A LDA-SVM Active Learning Framework For Web Service Classification

Xumin Liu[1], Shaleen Agarwal[1],Chen Ding[2], Qi Yu[1]
[1] *Golisano College of Computing and Information Science, Rochester Institute of Technology, USA*
[2] *Department of Computer Science, Ryerson University, Canada*

***Abstract*—Classifying web services and labeling them based on their functional features have played a major role in several fundamental service management tasks, such as service discovery, selection, ranking, and recommendation. Existing approaches leverage text mining techniques and follow a supervised learning process, which involves building a classifier from a training set of services and applying the classifier to other services. This process requires intensive human effort on labeling services in the training set. In this paper, we propose to leverage the idea of pool-based active learning to realize a scalable service classification approach. Instead of manually labeling a large number of services to construct a complete training set, the approach starts with a base classifier with a small set of training set and iteratively asks for the labels of the most informative services outside of the initial training set. By doing this, the classifier can achieve comparable accuracy compared to traditional classification method with much smaller size of training set. We use SVM as the base classifier due to its effectiveness in text classification. We also incorporate probabilistic topic models to address the issues caused by sparse term vectors generated from service descriptions and reduce the dimensions to improve the efficiency. We conducted a comprehensive experimental study on real-world service data to demonstrate the effectiveness of the proposed approach.**

***Keywords*-web service classification; active learning; support vector machine; topic modeling**

I. Introduction

Driven by the benefits of web service technologies, the variety and number of web services available on the web have been significantly increased. This has brought great potentials for users to leverage online programming resources (e.g., computing utilities, data, and code), which boost software reuse and form loosely-coupled integrations among distributed software components. Meanwhile, this poses great challenges to many service management tasks centering around service discovery, such as service composition, selection, ranking, and recommendation. The standard way of service discovery is through UDDI registry, which is a web service itself and provides APIs for service providers to publish their service descriptions and query APIs for service consumers to search services based on searching criteria. The limitations of UDDI have been observed over years due to two major reasons. (1) it highly relies on service providers to provide and maintain service descriptions. Such a burden has greatly discouraged service providers to use UDDI. (2) Although its `tmodel` mechanisms allow extensible service descriptions, the search process is still keyword-based. This causes poor query result with low precision and recall rate. Early research efforts addressing these issues include web service search engines and semantic web services. Web service search engines automatically crawl service information from UDDI registry or web pages containing WSDL links to create a service repository, and provide query interface for users to look for their desirable services [1]. Semantic web service technologies, such as WSDL-S, OWL-S, and WSMO, are to enrich service descriptions with semantics to conquer the limitations of the keyword-based search [6]. However, they are only suitable for a small-scale service space due to the high cost on matchmaking and semantic reasoning.

In recent years, people have started tackling the service discovery challenges from a different angle. Related research efforts leverage text mining technologies to process and analyze the textual service descriptions to assist the discovery process. They mainly follow two directions: web service clustering and classification. Web service clustering is to generate homogenous service groups out of a large-scale service space, where services providing similar functionality are grouped together [7], [5], [13]. They propose different service modeling methods (e.g., TF/IDF vectors or topic modelings) and similarity measures, and adopt different data clustering algorithms (e.g., KMeans and Singular Value Decomposition or SVD Co-clustering) to cluster web services. This follows the same idea of clustering to support web search engines. That is, once a query is submitted, the query will be directed to the related service groups and only the services in that group will be examined if there is a match. Such service groups can not only narrow down the searching space, which significantly improve the efficiency, but also improve the accuracy of searching result since only relevant services will be returned. There are also some attempts on web service classification, i.e., labeling web services based on their functionality [8], [11]. They apply different data classification techniques, such as decision tree and support vector machine (SVM) to train a classifier to label a web service. Such service labels can be used to efficiently and effectively identify relevant services for a query. As web services are labeled as one of the predefined classes, such as travel, education, and music, the semantic meaning of these classes provide more useful guidance on service discovery compared to the result of service clustering, where only service groups are formed. Moreover, the trained classifier

978-1-5090-2675-3/16 $31.00 © 2016 IEEE
DOI 10.1109/ICWS.2016.16

can be applied on new services, while the result of service clustering cannot. This makes the benefits of service classification more appealing than clustering. However, the current service classification solutions have a major limitation since they always follow a supervised learning process. That is, the training of a classifier is performed on a large training set, where the labels of web services are known. This requires significant human effort to read those service descriptions and manually assign labels to them.

In this paper, we propose to leverage the idea of pool-based active learning to reduce the cost of manually labeling services for training a classifier [12]. The training process starts with a base classifier trained from a small size of labelled web services and then iteratively asks for the labels of the most informative services outside of the initial training set. The selection of those services is based on minimizing the feasible solution space (referred to as the version space). The process stops when the classifier achieves a desirable accuracy. We choose to use SVM as the base classifier due to its effectiveness in text classification. The traditional way of text classification is to model each document as a term vector, where each entry represents the frequency (or tf-idf) of the corresponding term. Considering that most service descriptions are comprised of very limited terms, we propose to adopt topic modeling technologies to model each service description as a vector in the topic space, where each entry corresponds to the proportion of the corresponding topics. This can not only effectively address sparse term vectors issue but also reduce data dimensions to improve the efficiency.

