Predicting Popularity of Posts on Hacker News
CS229 Autumn 2016 Project Final Report

Zhenglin Geng
zhenglin@stanford.edu

Yang Yuan
yyuan16@stanford.edu

Chao Wang
cwang17@stanford.edu

Abstract— **In this project, we try to find the best popularity predictor on the dataset of Hacker News posts. Features including n-gram, simple counts, tf-idf, word2vec and topic models and classifiers like SVM, Naive Bayes, etc are experimented. Data resampling is applied to combat the imbalance of the dataset. By combining different features, classifiers and resampling techniques, we are able to find predictors that achieve significantly better performance than random predictors. We also discuss the reason why this task is hard and the tradeoff between precision and recall on our dataset. As a by-product of the project, our visualization tool is demonstrated which helps us better understand the data.**

I. INTRODUCTION

Hacker News is a social news website created in 2007 focusing on computer science and entrepreneurship. All news are submitted by users who volunteer to collect news from all kinds of sources that they suppose good hackers would find interesting. There is only an up-vote submission option compared to traditional forums where there could also be down-votes. [1]

The number of votes for a news on Hacker News indicates its popularity among readers. In fact, only a small portion of news could gain more votes, which brings them to the site's front page and means they do bring considerable values both to readers and to content providers. In more general settings, predicting news popularity in advance is essential for authors to create better content, for news website to provide better recommendations, and for advertisers to get more profits.

Our goal of this project is to build a model to predict whether a news post on Hacker News will be popular given its content as input. Although the content of a news post plays a crucial role in its popularity, it's difficult to predict solely from the super noisy content and without knowledge about context outside the web around the publishing moment. And it's even more complicated if the prediction is performed at cold start, which means no early-stage feedbacks are available. Moreover, datasets in reality are uneven and that challenges our design and evaluation metrics.

We obtain our original dataset from web and preprocess to get training and test set and apply resampling strategies in the training process. We try out different combinations of features, namely n-gram, simple counts, tf-idf, word2vec, topic, and classifiers to perform the classification task. Our experiments show that it is possible to get some classifiers that can estimate whether a news post will be popular better than random classifiers.

In this report, we first review previous work related to news popularity prediction in Section II. Then we introduce how we get our dataset and features in Section III and what models we adopt for the classification problem in Section IV. All the details of experiments and the corresponding results and discussions are in Section V. Finally in Section VI, we make conclusions and discuss future possibilities for this research.

II. RELATED WORK

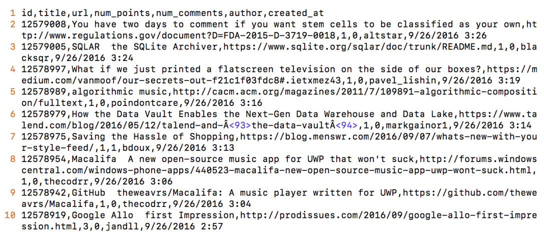
Although there is not much published work about predicting news popularity solely based on content, the concept of predicting popularity has been explored widely and other features for predicting news popularity have been put forward. People have obtained some methods to predict the popularity of movies, songs, news and so on. Early work followed a similar approach of using a feature set and some machine learning algorithms to predict popularity. The main steps were data extraction, data preprocessing, data transformation, feature selection and classification. In order to get better result of predicting, people developed different feature sets and learning algorithms, which were the main differences between these previous work.

Muhammad Hassan Latif and Hammad Afzal [1] used logistic regression, multilayer perceptron and Naive Bayes models to predict the popularity of the movies based on some independent variables, including rating, awards, screens, opening weekend, metascore, number of votes and budget.

Ioannis Arapakis, B. Barla Cambazoglu, and Mounia Lalmas [2] chose time, source and length of articles as the features to predict news popularity. The authors characterized the online popularity of news articles by two different metrics and tried to predict them using machine learning techniques such as NB, SVM, bagging and decision trees. Their result showed that popular articles cannot be predicted and surfaced to a large extent.

