vector Machines (SVMs) often fail when the feature space is sparse. In contrast to SVMs, Factorization Machines (FM) (Rendle 2012) model the interactions between variables using factorized parameters. Let $\mathbb{X} \in \mathbb{R}^{n \times p}$ denote the feature matrix of n instances for prediction in the training set, where the ith ($1 \leq i \leq n$) row $\mathbf{x}_i \in \mathbb{R}^p$ of $\mathbb{X}$ describes the the corresponding feature vector of the ith instance. Also, $\mathcal{Y} \in \mathbb{R}$ denotes the predication labels for the n instances, where $\mathcal{Y}_i$ ($1 \leq i \leq n$) describes the predication label for the ith instance. Factorization machines (FM) (Rendle 2012) model all nested interactions up to order d between the p input variables in $\mathbf{x}$ using factorized interaction parameters. The factorization machines (FM) model of order $d = 2$ is defined as,

$$\widehat{y}(\mathbf{x}) := w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} x_j x_{j'} \sum_{f=1}^{k} v_{j,f} v_{j',f} \quad (1)$$

where k is the dimensionality of the factorization and the model parameters $\Theta = \{w_0, w_1, \ldots, w_p, v_{1,1}, \ldots, v_{p,k}\}$ are, $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^p$, $\mathbf{v} \in \mathbb{R}^{p \times k}$. The first part of the FM model contains the unary interactions of each input variable x_j with the prediction exactly as in a linear regression model. The second part with the two nested sums contains all pairwise interactions of input variables, that is, $x_j x_{j'}$. The important difference to standard polynomial regression is that the effect of the interaction is not modeled by an independent parameter $w_{j,j'}$ but with a factorized parametrization $w_{j,j'} = \mathbf{v}_j{}^T \cdot \mathbf{v}_{j'} = \sum_{f=1}^{k} v_{j,f} v_{j',f}$ which corresponds to the assumption that the effect of pairwise interactions has a low rank. This allows FMs to estimate reliable parameters even in highly sparse data where traditional models fail.

To learn the parameters Θ of model for binary classification tasks, we define our objective function as,

$$\begin{aligned} &OPT(S, \lambda) \\ &:= \underset{\Theta}{argmin} \left(- \sum_{(\mathbf{x},y) \in S} \ln \sigma \left(\widehat{y}(\mathbf{x}|\Theta) \cdot y \right) + \sum_{\theta \in \Theta} \lambda_\theta \theta^2 \right) \end{aligned} \quad (2)$$

where $\widehat{y}(\mathbf{x}|\Theta)$ is the regression value, y is the label (+1/-1), $\sigma(x) = \frac{1}{1+e^{-x}}$ is a logistic function and $\lambda_\theta \in \mathbb{R}^+$ is the regularization value of parameter θ. To avoid overfitting, L2-regularization is used in the objective function. This item limits the number of model parameters. With this definition, we can employ several optimization algorithms (e.g., Stochastic Gradient Descent) to minimize the losses over data.

The most simple kernel in SVMs is linear kernel, and it is identical to a FM of degree $d = 1$ (without intersection items). The complex kernels allow SVMs to model interactions between features by learning a weight for each $x_i x_j$, but it is difficult to estimate reliable parameters in the scenarios where data is hugely sparse. This is the reason why linear kernel performs well in short text (e.g., tweets) classification problem. In contrast to SVMs, the FMs' factorized parameters ($\mathbf{v}_i{}^T \cdot \mathbf{v}_j$, $\mathbf{v}_i{}^T \cdot \mathbf{v}_l$) overlap and share parameters($\mathbf{v_i}$). Thus the latent relations can be used by factorization to relieve the feature sparsity. Therefore it is more robust to data sparsity than SVM.

Factorization Machines combine the advantages of SVMs with factorization models. Thus, they are able to estimate interactions even in the problems with huge sparsity. But unlike other state-of-the-art factorization models, FMs are applicable for general prediction tasks working with any real valued feature vectors of the input data. FMs also can be optimized in the primal directly and have linear complexity, hence FMs can deal with large datasets. Factorization Machines are originally used in collaborative filtering, but it is a general predictor which can be used for classification. To the best of our knowledge, we are the first to introduce FMs to the Natural Language Processing research community. The feature space in text classification is very sparse, especially the tweets due to the short string length and diversity of word usage. FMs break the independence of the interaction parameters by factorizing them, hence we can better utilize the data to learn the interaction or similarity between features even in the sparse settings.

Features

The set of features mainly contain three parts, which are unigram (i.e. bag-of-word), #hashtag, and template features.

