CIS 520 Fall 2020, Project Report

Predicting the Popularity of The Daily Pennsylvanian's Articles

Team Members:

- Baile (Peter) Chen; Email: `cbaile@seas.upenn.edu`
- James McFadden; Email: `jmcfa@seas.upenn.edu`
- Ali Mohammad; Email: `alism@seas.upenn.edu`
- Isaac Tham; Email: `iztham@sas.upenn.edu`

Abstract

In this project, we aim to predict the popularity of articles from Penn's student newspaper, The Daily Pennsylvanian (The DP), using a corpus of 29000 articles from 2012 to 2020. Our project serves to provide The DP with a reliable way to predict the performance of articles before they are published, aiding the company in its website and social media strategy. We trained supervised classification models to predict whether an article will be in the top 25% of articles for the month, using text and metadata features. Unsupervised topic modelling methods, LDA and NMF, were used for topic modelling. We also perform deep learning methods, such as recurrent neural networks and BERT. Finally, we combined predictions from all individual classifiers as features in a final ensemble model. We found that the ensemble model achieved 87.58% accuracy overall, compared to 70.27% for the RNN model and around 68% for most of the individual machine learning classifiers.

1 Motivation

The members of this project team have extensive connections with Pennâs student newspaper, The Daily Pennsylvanian (DP). Everyone at one point has worked as staff in the paperâs data analytics department; one member ran the department in 2019. During our time at the DP, we realized the potential of machine learning to improve the operations of the paper, which is increasingly moving towards a digital-first format as it struggles to find a viable business model in the twenty-first century. The DP is striving to establish an online readership base in order to remain relevant to students and continue earning revenue from advertisers. This trend towards digitization has accelerated this past year due to the ongoing COVID-19 pandemic; the DP has halted regular print production for the foreseeable future, and may never fully reinstate it.

Surviving this shift requires the paper to understand its audience and maximize its online presence, which in turn requires machine learning models to provide the necessary information. To aid in this process, we have set out to create an article classifier that can accurately predict the popularity of an article before it is released. A model like this can be used to optimize the paper's social media posting strategy, curate its daily and weekly newsletters, and wisely position articles on its website. This will help the paper increase total article pageviews, keeping its work a part of the campus conversation and attracting more revenue opportunities.

It is our hope that this model (or future iterations of it) will actually be used by the paper to make important decisions. The DP does not yet have a strong data culture, even if it is forming. The editorial side of the paper is dominated by liberal arts majors who often do not have an education in data science and machine learning techniques. At seven years old, the data analytics department is still a new feature in the paper's company structure. The DP is a place of tradition, and as of yet that tradition does not include using machine learning models to make emotional decisions related to whose work gets promoted. The important takeaway from all this is that our model will not automatically get an eager reception from editors and department heads. It cannot just be accurate; it must also be user friendly, and integrate well into daily workflow. Only then does it stand a chance of actually being used on a regular basis.

Keeping this reality in mind, we have carefully considered the cases in which this model would be used. One case is for "digital ranking," a nightly meeting between editors to decide which newly published articles will appear on the main page of the website, and in how prominent of a position. Another case is for the setting of the daily social media posting schedule by the audience engagement editor, who decides what articles get shared at the best times on Facebook and Instagram. The third case we considered is for the curation of the newsletters that get emailed to the Penn community by the audience engagement editor on a regular basis. Machine learning will not instantly take over any of these processes. It will start out as a supplement, something that is considered alongside the instinct of the editors as well as the current heuristics they use to make these decisions. Thus, our model will be designed not overload the user with information they won't ever act on.

It will be easy to explain to editors the usefulness of this type of binary classifier since it matches the type of decisions they make on a daily basis. It is our hope that through the introduction of models like this, carefully constructed and cheerfully explained, the Daily Pennsylvanian as a company will come to see the benefit of integrating machine learning into editorial decision-making.