Roja Bandari [3] applied linear regression, KNN regression and SVM regression for the prediction problem. The authors used a measure of popularity based on the number of times a news article is shared on Twitter. They devised a machine learning framework using some basic features including news source, genre, subjectivity of the language, and entities in the articles. However, those classifiers were

[1]https://en.wikipedia.org/wiki/Hacker_News



```
id,title,url,num_points,num_comments,author,created_at
12579008,You have two days to comment if you want stem cells to be classified as your own,ht
tp://www.regulations.gov/document?D=FDA-2015-D-3719-0018,1,0,altstar,9/26/2016 3:26
12579005,SQLAR  the SQLite Archiver,https://www.sqlite.org/sqlar/doc/trunk/README.md,1,0,bla
cksqr,9/26/2016 3:24
12578997,What if we just printed a flatscreen television on the side of our boxes?,https://m
edium.com/vanmoof/our-secrets-out-f21c1f03fdc8#.ietxmez43,1,0,pavel_lishin,9/26/2016 3:19
12578989,algorithmic music,http://cacm.acm.org/magazines/2011/7/109891-algorithmic-compositi
on/fulltext,1,0,poindontcare,9/26/2016 3:16
12578979,How the Data Vault Enables the Next-Gen Data Warehouse and Data Lake,https://www.ta
lend.com/blog/2016/05/12/talend-and-Â<93>the-data-vaultÂ<94>,1,0,markgainor1,9/26/2016 3:14
12578975,Saving the Hassle of Shopping,https://blog.menswr.com/2016/09/07/whats-new-with-you
r-style-feed/,1,1,bdoux,9/26/2016 3:13
12578954,Macalifa  A new open-source music app for UWP that won't suck,http://forums.windows
central.com/windows-phone-apps/440523-macalifa-new-open-source-music-app-uwp-wont-suck.html,
1,0,thecodrr,9/26/2016 3:06
12578942,GitHub  theweavrs/Macalifa: A music player written for UWP,https://github.com/thewe
avrs/Macalifa,1,0,thecodrr,9/26/2016 3:04
12578919,Google Allo  first Impression,http://prodissues.com/2016/09/google-allo-first-impre
ssion.html,3,0,jandll,9/26/2016 2:57
```

Fig. 1. Original Dataset

actually biased to learn unpopular articles due to the imbalanced class distribution which was pointed out in [4].

DaviD m. Blei [5] used probabilistic topic models to find the theme that people are interested in and then examined the documents related to that theme. The author provided a statistical solution to the problem of managing large archives of documents. Topic modeling algorithms can be adapted to many kinds of data and applied to massive collections of documents, which is appropriate for our dataset. Their findings suggest that popularity is disconnected from the inherent structural characteristics of news content and cannot be easily modeled. In our opinion, their work is relatively efficient and clever.

III. DATASET AND FEATURES

A. *Original Dataset*

We obtained our dataset from Kaggle[2], which contains Hacker News posts information in one year (up to September 26 2016). Each post has 7 data fields, including:

- id: identical number of the post,
- title: title of the post,
- url: url of the news source,
- num_points: number of votes the post received,
- num_comments: number of comments the post received,
- author: the name of the account that made the post,
- created_at: the date and time the post was made.

B. *Preprocessing*

The content of a news post is the most powerful source for predicting its popularity. In preprocessing, we first grab news content from html files by applying a fast python port readability[3] using url field in the dataset.

Most outputs of the readability algorithm are neat. But there are quite a few situations where it fails to extract correct news content due to failure of connection, running into a dynamically load website, running into a video site, deficiency in readability algorithm itself, etc. So we further filter bad outputs and remove documents that barely contains any content to maintain a clean contents of our data.

Considering our limited computational resource, We perform our experiments on a subset of the dataset after preprocessing. The training set has 21,000 posts, and test set

[2]https://www.kaggle.com/hacker-news/hacker-news-posts

[3]https://github.com/buriy/python-readability

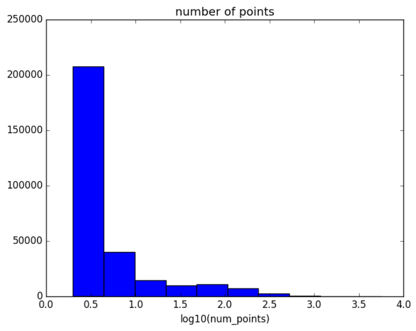


Fig. 2. Distribution of number of points

9,000 posts, which are created from February to September 2016.

