Predict Closed Questions on StackOverflow

© Galina E. Lezina
Ural Federal University
galina.lezina@gmail.com

Artem M. Kuznetsov
Ural Federal University
whoisnexta@gmail.com

Abstract

”Millions of programmers use StackOverflow to get high quality answers to their programming questions every day. There has evolved an effective culture of moderation to safe-guard it. More than six thousand new questions is asked on StackOverflow[1] every weekday. Currently about 6% of all new questions end up ”closed”. The goal of this paper is to build a classifier that predicts whether or not a question will be closed given the question as submitted, along with the reason that the question was closed.

1 Introduction

In recent time question-answer services like StackOverflow are becoming more popular. Knowledge of such services has been steadily growing so it requires more resources to moderate. Some automation of this process would ease this task. The problem solved in this paper is a small step in this direction. The task was a contest organized by kaggle[2]. It has two stages: public and private, for public and private datasets accordingly. In first one user who submitted solution could see their results immediately, but they could do it no more than 2 times a day. In private stage users were doing prediction for private dataset but results can be seen only after the competition. The results of all participants can be seen in public leaderboard, but those who submitted post-deadline are not shown there. We submitted our solution after deadline and the best position we've got is 5th with 0.31467 points.

StackOverflow is a service where users ask questions about programming and it belongs to StackExchange network which contains many thematic websites. Questions on StackOverflow can be closed as off topic (OT), not constructive (NC), not a real question (NRQ), too localized (TL) or exact duplicate. Exact duplicate reason was excluded from competition because it depends on posts history. Posts history actually is present in StackOverflow database dump but its size is about 6GB in xml format, which requires many resources to analyze.

Off topic is a question that is not on-topic of the site or is related to another site in Stack Exchange network.

Example: Is there a way to turn off the automatic text translation at the MSDN library pages ? I do prefer English text but due to having a German IP address Microsoft activates the automatic translation on every new page load which gives me a yellow box with a German translation of the text I am currently hovering over with the mouse.

This happens regardless what language is initially set in the right upper corner and regardless of whether I am logged in or not. I can't tell how annoying this is !! Any ideas, anyone ?

Too localized is a question that is unlikely to be helpful for anyone in the future; it is only relevant to a small geographic area, a specific moment in a time, or an extraordinary narrow situation that is not generally applicable to the worldwide audience of the internet.

Example: Is it time to start using HTML5? Someone has to start sometime but is now the time? Is it possible to use the new HTML5 tags and code in such a way as to degrade gracefully?

Not constructive is a question that is not a good fit to Q&A format. It is expected that the answers generally involve facts, references, or specific expertise; this question will likely solicit opinion, debate, arguments, polling, or extended discussion.

Example: What is the best comment in source code you have ever encountered?

Not a real question is a question when it's difficult to tell what is being asked here. This question is ambiguous, vague, incomplete, overly broad or rhetorical and cannot be reasonably answered in its current form.

Example: For a few days I've tried to wrap my head around the functional programming paradigm in Haskell. I've done this by reading tutorials and watching screencasts, but nothing really seems to stick. Now, in learning various imperative/OO languages (like C, Java, PHP), exercises have been a good way for me to go. But since I don't really know what Haskell is capable of and because there are many new concepts to utilize, I haven't known where to start. So, how did you learn Haskell? What made you really ”break the ice”? Also, any good ideas for beginning exercises?

The process of question closing includes user voting. Thus users with a certain reputation can vote a question to be closed with one reason. When question gains enough close votes it is closed by moderator. So this can be automated if it will be possible to predict which question will be closed.

2 Dataset

For this task the data was provided by kaggle and it includes train data which contains 3664927 posts and train

Proceedings of the Ninth Spring Researcher's Colloquium on Database and Information Systems, Kazan, Russia, 2013

[1] http://stackoverflow.com
[2] https://www.kaggle.com

sample data consisting of 178 351 posts. Full train data and sample train data distribution on closed reasons is shown in table 1 .

Table 1: Training data distribution over categories

Dataset	NRQ	NC	OT	Open	TL
Train	38622	20897	20865	3575678	8910
Sample	38622	20897	20865	89337	8910

Also StackOveflow database dump of august 2012 was available. Database dump contains all users and posts information including history of the post editing, commenting and many other information.