The remainder of the paper is organized as follows. Section 2 gives an overview of the proposed active learning service classification framework. Section 3 presents the details of the proposed LDA-SVM active learning service classification approach. Section 4 describes the experimental results over real-world service description data to demonstrate the effectiveness our approach. Section 5 reviews related works and Section 6 concludes the paper.

II. The Active Learning based Service Classification Framework

Labeling of web services is the task of automatically categorizing web services into a subset of predefined classes. Let us denote the service descriptions in the training dataset as $d_1 ... d_D$ and let the number of classes be $1...J$. The goal is to construct a classifier, which is a function $y(a)$ that is able to assign a label c_j to a new service a, where $c_j > 0$ denotes that a belongs to the j-th class and $c_j < 0$ otherwise.

We use the idea of SVM-based active learning for the classification of web services. We consider SVM as our base classifier because it has been successfully applied to various text classification tasks. Besides using the standard term based vector space representation of services, we also exploit Latent Dirichlet Allocation (LDA) based topic models [3] to extract high-level topics, which are expected to capture the underlying semantics of the services. After the topics are extracted, each service can be represented as a vector in the topic space. Combining LDA based topics models with SVM offers two major advantages, which particularly benefit service classification tasks. First, as most services are described using very limited terms, the term based service representation will lead to very sparse term vectors. Meanwhile, a SVM classifier essentially assigns class labels to a new service based on their similarity with the discovered support vectors (through a kernel function) from different classes in the training data. As the sparse term vectors may share very few entries, their similarity will be low under such a representation. As a result, the kernel function will output a value close to zero, which leads to low classification accuracy and large uncertainty. Intuitively, since $y(a) = 0$ is the decision boundary, a close to zero y value implies that the data point is very close to the decision boundary and hence uncertainty of the classification result is large. The second benefit is that representing services in the topic space achieves the effect the dimensionality reduction as the size of the topic space (say K) is typically much smaller than the size of the term vocabulary (say V). This will help reduce the computational cost. Since the SVM classifier works most naturally with the two-class problems, we adopt the one-versus-the-rest scheme and train the classifier for each possible class against the rest of the classes. We then aggregate the output of all the binary classifiers to determine the final labels of the given web service.

To reduce the cost of manually labeling services to build the training set, we integrate the pool-based active learning strategy with the above described LDA-SVM classifier. The idea is to assume that we are given a pool of partial labeled data A^L and the remaining data is unlabeled A^U. Initially, an SVM classifier is trained using A^L. Depending upon this classifier, the active learner will randomly select a sample from A^U and queries for the true labels of the sample data. Then, the newly labeled data is added taken from A^U and put into A^L. The entire training and labeling process runs iteratively after a threshold when the classifier reaches a particular and sufficient classification accuracy. One of the major problem with active learning is how to choose the most informative web services for labeling. This problem is also know as selection sample strategy.

In sum, the research problem is to train a LDA-SVM web service classifier by using a minimum number of labeled services with a principled selection sample query strategy. The proposed LDA-SVM active learning framework has the potential to extend service classification to a very large-scale service repository, which meets the demand of organizing the fast increasing service and API space on the web. In what follows, we describe the key components of the framework:

1) The framework applies standard text processing techniques, including tokenization, stop word removal, and

stemming, to extract important features (in the form of terms) from service description files.
2) LDA topic modeling is applied to extract a number of topics, which are expected to align well with the key functionalities provided by those services.
3) We manually label a small subset of the services and this labeled dataset will act as our base dataset.
4) We train a base LDA-SVM classifier using the initial training set.
5) We apply active learning to choose the most informative services from the unlabeled ones to manually assign class labels and refine the base classifier. This step continues until the classification accuracy over a validation set is improved to a satisfactory threshold.

III. LDA-SVM Active Learning for Service Classification

In this section, we start by presenting how to compute the topic based representation of service descriptions. We then describe how to integrate the topic based service representation into a SVM to achieve a LDA-SVM service classifier. We present the pool-based sampling and sample selection strategy with the LDA-SVM classifier in the end.

A. Computing Topic based Service Representation

We adopt Latent Dirichlet Allocation (LDA) [3], which a probabilistic topic model, to discover service topics, allowing us to represent service descriptions in the topic space. These topics are expected to capture the underlying semantics (or functionalities) of the service space. LDA will convert a service description d represented by a vector of terms into a mixture of multiple topics. The topic proportion in d is captured by parameter $\boldsymbol{\theta}_d$, which is a K dimensional vector with the k-th entry θ_{dk} denoting the probability of the k-th topic in service description d. K is the total number of topics and $\sum_k \theta_{dk} = 1$. Being a Bayesian model, LDA places a Dirichlet prior over $\boldsymbol{\theta}_d$'s: $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_K\}$ is the hyper parameter that can encode any prior knowledge about the model.

