News Article Preference and Recommendation via Naïve Bayes Classification

Brian Brunner
brian@snackr.com

Abstract

With the rise of the Internet, we have seen a massive increase in the total amount of news available to consumers. Major old-media players, such as The New York Times, The Wall Street Journal, and CNN, stepped on to the Internet in full force. Social media and blogging have made everyone with a laptop and an Internet connection a journalist. This has led to an unbelievably large amount of content that users are required to manually sort through. To solve this problem, we can use context-aware tagging to automatically categorize articles and then both ask users up front what tags they are interested in as well as track the articles users rate as they interact with the system. We can then train a Naive Bayes Classifier on these examples with our tags as features and store the classifier, rebuilding it at a later time to filter new articles in order to give the user a more engaging experience.

1 Introduction

The Internet has lowered the barriers to entry for journalism and has caused a surge in the number of sources that are producing unique and interesting content on both mainstream and long-tail subjects. Subjects that were historically too niche to gain coverage now have massive outlets with many more people reporting on them and giving out their opinions. This excess of content, while beneficial to the consumer in many ways, forces users to do an overwhelming amount of manual filtering to find the content they enjoy [1].

In response to this, automated news personalization has become a growing consumer demand. 40% of users say an important feature of news websites is the ability to customize the news they receive from the site [2]. However, very few users rely on just one website, only 21% [2]. This means that users either must visit all sites they are interested in, or use some form of aggregator, whether that be an RSS reader or a website such as Digg or Reddit. RSS readers alleviate but don't solve the problem of filtering, as one must still scan all headlines delivered to them, and sites based on social recommendation only work for users whose interests line up with those of the community.

We can solve this problem with a simple news article likability system relying on a binary Bayes classifier with two classes: like and dislike. We can model each story as a collection of tags that represent the high-level genres of the article, such as Sports, Health, or Politics, the source of the article, such as CNN, WSJ and NYTimes, and the important or notable people, places and objects in the article. These articles will be automatically tagged using a commercially available content tagger. As naive Bayes classification is a fairly standard algorithm, we will be using the

open-source Bernoulli Naive Bayes classifier available as part of the scikits.learn python library.

The commercial potential of this application lies in its incorporation into the Snackr mobile application. Snackr is a mobile app that, on the backend, formats then synthesizes written articles into audio stories and then, on the front end, plays stories back in a customizable stream. To sum the application up in a metaphor, it is, quite literally, a Pandora for audio news.

2 Dataset

The dataset is made up of ratings from a small group of 10 users on 200 stories, which were selected evenly from amongst 20 different news outlets. The was created via an online system that presented the news source and headline for each story to the user and allowed the user to click through to the story if they desired.

3 Description of Algorithm

3.1 Data Preprocessing

We receive articles from different sources via an in-house developed scraping routine. This routine removes any HTML tags or other undesirable features from the article's text and then feeds the text through the autotagger. The autotagger returns a set of content tags for the people, places and objects mentioned in the article as well as the high-level genres of the article. We then add a tag for the source of the content as well as any genre tags that are associated with that source, e.g. a "Technology" genre tag for Mashable. We then store these tags as well as metadata for the story to our database. After this, the users perform their ratings on the stories retrieved by the scraping routine. The ratings are stored to the database, with a positive rating being represented by a 1 and a negative rating being represented by a 0.

