SOLinker: Constructing Semantic Links between Tags and URLs on StackOverflow

***Abstract*—Thanks to the strength of crowdsourcing, there is a lot of useful information on StackOverflow, the most popular Question and Answer (Q&A) platform in software engineering area. These information can be treated as numerous URLs (Uniform Resource Locators), which can be categorized into URLs of Q&As and URLs in Q&As. The domain of former ones is StackOverflow itself, while domains of latter ones are miscellaneous, such as from some personal blogs and so on. Although each Q&A has been manually assigned tags, relations between URLs and tags are not clear enough. In this paper, we propose SOLinker, a method to build semantic links between various URLs and tags. Firstly, SOLinker identifies proper relations from a predefined relation set between tags and URLs, which is modeled as a text classification problem. Features are extracted from content of Q&A, the URL and the tags list, and classification algorithms are Logistic Regression and Gradient Boosting Decision Tree, depending on the category of URLs. Secondly, there exists a partial tagging problem, which means for a URL in a Q&A, there are only a part of tags of the Q&A relating to the URL. To address this problem, we propose a semantic analysis method to analyze context of this URL and the URL itself from both implicit and explicit aspects. Then SOLinker will infer proper tags by the label propagation technique. Results show that our method is feasible and practical in constructing semantic links between tags and URLs in/of Q&As. In particular, the F-Score of semantic relation identification is around 77%, 5% higher than the other existing method, and F-Score of partial tagging solving is around 87%.**

I. Introduction

Nowadays, software developers do not create alone. Experienced programmers share their knowledge by writing personal blogs or answering questions in some online communities. All these knowledge is good resources for programmers to further promote themselves. Although these resources are separately located on the Internet, StackExchange networks are a good knowledge repository storing many of them [1]. On one hand, programmers write solutions of questions asked by others directly; on the other hand, they answer questions by listing some URLs, like their blogs, in which question askers can find solutions. One of the most popular StackExchange sites is StackOverflow[1], which owns more than 4 million registered users and 11 million questions[2].

However, these resources on StackOverflow lack of organization. Although each URL of Q&A on StackOverflow has already been linked to tag(s) by human-beings, these links lack of semantic interpretations; for instance, some Q&As describe how to solve bugs, while some explain how a system works. On the other hand, for those URLs in Q&As, there are even no proper links between tags and them. An example of semantic linkage is illustrated in Fig. 1. In Fig. 1, firstly, the URL of the question is on the top of the figure. Because the asker wants to seek some libraries of Machine Learning, the content of this Q&A is labeled to *Seeking-Something* (*S*), and then linked to tags of this Q&A. Secondly, there are two URLs in the answer of the question, *PyBrain* and *Theano*, which are two famous machine learning libraries of python. However, there are three extra uncorrelated tags: *c*, *c++* and *java*, in the tags list of the question. This is called partial tagging problem. SOLinker will choose proper tag(s) in the tags list for corresponding URL, and then construct semantic links. As both *PyBrain* and *Theano* are official documents (*O*), SOLinker marks relations between tags and these two URLs as *O*.

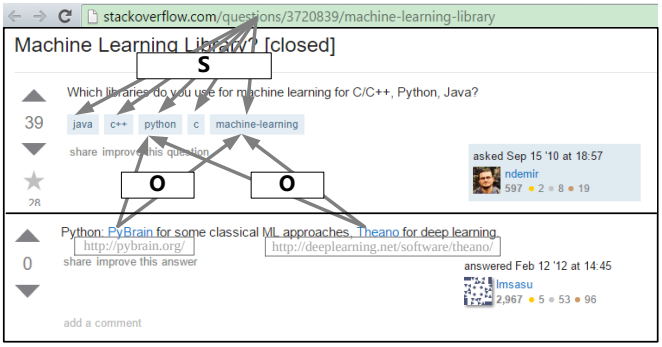

Fig. 1. Example of Constructing Semantic Links and Partial Tagging Problem.

There are many advantages of building semantic links between URLs in/of Q&As and tags, and three of them are listed: (1) Semantic Queries Supporting: For URLs of Q&As, users have one more criterion to filter search results by their usages. For URLs in Q&As, without semantic links, the search engine of StackOverflow treats these URLs as common text, regardless of latent meanings in them. Connecting these URLs to tags can make search engine smarter by knowing the hidden meanings in these URLs. (2) Understanding Tags Behaviors: After constructing semantic links between tags and URLs on StackOverflow, we can study behaviors of tags; for instance, users have less questions about how to debug for tag *machine-learning* than tag *c++*. (3) Semantic Annotation: Constructing links between URLs and tags on StackOverflow is a kind of semantic annotation for StackOverflow. On the other hand, there are also relations between tags. If an ontology, described relations between tags [2], is constructed, these semantic links can be an enhancement of ontology, the other kind of semantic annotation.

Due to increasing requirements, in recent years, the task has

[1]http://stackoverflow.com/
[2]https://en.wikipedia.org/wiki/Stack_Overflow

attracted extensive attentions. Lucas *et al.* proposed a method to classify URLs of Q&As from usages of them, and then obtained a conclusion that this criterion can promote users' search experience greatly [3]. However, features extracted in their method are manually selected keywords. Carlos *et al.* have labeled some URLs in Q&As manually to study how software developers discover and disseminate innovations [4]. However, they have not proposed a method to labeled these URLs automatically. To constructed semantic links between tags and URLs in/of Q&As, conclusively, there are three challenges: (1) For URLs of Q&As, we should consider how to extract proper features to build a classification model; (2) For URLs in Q&As, because SOLinker does not know content in webpages of them (crawling pages of these URLs is time-consuming), we can only infer the type of them from these URLs themselves and contexts of them appearing in the corresponding Q&A. (3) For URLs in Q&As, as the example shown in Fig. 1, there exists a partial tagging problem, and we should address this problem when linking these URLs to tags.

To address these challenges, we propose SOLinker to create semantic links between tags and two kinds of URLs on StackOverflow. Our method consists of three steps: (1) Semantic Linkage Between URLs of Q&As and Tags. This step is the easiest one, because we can obtain the full content of Q&As and each Q&A has already labeled by their askers. The main work in this step is to construct a text classification model to classify these Q&As to predefined relations. (2) Semantic Relations Identification for URLs in Q&As. SOLinker also models this step as a classification problem, but differ to step (1), for each URL in Q&A, we must extract features from the URL itself and the context of the URL in the Q&A as we cannot obtain the specific content in the webpage of the URL. (3) Solving Partial Tagging Problem for URLs in Q&As. We model this problem as a subset selection problem, which selects the most relevant subset of tags from the tags list of the Q&A. SOLinker will analyze the URL, the title and context of the URL in the Q&A implicitly and explicitly to obtain some initial tags. Then based on these initial tags, SOLinker will infer the most proper tags by label propagation technique. In particular, F-Score of semantic relations construction is around 77%, 5% higher than the other existing method, and F-Score of partial tags solving is around 87%. Our main contributions can be summarized as follows: (1) Proposing a text classification algorithm to identify semantic relations between tags and URLs of Q&As. (2) Proposing a URL classification algorithm to identify semantic relations between URLs in Q&A and tags without knowing contents of these URLs. (3) Proposing a semantic analysis method to solve the partial tagging problem appearing in semantic linkage between URLs in Q&As and tags.

