Model for Ranking in Hackernews

Yash Bajaj,Final year CSE,IIT Kharagpur

Index

1.Features selection :

I have chosen 5 features based on intuition -

a)The number of unigrams in the title.It's often a case where we tl;dr long titles and don't read them.

b)Number of votes.It should be directly proportional to the popularity

c)The user reputation.We might think that a super user's (with very high karma) post will be more visible than a normal user with low reputation

d)The time difference of current time and the time at which it was posted. Pretty intuitive.

e)Number of comments on the post. More discussed posts might become popular and appear on the top.

2)Training data :

I have scraped three pages of hackernews for training purposes.For every post,I extracted features as follows :

a)Counted the number of unigrams in the title
b)Saw the number of votes on the post
c)Right cliked on the user who has posted,went to his profile and found out the reputation of the person
d)Saw the time before which it was posted
e)Saw the number of comments on the post

I made a file for 90 such posts
Taking integers from beforehand itself reduced the cost of mapping strings to integers.
I then made the input file for weka with the data and attributes section.

3)Choice of ML Tool :

I used weka for the extensive class of algorithms that it supports and I'm comfortable with it.

4)Models Used :

a)I had started with Least trimmed squared linear regression model . It takes slightly higher time to build the model but in linear regression model of weka if a column doesn't contribute much to the rank,it simply gives a weight of 0 to it unlike Least trimmed squared model.

So,the model builds and looks like this -

=== Classifier model (full training set) ===

Linear Regression Model

rank =

```
      0.4141 * unigramLength +
     -0.2365 * votes +
     -0.0002 * userReputation +
      2.9106 * time +
      0.1557 * comments +
      6.3352
```

Time taken to build model: 1.89 seconds

b)**I then went for Linear regression model as I felt it would perform better.**

The model build for linear regression looked like this -

=== Classifier model (full training set) ===

Linear Regression Model

rank =

-0.0851 * votes +
 2.2986 * time +
14.2949

Time taken to build model: 0 seconds

According to this model,only votes and time contribute to the rank of a particular post. The different testing phases will show where it outperformed a)

c)I also came across a model which uses a Neural networks based approach called Multilayer Perceptron Model.

The model goes as follows on our features -

=== Classifier model (full training set) ===

Linear Node 0

Inputs	Weights
Threshold	-0.7578371288104592
Node 1	1.9376552757942842
Node 2	1.3754116863872254
Node 3	0.883231253601369

Sigmoid Node 1

Inputs	Weights
Threshold	-2.102643302638737
Attrib unigramLength	1.3234629290899307
Attrib votes	1.9888891202255556
Attrib userReputation	0.020419233672120833
Attrib time	2.111120717041645
Attrib comments	-2.363159493040288

Sigmoid Node 2

Inputs	Weights
Threshold	-1.267989052451165
Attrib unigramLength	-1.4299517189687507
Attrib votes	-12.921982511034122
Attrib userReputation	-2.217767092597609
Attrib time	8.293346809600822
Attrib comments	8.101618882282132

Sigmoid Node 3

Inputs	Weights
Threshold	0.113119717394148
Attrib unigramLength	-3.031655643707135
Attrib votes	1.5139641489661575
Attrib userReputation	6.97807899691418
Attrib time	3.2714981876674125
Attrib comments	-1.5710299570899864

Class

Input	
Node 0	

Time taken to build model: 0.14 seconds

5)Testing data :

For testing purposes I scraped three more pages of data with their ground truths i.e ranks.

6)Evaluation metric :

After a literature review on ranking based algorithms.I fixed on using **corelation coefficient and R.M.S error** as the evaluation metric. It lies from -1 to +1

Note that the k for which the k-fold cross validation is obtained for which the corelation coefficient is maximum.I got this by trial and error by changing k from 2 to 10.

a)The testing for Least trimmed square model :

i)Evaluation on training set :

Correlation coefficient	0.6164
Root mean squared error	22.2948

ii)3-fold cross validation

Correlation coefficient	0.4741
Root mean squared error	28.008

iii) Percentage spilt – 66 % train. 33% test

Correlation coefficient	0.7455
Root mean squared error	17.3249

b)The testing for Linear Regression model :

i)Evaluation on training set :

Correlation coefficient	0.693
Root mean squared error	18.7292

ii)3-fold cross validation :

Correlation coefficient	0.6407
Root mean squared error	19.9738

iii) Percentage spilt :

Correlation coefficient	0.7605
Root mean squared error	17.5348

c)The testing for Multilayer Perceptron Model :

i)Evaluation on training set :

Correlation coefficient	0.7719
Root mean squared error	16.8359

ii)2-fold cross validation :

Correlation coefficient	0.677
Root mean squared error	19.6634

iii) Percentage spilt :

Correlation coefficient	0.6819
Root mean squared error	19.0485

So,from the above metrics we come to know that **Least trimmed square model** gives the worst results for all. **Linear regression** gives best results in **Percentage split** and **Perceptron** gives best results on evaluation on **training** set.

So,I evaluated the **testing dataset which I made** on Linear Regression model and these were the results :

Correlation coefficient	0.7698
Root mean squared error	20.7752