In LDA, each topic is denoted by a multinomial distribution over the term dictionary (i.e., the set of distinct terms extracted from all service descriptions). We use $\boldsymbol{\beta}_k$ to specify the term proportion of the k-th topic, where β_{kw} denotes the probability of term w in the k-th topic. Typically, only a small number of terms are likely to appear in a given topic where most other terms with probability close to zero to appear in the topic. This will allow us to represent a topic using its more probable terms. For example, in the service description dataset used in our experiments, we discover topics such as *Education* with top terms including college, student, admission, etc. and *Business* with top terms including money, stock, market, etc (see Table II for details).

Similar to the topic proportion parameters, LDA also places a Dirichlet prior over $\boldsymbol{\beta}_k$'s: $\boldsymbol{\beta}_k \sim \text{Dir}(\boldsymbol{\lambda})$, where $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, ..., \lambda_V\}$ is the hyper parameter. Table I summarizes the major notations used in this section.

Table I: Summary of Notations

Notation	Description
$\boldsymbol{\beta}_k$	Term proportion vector of topic k
$\boldsymbol{\theta}_d$	Topic proportion vector of service description d
$\boldsymbol{w}_d$	All the words in service description d
$\boldsymbol{z}_d$	Topic indicator vector of service description d
$\boldsymbol{\alpha}$	Hyper parameter of the Dirichlet prior over $\boldsymbol{\theta}_d$
$\boldsymbol{\lambda}$	Hyper parameter of the Dirichlet prior over $\boldsymbol{\beta}_k$
$\boldsymbol{\phi}^d$	Parameter of the variational multinomial distribution $q(\mathbf{z}_d)$
$\boldsymbol{\gamma}_d$	Parameter of the variational Dirichlet distribution $q(\boldsymbol{\theta}_d)$
$\boldsymbol{\pi}_k$	Parameter of the variational Dirichlet distribution $q(\boldsymbol{\beta}_k)$

Due to the generative nature of the LDA model, a simple process as described below can be used to generate the service descriptions:

1) For the k-th topic, draw $\boldsymbol{\beta}_k \sim \text{Dir}(\boldsymbol{\lambda})$, $k = 1..K$
2) For the d-th service description, draw $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha})$, $d = 1..D$
3) For the i-th word in service description d:
   a) Draw a topic indicator $z_{di} \sim \text{Multinomial}(\boldsymbol{\theta}_d)$
   b) Draw a word $w_{di} \sim \text{Multinomial}(\boldsymbol{\beta}_{z_{di}})$

The descriptive generative process can help us conveniently put down the joint distribution of both hidden ($\boldsymbol{\theta}_d, \mathbf{z}_d, \boldsymbol{\beta} = (\boldsymbol{\beta}_1..\boldsymbol{\beta}_K)$) and observed variables ($\mathbf{w}_d$, which denotes all the words in service description d) of the model given the hyper parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$:

$$p(\boldsymbol{\theta}_d, \mathbf{z}_d, \mathbf{w}_d, \boldsymbol{\beta}|\boldsymbol{\alpha}, \boldsymbol{\lambda}) = p(\boldsymbol{\beta}|\boldsymbol{\lambda})p(\boldsymbol{\theta}_m|\boldsymbol{\alpha}) \prod_{i=1}^{N_d} p(z_{di}|\boldsymbol{\theta}_d)p(w_{di}|z_{di}, \boldsymbol{\beta}) \quad (1)$$

By integrating the hidden variables, we can obtain the marginal distribution of a service description d:

$$p(\mathbf{w}_d|\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \int \int \sum_{\mathbf{z}_d} p((\boldsymbol{\theta}_d, \mathbf{z}_d, \boldsymbol{\beta}, \mathbf{w}_d|\boldsymbol{\alpha}, \boldsymbol{\lambda}) d\boldsymbol{\theta}_d d\boldsymbol{\beta} \quad (2)$$

Based on the observed service descriptions, $d_1, ..., d_D$, the model parameters can be estimated by maximizing the total log likelihood of all the descriptions:

$$\mathcal{L} = \sum_d \log p(\mathbf{w}_d|\boldsymbol{\alpha}, \boldsymbol{\lambda}), \quad (3)$$

Directly maximizing the log likelihood is challenging because the marginal distribution term in (2) is intractable due to the interaction between the hidden variables. Variational inference has been widely adopted, which instead maximizes a lower bound of the marginal distribution (or its log likelihood) by finding a variational distribution that approximates the true posterior, $p(\mathbf{z}_d, \boldsymbol{\theta}_d, \boldsymbol{\mu}|\mathbf{w}_d, |\boldsymbol{\alpha}, \boldsymbol{\lambda})$. The variational distribution usually assumes a fully factorized

form, $q(\mathbf{z}_d, \boldsymbol{\theta}_d, \boldsymbol{\beta}) = q(\mathbf{z}_d)q(\boldsymbol{\theta}_d)q(\boldsymbol{\beta})$, to allow efficient optimization. Both the form and the parameters of the component distributions are obtained by maximizing a lower bound of the log likelihood $\log p(\mathbf{w}_d|\boldsymbol{\alpha}, \boldsymbol{\lambda})$:

$$\mathcal{F}(q(\mathbf{z}_d), q(\boldsymbol{\theta}_d), q(\boldsymbol{\beta})) = \mathbb{E}_q[\log p(\mathbf{w}_d, \mathbf{z}_d, \boldsymbol{\theta}_d, \boldsymbol{\beta}, |\alpha, \boldsymbol{\lambda})] - \mathbb{E}_q[\log q(\mathbf{z}_d, \boldsymbol{\theta}_d, \boldsymbol{\beta})] \quad (4)$$