II. Related Work

In this section, we first introduce methods of semantic linkage, and then introduce methods of text classification and some similar work to solve the partial tagging problem.

A. Semantic Linkage

To learn semantic lexicons from massive corpus, there are two kinds of approaches: Pattern-based and Classification-based. (1) Pattern-based methods focus on extracting semantic relations based on predefined patterns [5][6] or extracted patterns [7][8][9][10]. Hearst *et al*. proposed a method which defined patterns such as *X like Y* to find synonymy between *X* and *Y* from texts [5]. Then, based on these patterns, Kozareva *et al.* recursively applied them to further increase the recall [6]. To learn some new patterns from existing patterns and corpus, Michael *et al.* proposed a method, Basilisk [7], which hypothesized the semantic class of a word based on collective information over a large body of extraction pattern contexts. Richard *et al.* proposed a set expansion-based method, in which patterns were used to extract semantic relations from semi-structured documents written in any markup language and in any human language [8]. (2) Classification-based methods focus on classifing words or documents into some predefined types. Nadeau *et al.* summarized a serious of works for entity-typing, which listed a lot of supervised classification models and features [11]. Turian *et al.* propsed a feature-level semi-supervised learning approach to learn the type of documents or words, in which unsupervised word feature derived from a large corpus is used to improve performance of existing supervised models [12]. Ling *et al.* proposed a method which modeled the semantic typing problem as a multi-class multi-label classification task [13]. There are also many other approaches to do the semantic linkage; for instance, graph-based methods [14], label propagation-based methods [15], clustering-based methods [16], knowledge-based methods [17] and so on. We borrow the idea from classification-based methods that firstly extract features from data and then apply a classification algorithm to obtain the semantic type of data.

B. Text Classification

Text classification [18] is a very practical technique widely used in tags recommendation [19], application reviews mining [20] and so on. Features in text classification [18] are the n-gram of words presented by *tf-idf* measurement or Bag-Of-Word (BOW) model. To select or extract important features from these raw features to further improve performance of classification, supervised methods, like Mutual Information (MI) selection, and unsupervised methods, like Principal Component Analysis (PCA) and Latent Dirichlet Allocation (LDA), are widely used [18]. With text data on StackOverflow, Lucas *et al.* proposed a text classification to classify Q&A pairs to five classes to assist software development [3]. However, most features used in this method are all keywords extracted manually, so their method is not fully automatically. When constructing semantic links for URLs of Q&As, we borrow their idea, but our method extracts a variety of features and then applies dimensionality reduction techniques to learn some hidden feature representation rather than manually selected keywords.

URL classification is a special case of text classification, which is widely used in identifying specific type of page [21], like research homepage [22], and malicious web sites detection [23][24][25]. Features to classify URLs are extracted from URL itself [21], such as top n-gram in URL, IP address and so on, or/and from contents in webpages of URLs [22]. For URLs in Q&As, Carlos *et al.* manually labeled some of them and then studied innovation diffusion [4]. We borrow the idea of some URL features extraction methods from Kan *et al.*'s work[21] as we cannot obtain contents of URLs in Q&As.

We also borrow the idea from labeling work in [4], but our approach is fully automatic.

C. Partial Tagging Problem

Partial tagging problem of URLs in Q&As is a new problem proposed in our paper, so there is no existing work directly targeting in this problem. However, there are some relative works: tags recommendation and aspect extraction. (1) Tags Recommendation focuses on recommending several tags to a given document. Wang *et al.* proposed a tags recommendation system called EnTagRec [19], in which it combines the Bayesian Inference, Frequentist Inference and spreading activation technique [26] to recommend tags for StackOverflow. Mo *et al.* proposed a cross-sites tags recommendation algorithm to recommend tags on StackOverflow to repositories on Github [27]. (2) Aspect Extraction. Hu *et al.* proposed a method to extract implicit and explicit aspects from reviews to do sentiment analysis [28]. To solve partial tagging problem, our method borrows the idea from these two kinds of work. Firstly, our approach analyzes the URL, the title and context of the URL in the Q&A implicitly and explicitly to obtain a set of initial tags. Then, based on these tags, our approach adopts spreading activation technique [26] to obtain more relevant tags before selecting a most proper set of tags to construct semantic links.

III. Problem Definition

Our method treats the semantic linkage problem as text classification problem. To do text classification, the label set is defined as $\boldsymbol{Y}$, and features extracted from URLs and Q&As' content are denoted by $\boldsymbol{X}$. The goal of text classification is to construct a classifier $f(\mathbf{x}) = argmax_y P(y|\mathbf{x})$ from some labeled data, donated by $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)\}$, where $(\mathbf{x_i}, y_i) \subset X \times Y, (i = 1, ..., n)$. For an unseen instance, represented by $\mathbf{x}'$, where $\mathbf{x}' \subset X$, this classifier $f(\mathbf{x}')$ can predict the right type of the instance in a relative high probability (at least higher than random guessing). To construct semantic links by text classification, from the content of websites, we define a set of semantic relations between URLs and tags for both URLs of Q&As and URLs in Q&As. We borrow the idea in [3], considering four categories: *How-to-do-it* (*e.g.* how to solve a bugs or use an API, etc.), *Conceptual* (*e.g.* definition of concepts, best practices for a given technology, etc.), *Seeking-Something* (*e.g.* book, tutorial, advice, recommendation, etc.) and *Others*. This classifier is called URL Content Classifier. For URLs in Q&As, borrowing the idea in [4], we define the other set of semantic relations between URLs and tags considering website type: *Official Document*, *Blog*, *Q&A Post*, *Wiki*, *Code Repository* and *Others*. This classifier is called URL Type Classifier.

On the other hand, for the partial tagging problem, we defined it as a subset selection problem. For a URL $\boldsymbol{u}$ in a Q&A, there is a tags list T in the Q&A. The most suitable tags list for $\boldsymbol{u}$ are $T_u \subseteq T$. This problem is solved by Partial Tagging Solver in SOLinker.

Fig. 2 is the overview of SOLinker. To construct semantic links between tags and URLs in/of Q&As, the key issue is text classification. (1) For a URL of Q&A, word-based and non-word-based features are extracted from its tags list, title and body. To overcome the curse of dimensionality problem [29], SOLinker reduces the dimension of word-based and tags features before classifying by URL Content Classifier. Then, semantic links between the URL and tags in the tags list are constructed by Sematic Linker if the URL is not classified to *Others*. (2) For a URL in Q&A, the first step is extracting features from the URL itself, tags list and its context and then reducing feature dimensions before classifying by two classifiers, one by URL Type Classifier and the other by URL Content Classifier. The second step is solving partial tagging problem by Partial Tagging Solver, which applies explicit analysis, implicit analysis and label propagation technique to obtain a subset of tags list as target tags. Finally, semantic links are constructed between this URL and these target tags by Semantic Linker if it is not classified to *Others*.