3 Related work

User interaction analysis in social media. As was mentioned above questions on StackOverflow are closed by user voting. So user's feedback is very important component also it's a valuable source of post quality. In [1] user relationships were analyzed to gain significant amount of quality information. Authors applied link-analysis algorithms for quality scores propagating; the main idea was that "good" answerers write "good" answers. This idea can be propagated onto the questions that peoples who do not asks "bad" questions are less unlikely to do so in the future. In process of link-analysis user-user graph was built to represent those relationships. This graph can be noted as $G = (V, E)$ in which V is a set of vertices stands for users set, and E represents relationships between the users. In [4] authors classified questions as conversational and informational. In their work they divided peoples into two categories: answer people, who answers many questions and discussion peoples who interact often with other discussion people. To do so they also analyzed user's question answers ego-network. Authors of [7] showed that almost the same user interaction features are significant during classification of a question as social and non-social.

Text content quality analysis. In [1] were presented features to represent grammatical properties of the text. In their work they also take into account punctuation and typos, syntactic and semantic complexity. It's important because this content is generated by the users. Their features for text quality analysis were helpful for us because one of the close reason - not constructive - is essentially a conversational question.

4 Used methods

We've compared three methods during our research.

4.1 Random forest

We took baseline's scikit-learn random forest with 50 estimators implementation and used it with our new features to see how these new features may affect the result predictions.

4.2 Support Vector Machine

We've used liblinear[3] library support vector machine implementation. Support Vector Machines (SVM) have been shown to be highly effective at traditional text categorization [5]. We chose this because of amount of data. As mentioned above it is slightly less than 4 millions of samples and we didn't balanced data as we did for random forest classifier. Liblinear do not use kernels and is trained very quickly. Liblinear also provides an option to select regularization parameter C. The value for C parameter that we found to be optimal for our dataset is 1.

4.3 Vowpal Wabbit

Vowpal Wabbit (VW)[4] is a library and algorithms developed at Yahoo! Research by John Langford. VW focuses on the approach to stream the examples to an online-learning algorithm [6] in contrast of parallelization of a batch learning algorithm over many machines. The default learning algorithm is a variant of online gradient descent. The main difference from vanilla online gradient descent is fast and correct handling of large importance weights. Various extensions, such as conjugate gradient (CG), mini-batch, and data-dependent learning rates, are included. We found that default algorithm works much better on our dataset. We trained VW with samples in chronological order and for reasons of clarity in shuffled order and the result for the shuffled data were much worse - 0.3340 versus 0.31467 for ordered dataset in condition that we used the same feature set for both of them.

As mentioned above the algorithm used in Vowpal Wabbit is a modified stochastic gradient descend algorithm. Unlike the typical online-learning algorithms which have at least one weight for every feature the approach used in VW allows to induce sparsity in learned feature weights. The main idea of truncate gradient is that it uses the simple rounding rule of weight to achieve the sparsity. The most of the methods rounds small coefficients by threshold to zero after a specified number of steps. In truncated gradient amount of shrinkage is controlled by a gravity parameter g_i. Weights are updated in according with update rule $f(w_i)$:

$f(w_i) = T_1(w_i - \eta \nabla_1 L(w_i, z_i), \eta g_i, \theta)$

where $T_1(v, \alpha, \theta) = [T_1(v_1, \alpha, \theta), ..., T_1(v_d, \alpha, \theta)]$ with

$$T_1(v_j, \alpha, \theta) = \begin{cases} max(0, v_i - \alpha) & \text{if} v_j \subseteq [0, \theta] \\ min(0, v_j + \alpha) & if v_j \subseteq [-\theta, 0] \\ v_j & otherwise \end{cases},$$

θ is a threshold,

g_i is a gravity parameter so $g_i = 0$ if $\frac{i}{K}$ is not an integer and $g_i = K$ if $\frac{i}{K}$ is an integer. Here K is the number of steps after which the weights are updated. g_i is used with θ to control sparsity.

η_e is a step size and calculated as

$\eta_e = \frac{l d^{n-1} i^p}{(i + \sum_{e' < e} i_{e'})^p}$,

where l is a learning rate, d is a decay learning rate, i is an initial time for learning rate, p is a power of learning rate decay.

