News Recommendation System Using Logistic Regression and Naive Bayes Classifiers

Chi Wai Lau

December 16, 2011

Abstract

To offer a more personalized experience, we implemented a news recommendation system using various machine learning techniques. We learned that Logistic Regression worked a lot better than Naive Bayes. Also surprisingly, for both algorithms, more training data did not necessarily lead to better results; however, the performance became better as the ratio of positive training samples increased.

Introduction

Most readers go through different headlines just to identify the stories that are truly interesting to them. If the readers were getting their news from online sources, we could easily follow their reading patterns and thus offer a much more personalized experience by suggesting engaging news stories using machine learning techniques.

Given a reader and a story, our news recommendation system would predict the interest level of the reader towards the story. To acquire a real-world training and testing dataset for our system, we are in collaboration with Pulse, a local startup that offers a news reading application for mobile devices. We were given 3 sets of data captured from 1000 random users throughout August 2011:

- Stories: All news stories that were available
- Impressions: Stories that a user accessed the summary
- Click-throughs: Stories that a user accessed the full article

To access the full article of a story, the readers are required to perform 2 clicks on the application: one for selecting a story to view its summary and another for accessing the full article. Due to the dual manual filtering, we believe accessing the full article is a strong indicator that the story is truly interesting to the reader. Based on that assumption, we labelled Click-throughs as the positive samples and Impressions as the negative samples.

Our samples would be represented via a feature vector based on a dictionary of words. To build that dictionary, we extracted all alphanumeric title strings from Stories as our text corpus.

To evaluate the performance of our algorithms, we ran tests using k-Fold Cross Validation. Due to the large amount of data, we were only using 4 folds. For each run, we computed measurements such as accuracy, precision, recall, and f-score. The better algorithm would have a larger measurement values.

	Accuracy	Precision	Recall	F-Score
Naive Bayes (Multivariate Bernoulli Event Model)	0.769	0.989	0.035	0.028
Naive Bayes (Multinomial Event Model)	0.777	0.743	0.076	0.082
Logistic Regression	0.713	0.298	0.262	0.276
Logistic Regression with TF-IDF Weighting	0.709	0.298	0.272	0.281

Table 1: Results for logistic regression and Naive Bayes classifiers on unstemmed story titles

Algorithms

Naive Bayes

The first algorithm we tried was Naive Bayes because it has been used for text categorization such as spam-mail filtering and it could take each reader's individual preferences for specific features into account. Moreover, it is efficient and capable of scaling to millions of readers.

We implemented a Naive Bayes classifier based on the *Multivariate Bernoulli* event model. We used only the story titles and represented them via a feature vector using the dictionary generated from our corpus. Table 1 shows that the classifier had a fairly good accuracy and very high precision, but the recall and f-score were near zero. The classifier was not good at recognizing interesting stories and categorized almost everything as uninteresting.

We implemented another Naive Bayes classifier based on the *Multinomial* event model. The classifier ran much faster because the algorithm takes O(n) time, where n is only the number of words in the story title (as opposed to the length of the feature vector). The classifier did twice as good in recall and f-score, but it is still not good enough for production use.

Logistic Regression

Using the same feature vector, we then tried to fit our training data with logistic regression, which is also widely used for text classification. We trained our parameters using gradient

descent and the training time generally took longer than the Naive Bayes classifiers. The result classifier had a lower precision due to the increased number of false positives, but it had a much better recall and f-score.

TF-IDF

TF-IDF is a weight often used to measure how important a word is to a document in a corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus [1]. We pre-computed the document frequency (i.e. idf) of each word from our corpus. For each training story, we computed the TF-IDF for each word in the title and then use it as a local weight in Logistic Regression to train our parameters. The weighted classifier produced a slightly better overall results (i.e. lower accuracy, constant precision, higher recall and f-score).

Porter Stemming

The feature vector we used above had a length of 79855. In an effort to make our system run more efficiently, we ran our text through the Porter stemming algorithm [2] to reduce the vector length down to 63409. Stemming seems to have a positive performance impact on our classifiers (see Figure 1).

