On the Feasibility of Predicting News Popularity at Cold Start

Ioannis Arapakis, B. Barla Cambazoglu, and Mounia Lalmas

Yahoo Labs, Barcelona, Spain
{arapakis,barla,mounia}@yahoo-inc.com

Abstract. We perform a study on cold-start news popularity prediction using a collection of 13,319 news articles obtained from Yahoo News. We characterise the online popularity of news articles by two different metrics and try to predict them using machine learning techniques. Contrary to a prior work on the same topic, our findings indicate that predicting the news popularity at cold start is a difficult task and the previously published results may be superficial.

Keywords: News popularity prediction, cold-start prediction.

1 Introduction

So far, some research effort has been made to address the problem of news popularity prediction relying on early-stage measurements and user-generated content associated with the articles. The cold-start prediction scenario has been investigated, for the most part, in the context of recommender systems. To our knowledge, the only exception is the recent, widely cited work of Bandari et al. [2], who investigate the problem using exclusively content-based features available at cold start. The performance results reported by the authors imply that cold-start popularity prediction may be feasible.

Our work challenges the positive interpretation of high accuracy values reported in [2]. To this end, we first try to reproduce the performance results reported in [2] by following their experimental setting and methodology. We then improve their methodology and integrate the right performance metrics in a step-by-step fashion. Our work introduces a large number of new features (including those used in [2]) which may further help predicting future article popularity. As the popularity metric, in addition to tweet counts (the metric used in [2]), we also use the view counts of article pages.

Although we could mostly reproduce the findings of [2] and obtain similar results, our final findings, which are obtained after a more rigorous evaluation and interpretation, indicate that predicting the popularity of news articles at cold start is not really a viable task with the existing techniques. We point at the high skewness in the popularity distribution as the source of the problem (i.e., large number of unpopular articles and very few popular articles). We show that the techniques are biased to predict the large class of unpopular articles more accurately than the small class of popular articles (a common phenomenon in machine learning). Therefore, popular articles, which are more important to detect early, cannot be predicted and surfaced to a large extent.

2 Related Work

Some research efforts have addressed the cold-start prediction problem in the context of recommender systems. In [6], the authors present an approach to identify representative users and items using representative-based matrix factorisation. The authors of [5] discuss a hotel recommender system that employs context-based features. The authors overcome the cold-start problem by mining contextual information and analysing it for common traits per context group. In [3], the authors demonstrate how cold-start book recommendations based on social-tags can be combined with traditional collaborative filtering methods to improve performance. Finally, the work in [8] addresses the problem of cold-start social event recommendation, using the home location of the mobile phone users and the social events they have attended in the past.

To the best of our knowledge, the only work that has tackled the cold-start popularity prediction problem in the context of online news is the work of Bandari et al. [2]. In their work, the authors use a measure of popularity based on the number of times a news article is shared on Twitter. They devise a machine learning framework using some basic features including news source, genre, subjectivity of the language, and entities in the articles. The performance results reported by the authors suggest that popularity prediction is possible using only the limited information available before a news article is published. In this work, we reproduce the experimental results of [2] and demonstrate, using a more uniform dataset and a larger set of features, that predicting the news popularity at cold start is not a viable task with the existing techniques. Contrary to the findings of Bandari et al., we show that an article's popularity cannot be accurately estimated, solely on the basis of content features without incorporating any early-stage popularity information.

3 Data, Setup, and Characterisation of Metrics

Our analysis was conducted on a dataset consisting of $13,319$ news articles taken from Yahoo News. We opted for a single news portal to be able to extract features that are consistent across all articles. The dataset was constructed by crawling news articles over a period of two weeks. During the crawling period, we connected to the RSS feed API of the news portal every 15 minutes and fetched newly published articles. Each article was identified by its unique URI and stored in a database, along with meta-data like genre (e.g., politics, sports, crime), publication date, and article's HTML content at the time of publication.