$\mathcal{F}$ is commonly known as the Evidence Lower Bound (ELBO). It can be verified that $\log p(\mathbf{w}_d|\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \mathcal{F}(q) + \mathrm{KL}(q||p)$, where p and q denote the true posterior and the variational distributions, respectively. $\mathrm{KL}(q||p)$ is the KL divergence between the variational and the true posterior distributions, which always takes a nonnegative value [2]. Therefore, we have $\mathcal{F}(q) \leq \log p(\mathbf{w}_d|\boldsymbol{\alpha}, \boldsymbol{\lambda})$, which justifies that it is a lower bound of the log marginal likelihood.

As it is difficult to directly optimize $\log p(\mathbf{w}_d|\boldsymbol{\alpha}, \boldsymbol{\lambda})$, variational inference seeks for component distributions $q(\mathbf{z}_d), q(\boldsymbol{\theta}_d)$, and $q(\boldsymbol{\beta})$ that maximize the lower bound $\mathcal{F}(q)$. This optimization problem can be solved through coordinate ascent, which iteratively optimizes one component while keeping the other two fixed. The resultant component distributions are:

$$q(z_{di} = k) = \phi^d_{ik} \quad (5)$$

$$q(\boldsymbol{\theta}_d) = \mathrm{Dir}(\boldsymbol{\theta}_d; \boldsymbol{\gamma}_d) \quad (6)$$

$$q(\boldsymbol{\beta}_k) = \mathrm{Dir}(\boldsymbol{\beta}_k; \boldsymbol{\pi}_k) \quad (7)$$

where $\boldsymbol{\phi}^d \in \mathbb{R}^{|\mathbf{w}_d| \times K}$ with $|\mathbf{w}_d|$ denoting the number of distinct terms in service description d, $\boldsymbol{\gamma}_d \in \mathbb{R}^K$ and $\boldsymbol{\pi}_k \in \mathbb{R}^V$ are Dirichlet parameters. They take the following optimal values:

$$\phi^d_{wk} \propto \exp\{\mathbb{E}_q[\log \theta_{dk}] + \mathbb{E}_q[\log \beta_{kw}]\} \quad (8)$$

$$\gamma_{dk} = \alpha_k + \sum_w \phi^d_{wk} \quad (9)$$

$$\pi_{kw} = \lambda_w + \sum_d \phi^d_{wk} N^d_w \quad (10)$$

where N^d_w denotes the number of times that term w appears in service description d. Expectations $\mathbb{E}_q[\log \theta_{dk}]$ and $\mathbb{E}_q[\log \beta_{kw}]$ can be evaluated as

$$\mathbb{E}_q[\log \theta_{dk}] = \Psi(\gamma_{dk}) - \Psi(\sum_{j=1}^K \gamma_{dj}) \quad (11)$$

$$\mathbb{E}_q[\log \beta_{kw}] = \Psi(\pi_{kw}) - \Psi(\sum_{j=1}^V \pi_{kj}) \quad (12)$$

It is clear from Equations (8), (9), and (10) that the expression of the three parameters $\boldsymbol{\phi}, \boldsymbol{\gamma}$, and $\boldsymbol{\pi}$ are interleaved. This requires an iterative update procedure given by Algorithm 1.

Algorithm 1 Compute Topic Based Representations

```
Initialize π_kw = λ_k + 1/V, k = 1..K and w = 1..V
while not converged
    for d = 1..D
        Initialize γ_dk = α_k + N_d/K, k = 1..K
        while γ_dk is not converged
            Update φ^d_wk using Eq. (8)
            Normalize φ^d_wk's so that Σ_{k=1}^K φ^m_wk = 1
            Update γ_dk using Eq. (9)
        end while
    end for
        Update π_kw using Eq. (10)
end while
Compute the topic based representation using Eq. (13)
```

The topic learning process will result in three component distributions, $q(\mathbf{z}_d), q(\boldsymbol{\theta}_d)$, and $q(\boldsymbol{\mu})$. These are functions of $\mathbf{w}_d$ since $\mathcal{F}$ is maximized for a set of fixed $\mathbf{w}_d$'s (i.e., service descriptions). Thus, they can be regarded as the approximations of the true posteriors. In particular, $q(\boldsymbol{\theta}_d)$ approximates the posterior of the topic mixture proportion of service description $\mathbf{w}_d$. We can use $\mathbb{E}_q[\boldsymbol{\theta}_d]$, which is the mean of the topic proportion over $q(\boldsymbol{\theta}_d)$, as the new representation of the service description in the topic space:

$$\mathbb{E}_q[\theta_{dk}] = \frac{\gamma_{dk}}{\sum_{j=1}^K \gamma_{dj}} \quad (13)$$