Conclusion

From our experiments, we learned that the Naive Bayes classifiers did not perform as well as the logistic regression classifiers. As Figure 1

Figure 1: Impact of stemming

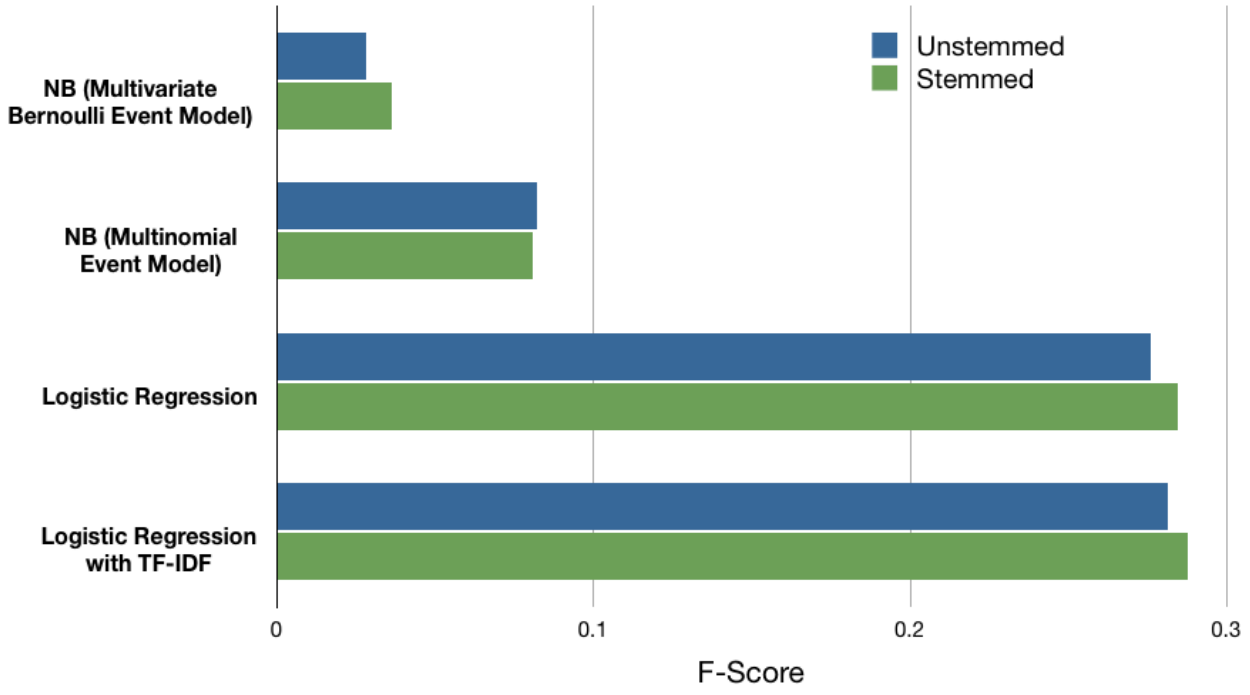
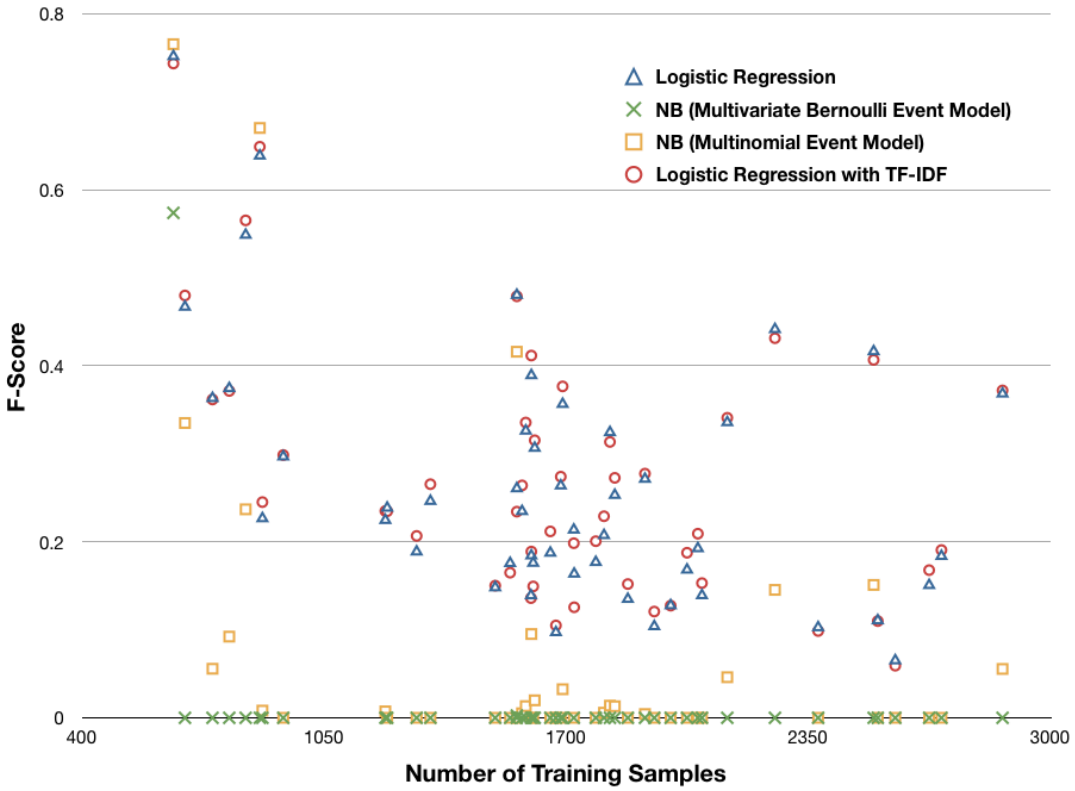