We model our problem as a classification problem and intuitively choose number of points as the indicator of posts popularity. Based on the distribution of number of points (Figure 2), we label posts with more than 30 votes 1(popular) and others 0(unpopular). For our training set, the ratio between class 1 and class 0 is approximately 1:9. In case binary classification simplifies our problem too much in that its possible that news with exactly 1 votes are quite different from those that have more than 10 votes, we try to split our data further into 5 classes using thresholds 1, 2, 3 and 30 to see if we can get better results.

C. *Resampling*

The distribution of our data is very uneven, which has a significant influence on the classification result. In order to reduce the impact of the imbalance and improve the performance of predicting minority class, we take the strategy of resampling in our training process, which undersamples the majority class(unpopular) or oversamples the minority class(popular), before feeding our features to the machine learning algorithms.

D. *Features*

1) n-gram: We perform number, symbol and stop word removal on news content and then tokenize and stem text using NLTK [6] to get stemmed tokens of each post. We then use a contiguous sequence of n tokens as n-gram feature for each news.

2) counts / tf-idf: We make use of n-gram feature in two ways, either use simple counts of n-gram patterns or transform the counts into tf-idf (term frequencyinverse document frequency) to better represent the importance of n-gram pattern to certain news in our corpus.

3) word2vec: Word2Vec is the algorithm that takes corpus of text as input and word vectors as output. The distance of the word vectors is a measure of the similarities between words. For example Table I is the top ten nearest words to ”Tensorflow” with respect to cosine distance. We used [7] to train our word2vec model on our corpus. Then the document vectors are calculated by calculating the mean of word vectors or the weighted sum of word vectors similar

TABLE I

10 Nearest Words to "Tensorflow"

theano	0.176906192559
kera	0.179778250692
caffe	0.248036194784
resnet	0.325316752004
vgg	0.340320447299
scipy	0.351462844066
numpy	0.355779952979
torch	0.357693275487
xgboost	0.359064479706
tflearn	0.363983654973

TABLE II

Selected Topics Extracted by NMF

Topic 1	js javascript node react component python config browser
Topic 2	pple iphone ipad fbi mac cook sir headphone watch steve
Topic 3	invest startup fund venture capital entrepreneur tech raise
Topic 4	game pokemon vr nintendo virtual oculus headset steam
Topic 5	email clinton website inbox gmail slack spam campaign
Topic 7	tesla vehicle musk autopilot elect autonomous battery road
Topic 8	facebook twitter advertise zuckerberg messenger snapchat
Topic 11	docker deploy cloud kubernet swarm cluster microservices
Topic 12	ai intelligence algorithm neurual deep alphago deepmind
Topic 14	uber ride city lyft tax vehicle hail self did transport kalanick

to tf-idf. Different vector sizes are experimented to find the best performance.

4) topic: A topic is a distribution over vocabulary. We use non-negative matrix factorization [8] to extract topic models in our corpus. Table II is some selected topics. We also visualize topic models in a web browser using [9]. Figure 3 Latent Dirichlet Allocation[10] is also experimented but does not perform as good as non-negative matrix factorization. We then used document's relevance with each topic as our feature vector for classification task.

IV. Methods

We apply both binary and multinomial classifiers. Binary classifiers include: linear support vector classification with L2 loss, L1, L2 or elastic-net regularization; Stochastic gradient descent SVM with L1 or L2 regularization; Multinomial and Bernoulli Naive Bayes, Ridge, Perceptron, Passive Aggressive, KNN, Random Forest Tree. Multinomial classifiers include: Inherent multiclass models: Multinomial Naive Bayes, Decision Tree; One Vs Rest Classifier using linear support vector classification and SGD SVM as classifier for each class against the rest. Here we focus on four important methods: NB, SVM, Ridge and one-vs-rest.

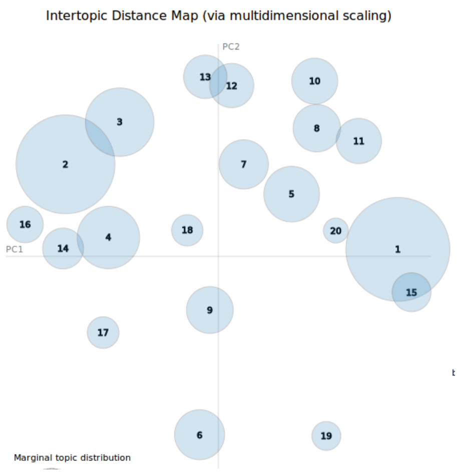


Fig. 3. Visualization of Topic Models. PCA is used to visualize topic models on 2D plane. The area of the circle represents the number of relevant news posts.