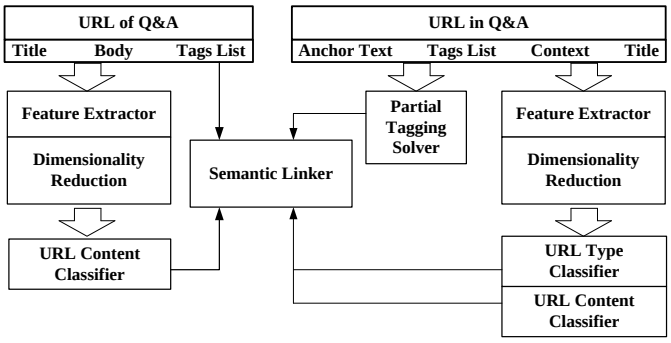

Fig. 2. The Overview of SOLinker

IV. Semantic Relation Identification

To identify semantic relations between Tags and URLs, SOLinker classifies URLs of/in Q&As to some predefined labels mentioned in Section III. This is a text classification task. In this section, we will introduce how to extract features from raw data, how to do the dimensionality reduction and how to classify. For URLs of Q&As, features are extracted from contents and tags of Q&As, while for URLs in Q&As, features are extracted from contexts of URLs appearing in Q&As, URLs themselves and tags of Q&As (Because SOLinker does not crawl webpages of these URLs, contents in these pages are unknown.).

A. Feature Engineering for URLs of Q&As

Because for URLs of Q&As, we know contents of them, features can be extracted from their tags lists and contents.

1) Features from Tags Lists: For each Q&A, there is a tags list constructed by the question asker manually. In Fig. 1, the tags list is: "*java*, *c++*, *python*, *c*, *machine-learning*". Firstly, we get tags lists from all Q&As, and then do the tag cleaning work to normalize tags in each list. The process of tag cleaning is the same as Mo *et al.*'s work[27]: (1) Removing Tag Version (e.g. *ios-4.2* or *ios5.1* will be rewritten to *ios*.), (2) Coping with Synonyms[3] (e.g. *io.js* will be rewritten to *node.js*.), and (3) Splitting Complicated Tags (e.g. *google-maps-sdk-ios* will be rewritten to "*google-maps*, *sdk*, *ios*".). Then a tags dictionary can be constructed from all clean tags lists, in which there are

[3]The synonyms relations are in http://stackoverflow.com/tags/synonyms

TABLE I. AN EXAMPLE OF URL PARTITION

Sample URL	http://www.cwi.nl:80/%7Eguido/Python.html
Partition by (1)	(http, www, cwi, nl, 80, guido, python, html)
Partition by (2)	(http, www.cwi.nl:80, /%7Eguido/python.html, , ,)

totally 657 tags. Then we represent each tags list to a vector by BOW (Bag-Of-the-Word) model, which means, if a tag appears in the list, then the corresponding dimension of its vector is 1; otherwise 0.

2) Features from Content: Our method divides content of Q&A into two parts: (1) the title of Q&A and (2) the body of Q&A. In Fig. 1, the title is "Machine Learning Library?" ("[closed]" is the title status, which is ignored in our method.) and rest text in question and answer is all body. The reason why we distinguish two kinds of text is that we consider to classify a Q&A, the degree of importance for text appearing in the title is not equal to that appearing in the body. Despite two kinds of text, the process of transforming them to vector form is the same. This transformation process contains following steps: (1) For text in a Q&A, tokenize it and then remove all punctuation characters, code fragments and xml labels (content of Q&As is all written in xml format) appearing in both title and body. (2) Do the stem operation[4] for each word to get the root of the word. (3) Select high-frequency bigrams and trigrams (appearing at least 10 times) among all documents. (4) Remove stopwords[5] and low-frequency unigrams (appearing less than 10 times in all documents). (5) Transform each document by BOW model to the vector. After these five steps, for each Q&A, two vectors are constructed, one for title, and the other for body. Because after transformation, each dimension in these vectors is corresponding to a word or a phrase, we call this kind of features as word-based features.

On the other hand, SOLinker not only extracts word-based features, but also 11 non-word-based features from content: (1) a binary feature indicating whether the title is a question or a statement; (2) two binary features indicating whether there exists code fragment, wrapped by $< code >$ in xml format, in question body and in answer body ; (3) a feature counting the number of URLs in the body; (4) three features counting the number of words in the title, the question body and the answer body; (5) two features counting the number of paragraphs, wrapped by $< p >$, in the question body and the answer body; (6) two features counting the number of elements in list, wrapped by $< li >$, in the question body and in the answer body.

B. Feature Engineering for URLs in Q&As

Because for URLs in Q&As, content of them is unknown, as the crawling process is time-consuming. Features can be extracted from their tags lists, corresponding Q&As' content and URLs themselves.

1) Features from Tags Lists and Q&A Content: Processes of feature extraction from tags lists for URLs in Q&As are the same as processes in Section VI A 1). We do not extract features from title, because they have negative effect for classifying URLs in Q&As, which will be discussed in the experiments section. To extract word-based features from the body, we consider the context of the URL rather than the whole body, because text far from this URL is noise (this is shown in the experiments section). As for non-word-based features, they are extracted from the whole body as same as Section VI A 2).

Next, we define the context of a URL. The body of question and answer is written by xml format and can be thus serialized by depth first traversing. In this sequence, we remain $< p >$ (paragraph) and $< a >$ (URL) nodes and remove other nodes. After that, the body can be represented by $(n_1, n_2...n_k), n_i \in \{p, a\}$. The w-window context of a URL node a is defined as: all text in p with w forward and backward steps in the sequence. For instance, for a sequence $(p_0, p_1, a_0, a_1, p_2)$, the 1-window context of a_0 and a_1 are the same, which is text in p_1 and p_0. In our work, SOLinker only considers 1-window context. After extracting the context, the feature vector for the context is also constructed by the process described in Section VI A 2).

2) Features from URLs: Although the content of the webpage of URL is not available, features can be extracted from the URL itself. We extract URL features from two ways:

(1) **Word-based URL Features**. This feature extraction method treats each URL as a sequence of words. However, tokenizing a URL is different from tokenizing text written in natural language. We define separators to split URL as non-digital characters and non-alphabet characters. Besides, another separator is a Regular Expression (RE) *%.2*, for removing characters by some special encodings. For example, in TABLE I, the first row is a sample URL and the second row is splitting it by this way (noted that *%7E* is the RE separator). After tokenizing, high-frequency bigrams and trigrams (appearing at least 10 times among all URLs) are selected and low-frequency unigrams (appearing less than 5 times among all URLs) are removed. Then we construct URL word-based feature vector by BOW model.