The update rule parameter were chosen empirically and it's values is: *logistic loss function*, p = 0.5, l = 1, d = 1

Also VW provides online latent Dirichlet allocation algorithm which we used for 200 topics. 200 topics were

[3] www.csie.ntu.edu.tw/čjlin/liblinear

[4] https://github.com/JohnLangford/vowpal_wabbit/

optimal for our data. We've tried for 50, 100, 200 and 300 values, but 200 gave the best result.

4.4 Baseline

Baseline model was provided by kaggle. It includes six features to represent each post as a vector of features:

OwnerUndeletedanswersAtPostCreation. This is the count of answers posts the user had made that were undeleted when that row's question was submitted.

BodyLength. This is the initial body length including its code blocks lenth.

ReputationAtPostCreation. User reputation at post creation time.

NumTags. Number of tags that the assigns to the post. Its maximum value is 5 per post.

TitleLength. Title length of the post.

UserAge. This is the system user age. Not actual user age which user fills in his profile, but the time elapsed from the time a user logs into the system.

Classification was carried out using scikit-learn random forest (RF) implementation (50 estimators) for training data sample. The output is a raw prediction which then is updated to get the posterior probability because it was trained on balanced data. The posterior probability for the modified model is

$P(C|D,T) = P(C|D,S)\frac{P(C|T)}{P(C|S)}$,

where $P(C|S)$ is the old prior (frequencies of closed questions in balanced sample), $P(C|D,S)$ is the classification outputs (posterior) from the model which is based on S and $P(C|T)$ is the new prior (frequencies of closed questions in unbalanced train data). Here S means trained on balanced sample data and T means training on unbalanced train data.

So classifier infers the probability of class C (distribution over a parameter C - close reason in our case) which is denoted as $P(C|D,S)$ based on explicit data D (the data to be predicted) and on a prior balanced data S. We need to modify this prior to get the probability of class C for modified model $P(C|D,T)$ based on data D and on unbalanced data T.

Baseline does not take into account the content of the post, so we decided to focus on this. We presented text in vector from using two techniques. First is tf-idf weighting technique; the second is Latent Dirichlet Allocation (LDA) [2]. We used these techniques with different previously described methods. For building LDA model for train data we've been using GibbsLDA++[5] implementation using Gibbs Sampling technique for parameter estimation and inference. At the same time Vowpal Wabbit has its own online LDA implementation.

5 Features

As was mentioned above along with baseline features we used our features to represent data as vectors. All these features can be divided into three categories. We describe just some interesting features. Full list of features can be found in the full version of paper which may be obtained by e-mailing[6] to the author.

[5]http://gibbslda.sourceforge.net/

[6]galina.lezina@gmail.com

5.1 User Features

User features describes user parameters on StackOverflow server such as reputation, personal information completeness and interaction between all users. To take into account interaction between users we calculate some features by building user interaction graph. Along with baseline user features we calculated are listed below:

Reputation. User reputation at the time of the StackOveflow database dump creation. The idea is that users with high reputation ask incorrect questions less often than users with low reputation.

AgeFilled, *AboutMeFilled*, *LocationFilled*, *WebsiteFilled*, *AllInfoFilled.* These features are binary and correspond to filled information in user's profile. We tested if users with fully filled profile more

UpVotes. Votes up the user received for his posts.

DownVotes. Votes down the user received for his posts.

CVInDegree, *CVOutDegree.*. Close votes received by the user for his posts, Close votes given by the user for other user's posts.

QAInDegree. Number of answers received by the user.

QAOutDegree. Number of answers given by the user.

QAClustCoef. The clustering coefficient of the question-ask ego network. The hypothesis was that users with high QA clustering coefficient are more communicable. In [4] it was summarized that users asking conversational questions have more densely interconnected than users asking informational question. In our case not constructive category questions are conversational in fact.

5.2 Features of a Post

Post features include information about the title and body contents. In addition to direct representation the text as a vector using tf-idf or LDA we calculate so-called text parameters along with post features from baseline:

CBCount. Number of code blocks in the post's body.

LinkCount. Number of links in the post's body.