A. Binary Classifiers

1) Naive Bayes: The Naive Bayes algorithm utilizes Bayes rules to predict a label given a feature set. Given a dataset, assume that x_is are conditionally independent given y. Then,

$$p(x_1, ..., x_n) = p(x_1|y)p(x_2|y, x_1)...p(x_n|y, x_1, ..., x_n - 1)$$
$$= p(x_1|y)...p(x_n|y) = \prod_{i=1}^{n} p(x_i|y)$$

The model is parameterized by $\phi_{i|y=1} = p(x_i = 1|y = 1)$, $\phi_{i|y=0} = p(x_i = 1|y = 0)$, and $\phi_y = p(y = 1)$. Given a training set $\{(x^{(i)}, y^{(i)}); i = 1, ...m\}$, the joint likelihood of the data is:

$$L(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) = \prod_{i=1}^{n} p(x^{(i)}, y^{(i)})$$

.

Having fit all these parameters, to make a prediction on a new example with features x, simply calculate

$$p(y = 1|x) = \frac{p(x|y = 1)p(y)}{\sum p(x|y)p(y)}$$

and pick whichever class has the higher posterior probability.

2) Support Vector Machine: SVMs are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. SVM attempts to find the maximum-margin hyperplane that separates the dataset in a higher dimensional feature space. Finding this optimal margin reduces to solving the following convex optimization problem:

$$min_{\gamma,\omega,b} \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{m} \xi_i$$

Subject to the constraints:

$$s.t. y^{(i)}(\omega^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, .., m$$

$$\xi_i, i = 1, .., m$$

3) Ridge: Ridge regression estimator is:

$$\hat{\beta}_{ridge} = (X'X + \lambda I_p)^{-1} X'Y$$

where X is training data, Y is test data. Ridge regression places a particular form of constraint on the parameters:$\hat{\beta}_{ridge}$ is chosen to minimize the penalized sum of squares:

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

which is equivalent to minimization of $\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2$ subject to, for some $c > 0, \lambda \sum_{j=1}^{p} \beta_j^2 < c$, i.e. constraining the sum of the squared coefficients.

B. *One-VS-Rest for Multiclass Classification*

The one-vs-rest strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a class label; discrete class labels alone can lead to ambiguities, where multiple classes are predicted for a single sample.

V. EXPERIMENTS, RESULTS AND DISCUSSION

A. *Experiments*

1) Features:

a) n-gram: n is set from 2 to 9. Simple counts and tf-idf are experimented.

b) counts / tf-idf: The experiments are done on both the entire token set and a cleaned token set($5 \leq$ term frequency ≤ 6000)

c) word2vec: The dimension of word vector is set to be 100,200,300,400,500,1000,1500,2000,2500. Simple mean and tf-idf are experimented.

d) Topic Models: The number of topic models is set to be 20.

2) Binary Classifiers:

a) Ridge: Regulation strength is set to 1.

b) Perceptron: Constant before regularization is 0.0001. The number of passes over training data is 5.

c) Passive Aggressive [11]: Maximum step size is set to 1. Hinge loss is used.

d) Linear Support Vector Classification : L2 loss is used. L1 and L2 regularization are experimented. The max step size is 1.

e) Stochastic Gradient Decent SVM: Learning rate is set to optimal[12]. L1, L2 and elastic-net regularization are experimented. The number of passes over training data is set to be 50.

f) Multinomial Naive Bayes: Laplace smoothing factor is set to be 0.01.

g) Bernoulli Naive Bayes: Laplace smoothing factor is set to be 0.01. In some cases, the factor is also set to be 0.1, 0.001.

3) Multinomial Classifiers:

a) Multinomial Naive Bayes: Laplace smoothing factor is set to be 0.01.

b) Decision Tree: All hyperparameters set to default values.