(2) **Non-word-based URL features**. This feature extraction method applies the urlparse[6], to break URL string up in 6-tuples (scheme, netloc, path, params, query, fragment). The third row of TABLE I illustrates an example of splitting the URL in first row in this way. For scheme, we use one feature to indicate different kinds of schemes; for netloc, we select top-frequency (at least appearing 10 times in all URLs) ones and then for each top netloc, we give it a binary indicator; for path, we use one feature to indicate the type of file extension; for params, we give a binary indicator to indicate whether there exists parameters in URL (the same with query and fragment). Besides, there are three other non-word-based features: (a) the length of URL; (b) whether this URL is in a list (wrapped by $< li >$); (c) whether text in last $< p >$ end with colon.

C. Dimensionality Reduction

Because the number of dimensions of tags feature vectors and word-based feature vectors are very large, it will lead to the curse of dimensionality problem when classifying [29]. To overcome this problem, before classification, SOLinker will do the dimensionality reduction process. To reduce the

[4]The stemming package is in http://www.nltk.org/api/nltk.stem.html.

[5]The stopwords list from http://www.textfixer.com/resources/common-english-words.txt

[6]https://docs.python.org/2/library/urlparse.html

dimension of these vectors, there are two kinds of methods: (1) Feature Selection, which tries to find a subset of the original features; and (2) Feature Extraction, which focuses on extracting some hidden features from feature vector rather than simply selection. SOLinker firstly applies feature selection techniques to select important features and then applies feature extraction techniques to extract hidden features.

1) Feature Selection: In this step, SOLinker calculates the Mutual Information (MI), derived from information theory, for each dimension, and then filters the dimension with low MI. We calculate the maximum values of mutual information, shown in Eq. (1), over different class for one dimension. In Eq. (1), y is a specific class and x is a specific dimension. Because the Mutual Information selection can be only adapt to discrete feature, we use BOW model to represent tags list features and word-based features rather than *tf-idf* model.

$$MI(x) = max_y \log \frac{p(x|y)}{p(y)} \tag{1}$$

2) Feature Extraction: After feature selection, SOLinker applies the PCA (Principle Component Analysis) technique [30] to further extract some hidden features. PCA is one of popular unsupervised feature extraction techniques. The insight of it is using an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. To do the PCA, we use the API provided by a famous python machine learning library scikit-learn[7]. SOLinker also tries to apply LDA (Latent Dirichlet Allocation) [31], a popular topic model to do the dimensionality reduction for text data, to do feature extraction work, but the performance of LDA is worse than that of PCA (shown in the experiments section).

D. Classification Algorithm

We performed a comparison between different classification algorithms to find the best one. In the classification process, we experiment totally six classification algorithms: Logistic Regression with L2-norm (LR), Support Vector Machine with Radial Basis Function Kernel (SVM), Naive Bayes, Decision Tree (C4.5), Random Forest (RF) and Gradient Boosting Decision Tree (GBDT). APIs of all these algorithms we used are all provided by scikit-learn.

V. Partial Tagging Solver

Partial tagging problem exists in semantic linkage for URLs in Q&As, because for URLs of Q&As, tags of them are all assigned manually. However, the tags list of a URL of Q&A is for the whole Q&A rather than for a specific URL in this Q&A. To link URLs in Q&As with tags, selecting target tags is important work. This problem can be solved by tags classification [19] [27], which uses features extracted from the URL, its context and its title, to predict a set of suitable target tags. We tried this method, but performance of it is not promising (shown in the experiments section). By analyzing data, if we use 1-window context, the context feature will be very sparse as there are only a few words in context. Although we can use mutual information selection to solve this problem, as the number of labels we want to predict (all tags on StackOverflow) is very large, the mutual information for each dimension will be too low. However, if we use large-size windows context, there will be much noise, as context far from the URL has low semantic relation with it.

To address this problem, we propose a semantic analysis method. Firstly, SOLinker analyzes 1-window content, the question title and the URL explicitly, then applies tags classification technique to analyze them implicitly, and finally applies label propagation technique [26] to find target tags.

A. Explicit Analysis

For a URL in Q&A, we consider $u = \{C, U, A, QT\}$, in which C is 1-window context, U is the URL itself, A is the anchor text and QT is the question title. Explicit analysis tries to apply string similarity techniques to obtain a set of obvious tags. Firstly, we tokenize all elements of u. Content C, the anchor text A and the question title QT are written by natural language, and thus they can be tokenized directly. As for the URL U, we apply the tokenized method mentioned in Section IV B. Then, for each token, we apply the ratio of Levenstein distance[8], based on the Levenstein distance and in range $[0, 1]$, to calculate the similarity between all tokens and all tags on StackOverflow. If the similarity between a token and a tag is high (greater than 0.8), then this tag is a candidate. Besides, SOLinker considers the unigram, bigram and trigram form of tokens to calculate the similarity at the same time. After explicit analysis, SOLinker can obtain a set of candidate tags.

B. Implicit Analysis

Sometimes, SOLinker cannot obtain any candidate tags from explicit analysis, so SOLinker will do implicit analysis. Implicit analysis tries to apply tags classification technique to obtain a distribution among all tags. Based on tokens in the explicit analysis step, SOLinker constructs a feature vector by combining four word-based vectors extracted from context tokens, URL tokens, title tokens and anchor text tokens, by method mentioned in Section IV A. As we analyze, there are too many labels; therefore, the performance of mutual information feature selection is poor. We apply PCA directly to reduce the dimension of the feature vector before applying Naive Bayes to get a tags distribution for each URL.

C. Label Propagation

After explicit analysis and implicit analysis, we can obtain a set of candidate tags and a tags distribution and they should be merged. Our composition strategy is that for each tag in candidate tags, SOLinker will add $1/n$ to corresponding dimension in tags distribution, where n is the total number of candidate tags, and then the new tags vector denoted by TD. For example, in Fig. 1, for URL *PyBrain*, through explicit analysis, we can obtain a set of candidate tags {*python*, *pybrain*, *machine-learning*}, and through implicit analysis, supposed, we can obtain a tags distribution {*python*: 0.25, *pybrain*: 0.25, *machine-learning*:0.25, *deep-learning*:0.25}. After

[7] http://scikit-learn.org/stable/

[8] The library we used to calculate this distance is in https://github.com/ztane/python-Levenshtein/

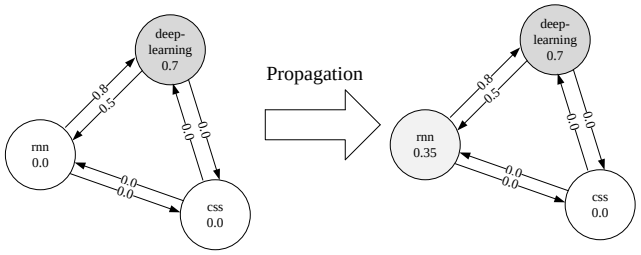

Fig. 3. An Example of Label Propagation in One Iteration

composition, the vector changes to {*python*: 0.58, *pybrain*: 0.58, *machine-learning*:0.58, *deep-learning*:0.25}.