To quantify the online popularity of news articles, we opted for two different metrics: the number of times an article was posted or shared on Twitter (`Tweets`) and the number of times an article page was viewed by the users (`Pageviews`).

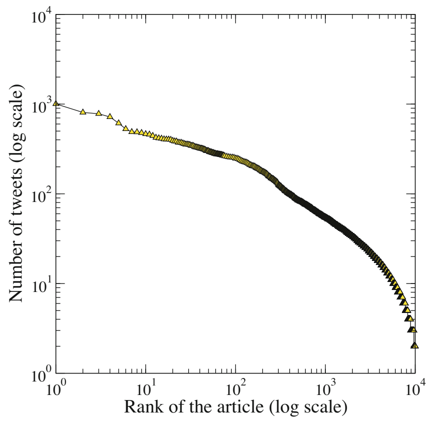

Fig. 1. Tweet counts of articles

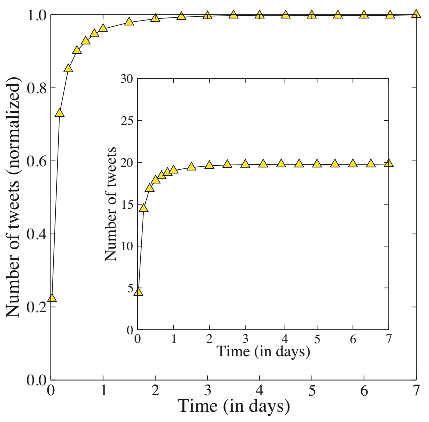

Fig. 2. Tweet counts over time

The choice of `Tweets` was informed by the fact that, nowadays, an increasing number of users are interacting with social media applications and exchanging content. Online communities, such as Twitter, serve as conduits for information flow and can thereby help to assess the virality of online content. We also include `Pageviews` as a metric since it is commonly used as a proxy for website engagement and online content popularity.

In our setup, every request to the RSS feed API was followed by a request to a public Twitter API to collect sample values for the `Tweets` metric across time. For all articles stored in the database, the metric values were sampled every half an hour, over a period of one week after the articles' publication. This resulted in 337 observations per article. In addition, we sampled data about the page views, again every half an hour, from the access logs of the news site.

Fig. 1 shows the tweet counts of articles in decreasing order of counts. As expected, the distribution is heavily skewed, i.e., most articles are tweeted a few times while very few articles are tweeted many times. Consequently, the problem of identifying soon-to-be-popular news articles becomes a challenging task as we will see in Section 5. Fig. 2 shows the increase in `Tweets` over time (the values are averages over all articles). We report both the original values (inner plot) and normalized values (outer plot). According to the figure, the increase in the `Tweets` metric saturates after two days. About 90% of the tweets happen in the first 12 hours after an article is published.[1]

4 Features

We use a larger number of features that we extract from the content of the news articles as well as external sources. Our features are categorised under ten main headings, depending on how or where the feature is obtained from. In what follows, we explain each feature category separately.

[1] `Pageviews` were omitted in Figs. 1 and 2 due to the confidential nature of this metric.

Time. We use features related to the time of news publication. Our choice is motivated by [1,7], where the authors successfully employ date and time information as features for their prediction tasks.

News Source. Similar to [2], we use the news source as a feature. In our study, the articles are obtained from five news distributors. A large portion of the articles are delivered by two major distributors, Reuters and Associated Press, while the share of the remaining agencies in the news volume is much smaller.

Genre. In [2], Bandari et al. use meta-data about the article category (i.e., genre) as one of the features. The authors observe that news related to certain genres have a more prominent presence in their dataset and most likely in the social media as well. Based on their results, we use specific genres as features.

Length. Our length features include the number of characters, words, and sentences in the body of the articles. The first two types of features are also computed for the titles of the articles.

NLP. We also use linguistic features which may have an effect on the online popularity of news articles. Our approach involves computing the distribution of nouns, adverbs, and verbs in the title and body of news articles. Our motivation for applying text analysis, even at this basic level, is that linguistic features can provide insights into certain aspects of the textual meaning or the impact on the reading experience.

