A Discriminative Model Approach for Suggesting Tags Automatically for Stack Overflow Questions

Avigit K. Saha* Ripon K. Saha† Kevin A. Schneider*
*University of Saskatchewan, Canada
avigit.saha@usask.ca, kevin.schneider@usask.ca
† The University of Texas at Austin, USA
ripon@utexas.edu

***Abstract*—Annotating documents with keywords or 'tags' is useful for categorizing documents and helping users find a document efficiently and quickly. Question and answer (Q&A) sites also use tags to categorize questions to help ensure that their users are aware of questions related to their areas of expertise or interest. However, someone asking a question may not necessarily know the best way to categorize or tag the question, and automatically tagging or categorizing a question is a challenging task. Since a Q&A site may host millions of questions with tags and other data, this information can be used as a training and test dataset for approaches that automatically suggest tags for new questions. In this paper, we mine data from millions of questions from the Q&A site Stack Overflow, and using a discriminative model approach, we automatically suggest question tags to help a questioner choose appropriate tags for eliciting a response.**

***Index Terms*—Machine learning, automatic tagging, discriminative model**

I. Background and Motivation

In information systems, tagging is a popular way to categorize information and to search content. Therefore, almost all online newspapers, blogs, question-answer communities, and other similar sites make use of tags to categorize articles, posts, questions, answers, and so on. Similarly, Stack Overflow uses tags to categorize programming questions so that their users can find similar questions on the same topic or find questions that they may be able to answer. On Stack Overflow, a questioner can add up to five tags to categorize a question using existing tags or using new tags they create. However, a user must have a certain level of reputation to create a new tag, which is determined by the site using a reputation score. Appropriate tagging of questions may be useful to get a quick answer because potential responders are able to be notified when a question is posted with a tag related to their interests.

In order to investigate the current state of tagging questions on Stack Overflow, we first determined how fully questions are tagged. Our results (in Table I) show that almost 87% of the questions have less than five tags and 38.38% of questions have only one or two tags. We further investigated whether the number of tags and the number of times a question is viewed are related, since the chance of getting an answer may improve the more a question is viewed by the community. To this end, we calculate the average number of views per question for each group and plotted them on a graph, where the x-axis represents the number of tags and the y-axis represents the number of views per question (cf. Figure 1). Our results show that the average number of views per question gradually increases with the number of tags. We observe that the average number of views per question is about 555 when there is only one tag, but it is above 800 when there are four tags, and it is very close to 800 when there are five tags. We also observed that the more tags there are, the more often a question has an 'accepted' answer. A questioner can mark one of the answers to their question as 'accepted' when they consider the answer to be the most helpful for them personally. 62% of questions with one tag had accepted answers whereas 68% of questions with five tags had accepted answers. The increased viewing by number of tags and the increased accepted answers by number of tags motivated us to build an automatic system for tagging Stack Overflow questions appropriately.

There are three main advantages to having an automatic tagging system. First, it can be used to tag existing questions that have less than five tags. Second, it can help new questioners by suggesting appropriate tags. Third, it can warn questioners if they tend to select an inappropriate tag. In this paper, we mine data from millions of questions [1] from

TABLE I
Proportion of Questions by Number of Tags

Number of Tags	Number of Questions	[%]
1	438,475	12.70%
2	887,037	25.68%
3	997,906	28.89%
4	693,234	20.07%
5	437,090	12.66%

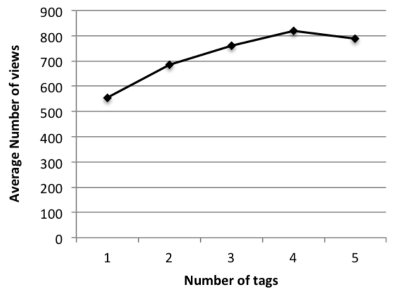

Fig. 1. Average views per quesion for the number of tags

the Q&A site Stack Overflow, and leverage recent advances on using discriminative models for information retrieval to suggest appropriate tags for eliciting a response. Based on an evaluation of 834 tags, we show that our approach can suggest tags with a mean accuracy of 68% ranging from 50% to 100% for individual tags.

The rest of the paper is outlined as follows. Section II describes our approach including how we process a dataset and build discriminative training models to suggest tags. In Section III we present our results showing the effectiveness of our automatic tagging system. Finally, we conclude our paper in Section IV with our directions for future research.

II. Our Approach

In general, there are three main steps in our approach, converting questions into vectors, training a discriminative model and suggesting tags for a given question.