To infer more associated tags, we apply a technique named label propagation [26][27]. The inputs of label propagation are a similarity matrix S, where each element $s_{i,j}$ in S is the similarity between the tag_i and the tag_j, and the initial state of tags which is TD in our context. During the process of label propagation, for each tag in TD, in one iteration, it propagates its weight to other tags with corresponding similarity. Fig. 3 shows an example of an iteration of label propagation. Supposed we have a document labeled by {*deep-learning*: 0.7, *rnn*: 0.0, *css*: 0.0} and tags similarities are shown in Fig. 3. After an iteration, the *deep-learning* propagates to *rnn* as they have 0.5 in similarity, and thus the vector becomes {*deep-learning*: 0.7, *rnn*: 0.35, *css*: 0.0}. To construct the similarity matrix, we use the conditional probability between two tags, shown in Eq. (2). In Eq. (2), $P(tag_i, tag_j)$ is the co-occurrence probability that tag_i and tag_j appear in the same tags list, and $P(tag_j)$ is the probability that tag_j appears among all tags lists. On the other hand, to overcome over propagation problem, if the similarity between two tags is lower than 0.3, then we set it to 0. And the propagation step is set to one.

$$s_{i,j} = P(tag_i|tag_j) = \frac{P(tag_i, tag_j)}{P(tag_j)} \tag{2}$$

D. Target Tags Selection

After label propagation, SOLinker will obtain a final tags distribution. To solve the partial tagging problem, firstly, we select the tag probability for each tag in tags list of the Q&A from the final tags distribution, and then normalize them. Finally, we select tags with high probability (greater than 0.30) from the tags list as target tags of the URL. For instance, in Fig. 1, there are two URLs in the Q&A, *PyBrain* and *Theano*. For *PyBrain*, it is labeled to {*python*: 0.78, *pybrain*: 0.78, *machine-learning*:0.78, *deep-learning*: 0.32, *rnn*: 0.125} through label propagation. The tags list of the Q&A is "*java*, *c++*, *python*, *c*, *machine-learning*", so selected tags are {*python*: 0.78, *machine-learning*:0.78}. After normalization, the vector becomes {*python*: 0.50, *machine-learning*:0.50}, and both probabilities of elements are greater than 0.30; therefore, target tags are *python* and *machine-learning*.

Algorithm 1 describes how to solve partial tagging problem. In line 1, it does the explicit analysis and then obtains a set of candidate tags TC; then, in line 2-4, it extracts word-based features and does the dimensionality reduction before doing implicit analysis. Line 5 merges TC to TD, and tags in the tags list are selected from TD in line 6. After normalization (line 7), high-probability tags are selected as target tags TT, in line 8.

Algorithm 1 Partial Tagging Solver

Input:
URL Information $u = \{C, U, A, QT\}$;
The Similarity Matrix S;
Naive Bayes Classifier NB;
The Tags List of Q&A TL;
Output:
Target Tags TT
1: $TC = explicitAnalysis(u)$
2: $FE = wordBasedFeatureExraction(u)$
3: $FE \leftarrow PCA(FE)$
4: $TD = implicitAnalysis(FE, NB)$
5: $TD = composition(TC, TD)$
6: $T \leftarrow tagSelection(TD, TL)$
7: $T \leftarrow normalize(T)$
8: $TT = highFreq(TT)$
9: **return** TT;

VI. Semantic Linker

After obtaining semantic relations between tags and URLs of/in Q&As, and solving partial tagging problem for URLs in Q&As, semantic links can be constructed. (1) For a URL of Q&A, it is linked to tags in its tags list with the same semantic relation, which is classified by URL Content Classifier. (2) For a URL in Q&A, it is linked to tags selected by Partial Tagging Solver with same semantic relation(s). Because for URLs in Q&As, SOLinker totally trains two classifiers, URL Content Classifier and URL Type Classifier, some links may contain two kinds of semantic relations. If URLs in/of Q&As are classified to *Others*, then SOLinker will remove this kind of semantic links.

For example, in Fig. 1, the Q&A is classified to *Seek-Something* (*S*); therefore, the URL of it will be linked to all tags in the tags list with *S*. For URLs in this Q&A, *pyBrain* and *Theano*, they are both classified to *Official Document* (*O*) by URL Type Classifier, and *Others* by URL Content Classifier, and tags of them are *python* and *machine-learning* selected by Partial Tagging Solver; therefore, both these two URLs will be linked to *python* and *machine-learning* with *O*, while the *Others* label will be ignored.

VII. Experiments

In this section, we first present our experimental settings and then analyze experiment results. In SOLinker, we concern performances of (1) URLs of Q&As classification, (2) URLs in Q&As classification, and (3) Partial Tagging Solver.

A. Experimental Settings

We select totally 1,200 Q&As from the data dumps of StackOverflow[9], to conduct experiments. In these Q&As, there are totally 1,000 URLs. To validate our approach, the first