TABLE III
CONFUSION MATRIX OF RANDOM CLASSIFIER

Predicted \ True	0	1	Sum
0	TN: $(1-\epsilon)(1-p)$	FN:$(1-\epsilon)p$	$1-\epsilon$
1	FP:$\epsilon(1-p)$	TP:ϵp	ϵ
sum	$1-p$	p	1

c) One Vs Rest Classifier: Using linear support vector classification with L2 regularization, SGD SVM with L2 and elastic-net regularization and Ridge as classifier for each class against the rest.

To sum up, we have 16+4+18+1=40 different sets of features, 10 different binary classifiers and 6 different multinomial classifiers. For each set of features and classifiers, we run experiment with no sampling, simple sampling and over sampling. There are $40 \times 10 \times 3 = 1200$ binary classification results. It is impossible for us to list all the results in the report. So we only list some of our best results here in Table IV.

B. *Results*

1) evaluation metrics: In practice, the objective of content providers is to find popular posts in advance. So were focusing our evaluation on popular class in experiment. And since the problem is hard to solve in nature, we use statistical significance to measure the effectiveness of our classifier against random classifiers. Binary classification and multiclass classification are similar in this point, so we base our discuss only on binary classification in evaluation metrics part.

Suppose the percentage of class 1 is p in test set, the probability for a random classifier to classify a post into class 1 is ϵ. The confusion matrix of such a random classifier is Table III.

And we can calculate the precision and recall for class 1 as a function of ϵ and p:

$$precision = \frac{TP}{FP+TP} = \frac{\epsilon p}{\epsilon} = p$$

$$recall = \frac{TP}{FN+TP} = \frac{\epsilon p}{p} = \epsilon$$

Therefore, the precision of a random classifier is determined by the percentage of class 1 in test set and has nothing to do with classifiers at all, while the recall is determined by the probability for the random classifier to predict 1.

We set significance level = 0.05, and get p-value = 0.1098 after 100,000 predictions of random classification. The distribution of precision is shown in Figure 4 It means that any classifier that achieves precision higher than p-value should be considered better than random guess.

2) binary classification results: In our test set, we have 844 positive cases, 8156 negative test cases. The ratio between these two is roughly 1:9.1. The imbalance of the dataset makes it very hard for the precision to be very large. V-B.2 is some of our best results.

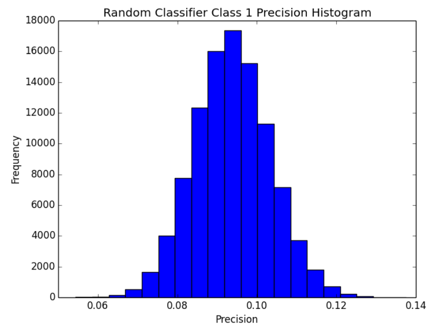


Fig. 4. Precision Distribution of Random Classifier

3) multiclass classification results: We perform multiclass classification on test sets using 1-gram, 2-gram, 3-gram tf-idf feature and undersampling. The results we get from all multinomial classifiers are no better than random classifiers, with the corresponding precision for popular class in this case ranging from 0.09 to 0.13. And for other 4 groups, the precisions are also approximately the percentage of that class in test sets. So we stop at this point rather than perform more multiclass classifications, for the results don't improve much.

C. Discussion

Predicting the popularity at cold start in hard in nature as mentioned in [4]. And predicting based on news context is also challenging. On one hand, the data is very noisy even if we did a lot of data cleaning before classification. On the other hand, this task is even hard for human because the popularity is dependent on many random factors. Furthermore, we have a very imbalanced dataset.

Our binary classifiers achieve better results than multinomial ones, which means binary classifier models our popularity prediction problem successfully enough. While most of our binary classification models have very high absolute value of precision on class 0 and low precision on class 1, we still have many models that are significantly better than random predictors.

We mainly focus on class 1 because we only care about the popular news in the corpus. According to TableV-B.2, Bernoulli Naive Bayes and SVM are among the best classifiers. In the meantime, we can observe that the lower the precision is, the easier the model can achieve high recall value.

Meanwhile, we have very good results on topic modeling. From TableII, We found out that a very large portion of posts on Hacker News are about javascript and nodejs. Also because our dataset is collected from Feburary 2016 to September 2016, we can clearly see a topic about "Pokemon, Nintendo, game" and "Clinton, email, gmail" reflecting the popular event happened in that period. At the same time, we also build a visualization tool which can help us understand the distribution of topics and easily find news posts under the same topic, which is very meaningful for people who want to understand what is happening in this community.