Sentiment Analysis. For sentiment analysis, we use SentiStrength, a lexicon-based sentiment analysis tool [9]. We compute a sentimentality score and a polarity score for an article by averaging the positive/negative sentiment scores returned by SentiStrength for the individual sentences in the article (as described in [4]). We compute the two scores also for the article's title, treating it as a single sentence.

Entity Extraction. Similar to [2], we use an in-house software to extract named entities from the news articles. Here, in particular, we were interested in observing if the number of named entities in a news article affects its popularity. In general, we observed that articles that mention at least one entity are more likely to become popular than articles that do not mention any entity.

Wikipedia. For each named entity extracted from the article, we retrieve the popularity of the entity in Wikipedia.[2] Title- and body-level popularity values are then computed by summing the popularity values of all entities extracted from the title and article body, respectively. Other aggregation techniques, like averaging, yield inferior performance.

[2] `http://www.mediawiki.org/wiki/API:Main_page`

Twitter. To determine the short-term popularity of articles, we compute the popularity of named entities in Twitter. For each entity, we track the volume of tweets referring to the entity starting one hour, one day, and one week before the article's publication date.[3]

Web Search. We repeat the same technique on a large sample of queries submitted to the front-end of a popular web search engine and compute the frequency of entities in the sample. Again, the popularity of an entity is computed at three different time intervals and the aggregate search popularity for an article is determined as before.

5 Experiments

We start our experiments by reproducing the classification results presented in [2] by Bandari et al. for `Tweets`. To this end, we split two weeks of articles (13,319 articles) into three classes based on their tweet counts: `A` (low popularity), `B` (medium popularity), and `C` (high popularity). Adopting the choice made in [2], the tweet count ranges are set to $[1, 20]$, $(20, 100]$, and $(100, \infty)$ for the `A`, `B`, and `C` classes, respectively. Articles that are not tweeted are removed from the data and not included in set `A`. We experiment with the same classifiers used in [2]: naive Bayesian (`NB`), bagging (`Bagging`), decision trees (`J48`), and support vector machines (`SVM`). Moreover, for comparison purposes, we include a baseline classifier `Baseline` that always predicts the majority class in the training data. We make predictions for one hour, one day, and one week after an article is published. We perform log transformation on features exhibiting a skewed distribution.

Despite our efforts to create a similar setup, there are two minor differences between our setup and the setup in [2]. First, the articles used in [2] (10,000 articles) are obtained from a large number of news sites while our collection is obtained from a single, relatively major news site. Second, in [2], the popularity of articles are assumed to saturate after four days. In our case, as the closest value, we can use the popularity values obtained after one week. Nevertheless, since the features used in our study form a more powerful superset of the features used in [2], we expect to attain better or at least similar classification performance.

In Table 1, we report the performance in terms of the accuracy metric, i.e., the fraction of test articles whose class is correctly predicted by the classifier.[4] The reported results are obtained by performing cross-validation with ten folds, again adopting the choice made in [2]. According to the table, for the (`Tweets`, `Week`) case, the best performing classifier (`SVM`) achieves an accuracy of 79.7%, which is a bit lower than the best accuracy value (83.96%) reported in [2] (achieved by `Bagging`). However, when we observe the relative improvement with respect to the baseline (79.7%−70.3%=9.4%), we find it to be slightly higher in our case. Although it is not directly reported in [2], the relative improvement in their case

[3] We use Topsy's Otter API, available at `http://code.google.com/p/otterapi/`

[4] We do not report the classification accuracies for the `Pageviews` metric as this may reveal confidential information about the distribution of page views.