[9]https://archive.org/details/stackexchange

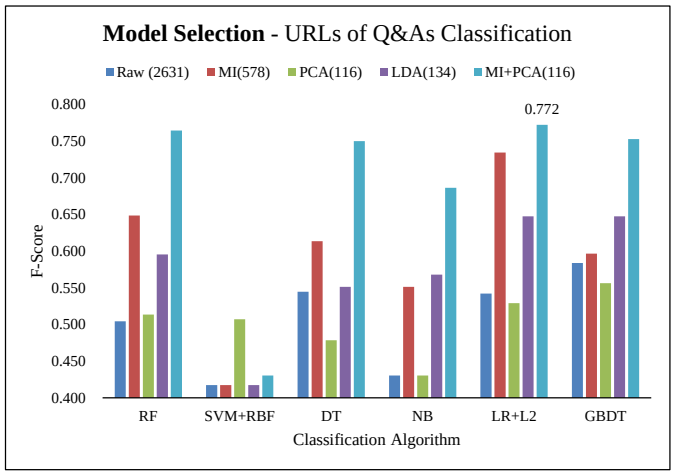

Fig. 4. Results of Model Selection for URLs of Q&As Classifier

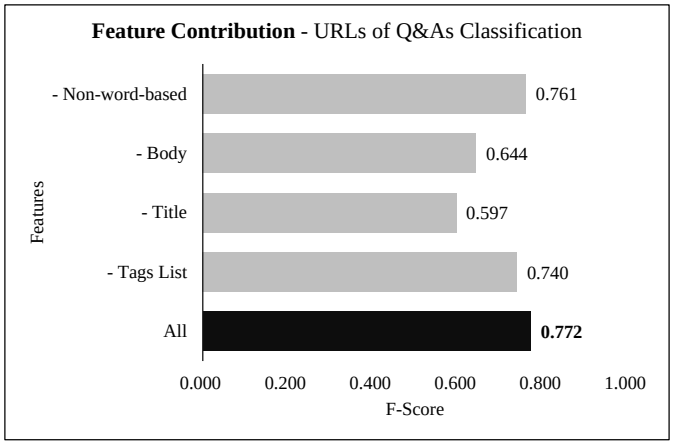

Fig. 5. Results of Feature Contribution for URLs of Q&As Classifier

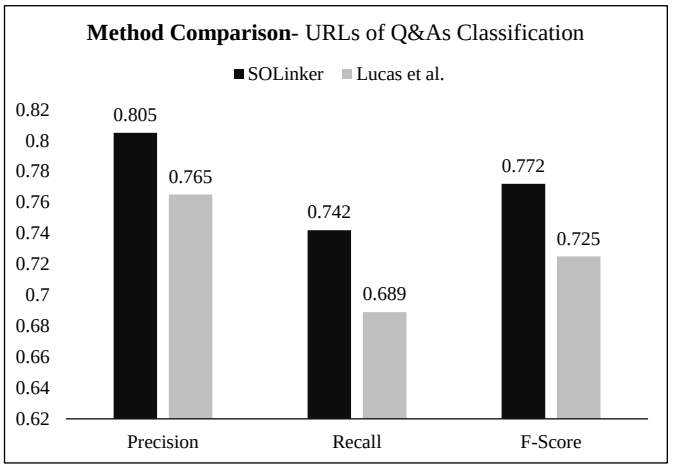

Fig. 6. Results of Model Comparison for URLs of Q&As Classifier

thing is to construct ground-truth data. We asked ten master students major in computer science to label these URLs of/in Q&As. As for the evaluation of partial tagging problem, these students are asked to select a subset of tags in the tags list of the corresponding Q&A, for each URL in it. We use Precision, Recall and F-Score to measure the performance of our method. Besides, results in the following section are all from the average of 5-cross validation.

B. URLs of Q&As classification

The experiment consists of three sub-experiments: model selection, feature contribution and model comparison.

1) Model Selection: Model selection focuses on comparing performances of different dimensionality reduction algorithms and classification algorithms. For dimensionality reduction, we compare performances of three algorithms: Mutual Information (MI) selection, Principle Component Analysis (PCA) and Latent Dirichlet Allocation (LDA). For classification algorithms, we compare performances of six algorithms: Random Forest (RF), Support Vector Machine with Radial Basis Function Kernel (SVM+RBF), Decision Tree C4.5 (DT), Naive Bayes (NB), Logistic Regression with L2-norm (LR+L2) and Gradient Boosting Decision Tree (GBDT). Results are illustrated in Fig. 4, in which horizontal axis donates different classification algorithms, vertical axis for the value of F-Score and bars with different color represent different dimensionality reduction techniques. In Fig. 4, results of raw features (features without any dimensionality reduction) among all classification algorithms are not promising. However, after adopting Mutual Information (MI) as criteria to select some word-based features and tags features, the dimension reduces from 2631 to 578, and F-Scores of all models increase sharply, except the SVM+RBF classifier. Then, we try to adopt PCA and LDA to reduce the dimension for tags features and word-based features directly. It is clear from the figure that performances of RF, DT and LR+L2 are worse than those reduced by MI, while, performances of PCA in SVM+RBF and LDA in NB and GBDT are better than those reduced by MI. Next, we use MI as criteria to select some features and then apply PCA to extract some hidden features. The number of dimensions is reduced to 116, far less than that of dimensions of raw data, but performances of almost all models increase dramatically except SVM+RBF. Finally model with the best performance is trained by Logistic Regression with L2-norm (0.772 F-Score), whose features are reduced by MI and then by PCA. Conclusively, dimensionality reduction is an important process, which not only reduces the dimension of features to decrease the training time, but also improves the performance of classification.

2) Feature Contribution: This experiment focuses on studying how much each feature contributes to the classification (features are selected and extracted by MI and PCA). To classify URLs of Q&As, there are totally four kinds of features, which are non-word-based features, features extracted the tags list and word-based features extracted from body and title. Because model with best performance for classifying URLs of Q&As is Logistic Regression with L2-norm, results in this experiment, shown in Fig. 5, are all obtained by this model. In Fig. 5, the bar in black is the F-Score of model with all four features and grey bars are F-Scores of models with features except the feature in vertical axis. Therefore, the larger the difference between value of the black bar and that of a grey bar, the more important that feature is. It can be seen from Fig. 5, word-based features extracted from title is the most important, followed by ones extracted from body. Although non-word-based features and tags features have less contributions than

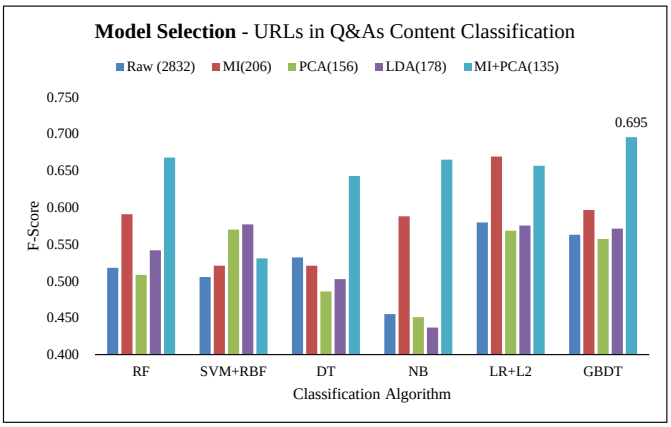

Fig. 7. Results of Model Selection for URLs in Q&As Content Classifier

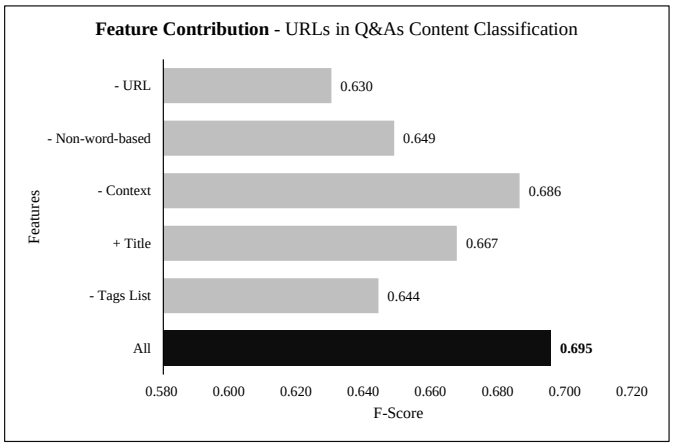

Fig. 8. Results of Feature Contribution for URLs in Q&As Content Classifier

features extracted from title and body, they also have positive effect for classification.

3) Model Comparison: This experiment compares the performance of SOLinker and that of the other method proposed by Lucas *et al.* [3], which only uses some manually selected keywords features and a few features extracted from xml. Features in SOLinker are extracted from various information, and the keywords selection is done by the MI selection step automatically. Also, SOLinker applies PCA to obtain some latent features. Fig. 6 illustrates results of model comparison by Precision, Recall and F-Score. In Fig. 6, black bars show results of SOLinker, while grey bars for those of Lucas et al.'s method. Because features they used are a subset of ours, values of Precision, Recall and F-Score of our method are all higher than those of Lucas et al.'s method. To sum up, the F-Score of semantic relation construction by SOLinker is around 77%, about 5% higher than Lucas *et al.*'s method.