Unigram features: The bag-of-words features capture clues between the lines, these content based features are simple but effective in most of settings. Therefore, we use them as the baseline.

#hashtag features: The second set of features are #hashtag features, which are unique and informative. We observe that some #hashtags are used to make a suggestion. There are many other social-related factors such as friends structure, retweet (i.e. RT) and mention (i.e. @) action, but they do not make much sense in the task of suggestion classification since this is mainly a content-based problem.

Template features: We notice that although the word usage of detail content is diversified, there are some indicator words which are much more discriminative and less sparse. So we employ these indicator words as suggestion templates which can strongly indicate whether the tweets are suggestions or not. Manually writing templates can achieve good precision but a low recall since it is difficult to write most of the corresponding templates even for native speakers. In our work, we extract the templates from background data automatically. We use the feedback data crawled from Windows Phone's official web site as background data set. Users can post their suggestions about new features, applications and advertising strategies in the web sites, and the other users can vote and comment for them. First we tokenize the suggestions to sentences, and treat each of them as a word sequence. To extract domain-independent templates from these feedbacks, we filter the domain-related words (e.g. "windows", "phone" and "lumia") in the feedback data set. We employ the PrefixSpan (Prefix-projected Sequential Pattern mining) (Pei et al. 2004) algorithm which is used to find frequent sequential patterns in sequential data. Through setting the minimum and maximum length as well as the minimum support (occurrence frequency), we can get sequential patterns whose occurrence number is no less than minimum support. Then we remove the patterns which only contain stop words since they are useless in the suggestion classification. After these steps, we can get the suggestions templates. The outline of this method can be found in Algorithm 1.

Algorithm 1 Automatic Suggestion Template Extraction

```
Input:
    Feedback data set;
    Parameters: minimum length, maximum length, minimum support;
Output:
    Suggestion templates;
 1: Tokenize the feedback data set into sentences
 2: Filter domain-related words //"lumia", "windows" as examples in our experiments
 3: Remove sentences less than 4 words in length
 4: template_list ← PrefixSpan(minimum length, maximum length, minimum support)
 5: for template_it in template_list do
 6:     if (template_it are all stopwords) then
 7:         Remove template_it from template_list
 8:     end if
 9: end for
10: return template_list;
```

The length of tweet is very short (no more than 140 characters) and the word usage is hugely diversified, which makes the feature space very sparse for classification. To overcome sparsity, we group some of features which have the same semantic information and syntactic roles to one dimension. In the manually feature grouping process, we select 45 high frequency template features learned from feedback data, and group the similar ones. For example, "I would like" and "I would love" have the similar meaning and usage, we can use them interchangeably without changing the meaning of original sentence. By grouping them, there are more samples for this dimension of feature space and hence we can estimate more reliable parameters.

Imbalance classification with Factorization Machines

By experiment, we find that FMs are sensitive to the imbalance of training data. So we use both Oversampling and Thresholding method to make FMs cost-sensitive.

Oversampling aims at redistributing the training data set before training the classifier, such that the number of positive items and negative items can be same. We also try Undersampling, but its result is not as good as Oversampling since it does not fully use the information.

Thresholding method can make FMs cost-sensitive without modifying the original models. After we get the prediction probability $p \in [0, 1]$, we compare it with the threshold τ which we set. If $p > \tau$, then this item is positive; and is negative otherwise. In practice we can get τ by greedy searching and cross-validation instead of setting $\tau = 0.5$. (Sheng and Ling 2006) indicate that Thresholding has the least sensitivity on the misclassification cost ratio and it almost always produces the lowest misclassification cost comparing with other existing cost-sensitive meta-learning methods.

We use the combination of these two methods in our work. Oversampling lies in training data preprocessing, and Thresholding lies in the prediction step. They all need not to modify the original model and learning process, so the framework is very flexible as well as easy to implement.

Experimental study

Dataset Description

In order to build the suggestion classification data set [2], we use the Twitter API to randomly extract the tweets which are all talking about *Windows Phone 7* during September 2010 to April 2012. Then, two separate persons annotate each tweet with a "1" if it is a suggestion tweet, and with a "-1" otherwise. We only keep 3,000 tweets which are annotated with the same labels by annotators. In total, we get 238 (/3,000≈7.93%) tweets which are annotated as suggestions. Unlike most of data sets, we retain the original data distribution and regard it as a imbalanced problem. This ensures our method is practical in the real-world setting.

In the following sections, we present the experiment and comparison results of the proposed approach and baseline methods using different feature sets. We also present extensive studies of the automatic template extraction employed in our experiments.

[2]http://goo.gl/hXtRv

Evaluation of suggestion classification

We compare different methods and features in this section to evaluate the effectiveness of our methods. In the following experiments, we report results of 5-fold cross-validation.