Table 1. Accuracy (ten-fold cross validation, without zero-popularity articles)

	Tweets		
Technique	Hour	Day	Week
Baseline	0.840	0.710	0.703
NB	0.693	0.581	0.574
Bagging	0.858	0.749	0.741
J48	0.856	0.781	0.775
SVM	0.859	0.802	0.797

Table 2. Accuracy (training/test split, without zero-popularity articles)

	Tweets		
Technique	Hour	Day	Week
Baseline	0.839	0.706	0.698
NB	0.735	0.589	0.584
Bagging	0.858	0.737	0.74
J48	0.852	0.779	0.774
SVM	0.861	0.803	0.798

can be estimated as 83.96%–76% = 7.96% using the data the authors provided in Tables 5 and 6. In general, the reported results are comparable, and we believe that we were able to reproduce the results reported in [2] to a certain degree.

Next, we repeat the same experiment using a training/test split in the time dimension instead of cross-validation. This is because, in the latter approach, the classifiers are allowed to use future information. In a real-life setting, this is not meaningful since a model would be trained at a fixed point in time using features extracted from previously seen articles and then it would be applied to predict the popularity of new articles. Hence, we repeat the previous experiment by splitting the data into training and test sets. The training set contains articles published in the first week and the test set contains articles published in the following week. The two sets are roughly equal in size. According to Table 2, the classification performance is somewhat similar to that in the previous experiment, i.e., there was no positive bias in results due to the use of cross-validation. In the remaining experiments, we use the setup with a training/test split.

Another issue that we observe in the methodology followed in [2] is the artificial manipulation of the original news collection. Before conducting their experiments, the authors remove from the data every article that is not tweeted at all after it was published. This manipulation may lead to unfair results because, in a real-life setting, it is not possible to know whether an article will be tweeted or not before it is published. Hence, herein, we repeat the previous experiment after including zero-popularity articles in the `A` class. The results are reported in Table 3. We observe that the classification problem is now easier than before as the accuracy of the best performing classifier has increased in all scenarios. In particular, the best accuracy increases from 79.8% to 82.5% in case of the (`Tweets`, `Week`) scenario. On the other hand, the performance gap between the best performing classifier and `Baseline` gets smaller. As an example, in case of (`Tweets`, `Week`), the improvement drops from 10.0% to 8.5%.

All results reported so far indicate high classification accuracy. But, how meaningful or useful are these results in practice? Can we really distinguish article popularity through classification? The answer lies in the surprisingly good performance of the `Baseline` classifier, which always predicts the label of the majority class in the training data. This implies that high accuracy values could be due to the highly skewed nature of the popularity distribution and the resulting class imbalance. In such scenarios, the classifiers are biased to learn and predict the majority class, leading to superficial accuracies.

Table 3. Accuracy (training/test split, with zero-popularity articles)

	Tweets		
Technique	Hour	Day	Week
Baseline	0.871	0.746	0.740
NB	0.772	0.642	0.633
Bagging	0.886	0.780	0.769
J48	0.883	0.805	0.804
SVM	0.890	0.829	0.825

Table 4. Fraction of instances in each of the three popularity classes

	Tweets		
Class	Hour	Day	Week
A	0.871	0.746	0.740
B	0.125	0.227	0.231
C	0.004	0.027	0.029

Table 5. The confusion matrix for (Tweets, Week)

	Predicted		
Actual	A	B	C
A	4,698	247	0
B	728	812	0
C	98	96	0

Table 6. Root mean squared error (training/test split, with zero-popularity articles)

	Tweets		
Technique	Hour	Day	Week
BaselineR	1.701	1.931	1.950
LR	1.132	1.270	1.305
kNNR	1.537	1.720	1.753
SVM	1.135	1.278	1.315

But, how skewed is the class distribution at hand? In Table 4, we display the fraction of articles in the test set for each of the three classes (confirming Fig. 1). As we can see, the collection is dominated by the unpopular articles in class A. In all cases, class C (the class of most popular articles) constitutes less than 4% of the sample. In a real-life setting, it is much more important to distinguish the articles in class C from the rest. The question is then how good are we in predicting class C articles. To answer this question, one can look at the confusion matrices, showing the true and false positive rates per class. In Table 5, as a representative, we provide the confusion matrix for the (Tweet, Week) scenario (using the best performing classifier, SVM). According to the table, the classifier does quite well in correctly identifying class A articles. However, it fails to distinguish class C articles from class A and B articles. This result indicates that the accuracy numbers reported in [2] are very likely to be not useful either.