B. LDA-SVM Active Learning Service Classification

We consider the standard two-class classification problem. The training set consists of D service descriptions $\mathbf{x}_1, ..., \mathbf{x}_D$, where $\mathbf{x}_d \in \mathbb{R}^V$ is a term based vector, which can be obtained through standard text processing as described in Section II. Assume that all the labels of the training data are provided, which are $y_1, ..., y_D$, where $y_d \in \{-1, 1\}$. We will discuss how to leverage a partially labelled training set using active learning later in this section.

A linear classifier computes a hyperplane in the following form

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \quad (14)$$

that satisfies $f(\mathbf{x}_d) > 0$ for service descriptions having $y_d = 1$ and $f(\mathbf{x}_d) < 0$ for service descriptions having $y_d = -1$. Since there might be many hyperplanes that can separate the entire training dataset, a SVM classier seeks for a hyperplane that separates the training dataset with the largest margin. The maximum margin computation problem can be formally specified as follows. Given hyperplane $f(\mathbf{x}) = 0$, the distance between a data point $\mathbf{x}$ and the hyperplane is given by $|f(\mathbf{x})|/||\mathbf{w}||$. Assume that all the training data can be separated by the hyperplane, which implies that $y_d f(\mathbf{x}) > 0$, hence $y_d f(\mathbf{x}) = |f(\mathbf{x})|, \forall\ d \in [1..D]$. Hence, the distance of a service description $\mathbf{x}_d$ to the separating

hyperplane is

$$\frac{y_d f(\mathbf{x})}{||\mathbf{w}||} = \frac{y_d(\mathbf{w}^T\mathbf{x}+b)}{||\mathbf{w}||} \tag{15}$$

The margin is given by the distance between the separating hyperplane and the closest service description $\mathbf{x}_d$ from the training dataset. Therefore, a SVM classifier is obtained by computing the parameters $\mathbf{w}$ and b that achieves the largest margin from the training set:

$$\arg\max_{\mathbf{w},b} \quad \left\{\frac{1}{||\mathbf{w}||}\min_d[y_d(\mathbf{w}^T\mathbf{x}+b)]\right\} \tag{16}$$

$$\text{subject to} \quad ||\mathbf{w}|| = 1 \tag{17}$$

$$y_d(\mathbf{w}^T\mathbf{x}+b) > 0, d = 1, ..., D \tag{18}$$

where $||\mathbf{w}|| = 1$ is the norm constraint on $\mathbf{w}$. This constraint is added as we only care about direction of the hyperplane defined by $\mathbf{w}$ and its length is irrelevant, which is fixed to be 1. To see why this is valid, consider to scale both $\mathbf{w}$ and b to be τ times bigger (or smaller). Note that the distance defined by Eq. (15) is a constant of τ.

By solving the optimization problem in Eq. (16), we can compute $\mathbf{w}$ and b, which defines the max-margin separating hyperplane. In particular,

$$\mathbf{w} = \sum_{d=1}^{D} a_d y_d \mathbf{x}_d \tag{19}$$

where a_d is the Lagrange multiplier, which is incorporated to solve the constrained optimization problem in Eq. (16). To use the max-margin separating hyperplane to classify a new service description $\mathbf{x}$, we can plug it into Eq. (14), which gives

$$f(\mathbf{x}) = \sum_{d=1}^{D} a_d y_d \mathbf{x}_d^T\mathbf{x} + b \tag{20}$$

By checking the sign of $f(\mathbf{x})$, we can assign $\mathbf{x}$ into the corresponding class. It is also worth to note that only a small number of a_d will take nonzero values, which make them so called "support vectors". All other data points have no impact in classifying the new data point. By further examining Eq.(21), the label of a new data point is essentially the weighted sum of the labels of the support vectors:

$$f(\mathbf{x}) = \sum_{d\in\mathcal{S}} \beta_d y_d + b \tag{21}$$

$$\text{where } \beta_d = a_d \mathbf{x}_d^T\mathbf{x} \tag{22}$$

where $\mathcal{S}$ is the set of support vectors and β_d is the weight of the d-th support vector. As can be seen, β_d is computed based on the similarity between the two service descriptions through the dot product over their term vectors. As discussed earlier, due to the limited terms used by each service description, the term vectors of service descriptions are usually very sparse. This will make most weights close to zero. As a result, the predicted label will be close to zero (note that the two classes are labeled as 1 and -1, respectively), leading to error-prone predictions.

Having the topic-based service representation, we can replace the term vector $\mathbf{x}_d$ using a topic proportion vector $\mathbb{E}_q[\boldsymbol{\theta}_d]$. As the topics are computed to capture the high-level semantics of the service descriptions, topic proportion vectors are expected to better reflect the semantic similarity between service description and contribute positively to the classification accuracy. It is worth to note that the standard kernel trick can be applied to generate nonlinear classification boundaries, which will be useful for a set of service descriptions that is not linearly separable in the topic space. Furthermore, generalization to multiple classes can be achieved through the "one-versus-the-rest" scheme.