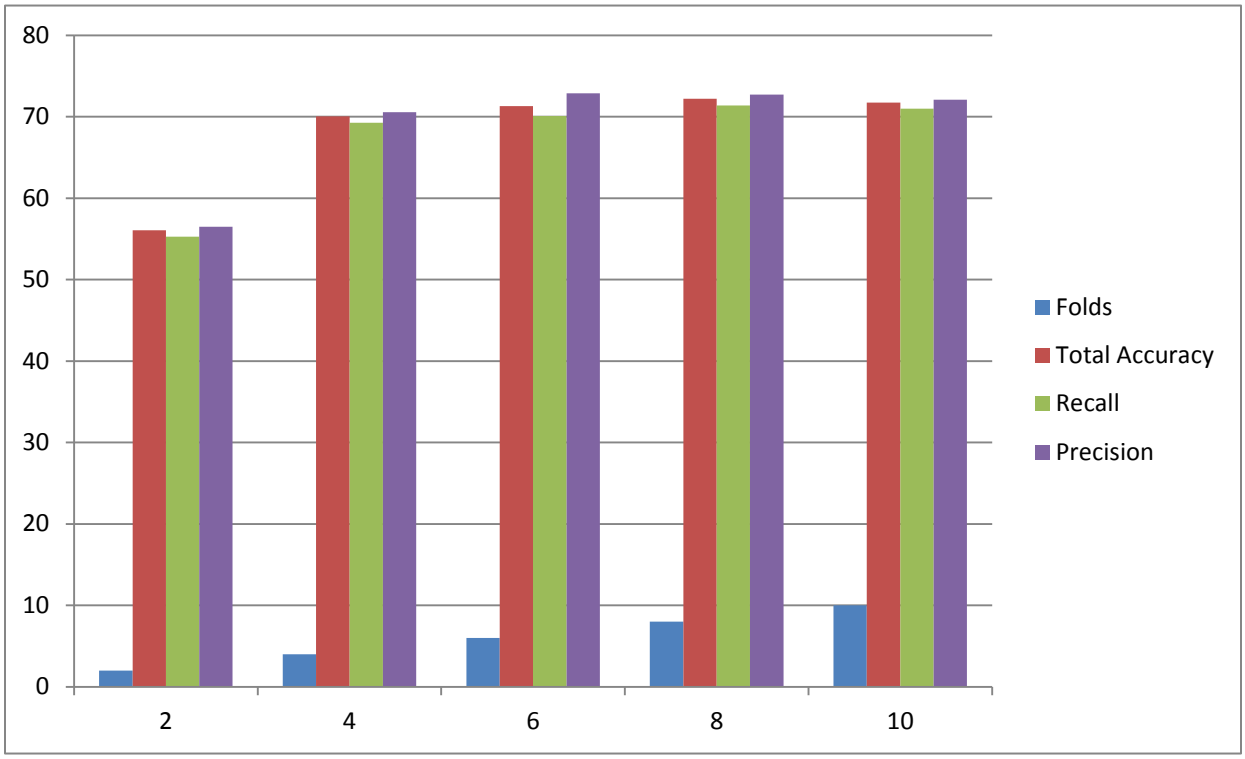
3.2 Bernoulli Naive Bayes Classification

Our next step is to build a Bernoulli Naive Bayes Classifier for each user. For the features of our classifier we use the presence or absence of a tag on a story. Due to the large number of tags a system like this might generate over the lifetime of a user, we limit ourselves to a subset of all of the tags on the rated stories, selecting only a small group of the most liked and least liked tags.

We then intersect the tags for each article with the tags we've selected to train our model on. This intersection, combined with a user's rating for the article, is fed into our classifier as a training example. Once we've finished building the classifier, we store its state to the database so that it may be quickly rebuilt in the future and used to rate new stories. In the live system, this process is repeated multiple times a day to keep the saved classifiers fresh.

3.3 Testing the Algorithm

To test the algorithm, we perform cross validation on each user's data. We'll perform our cross validation with varying numbers of folds, specifically with k folds where k=2,4,6,8,10. We'll examine both the results for individual users as well as global statistics about the system as a whole. Our key measurements are total system accuracy as well as positive precision and positive recall. We use these measures as the application dictates that we must serve users only the content they desire (positive precision) and are able to retrieve enough good content to make sure the users don't run out of fresh news (positive recall).



4 Results

The performance of the classifier was the best under 8-fold cross validation. The only notable jump in performance occurred from 2 to 4 folds. The total accuracy of the system maxed out at 72.18%, with positive recall and positive precision peaking at 71.36% and 72.86% respectively.

5 Conclusion

The implemented classifier performed moderately well on the collected data set, with a total accuracy of roughly 72%. The positive precision achieved, about 71%, is a good starting point and proves that the current tagging system is worth further exploration and expansion. Although 72% precision for selecting liked stories is below the threshold that we believe is necessary for a good user experience, current manual features, such as the ability to skip stories or select high-level content filters up front, allow the user to get much closer to a system that only selects stories that they will like. Additionally, social indicators, such as total number of likes for an article or if it covers trending subjects, have yet to be integrated into the classifier, and we believe these features will provide a very large boost to performance. Overall, the system has met and exceeded our expectations for a prototype for the hard problem of news recommendation. It has been demonstrated that it is based on valid methodology and worth developing further.

References

[1] http://mashable.com/2010/03/01/social-networks-source-news/
[2] http://www.pewinternet.org/Reports/2010/Online-News.aspx?r=1