C. URLs in Q&As classification

The experiment consists of three sub-experiments: the performance of URL Type Classifier, that of URL Content Classifier and how the number of context windows affects final results. Carlos *et al.* also labeled some URLs in Q&As and then studied the innovation diffusion [4], but labels are assigned manually, while SOLinker does this work automatically, so we cannot compare with their work.

1) Performance of URL Content Classifier: This experiment evaluates the performance of URL Content Classifier for URLs in Q&As. Fig. 7 illustrates results of model selection. Also, we compare the performance of different dimensionality reduction algorithms and classification algorithms. In Fig. 7, before doing dimensionality reduction, performances of all models are poor. MI feature selection still works well in reducing dimension. Finally, GBDT achieves the best performance, reaching 0.695 F-Score. When achieving this value, features are also first reduced by MI selection and then PCA. However, the F-Score of URLs in Q&A Content Classifier (0.772) is lower than that of URLs of Q&As Content Classifier (0.695). Because for URLs of Q&As, contents of them are known, which are Q&As themselves, while for URLs in Q&As, contents of them are unknown. To prevent crawling contents of them, which is time-consuming, SOLinker can only classify

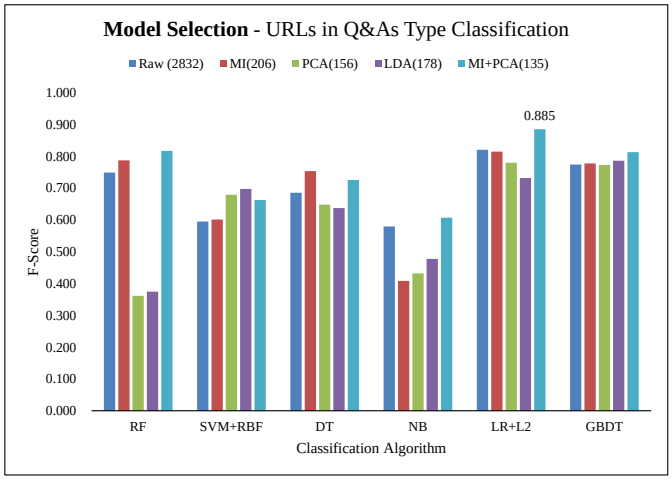

Fig. 9. Results of Model Selection for URLs in Q&As Type Classifier

them by their contexts and URLs. Therefore, information of URLs in Q&As providing to SOLinker to do the classification is less than that of URLs of Q&As, which is the reason that URL Content Classifier performs worse in URLs in Q&As than in URLs of Q&As.

Fig. 8 illustrates results of feature contribution. In URLs Content Classifier for URLs in Q&As, there are totally four kinds of features: tags features, non-word-based features, and two word-based features extracted from contexts and URLs. Compared with features to classify URLs of Q&As, title features are not used. Because from the Fig. 8, if title features are involved, the performance of model decreases from 0.695 to 0.667, which means that title features have negative contributions to classification. Because if there are many URLs in a Q&A, these URLs have the same title but may belong to different class, title will become noise from model learning. As for other features, URL features contribute the most, followed by non-word-based features and tags features.

2) Performance of URL Type Classifier: This experiment is used to evaluate the performance of URL Type Classifier. Fig. 9 illustrates results of model selection. It is clear that classifying URLs by their types achieve better performance than by their contents for URLs in Q&As. The best model

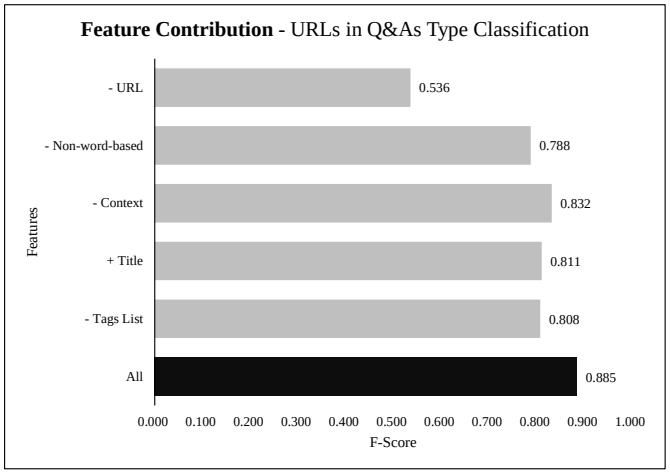

Fig. 10. Results of Feature Contribution for URLs in Q&As Type Classifier

TABLE II. How does the Windows Size Affect Final Results

n-window(s)	URL Type Classifier	URL Content Classifier
1	**0.885**	**0.695**
2	0.845	0.672
3	0.811	0.630
All	0.801	0.613

is Logistic Regression with L2-norm, at 0.885 F-Score. Also, the dimensionality reduction step greatly promotes the final performance of classification. Fig. 10 shows the feature contribution in URL Type Classifier. Same with URL Content Classifier, for URLs in Q&As, features extracted from title have negative contributions to model construction. However, features extracted from URL have dramatically positive affection, followed by non-word-based features. Because to classify URL by its type, the domain part of a URL plays an important roles. Through observing relations between features and types, domain "wikipedia" has a strongly correlation to label *Wiki*, domain "github" and "sourceforge" are related to label *Code Repository*, "blog" to label *Blog* and "stackoverflow" and "stackexchange" to label *Q&A Post*. Due to these strong correlated features, the performance of URL Type Classifier are better than that of URL Content Classifier.

3) Context Windows Affection: To classify URLs in Q&As, we extract word-based features from contexts. In Section IV, we define *n*-windows context. This experiment shows how this *n* affects the final performance. TABLE II shows results of this experiment, in which the first column is the value of *n* ("All" means we use full body of the Q&A), the second colum shows the F-Score of URL Type Classifier and the third column shows that of URL Content Classifier. From the table, when the value of *n* increases, the performance decreases in both URL Type Classifier and URL Content Classifier. From this experiment, we can conclude that context far from the URL has less semantic relations to it and is noise for classification. This is because if there are many URLs in a Q&A, and we use full body to extract word-based features, then all these URLs have a same feature vector. If labels of these URLs are different, then the same feature vector will correspond to different labels and thus becomes noise.