A. Converting Questions into Vectors

In this step, we convert each question into a high-dimension vector space where each dimension corresponds to a unique word from the documents and the dimension value represents the weight of the associated word. To this end, we first perform some lexical analysis to break down the question text into tokens. After tokenizing we remove stopwords. Stopwords are those words that are very common in most of the documents such as *a*, *an*, *the*, *is*, *at*, *which*, and so on. Finally, we use *term frequency (TF)*, a common information retrieval term-weighting scheme, to measure the importance of a term in a question. *TF* corresponds to the number of times the term appears within the documents. We use Mallet [2] to process documents and to extract features and values.

B. Training Discriminative Model

1) Training Dataset: We take more than 1.3 million questions to prepare the training datasets for 834 popular tags (a popular tag is a tag that is related to more than 2000 questions). A previous study [3] suggests that a discriminative model learner produces the best result for a class that has 60 positive documents and 1500 negative documents. They also ensured that each document had more than 800 characters to provide enough information to the model. Therefore, to build the dataset for each tag, we randomly choose 60 questions associated with that tag and 1500 questions from some different tags. Then we convert each document into a feature vector as described in Section II-A.

2) Test Dataset: In order to test our models, we create an oracle of 16,680 questions by randomly choosing 20 questions for each of the 834 tags. For each tag, we take 10 positive documents and 10 negative documents. We ensured that the test documents were not also part of the training dataset. We also ensured that each question had more than 800 characters to make the test dataset compatible with the training dataset. Then we processed the test documents in the same way we processed the training dataset.

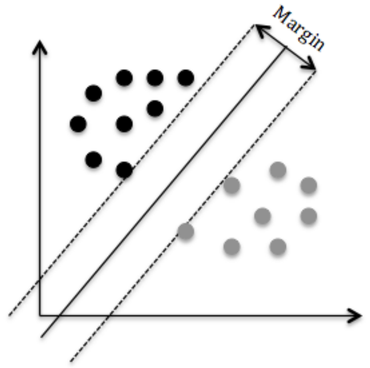

Fig. 3. Maximum-margin hyperplane for an SVM

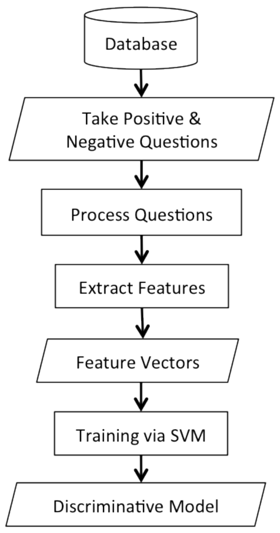

Fig. 4. Training Discriminative Model via SVM

3) Building a Discriminative Model via SVM: A Support Vector Machine (SVM) is a popular approach for building a discriminative model from a training dataset. A SVM constructs a hyperplane or a set of hyperplanes to classify patterns. Figure 3 shows a scenario using a maximum margin hyperplane for samples and Figure 4 shows the step by step flow chart to build a discriminative model from a given dataset. We use SVM^{light} [4], a popular implementation of SVM to build a discriminative model for each of the 834 tags that we considered based on the training dataset for that tag. Now these resulting discriminative models can be used to answer if a new question can fit with any of these existing tags.

C. Tag Suggestions

Once the discriminative models are built, our system is ready to suggest relevant tags for a new question. We use Algorithms 1 and 2 to suggest tags. Algorithm 1 retrieves and returns a list of probable tags for a question. It iterates over all models to calculate the similarity between a question and each model. On line 3 of Algorithm 1, Algorithm 2 is called to calculate the similarity between question Q and a given model m. Algorithm 2 extracts feature vectors for Q (line 2) and then, *SVMPredict* is invoked to obtain the predicted