Given that classification does not yield meaningful performance, we turn our attention to regression and observe the performance in predicting the actual popularity values rather than the popularity class values. To this end, we evaluate three regression approaches: linear regression (LR), k-nearest neighbor regression (kNNR), and support vector machines (SVM). For comparison, we also use a simple baseline (BaselineR) that always predicts the mean value in the training data. We perform a logarithmic transformation on the target values before regression.

In Table 6, the regression performance is reported in terms of root mean squared error. According to the table, LR is the best performing technique. Overall, the calculated errors are low and also there is considerable improvement with respect to BaselineR. As we go from Hour to Week, the error increases due to the larger variation in popularity. However, the improvement relative to BaselineR also increases since predicting the late-stage popularity is easier.

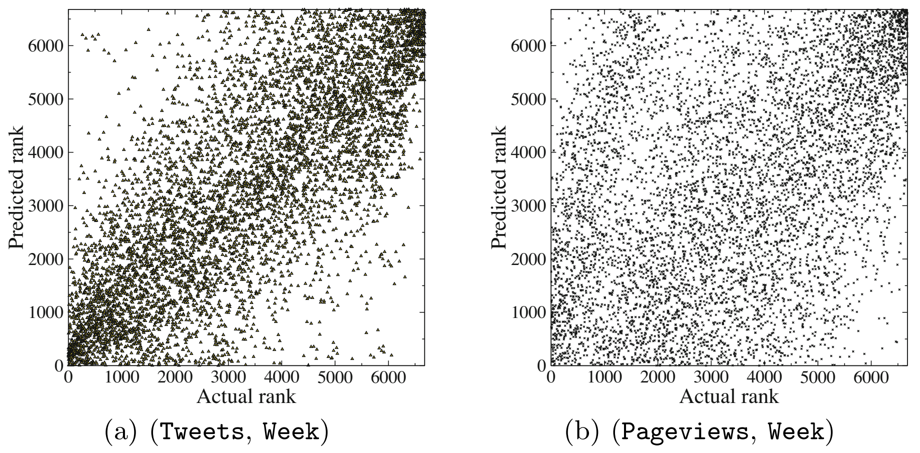

Fig. 3. Actual versus predicted ranks

Although the regression results give an idea about the prediction quality, they still do not tell us whether the predictions are biased towards unpopular articles. Moreover, in practice, accurate ranking of articles (in decreasing order of popularity) is more important than accurate prediction of their popularity. That is, given a popular and an unpopular article, the difference between the predicted popularity values is not of high importance as long as we can correctly rank the popular article above the unpopular article.

To visualize the ranking performance, in Fig. 3, we display the actual versus predicted popularity ranks of the articles (e.g., the article with the highest popularity is ranked first). A stronger correlation is observed between the actual and predicted tweet counts. In case of `Tweets`, we observe a stronger correlation between the actual and predicted values compared to `Pageviews`. The Kendal Tau (`KT`) correlation values in Table 7 are consistent with the plots.

Finally, we evaluate the performance focusing on the top ranked articles. This is important because, as we mentioned before, only a small fraction of articles gain visibility due to the limited space in web pages and the limited attention span of users. Therefore, it is vital to get the popularity ranking right especially at the high ranks by correctly identifying the most popular articles. To evaluate the performance at top ranks, we define the recall@k (`R@k`) metric (this is different than the traditional recall metric in information retrieval). Our metric basically selects the articles that are placed in the top k ranks by the prediction algorithm and then computes what fraction of them also appear within the top k ranks in the actual popularity ranking. We report `R@k` values for $k \in \{10, 100, 1,000\}$ in Table 7. For $k = 1,000$, even with the best prediction scenario (i.e., `Tweets`), we observe that about 45% of the articles in the top 1,000 ranks are not ranked among the top 1,000 articles in the actual popularity ranking. The results are even worse as the k value decreases. These final results illustrate the real difficulty of the problem and indicate the superficial nature of the previous results obtained through classification and regression [2].