Dates, *Times.* The number of occurrences of dates and time periods in the body of the post. As described in StackOverflow FAQ, the too localized posts is closed as time or place specific.

NumberOfDigits. Number of digits in the post body.

NumberOfSentences. Number of sentences in the post body excluding code blocks.

NumberOfSentencesStartsWithI. Number of sentences in the post body which start with "I".

NumberOfSentencesStartsWithYou. Number of Sentences in the post which start with "you". In [4] it was concluded that conversational questions are more often directed at readers by using word "you", while informational questions are more often focused on the asker by using the word "I".

UpperTextLowerTextRatio. The ratio of the number of upper letters to lower letters in the text. Besides of text it may give some answerer characterization like accuracy.

FirstTextLineLength. Length of the first text line. Usually first short line implies personal appeal or greeting. The former case is the most interesting if it is peculiar to one of the close categories.

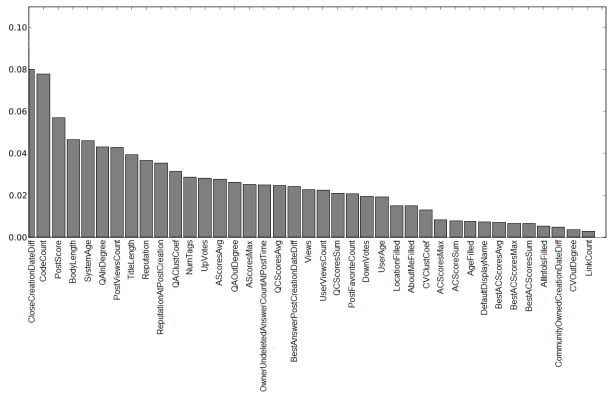

Figure 1: Relative feature importance.

NumberOfInterrogativeWords. Number of interrogative words such as "what", "where" and so on.

NumberOfSentencesStartsWithInterrWords. Number of sentences which starts with interrogative words. It provides an information on whether the post is a question.

NRQCloseRate, *NCCloseRate*, *OTCloseRate*, *OCloseRate*, *TLCloseRate.* We encounter close rate for each category. These values are calculated for every 10000 posts. For predicting samples we take latest close rate values.

The rest of features include such information as punctuation marks, indentation in code blocks, some features from post title and many others are described in paper linked at the beginning of this section.

5.3 Tag Features

During prediction we've been using information about the tags affixed to the posts by its owner. Every new question asked on service must be tagged with at least one tag. During classification we counted close frequencies for each tag and each category. Also we counted questions close frequency for every tag pair. The hypothesis is that tags reflect some topics of the post and a pairwise occurrence of some tags can mean that two topics that lead to disputes may occur in one post.

5.4 Feature Selection

To select most important features we used two approaches. The first method is based on estimating relative importance of features by constructing big amount of trees for randomly selected subsets of features and is described in [3]. The result was used while classifying data using random forest and support vector machine classifiers as the output of this method can be used to train any classifier.

The example is shown in figure 1. This figure shows relative importance for some features described earlier in the text. As we can see from the figure the most relevant feature is a time elapsed since the opening of the post before it was closed. The longer post is open the less likely that it will be closed. The next is the number of code blocks in the post, its score and so on. While selecting features we also measured importance in combination with LDA topics and tf-idf weights.

The second approach is a feature selection using the Vowpal Wabbit. VW has a small wrapper around it called vw-varinfo. VW-varinfo produces input variable names as an output and any other parameters including the relative distance of each variable from the best constant prediction - feature relevance score. So in final VW training we removed all features with zero relative scores.

6 Evaluation

Prediction results were evaluated for the public and private data provided by kaggle[7] using metric called multiclass logarithmic loss (logloss). Class labels for public and private data is hidden and still is not accessible for participants. So the result for these data can be calculated only on the kaggle's server using logloss metric.

6.1 Multiclass Logarithmic Loss

The metric is negative log likelihood of the model that says each test observation is chosen independently from a distribution that places the submitted probability mass on the corresponding class, for each observation. Multiclass logarithmic loss (MLL) is calculating by the following formula.

