Predicting closed questions on Stack Overflow

2012-10-05

This time we enter the Stack Overflow challenge, which is about predicting a status of a given question on SO. There are five possible statuses, so it's a multi-class classification problem.

We would prefer a tool able to perform multiclass classification by itself. It can be done by hand by constructing five datasets, each with binary labels (one class against all others), and then combining predictions, but it might be a bit tricky to get right - we tried. Fortunately, nice people at Yahoo, excuse us, Microsoft, recently relased a new version of Vowpal Wabbit, and this new version supports multiclass classification.

In case you're wondering, Vowpal Wabbit is a fast linear learner. We like the "fast" part and "linear" is OK for dealing with lots of words, as in this contest. In any case, with more than three million data points it wouldn't be that easy to train a kernel SVM, a neural net or what have you.

VW, being a well-polished tool, has a few very convenient features. A particularly useful one is automatic hashing - it means that you don't need to convert categorical features into sparse representations of one-hot vectors (how does it sound?) like you would in LibSVM format:

```
1 23:1 67:1 856:1 3372:1 4200:1 8345:1 23423:1
```

Instead, you just give VW the words:

```
1 | I'd like to have an argument, please.
```

Or maybe, after some pre-processing, in robot speak:

```
1 | id like to have an argument please
```

More readable than numbers, isn't it? Here is the game plan:

1. pre-process CSV data
2. convert it into VW format
3. run learning and prediction
4. convert output into submission format
5. profit

If you'd like to do some tweaking, validation is a must. Therefore, there would be two additional steps: split the training set into training and validation, and at the end compute the score. A metric for this contest is multiclass logarithmic loss, which is easy enough to implement - see the code.

Ready? Let's get pre-processing CSV files to get CSV files slightly more fitting our purposes:

```
extract.py train.csv train_.csv
```

```
extract.py public_leaderboard.csv test_.csv
```

Then, converting them into Vowpal Wabbit format. We will use words (title, body, tags) as features, and also user reputation and post count. If you're short on disk space, you can get rid of post bodies and still do OK.

```
csv2vw.py train_.csv train.vw
```

```
csv2vw.py test_.csv test.vw
```

Now it's time to train:

```
vw --loss_function logistic --oaa 5 -d train.vw -f model
```

It means: use logistic loss function, *one against all* classification with five classes, data from train.vw and save the resulting model in a file called *model*. At this point, you get some numbers:

```
average    since         example     example  current  current  current
loss       last          counter      weight    label  predict features
0.333333   0.333333          3       3.0          4        4       10
0.166667   0.000000          6       6.0          4        4       12
0.181818   0.200000         11      11.0          4        4        8
0.090909   0.000000         22      22.0          4        4        7
0.136364   0.181818         44      44.0          4        4       22
0.080460   0.023256         87      87.0          4        4        7
0.074713   0.068966        174     174.0          4        4       13
0.094828   0.114943        348     348.0          4        4       15
0.099138   0.103448        696     696.0          4        4       16
0.079741   0.060345       1392    1392.0          4        4        7
0.075790   0.071839       2784    2784.0          4        4       14
0.063218   0.050647       5568    5568.0          4        4       14
0.056668   0.050117      11135   11135.0          4        4       11
0.048184   0.039698      22269   22269.0          4        4        8
0.036980   0.025777      44537   44537.0          4        4       17
0.027696   0.018412      89073   89073.0          4        4        7
0.019512   0.011328     178146  178146.0          4        4       22
0.013685   0.007859     356291  356291.0          3        4       13
0.011331   0.008976     712582  712582.0          4        4        8
0.013039   0.014746    1425163 1425163.0          4        4        9
0.019277   0.025516    2850326 2850326.0          4        4       10

finished run
```

Probably the most important is the first column, called *average loss*. It shows the value of loss function as the learner goes along with example counter. Usually, the smaller the loss, the better. Here we're getting pretty low numbers, so let's see how it turns out. We'll use the model to get predictions:

```
vw --loss_function logistic --oaa 5 -i model -t -d test.vw -r raw_predictions.txt
```

The switches are: -i for model file, -t means "do not train, just predict", and -r specifies the file to output raw predictions. We need them raw because otherwise we'll only get a predicted class. The raw predictions look as follows - first a line with values for each class, then a line with post id (we saved post id earlier):

```
1:-1.58226 2:-2.73852 3:-4.86114 4:2.0419 5:-4.73838
 4
1:-5.56301 2:-7.46127 3:-6.26342 4:3.23339 5:-6.79117
 145
1:-5.25797 2:-6.82686 3:-5.11928 4:3.65003 5:-6.38332
 231
```

All we need now is to run these raw values through sigmoid function to obtain probabilities, and optionally normalize them so they sum up to one (Kaggle will do this automatically, but for validation the task is upon us).

```
sigmoid_mc.py raw_predictions.txt p.txt
```

```
4,0.150503851876,0.0536225867259,0.00678218723343,0.781431031164,0.00766034300002
145,0.00394282781518,0.0005926764116,0.00196092272855,0.992345861757,0.00115771128812
231,0.0052388258145,0.00109561525294,0.00601356189425,0.98594582605,0.00170617098785
```

All this takes a few minutes to run (a little longer to write from scratch) and will get you a score around 0.18. Submit and enjoy.