SVM with bag-of-words(SVM1). We use bag-of-word features as the baseline feature set, which is the most widely-used features in traditional text classification tasks . This is the baseline among all the methods. LIBSVM (Chang and Lin 2011) is used to train a SVM classifier with linear kernel. The SVMs are widely used for text classification problems and can achieve good results in most of settings. We use the default parameters to train the model. Due to Twitter's characteristics, we employ the tokenizer provided in (O'Connor, Krieger, and Ahn 2010) instead of standard ones. This tokenizer is developed specially for Twitter, and it treats #hashtags, @-replies, abbreviations, strings of punctuation, emoticons and unicode glyphs(e.g. musical notes) as tokens.

SVM with bag-of-words + cost-sensitive (SVM2). To overcome the imbalance between positive class (i.e. suggestion tweets) and negative class (i.e. non-suggestion tweets), we use both Oversampling and Weighting methods in our experiments. We randomly oversample the training data to make the number of instances in positive class as many as in negative class. Weighting is a method adjusting the cost of different class, which is belong to algorithm specific approaches. We can tune the parameter w_i in LIBSVM to implement Weighting, which is used to adjust the cost of different classes.

SVM with all features (SVM3). Besides the bag-of-word features in SVM1, we add templates features and #hashtag features to the feature set. As mentioned in the previous section, some of template features are grouped to one dimension by their semantic similarity to relive feature sparsity.

SVM with all features + cost-sensitive (SVM4). Compare with SVM2, the difference is that we add template features as well as #hashtag features in this setting. We also use both Oversampling and Weighting methods to overcome data imbalance, and the parameters are all same as in SVM2.

SVM with all features + cost-sensitive + polynomial kernel (SVM5). We employ the same settings as in SVM4, except using the polynomial kernel instead of linear kernel. The form of polynomial kernel is $K(\mathbf{u}, \mathbf{v}) = \left(\gamma * \mathbf{u}^T \cdot \mathbf{v} + \beta\right)^{deg}$, where we set $deg = 2$, $\beta = 1$ and default γ in LIBSVM.

FM with bag-of-words (FM1). By employing libFM [3], we train a 2-dimension FM for the suggestion classification task. We use Markov Chain Monte Carlo (MCMC) learning method, as MCMC is easy to implement and use (e.g. no learning rate, no regularization) (Salakhutdinov and Mnih 2008). In our experiment, we set the number of iteration as 4,000 to make sure the learning of Factorization Machines is convergent. We use bias and 1-way interaction items in our model, and the *stdev* for initialization of 2-way factors is set to 0.1, using the default settings. The bag-of-word features are used in this setting.

FM with bag-of-words + cost-sensitive (FM2). Compare with FM1, we randomly oversample the training data to make positive class as many as negative class. Thresholding is also used, it can convert any existing cost-insensitive algorithms into cost-sensitive ones. We set the threshold to 0.40 in terms of cross-validation. Instead of comparing the prediction probability with 0.5, we need to compare it with the threshold (here is 0.40). We classify a tweet into positive (i.e. suggestion) class if the probability is larger than the threshold, and to the negative (non-suggestion) class, otherwise.

FM with all features (FM3). The same parameters are used as in FM1, and the difference is that we add template features and #hashtag features in this setting.

FM with all features + cost-sensitive (FM4). We use both Oversampling and Thresholding methods to overcome data imbalance, and use the same parameters as in FM2. The only difference with FM2 is that we also use the template features and #hashtag features in addition to the bag-of-words features.

Table. 1 shows the classification results of the different classifiers. As the suggestion classification task is a highly imbalanced classification problem, we are more interested in the performance of suggestion tweets rather than the others. In other words, the evaluation scores of positive (suggestion) class are more important to us than negative (non-suggestion) class and overall accuracy.

First of all, give a glance at the experiment results of baseline (SVM1), we find that the SVM with bag-of-words feature can only achieve 56.96% precision and 52.18% recall of positive class. Nevertheless, the performances on negative class and overall accuracy are much better. Compared with the results of SVM1, the FM1's recall and F-1 score of positive class is low, whereas the precision is high, which indicates that the FMs are more sensitive to the imbalance of training data than SVMs. Imbalance also harms the performance of FM3. Therefore, it is necessary to introduce cost-sensitive learning methods in FMs. After using these techniques in FM2, there are significant gains of F-1 scores on positive class. Notably, the F-1 score of FM2 increases by 22.75% than FM1. At the same time, we do not harm the evaluation scores of negative class too much, and the overall accuracy increases by 5.28%. This illustrates the usefulness of the combination of Oversampling and Thresholding learning methods in FMs for dealing with imbalance classification. To compare with the SVMs, we also apply cost-sensitive learning methods in SVM2. The F-1 score of positive class in SVM2 rises by 2.56% than SVM1, and we also see an improvement of the overall accuracy. By employing the combination method, we can improve the recall of minority class and the overall accuracy.