Figure 2: F-score vs the number of training samples

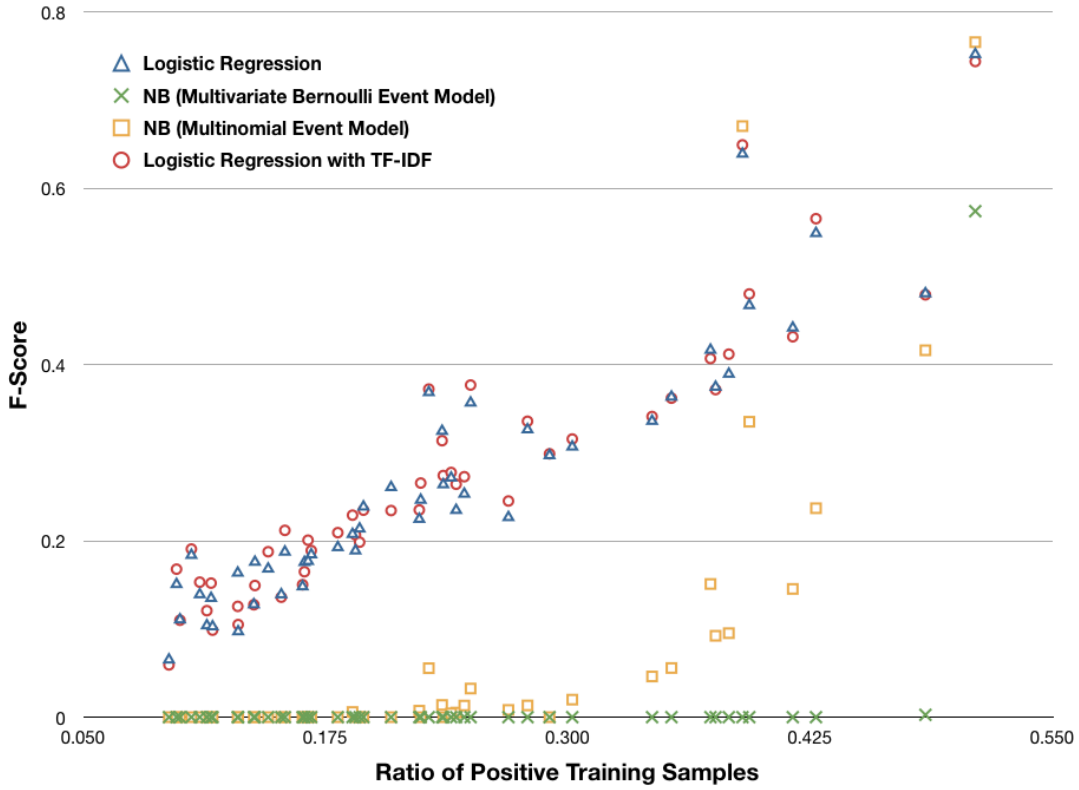


clearly shows, the f-score of the Naive Bayes classifiers with the *Multivariate Bernoulli* event model remained near 0 regardless how many training samples it was given; it indicates that the model is not very good at learning from our data and thus is likely not the right fit for our application. For the other classifiers, the number of training samples seemed to cause the f-score to vary, but opposite to the common wisdom, we notice that more training samples did not necessarily lead to better results and we do not observe any consistent patterns between f-score and the number of training data.

Interestingly, if we plot f-score against the ratio of positive training samples, we notice that f-score did get better as the ratio of positive training samples increased (see Figure 3). The pattern tells us that the classifiers were biased towards the negative samples because they were more dominant in the training dataset. To improve the performance of our classifiers, we may have to balance the distribution of positive and negative samples.

In summary, logistic regression yielded better results and the TF-IDF weights offered a slight performance boost. Stemming seemed to be quite effective; not only does it significantly reduce the vector length for a better run time, it also has a positive performance impact on our classifiers.

Figure 3: F-score vs the ratio of positive training samples



References

1. I. Paik and H. Mizugai, "Recommendation System UsingWeighted TF-IDF and Naive Bayes Classifiers on RSS Contents", Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 14 No. 6, 2010.
2. M. F. Porter, An algorithm for suffix stripping, Readings in information retrieval, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1997.