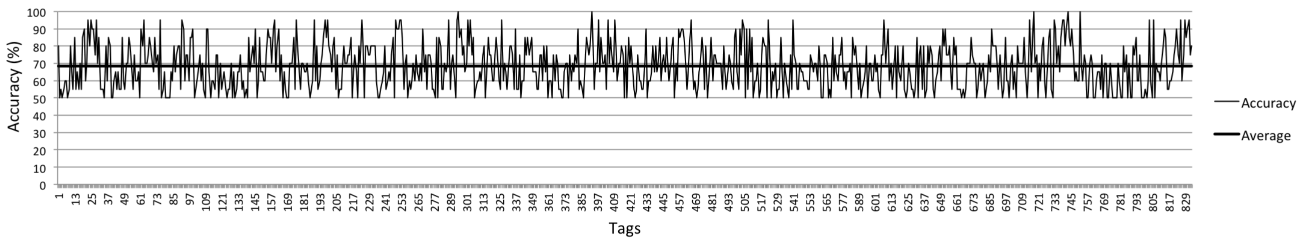

Fig. 2. Accuracy of Discriminative Modes for each Tag

Algorithm 1 Propose probable tags

```
Procedure proposeTags
Input: $Q$ : a new question; $M$ : all models
Output: $result$ : a list of suggested tags for $Q$
Body:
1: $listOfTags$ = a <key,value> map where key is a model tag $m$ and value is its $similarity$
2: for each model $m \epsilon M$ do
3:     $similarity = predict(Q, m)$
4:     add <$m$, $similarity$> to $listOfTags$
5: end for
6: sort $listOfTags$ in descending order of $similarity$
7: return sorted $listOfTags$
```

Algorithm 2 Calculate similarity between Q and a model

```
Procedure predict
Input: $Q$ : a new question; $m$ : a model
Output: $max$ : similarity between $Q$ and model $m$
Body:
1: $max = 0$
2: $vectors = featureVector(Q)$
3: $similarity = SVMPredict(vectors, m)$
4: return $MAX(max, similarity)$
```

probability from the discriminative model (line 3). On return to Algorithm 1 the similarity value for the question and model is added to the $listOfTags$. Finally, the $listOfTags$ is sorted in descending order of similarity and the sorted $listOfTags$ is returned.

D. Model Evolution

As programming languages are rapidly evolving with new features and technologies, questioners create new tags to ask questions on those topics. Therefore, our models should be updated periodically to incorporate those newly created tags into the system. The update process can be done automatically. A dedicated component of the system can monitor each of the newly created tags, and can build a discriminative model by creating a training dataset whenever a specific tag has 60 or more questions.

III. Results

In this section, we focus on following 3 research questions:

TABLE II
COMPARISON BETWEEN ORIGINAL TAGS AND SUGGESTED TAGS

Id	Original Tags	Suggested Tags
11744046	ios5, uiviewcontroller, uiscrollview	scrolling, uiscrollview, uikit, uiview, ipad, uiviewcontroller, cocoa-touch, uinavigationcontroller, ios5, textview, uitabbarcontroller, uitableviewcell, uiimageview, uitextfield
11709481	iphone, xcode, delegates, uitabbarcontroller, storyboard	uitabbarcontroller, uinavigationcontroller, tabs, xcode4.2, uikit, cocoa-touch, interface-builder, uiview, uiviewcontroller
11726620	ios, uibutton, uitabbarcontroller, tabbar, segue	uitabbarcontroller, tabs, uinavigationcontroller
11730588	iphone, storyboard, uitabbarcontroller, modalviewcontroller, uistoryboardsegue	uiviewcontroller, uiview, uitabbarcontroller, uinavigationcontroller, cocoa-touch, interface-builder
11738761	cocos2d-iphone, box2d-iphone	cocos2d, cocos2d-iphone, textview, xmpp
11749370	html, css	css
11738714	android, android-listview	android-listview, android-intent, checkbox, android-widget
11745574	cocoa-touch, uitableview, core-data, nsfetchedresultscontrolle	core-data, nsarray, ios4, iphone-sdk-4.0, xsd, uitableviewcell, iphone-sdk-3.0
11733482	datepicker, sencha-touch-2	extjs, datepicker, extjs4, sencha-touch
11737147	google-maps, google-maps-api-3	gwt, google-maps-api-3, google-analytics, shell, google-app-engine, nullpointerexception, jvm

A. How accurately do our models work for each tag?

To answer this question, first, we run our tagging system to build a discriminative model for each tag from the training dataset. Then we use that model on the corresponding test dataset to test each question for that tag. As we described in Section II-B2, the testing dataset contains 10 positive and 10 negative questions for each tag. We calculate the accuracy of each model with the following equation:

$$accuracy = \frac{number\ of\ correctly\ classified\ questions}{total\ number\ of\ questions} \quad (1)$$

For example, if a model can correctly recognize 16 questions from the test dataset for a given tag, the accuracy of the model is 80%. Figure 2 displays the accuracy (y-axis) of our model for each tag (x-axis). We see that the accuracy

TABLE III
MISSING TAGS

Id	Original Tags	Missing Tags
11748476	android	json
11733330	telerik	datetime, date
11732592	jquery	jquery-selectors, webforms
11718763	android	java
11699099	android	tcp,crash
11737147	google-maps, google-maps-api-3	gwt
11744046	ios5, uiviewcontroller, uiscrollview	uiview
11745574	cocoa-touch, uitableview, core-data, nsfetchedresultscontrolle	uitableviewcell

ranges from 50% to as high as 100%. The straight bold line represents the average accuracy, which shows that our system has an accuracy of 68.47%, on average, for all tags.