$MLL = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j} ln(p_{i,j})$,

where N is the number of observation, M is the number of class labels, $y_{i,j}$ is 1 if observation i is in class j and 0 otherwise, and $p_{i,j}$ is the predicted probability that observation i is in class j.

6.2 Outline the results

In table 2 we present results for different methods we used and for clarity best public leaderboard result also provided.

Table 2: MLL for used methods

Method	MLL	Leaderboard position
leader	0.29837	1
vw+uf+tf+lda200	0.31467	5
vw+uf+tf+tf-idf3+lda200	0.31994	5
vw+if+uf+tf+lda200	0.31795	5
vw+tf+lda200	0.31909	5
vw+uf+tf+tf-idf3	0.34141	15
vw+uf+lda200	0.44828	35
vw+lda200	0.45577	35
svm+tf-idf3+tf+uf+if	0.38152	24
svm+lda200+tf+uf+ if	0.41366	29
rf+tf-idf3+tf+uf+if	0.44045	34
rf+lda200+tf+uf+if	0.44846	35
baseline	0.46102	36

Here in table of results

uf means user features which are described in User Features section except user interaction features which is marked in results separately,

tf is a text features which are fully described in Post Features section and it also includes tag features,

lda200 is a representation of text as vector of topics probability distribution and includes post title and post body text along with tags attached to it,

[7]https://www.kaggle.com/c/predict-closed-questions-on-stack-overflow/data

if is a user interaction features and includes question-answer and close vote features such as in-degree, out-degree and clustering coefficient.

rf is a scikit-learn random forest implementation used with 50 estimators,

svm is a support vector machine

vw is a vowpal wabbit library.

As we can see from the table of results for vowpal wabbit user interaction features worsen outcome for a small value. And if we compare using vw with user features and text features (along with lda200 in both cases) we will see that text features contribute much more to the result than user features.

The model of Vowpal Wabbit which gave us the best result utilizes a logistic loss function and one against all classification. Also we measured the result for tf-idf vectors counted for 3-grams and it is interesting that it didn't outperformed our result for the model which uses only LDA for 200 topics. Also the SVM and RF classifiers require preliminary LDA model construction which requires a lot of resources while VW has online LDA implementation so it can build it while training.

As we mentioned earlier baseline doesn't take into account the content of the post and as we've seen the text feature is very informative. So our solution benefits from the post context.

7 Future work

After some manual analysis we've noted that some questions' status is open but it actually should be closed. Sometimes it's reflected in the comments and it would be useful to consider such close recommendations from comments. Such information can be taken from StackOveflow database dump. So we are planning to make better use of the database dump not only for further text feature extraction but for seeing user communication. Posts can be edited by different users depending on who owns the post - community or owner, so we can watch changes which make "bad" post (which received some votes for closing) to be a "good" one.

Also it's very hard to determine too localized question in some case because sometimes it can be seen from the text of the post, and sometimes it is enough to look at the code included in the post body. But we didn't analyze the content of the code in any way during classification. It is nontrivial task to determine if the code works for specific conditions and will never be useful for anyone else in the future. Some code analysis might be helpful because in StackOverflow the code includes actually not only the code but some stack trace is also considered to be a code but when user posts full stack trace he can ask very specific question associated with his error. Determining type of code language may be helpful if we are trying to see if user compares the same thing in different language which is like to ask "what is better" but this is not constructive.

References

[1] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. *Proc. of the 2008 International Conference on Web Search and Data Mining*, pages 183–194, 2008.

[2] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.

[3] M. Draminski, A. Rada-Iglesias, S. Enroth, C. Wadelius, J. Koronacki, and J. Komorowski. Monte carlo feature selection for supervised classification. *Bionformatics*, pages 110–117, 2008.

[4] F. Maxwell Harper, D. Moy, and J. Konstan. Facts or friends?: distinguishing infromational and conversational questions in social q&a sites. *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 759–768, 2009.

[5] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *Proc. of the European Conference on Machine Learning (ECML)*, pages 137–142, 1998.

[6] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *The Journal of Machine Learning Research*, pages 777–801, 2009.

[7] E. Rodrigues and N. Milic-Frayling. Socializing or knowledge sharing?: characterizing social intent in community question answering. *Proc. of the 18th ACM Conference on Information and knowledge management*, pages 1127–1136, 2009.