TABLE IV
CLASSIFICATION REPORT AFTER OVERSAMPLING

	C[a]	P [b]	R[c]	F1	CM[d]	
SVM-2gram-EN [e]	0	0.91	0.98	0.94	8013	143
	1	**0.18**	**0.04**	**0.06**	813	31
Linear-4gram-L1-tfidf-u [f]	0	0.91	0.97	0.94	7931	225
	1	**0.17**	**0.06**	**0.08**	797	47
BernoulliNB-clean-o [g]	0	0.91	0.90	0.91	7376	780
	1	**0.16**	**0.18**	**0.17**	694	150
BernoulliNB-clean [h]	0	0.92	0.89	0.90	7253	903
	1	**0.16**	**0.20**	**0.18**	672	172
BernoulliNB-w2v(300) [i]	0	0.92	0.88	0.90	7165	991
	1	**0.15**	**0.21**	**0.18**	664	180
SVM-w2v(500)-o [j]	0	0.92	0.59	0.72	4795	3361
	1	**0.14**	**0.35**	**0.20**	550	294
BernoulliNB(0.1) [k]	0	0.92	0.71	0.80	5805	2351
	1	**0.14**	**0.44**	**0.21**	473	371
SVM-w2v(100)-u [l]	0	0.93	0.58	0.72	4743	3413
	1	**0.13**	**0.59**	**0.21**	350	494
SVM-w2v(3000)-o [m]	0	0.93	0.57	0.70	4609	3547
	1	**0.13**	**0.60**	**0.21**	337	507
Ridge-u [n]	0	0.94	0.43	0.59	3510	4646
	1	**0.12**	**0.72**	**0.20**	240	604

[a]Class
[b]Precision
[c]Recall
[d]Confusion Matrix
[e]SVM, 2-gram, elastic net penalty, no resampling
[f]Linear support vector classification with L2 loss, L1 penalty, tfidf, undersampled
[g]Bernoulli Naive Bayes, using cleaned tokens, oversampled
[h]Bernoulli Naive Bayes, using cleaned tokens, no resampling
[i]Bernoulli Naive Bayes, using word2vec, the dimension of word vector is 300, no resampling
[j]SVM, using word2vec, dimension of word vector is 300, oversampled
[k]Bernoulli Naive Bayes, Laplace smoothing factor set to 0.1, no resampling
[l]SVM, using word2vec, dimension of word vector is 100, undersampled
[m]SVM, using word2vec, dimension of word vector is 3000, oversampled
[n]Ridge, using simple counts, no resampling

VI. CONCLUSION

Although news popularity prediction based on news content is a hard task, we try a number of combinations of different features and models, apply resampling strategies in training process and finally get some binary classifiers including Bernoulli Naive Bayes and SVM that achieve better performance than random classifiers.

However, our classifiers are still far from perfect. To achieve better performance, we would consider fine tune different classifiers. And we could explore more features from both news text and other information in the original dataset, such as title, publication time and source domain, which would certainly require a larger dataset and more computational resources. If there's a way to get more context at or before the publishing time, we believe the prediction could be improved even more.

REFERENCES

[1] Muhammad Hassan Latif and Hammad Afzal. Prediction of movies popularity using machine learning techniques. *International Journal of Computer Science and Network Security (IJCSNS)*, 16(8):127, 2016.

[2] David M Blei and John D Lafferty. Topic models. *Text mining: classification, clustering, and applications*, 10(71):34, 2009.

[3] Roja Bandari, Sitaram Asur, and Bernardo A Huberman. The pulse of news in social media: Forecasting popularity. *arXiv preprint arXiv:1202.0332*, 2012.

[4] Ioannis Arapakis, Berkant Barla Cambazoglu, and Mounia Lalmas. On the feasibility of predicting news popularity at cold start. In *SocInfo*, 2014.

[5] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[6] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. " O'Reilly Media, Inc.", 2009.

[7] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

[8] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[9] Carson Sievert and Kenneth E Shirley. Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70, 2014.

[10] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[11] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.