A key challenge to scale service classification to a large service space is to construct a high-quality training dataset, which usually involves intensive human effort to label a large number of services. Active learning is a sub-field of machine learning, the study of teaching computer systems to learn from experience, which aims to improve the training experience. This type of learning algorithm is able to query users in an interactive manner, in order to obtain the desired results with new data points [10]. In the most widely used pool-based active learning, there are three major components: (f, q, X), where f is a classifier trained over a small subset of label data points X and q is the query function that decides which instance from the unlabeled pool U to query next. The objective of applying this algorithm is to label the web services in a way that can contribute to the maximum reduction of human effort.

The reduction of the human effort can be quantified by the size of the so called "version space", which can be better understood in the context of separating hyperplanes as discussed earlier. A SVM classier seeks for a max-margin separating hyperplane by solving an objective function in Eq. (16). In particular, any separating hyperplane (not necessarily the ones giving the max margin) should satisfy the two constraints given by Eqs. (17) and (18). This set of feasible separating hyperplane forms the version space. As the version space contains all the feasible solutions based upon the currently labelled data, the goal is to query the next unlabeled data point that can achieve the (approximately) largest reduction of the version space. It has been shown that the SVM margin can provide a good approximation on the size of the current version space [10]. Therefore, we choose a web service to label that can maximally reduce the size the current SVM margin and then recalculate the margin based on the addition of the newly labeled web service. This process continues until a satisfactory test accuracy is achieved. Calculating the margin requires to solve a quadratic programming problem. The complexity is at the cubic order of the number of variables. The optimization problem in Eq. (16) is typically converted into its dual

formulation to allow the “kernel trick" in SVM. The dual problem can be solved in $O(D_t^3)$, where D^t is the number of service descriptions selected into the training set by the active learner.

IV. Experimental Study

In this section, we conduct a set of experiments to evaluate the effectiveness of the proposed LDA-SVM active learning method for web service classification. The experiments were performed on a real world dataset. All the experiments have been carried out on a Windows OS with 2.7 GHz I7 Intel Core processor and 16GB DDR3 memory. We evaluated our approach on three aspects. First, we compared the classification accuracy comparison between term vector based service modeling and LDA-based modeling. Second, we compared the classification accuracy between traditional SVM classifier and our approach. Third, we compared the number of services required to be labeled by traditional SVM classifier and our approach. The experiment result suggested that our approach can achieve comparable classification accuracy with a much smaller size of training set.

A. Experiment Dataset

We used the WS-DREAM web service dataset [15]. After cleaning the dataset by removing duplicates and filtering out the ones that have broken links, we selected the services with informative descriptions and manually labeled them based on their functionality. We chose ten categories that have a good number of services, including *activities, authentication, education, finance, health, human resource, messaging, news, search, and social media*. For each category, we selected the services whose descriptions are informative. These categories contain 900 services in total, where the number of services in each category ranges from 33 to 227.

B. Performance Study

Learning topics from WSDL files. We applied the LDA algorithm to all the selected service descriptions using different K values, ranging from 20 to 100. The LDA is trained on 70% of the documents and the perplexity is computed on the rest 30% documents. The perplexity is widely used to measure the quality of topic modeling result [4]. It measures the log-likelihood of unseen documents using the learned topic model. The lower the perplexity it is, the better the topic model is. The change of perplexity is described in Figure 1. We can see that the perplexity flats out at the lowest value when K equals to 55, which suggests an optimal value of K. Therefore, in our experiment, we use 55 as the number of topics.

We presented five selected topics and their top 5 terms in Table II. Those terms have the highest probability of belonging to the corresponding topics. As the terms suggested, topic 1 is related to education, topic 2 is related to News,

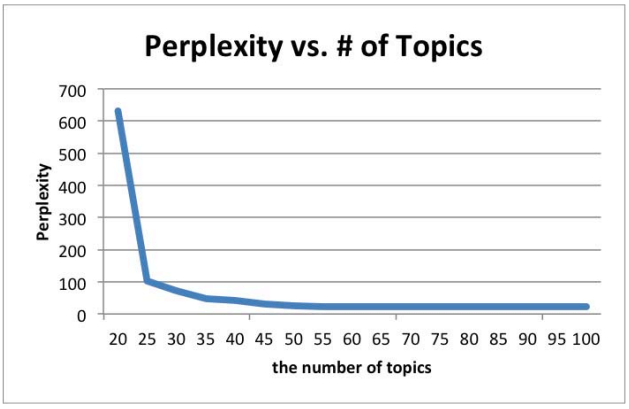

Figure 1: Perplexity vs. the number of topics

Table II: Top terms in selected topics

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
college, student, admission, courses, events	blog, feeds, update-newsdata, webnews, reporting	sms, gateway, outputmessage, unicell, sendbulksms	jobs, career, employee, salary, company	money, stock, market, business, account

topic 3 is related to messaging, topic 4 is related to human resources, and topic 5 is related to finance.