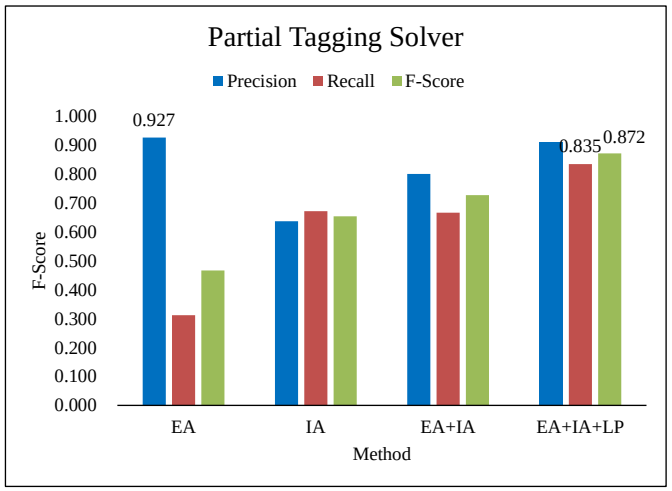

Fig. 11. Results of Different Algorithms when Obtaining Target Tags

D. Partial Tagging Solver

This experiment is to validate the performance of partial tagging solver. In partial tagging solver, firstly we apply Explicit Analysis (EA) and then Implicit Analysis (IA) before applying Label Propagation(LP) to obtain associated tags. Fig. 11 illustrates results of this experiment. From Fig. 11, if we only apply EA to solve this problem, it can achieve a high Precision (0.927) but low Recall (0.312). This is because, in many cases, there is no tag appearing in context, URL, title or anchor text directly. IA apply Naive Bayes to predict tags, but the performance of it is not very high. By combining IA and EA, the Recall is still not good. However, if we apply LP after EA and IA, the performance has a dramatic rise, with the highest Recall (0.835) and F-Score (0.872). This is because sometimes tags find by EA and IA are not in the tags list, but they have high similarity with tags in the tags list, which can be propagated from other tags by label propagation.

VIII. Conclusion and Future Work

A. Conclusion

In this paper, we propose SOLinker, a semantic linkage constructor between tags and URLs in/of Q&As. Firstly, SOLinker identifies proper relations from predefined relations set between tags and URLs, which is modeled as a text classification problem. Features are extracted from text of Q&A, the URL and tags, and classification algorithms are Logistic Regression and Gradient Boosting Decision Tree, depending on the category of URLs. Secondly, to address the partial tagging problem, we propose a semantic analysis method to analyze context of this URL and the URL itself from both implicit and explicit aspects. Then SOLinker will infer proper tags by the label propagation technique. Results show that our method is feasible and practical in constructing semantic relations between tags and URLs on StackOverflow.

B. Future Work

Our work can be further improved from following four aspects: (1) **More Semantic Relations**. The set of relations

can be further refined to construct more semantic annotations, which can be done by adding more labels to URL Content Classifier and URL Type Classifier or constructing other classifiers. (2) **Partial Tagging Problem of Relations**. SOLinker constructs the same semantic link between a URL and corresponding tags list. This step can be further refined by linking the URL to each tag in the tags list semantically. For instance, in Fig. 1, the question asker want to seek some libraries of machine learning; therefore, the *seek-something* link should be constructed only between the URL and the tag *machine-learning*. (3) **Multi-relations between a Tag and a URL for URLs of Q&As**. SOLinker only links the URL to a tag with one semantic relation. However, sometimes, there are multi-relations between a tag and a URL; for instance, some questions contains several sub-questions, some of them are *Seeking-Something* while some are *How-to-do-it*. (4) **Semantic Querying**. Once semantic annotations are constructed, they can promote the searching experience by semantic querying.

REFERENCES

[1] Bogdan Vasilescu, Alexander Serebrenik, Prem Devanbu, and Vladimir Filkov. How social q&a sites are changing knowledge sharing in open source software communities. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 342–354. ACM, 2014.

[2] Jiangang Zhu, Beijun Shen, Xuyang Cai, and Haofen Wang. Building a large-scale software programming taxonomy from stackoverflow. In *SEKE'2015: 27th International Conference on Software Engineering and Knowledge Engineering*, pages 391–396. Knowledge Systems Institute Graduate School, 2015.

[3] Lucas BL de Souza, Eduardo C Campos, and Marcelo de A Maia. Ranking crowd knowledge to assist software development. In *Proceedings of the 22nd International Conference on Program Comprehension*, pages 72–82. ACM, 2014.

[4] Christopher Gomez, Brendan Cleary, and Leif Singer. A study of innovation diffusion through link sharing on stack overflow. In *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, pages 81–84. IEEE, 2013.

[5] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[6] Zornitsa Kozareva and Eduard Hovy. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1482–1491. Association for Computational Linguistics, 2010.

[7] Michael Thelen and Ellen Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 214–221. Association for Computational Linguistics, 2002.

[8] Richard C Wang and William W Cohen. Language-independent set expansion of named entities using the web. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 342–350. IEEE, 2007.

[9] Olena Medelyan, Ian H Witten, Anna Divoli, and Jeen Broekstra. Automatic construction of lexicons, taxonomies, ontologies, and other knowledge structures. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(4):257–279, 2013.

[10] Sonal Gupta and Christopher D Manning. Improved pattern learning for bootstrapped entity extraction. *CoNLL-2014*, page 98, 2014.

[11] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[12] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[13] Xiao Ling and Daniel S Weld. Fine-grained entity recognition. In *AAAI*. Citeseer, 2012.

[14] Samuel I Daitch, Jonathan A Kelner, and Daniel A Spielman. Fitting a graph to vector data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 201–208. ACM, 2009.

[15] Bonan Min, Shuming Shi, Ralph Grishman, and Chin-Yew Lin. Ensemble semantics for large-scale unsupervised relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1027–1037. Association for Computational Linguistics, 2012.

[16] Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R Voss, and Jiawei Han. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 995–1004. ACM, 2015.

[17] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *Knowledge and Data Engineering, IEEE Transactions on*, 27(2):443–460, 2015.

[18] Charu C Aggarwal and ChengXiang Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[19] Shaowei Wang, Daniel Lo, Bogdan Vasilescu, and Alexander Serebrenik. Entagrec: an enhanced tag recommendation system for software information sites. In *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*, pages 291–300. IEEE, 2014.

[20] Walid Maalej and Hadeer Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pages 116–125. IEEE, 2015.

[21] Min-Yen Kan and Hoang Oanh Nguyen Thi. Fast webpage classification using url features. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 325–326. ACM, 2005.

[22] Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C Lee Giles. Researcher homepage classification using unlabeled data. In *Proceedings of the 22nd international conference on World Wide Web*, pages 471–482. International World Wide Web Conferences Steering Committee, 2013.

[23] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254. ACM, 2009.

[24] Anh Le, Athina Markopoulou, and Michalis Faloutsos. Phishdef: Url names say it all. In *INFOCOM, 2011 Proceedings IEEE*, pages 191–195. IEEE, 2011.

[25] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 681–688. ACM, 2009.

[26] Fabio Crestani. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11(6):453–482, 1997.

[27] Mo Wenkai, Shen Beijun, Chen Yuting, and Zhu Jiangang. Tbil: A tagging-based approach to identity linkage across software communities. In *22st Asia-Pacific Software Engineering Conference, APSEC 2015, Volume 1: Research Papers*, pages 56–63. IEEE, 2015.

[28] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[29] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[30] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[31] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.