We also evaluate the performance of template features and #hashtag features. Without taking the imbalance problem into account in SVM3 and FM3, these two sets of features respectively achieve 7.65% and 19.10% improvement (F-1 score of positive class) than SVM1 and FM1. Specifically, the recall of positive class in FM3 rises by 18.52%. After considering the imbalance problem, we can still benefit from these features. Compare the SVM4 and FM4 with SVM2 and FM2, their performances are all improved significantly. This indicates the effectiveness of templates which

[3]http://libfm.org

Method	Suggestion Tweets			Non-suggestion Tweets			Acc.
	Prec.	Rec.	F-1	Prec.	Rec.	F-1	
SVM1	56.96	52.18	54.47	95.82	96.53	96.17	92.33
SVM2	56.79	57.27	57.03	96.33	96.27	96.30	93.21
SVM3	63.68	60.63	62.12	96.66	97.04	96.85	93.78
SVM4	63.76	65.35	64.55	97.02	96.85	96.93	94.49
SVM5	62.25	64.42	63.32	96.90	96.59	96.74	94.30
FM1	85.74	24.48	38.09	93.96	99.63	96.71	88.40
FM2	60.89	60.79	60.84	96.54	96.57	96.55	93.68
FM3	85.37	43.00	57.19	95.37	99.34	97.31	91.18
FM4	71.06	67.86	**69.42**	97.21	97.46	97.33	**94.86**

Table 1: Results of Suggestion Classification. The cost-sensitive FM with all features(FM4) achieves best performance.

we extract from background data (Windows Phone's feedback data) automatically.

In the SVM4 and FM4, we use all the features and cost-sensitive learning methods in the FMs and SVMs, both of them achieve better results than only using the unigram features. According to overall accuracies and F-1 scores of both classes in Table. 1, we find that FM4 performs best among different classifiers. For positive class, the FM4 achieves 14.95% improvement of F-1 score over baseline. Compare with SVM4, the FM4 can bring 4.87% improvement of F-1 score (positive class) in the same setting.

Finally, we compare the FMs with non-linear SVMs which have polynomial kernels. The performance of SVM5 is worse than SVM4 and FM4, this makes sense due to the high dimension of feature vector and the huge sparsity. In our experiment, we set the $deg = 2$ and $\beta = 1$ so that the forms of non-linear SVMs are similar to FMs (degree=2). For this case, polynomial kernel maps the original vector $\mathbf{x}$ to $(1, \sqrt{2}x_1, ..., \sqrt{2}x_n, {x_1}^2, ..., {x_n}^2, \sqrt{2}x_1x_2, ..., \sqrt{2}x_1x_n, \sqrt{2}x_2x_3, ..., \sqrt{2}x_{n-1}x_n)$. The model equation of primal form for SVMs is defined as,

$$\widehat{y}(\mathbf{x}) := w_0 + \sqrt{2}\sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} w_{j,j}{x_j}^2 + \sqrt{2}\sum_{j=1}^{p}\sum_{j'=j+1}^{p} w_{j,j'}x_j x_{j'} \qquad (3)$$

where the model parameters $\Theta = \{w_0, w_1, \ldots, w_p, w_{1,1}, w_{1,2}, \ldots, w_{p,p}\}$ are, $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^p$, $\mathbf{W} \in \mathbb{R}^{p \times p}$. The main difference with FMs is that parameters $w_{j,j'}$ are independent between each other. Consider the x_i and x_j co-occur in the same feature vector $\mathbf{x}$, as a result of the huge sparsity, we may have no enough (too few or even no) cases which they co-occur in training data. It is difficult to estimate good parameter $w_{i,j}$ for SVMs in this setting. However, the FMs take advantages from the ideas of factorization models, and the intersection parameters have overlaps between them which relieve the sparse problem.

We further investigate the reason why FM performs better than SVM in our task. Generally, the difference between SVM and FM is that FM is effective to model the correlations among the features using the intersection vectors even in the feature sparsity settings (Rendle 2012). The suggestion classification task presented in this paper falls exactly into this configuration. The intersection vector $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,k})^T$ can be regarded as a latent vector capturing the latent correlation among the features, which provides useful information for classification especially in the feature sparsity settings (Rendle 2012).