B. *How well can our system suggest tags for a given question?*

In the previous experiment, we validated our model for each tag. In this experiment, we investigate how well all the models in our system work together to suggest a set of tags for a given question. We chose questions randomly from the Stack Overflow dataset which have all the tags from the 834 tags that we considered. Then we invoke our tagging system to suggest tags for those questions. After obtaining the results, we manually compare all the suggested tags for a given question with the original tags. We intentionally did not make string based automatic comparisons because there are many different forms of the same tag due to the use of synonyms and other word separating characters such hyphen, underscore, etc. Furthermore, someone asking a question may not necessarily know the best way to categorize or tag the question. Table II shows some concrete examples from our experiment. From the results, we observe that in most cases our system chose the appropriate tags. For example, for the question *id* 11744046, the questioner chose *ios5*, *uiviewcontroller*, *uiscrollview*, which were also suggested by our system. Other samples in Table II show that our tagging system suggests sensible tags for most of the questions.

C. *Can our model suggest missing tags?*

Stack Overflow allows a questioner to add up to 5 tags per question. However, more than 438K questions have only one tag and more than 887K questions have just two tags. In this experiment, we investigate whether there are some more tags that would be appropriate to add. To this end, we chose questions with less than 5 tags randomly and invoke our tagging system to obtain tag suggestions. Finally, we manually investigate the questions to see if the suggested tags are appropriate. We found that there are many questions that could have more tags than they already have. Table III shows some of the examples that we investigated in our experiment. We see that the question *id* 11733330 had only the tag *telerik*. The title of the question is "Disabling Holidays in a Calendar". It is clear from the title that the question is about date and time. By further studying the full question we found that *datetime* and *date* are both important tags to add. Interestingly, they are both suggested by our tool. We also see that *datetime* and *date* are popular tags. There are 8794 questions that are tagged as *datetime* and 8770 questions that are tagged as date which are more than that of the *telerik* tag. Another example is the question 11737147 which has two tags: *google-maps* and *google-maps-api-3*. The title of the question is "GoogleMaps, calling method throws 'TypeError"'. While reading the question we see that the error provided by the IDE is `at com.google.gwt.core.client.JavaScriptException: (TypeError)`. So, it is clear that `gwt` (Google Web Toolkit) is an important tag for this question. From the suggested tags, we see that our system put this tag at the top of the list. We found many other questions with missing relevant tags. A questioner can miss relevant tags for a number of reasons. For example, they may be busy, may forget, may not find appropriate words for tagging, or may not be aware of some popular tags. We believe that an automatic tag suggestion systems like ours can be very helpful in these situations.

IV. CONCLUSION

Investigating millions of questions from a popular Q&A site, Stack Overflow, we see that almost 87% of the questions have less than the possible five tags. As well, over one-third of the questions (38.38%) have only one or two tags. We also see that the average number of views per question increases gradually with the number of tags, which increases the chance of getting accepted answers. We propose a discriminative model approach that assists questioners using Stack Overflow by automatically suggesting relevant question tags. Mining data related to millions of questions from the database, we collected training datasets and built a discriminative model for each of 834 popular tags. Using these models our tagging system suggests tags for new questions with an average accuracy of 68.47% on the test dataset. We show that our tagging system suggests tags closely related to those added by users. We also show that our tagging system is able to find missing tags (tags that are neglected to be added by questioners or other users). However, we found that although our tagging system shows more than 85% accuracy for specific tags (e.g., uitableview), it sometimes does not work well with general tags (e.g., java). Using a large number of good quality positive documents may help resolve this issue. Future work includes tuning the models by manual investigation to improve accuracy and further exploring the relationship between tags and responses.

REFERENCES

[1] A. Bacchelli, "Mining challenge 2013: Stack overflow," in *The 10th Working Conference on Mining Software Repositories*, 2013, p. to appear.

[2] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002, http://mallet.cs.umass.edu.

[3] J. Wang and B. D. Davison, "Explorations in tag suggestion and query expansion," in *Proceedings of the 2008 ACM workshop on Search in social media*, ser. SSM '08. New York, NY, USA: ACM, 2008, pp. 43–50. [Online]. Available: http://doi.acm.org/10.1145/1458583.1458592

[4] T. Joachims, "Making large-scale svm learning practical." in *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999, software available at http://svmlight.joachims.org/.