Performance comparison between SVM and LDA-SVM active learning. We set K's value as 55. We modeled each service description as a topic vector and then fed it to the classification process. We compared the performance of traditional SVM classification and LDA-SVM active learning in two aspects. First, we compared the classification accuracy achieved by those two approaches as shown in Figure 2. Second, we compared the number of services needed to be manually labelled by these two approaches given in Table III. We evaluated them over different sizes of datasets, ranging from 100 to 900 to study their scalability. For SVM, we used 70% of the WSDL files as the training set and the rest as the testing set and perform cross validation to compute the accuracy. For LDA-SVM active learning, we randomly chose 20% of the WSDL files to train the base classifier.

Performance comparison between term-based and LDA-based active learning. We compared two different modeling approaches, which provide the input to the classification process: term-based and LDA based. The term-based approach models each service as a term vector. We extracted 6,721 distinct terms from 900 WSDL files. As many of the terms only appear few times, they do not contribute to capture a service's feature but will act as noises. Therefore, we need to apply a certain frequency threshold to remove the useless terms. After several trials, we chose 12 as the frequency threshold and only kept the terms that appear at least 12 times. 1452 terms are selected and used to build

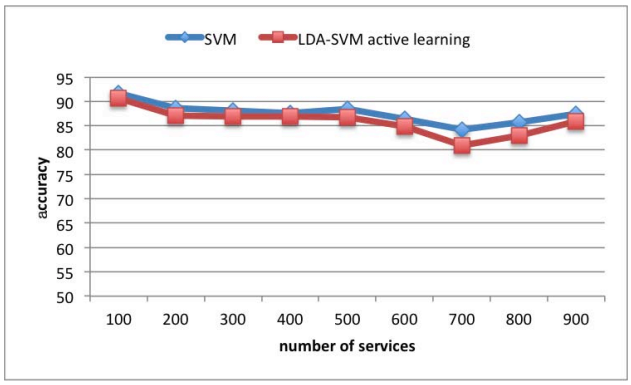

Figure 2: Accuracy comparison between SVM and LDA-SVM active learning

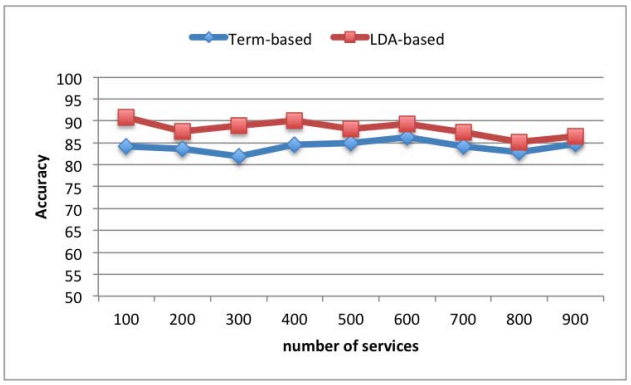

Figure 3: Accuracy comparison between Term-based and LDA-based active learning

Table III: Labeling cost comparison

# of services	# of manually labelled services in SVM	# of manually labelled services in LDA-SVM active learning	% reduction
100	70	44	37%
200	140	87	38%
300	210	141	33%
400	280	218	22%
500	350	272	22%
600	420	328	22%
700	490	385	21%
800	560	440	21%
900	630	489	22%

the feature space of web services. The topic-based approach models each service as a topic vector. Each entry represents the probabilistic proportion of the corresponding topic in the WSDL file. We set the number of topics as 55, as we got from the topic learning process. The accuracy comparison between these two approaches are shown in Figure 3. LDA-based approach clearly outperforms term-based approach since it better captures the topic-related features of web services.

V. Related Works

In this section, we describe several representative work related to using text mining techniques to assist service discovery. They can be classified into two categories: service clustering and service classification.

The idea of service clustering is to follow an unsupervised learning process, learning the similarity relationship among services and group those similar services together. In [7], descriptions of five features of web services are extracted from WSDL files, including type, content, message, port, and service name. The overall similarity between two services is measured as the weighted sum of the similarity measurement of the services on each feature. The approach then applies Quality Threshold (QT) clustering algorithm to generate service clusters based on the similarity matrix. In [5], probabilistic modeling techniques are used to model services based on their WSDL descriptions and tags. In [13], an approach was proposed to leverage the duality relationships between web services and operations to improve the accuracy of clustering. It models both services and operations as TF/IDF term vectors and generate a bipartite group between them. It then applies cosine similarity to build the similarity matrix between services and operations. A co-clustering algorithm, SVD, is apply to co-cluster services and operations. As a result, two clustering processes are performed simultaneously, where service clusters and operation clusters are formed. The two clustering process evolve together and provide input to each other, therefore, improve the accuracy of both clustering results. In [14], a latent factor model is adopted, which discovers a number of latent factors over a set of services and projects those services onto the latent factor space. Services that share similar latent factors can then be conveniently clustered. In particular, a Bayesian nonparametric model is adopted to automatically determine the optimal number of latent factors.