Conclusion

In this paper, we propose Factorization Machines based classification framework to automatically detect and classify users' suggestions from tweets. We investigate different strategies to overcome data imbalance and feature sparsity in developing the classifier. Particularly, we introduce cost-sensitive learning techniques in Factorization Machines to overcome data imbalance, and the automatically extracted suggestion templates are used as better representation of tweets for classification. Experimental results show that the proposed approach significantly improves the baseline approach.

There are many interesting directions for further research studies. For example, the detection of targets in suggestions is still an open question. Meanwhile, the automatic summarization of suggestions will be pretty useful for business intelligence. In addition, we are also interested in applying Factorization Machines in other tasks with similar settings, such as sentiment classification, etc.

Acknowledgments

We gratefully acknowledge helpful discussions with Dr. Steffen Rendle. This research was partly supported by Innovation Fund Project for Graduate Students of Beihang University (YCSJ-02-01), the National 863 Program of China (No. 2012AA011005), and Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20111102110019).

References

Akbani, R.; Kwek, S.; and Japkowicz, N. 2004. Applying support vector machines to imbalanced datasets. In *In Proceedings of the 15th European Conference on Machine Learning (ECML)*, 39–50.

Caragea, C.; Silvescu, A.; and Mitra, P. 2012. Combining hashing and abstraction in sparse high dimensional feature spaces. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 3–9.

Chang, C.-C., and Lin, C.-J. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2(3):27:1–27:27.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Mach. Learn.* 20(3):273–297.

Drummond, C., and Holte, R. C. 2003. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. 1–8.

Elkan, C. 2001. The foundations of cost-sensitive learning. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 973–978.

Estabrooks, A.; Jo, T.; and Japkowicz, N. 2004. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence* 18–36.

Gabrilovich, E., and Markovitch, S. 2005. Feature generation for text categorization using world knowledge. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, 1048–1053. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Gabrilovich, E., and Markovitch, S. 2006. Overcoming the brittleness bottleneck using wikipedia: enhancing text categorization with encyclopedic knowledge. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, 1301–1306. AAAI Press.

Goldberg, A. B.; Fillmore, N.; Andrzejewski, D.; Xu, Z.; Gibson, B. R.; and Zhu, X. 2009. May all your wishes come true: A study of wishes and how to recognize them. In *HLT-NAACL*, 263–271.

Hu, X.; Sun, N.; Zhang, C.; and Chua, T.-S. 2009. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, 919–928. New York, NY, USA: ACM.

Jin, O.; Liu, N. N.; Zhao, K.; Yu, Y.; and Yang, Q. 2011. Transferring topical knowledge from auxiliary long texts for short text clustering. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, 775–784. New York, NY, USA: ACM.

Kanayama, H., and Nasukawa, T. 2008. Textual demand analysis: detection of users' wants and needs from opinions. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, 409–416. Stroudsburg, PA, USA: Association for Computational Linguistics.

Lacoste-Julien, S.; Sha, F.; and Jordan, M. I. 2008. Disclda: Discriminative learning for dimensionality reduction and classification. In *NIPS*.

Ling, C. X.; Yang, Q.; Wang, J.; and Zhang, S. 2004. Decision trees with minimal costs. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, 69–. New York, NY, USA: ACM.

Nigam, K.; McCallum, A. K.; Thrun, S.; and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using em. *Mach. Learn.* 39(2-3):103–134.

O'Connor, B.; Krieger, M.; and Ahn, D. 2010. Tweetmotif: Exploratory search and topic summarization for twitter. In *ICWSM'10*.

Pang, B., and Lee, L. 2008. *Opinion Mining and Sentiment Analysis*. Now Publishers Inc.

Pei, J.; Han, J.; Mortazavi-Asl, B.; Wang, J.; Pinto, H.; Chen, Q.; Dayal, U.; and Hsu, M.-C. 2004. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. on Knowl. and Data Eng.* 16(11):1424–1440.

Phan, X.-H.; Nguyen, L.-M.; and Horiguchi, S. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, 91–100. New York, NY, USA: ACM.

Rendle, S. 2012. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3(3):57:1–57:22.

Saif, H.; He, Y.; and Alani, H. 2012. Alleviating data sparsity for twitter sentiment analysis. In *NSM*, 2–9.

Salakhutdinov, R., and Mnih, A. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, 880–887. New York, NY, USA: ACM.

Sheng, V. S., and Ling, C. X. 2006. Thresholding for making classifiers cost-sensitive. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, AAAI'06, 476–481. AAAI Press.

Zhou, Z.-H., and Liu, X.-Y. 2010. On multi-class cost-sensitive learning. *Computational Intelligence* 26(3):232–257.