Table 7. Performance in terms of the Kendal Tau and recall@k metrics

	Tweets			Pageviews		
Metric	Hour	Day	Week	Hour	Day	Week
KT	0.551	0.569	0.561	0.078	0.286	0.287
R@10	0.000	0.000	0.000	0.000	0.000	0.000
R@100	0.240	0.110	0.090	0.010	0.020	0.060
R@1000	0.578	0.557	0.548	0.212	0.173	0.245

6 Conclusion

In this work, we investigated the cold-start news popularity prediction problem. We measured the popularity of articles in terms of their tweet counts as well as page views. Using the content of news articles and external sources, we engineered a large number of features that may indicate the future popularity of news articles. Our work revealed that predicting the news popularity at cold start is not a solved problem yet. We observed that classifiers were biased to learn unpopular articles due to the imbalanced class distribution. Hence, highly popular articles could not be accurately detected, rendering the predictions not useful in most practical scenarios. Our findings suggest that popularity is disconnected from the inherent structural characteristics of news content and cannot be easily modelled. News popularity may be more accurately predicted if early-stage popularity measurements are incorporated into the prediction models as features. In general, increasing the duration of such measurements will increase the accuracy of predictions but decrease their importance, leading to an interesting trade-off.

Acknowledgments. This work was supported by the MULTISENSOR project, partially funded by the European Commission, under the contract number FP7-610411.

References

1. Ahmed, M., Spagna, S., Huici, F., Niccolini, S.: A peek into the future: Predicting the evolution of popularity in user generated content. In: Proc. 6th ACM Int'l Conf. Web Search and Data Mining, pp. 607–616 (2013)
2. Bandari, R., Sitaram, A., Huberman, B.A.: The pulse of news in social media: Forecasting popularity. In: Proc. 6th Int'l Conf. Weblogs and Social Media (2012)
3. Givon, S., Lavrenko, V.: Predicting social-tags for cold start book recommendations. In: Proc. 3rd ACM Conf. Recommender Systems, pp. 333–336 (2009)
4. Kucuktunc, O., Cambazoglu, B.B., Weber, I., Ferhatosmanoglu, H.: A large-scale sentiment analysis for Yahoo! Answers. In: Proc. 5th ACM Int'l Conf. Web Search and Data Mining, pp. 633–642 (2012)
5. Levi, A., Mokryn, O., Diot, C., Taft, N.: Finding a needle in a haystack of reviews: cold start context-based hotel recommender system. In: Proc. 6th ACM Conf. Recommender Systems, pp. 115–122 (2012)
6. Liu, N.N., Meng, X., Liu, C., Yang, Q.: Wisdom of the better few: cold start recommendation via representative based rating elicitation. In: Proc. 5th ACM Conf. Recommender Systems, pp. 37–44 (2011)

7. Marujo, L., Bugalho, M., da Silva Neto, J.P., Gershman, A., Carbonell, J.: Hourly traffic prediction of news stories. In: 3rd Int'l Workshop on Context-Aware Recommender Systems (2011)
8. Quercia, D., Lathia, N., Calabrese, F., Lorenzo, G.D., Crowcroft, J.: Recommending social events from mobile phone location data. In: 2010 IEEE 10th Int'l Conf. Data Mining, pp. 971–976 (2010)
9. Thelwall, M., Buckley, K., Paltoglou, G.: Sentiment strength detection for the social Web. J. Am. Soc. Inf. Sci. Technol. 63(1), 163–173 (2012)