Update - march 2013

We re-run the model on the final data. The numbers in training look slightly worse:

```
(...)
0.015177   0.012582     712582  712582.0         4         4       107
0.018332   0.021487    1425163 1425163.0         4         4        87
0.022005   0.025678    2850326 2850326.0         4         4        90

finished run
```

But good enough for 11th place, if we did this a bit earlier:

10	-	**Marco Lui**	0.32886	9	Thu, 01 Nov 2012 01:22:08
-		**Foxtrot**	0.33396	-	Wed, 06 Mar 2013 21:55:47
11	-	POST-DEADLINE ENTRY	0.33401	6	Sun, 28 Oct 2012 10:24:53
12	-	✖ If you would have submitted this entry during the competition, you would have been around here on the leaderboard.	0.33656	4	Thu, 01 Nov 2012 20:22:50
13	-		0.33672	8	Sat, 27 Oct 2012 08:10:24

Only recently have we learned that Marco Lui in 10th built on our approach and also shared his improvements and code.

We would like to thank Ariel Faigon and John Langford for help in understanding the workings of Vowpal Wabbit.

Posted by Zygmunt Z. 2012-10-05 Kaggle, VW, big-data, code, software

« Best Buy mobile contest - full disclosure Merck challenge »

Comments

anon • 2 years ago

What is the extract.py function that you use for "preprocess the csv file"? What does it do and where do I get it?

Reply • Share ›

ZygmuntZ Mod anon • 2 years ago

https://github.com/zygmuntz...

Reply • Share ›

Ilyas • 3 years ago

Hi! I used libsvm2vw.py on the data here:http://www.kaggle.com/c/wis...
It transforms the test set and crushes on the train set file
Any thoughts??

Reply • Share ›

ZygmuntZ Mod Ilyas • 3 years ago

There seems to be a line in the file consisting of "200" by itself.

Modified the script to skip malformed lines.

Reply • Share ›

Ilyas ZygmuntZ • 3 years ago

thank you!

Reply • Share ›

ZygmuntZ Mod • 4 years ago

Benjamin, this post was written before the competition entered its final phase. So at one time, the score was 0.18. This article still enjoys some interest and seems to be one of more popular ones on this site, so we will probably re-train the model on the final data and post the results.

Reply • Share ›

Benjamin Haley • 4 years ago

You say the score for this method is 0.18. The winning score on the leaderboard is 0.29. Are you implying that you beat the top posted score or am I missing something?

leaderboard: http://www.kaggle.com/c/pre...

^ | v • Reply • Share ›

Log0 • 4 years ago

Hi Zygmunt,

I run on the smaller train data (train-sample) but the average loss goes to about 0.33. Even if the training set is small, it shouldn't yield so high average loss?

I'm interested in learning more on ML but could not find a contact email from you. Appreciate if you could drop an email. Thanks!

^ | v • Reply • Share ›

FASTML - MACHINE LEARNING MADE EASY

Classifying text with bag-of-words: a tutorial

2 comments • 2 years ago •

Ros... — Thank you for the great article!It covers fundamental aspects of the text in machine learning in a clear and concise way. Do you think

Piping in R and in Pandas

2 comments • 3 months ago •

E... — The pipe operator is by Stefan Bache but really came into its own with dplyr. https://github.com/smbache/...

Project RHUBARB: predicting mortality in England using air quality data

1 comment • a month ago •

D... — Hello! If I may sugest something, probably an overkill.Once I found it useful to encode time using: sun zenith, sun azimuth,

Adversarial validation, part two

20 comments • a year ago •

Gustavo Pereira — I did something similar, but was able to get a better score by discarding the first 10% of train samples which looked least like

Subscribe Add Disqus to your siteAdd DisqusAdd Privacy DISQUS

Recent Posts

- Project RHUBARB: predicting mortality in England using air quality data
- Tuning hyperparams fast with Hyperband
- How to use pd.get_dummies() with the test set
- Data in, predictions out
- On chatbots
- Piping in R and in Pandas
- Deep learning architecture diagrams

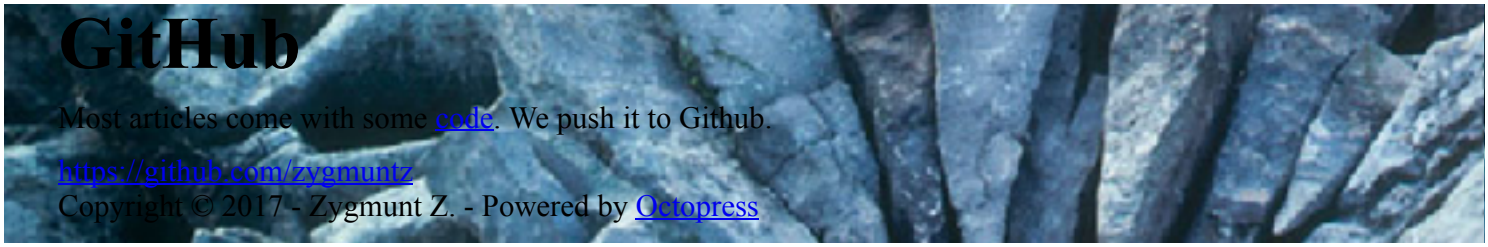
Twitter

Follow @fastml for notifications about new posts.

- Status updating...

Follow @fastml 4,928 followers

Also check out @fastml_extra for things related to machine learning and data science in general.



GitHub

Most articles come with some code. We push it to Github.

https://github.com/zygmuntz