The idea of service classification is to follow a supervised learning process to label web services using a set of predefined classes. A classifier is trained from a training set, where the services' labels are known. It can then be used to predict the label of a new service. In [8], an approach was proposed to combine both syntactic description and semantic description to model services using term vectors. It applies five classification methods, including decision tree, naive bayes, SVM, Nearest Neighbor, and RIPPER rule learner, and then compares their accuracy. The result suggests that SVM outperforms other methods. In [11], UNSPSC, one of the standard taxonomy for web services, are used to form the feature space of web services. The description of a service category in UNSPSC is used as a sample document for the category. The terms in these documents form a

feature space, which is used to model web services. It then applies SVM to classify services into the categories. Each service is modeled as a category vector In [9], an approach is proposed to classify web services based on their QoS values. Several QoS attributes are used to model services such as availability, reliability, price, throughput, response time, and security. Bayesian networks are applied to classify services into four categories: platinum, gold, silver, and booze, corresponding four levels of service quality. All of these works rely on a large portion of services as the training set, whose labels need the input from human. Our approach can significantly reduce the cost of labelling web services with comparable classification accuracy.

VI. Conclusions

In this paper, we present a LDA-SVM active learning framework, which has the potential to classify large-scale services. The proposed framework is expected to play a vital role in organizing the large-scale, heterogeneous, and fast-increasing services and APIs on the web. By combining LDA based topic models with a SVM classifier, the resultant classification model is able to more effectively deal with the sparsity issue of the term based representation of service descriptions. By further integrating with the pool-based active learning, the proposed LDA-SVM active learning framework exploits a principled selection strategy to only able the most informative unlabeled services in the pool. Since the margin of the LDA-SVM base classifier provides a good approximation of the size of the current feasible solution space, the next sample is chosen to reduce the size of the margin as much as possible. Experimental results over real-world service data clearly demonstrate the effectiveness of the active learning service classification framework.

Our future work will follow two directions. First, we will extend our experiment study on a larger dataset which consists of both structured and unstructured service description. We will study how the performance of the approach changes with the impact of the related factors. We will also explore advanced topic modeling and active learning approaches to improve our approach. Second, we will work on a comprehensive service discovery framework that leverage the service classification result to efficiently query web service in a large scale.

References

[1] Eyhab Al-Masri and Qusay H. Mahmoud. WSCE: A crawler engine for large-scale discovery of web services. In *2007 IEEE International Conference on Web Services (ICWS 2007), July 9-13, 2007, Salt Lake City, Utah, USA*, pages 1104–1111, 2007.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[5] Liang Chen, Liukai Hu, Zibin Zheng, Jian Wu, Jianwei Yin, Ying Li, and Shuiguang Deng. Wtcluster: Utilizing tags for web services clustering. In *Service-Oriented Computing*, pages 204–218. Springer, 2011.

[6] Jos de Bruijn, Dieter Fensel, Uwe Keller, and Rubén Lara. Using the web service modeling ontology to enable semantic e-business. *Commun. ACM*, 48(12):43–47, 2005.

[7] Khalid Elgazzar, Ahmed E. Hassan, and Patrick Martin. Clustering WSDL documents to bootstrap the discovery of web services. In *IEEE International Conference on Web Services, ICWS 2010, Miami, Florida, USA, July 5-10, 2010*, pages 147–154, 2010.

[8] Ioannis Katakis, Georgios Meditskos, Grigorios Tsoumakas, Nick Bassiliades, and Ioannis P. Vlahavas. On the combination of textual and semantic descriptions for automated semantic web service classification. In *Artificial Intelligence Applications and Innovations III, Proceedings of the 5TH IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI'2009), April 23-25, 2009, Thessaloniki, Greece*, pages 95–104, 2009.

[9] Ramakanta Mohanty, V. Ravi, and M. R. Patra. Classification of web services using bayesian network. *Journal of Software Engineering and Applications*, 5(4), 2012.

[10] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

[11] Hongbing Wang, Yanqi Shi, Xuan Zhou, Qianzhao Zhou, Shizhi Shao, and Athman Bouguettaya. Web service classification using support vector machine. In *22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010, Arras, France, 27-29 October 2010 - Volume 1*, pages 3–6, 2010.

[12] Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 917–926, New York, NY, USA, 2009. ACM.

[13] Qi Yu and Manjeet Rege. On service community learning: A co-clustering approach. In *IEEE International Conference on Web Services, ICWS 2010, Miami, Florida, USA, July 5-10, 2010*, pages 283–290, 2010.

[14] Qi Yu, Hongbing Wang, and Liang Chen. Learning sparse functional factors for large-scale service clustering. In *2015 IEEE International Conference on Web Services, ICWS 2015, New York, NY, USA, June 27 - July 2, 2015*, pages 201–208, 2015.

[15] Yilei Zhang, Zibin Zheng, and Michael R. Lyu. Wsexpress: A qos-aware search engine for web services. In *Proc. IEEE Int'l Conf. Web Services (ICWS'10)*, pages 83–90, 2010.