2 Related Work

The literature on using NLP methods to model article views is relatively sparse compared to other NLP supervised learning tasks of predicting ratings and sentiment. This is due to the relative unavailability of article views data, restricted to news agencies and not normally. open for researchers.

There was one extremely relevant paper by Hardt and Rambow (2017), which used a text dataset of 64k news articles from the Danish daily newspaper (Jyllands-Posten), and used text features to predict user views.

They structured this as a classification problem: with the label being whether a given article is in top 50% / 33.3% / 25% of clicks. Next, they used the following lexical features: Bag of Words - used count of word occurrences, weighted count (term frequency), and also TF-IDF values. After that, they used the following methods with the lexical features: SVMs with linear kernel, logistic regression, Random Forests.

For their results, they found that logistic regression outperformed the other ML methods, and that using term frequency as well as TF-IDF performed equally well, both outperforming simple counts of words. Lastly, they found that text features in the article body are the most important determinants of viewing behavior (surprising as one would think that the article title contains all the information that drives page views). The authors explain this with a hypothesis that the headline on its own gives users a lot of semantic information which are not captured with the simple BOW model - people then imagine the article before they read it and implicitly base their reading behavior on their expected article content.

3 Dataset

Feature and page view collection

We collected the page views in the first 3 days after publishing for all articles since 2012 by accessing relevant data from Google Analytics. To investigate the effect of article metadata on page views, we queried the DP article database and related APIs to gather several features, including titles, content, published date, authors, and tags. This sums to a total of 28897 articles.

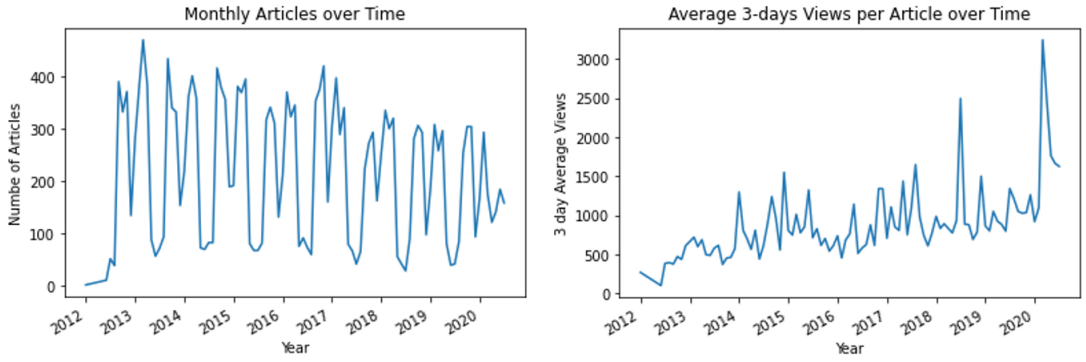
Moreover, under the impression that the longer an article appears on the front page, the more likely it is to get higher page views, we are also interested in knowing the duration of each article appearing on *thedp.com* home page. First of all, there exists a tagging system that defines the positions of articles. If an article is tagged *centerpiece*, then it appears as the first article on the home page. If an article is *top*, it appears as one of the two articles next to the centerpiece article. Understanding this, we can then compute the duration of an article on the home page by comparing the published dates of articles appearing under the same spot (with the same tag). For example, an article tagged *centerpiece*, once published, will stay on the

homepage until another article tagged *centerpiece* overwrites it. So the approximate duration of appearance is the difference between the published dates.

However, the fact that we are using articles that span a period of 10 years makes us rethink whether some of our preliminary features are plausible predictors, namely author names and duration of articles on the home page. Because all writers in the DP are students, and students leave the DP for various reasons (and they will have to leave DP after graduation), the group of writers today can differ significantly from the group of writers 2 years ago. As a result, author names are less helpful in making future predictions. Having said that, what does not change is positions/ titles of an author (e.g. news editor, beat reporter, general assignments). The intuition here is that articles written by more senior writers tend to have better appearances/ higher ranking compared to reporters or new staffers, and therefore more page views. With this in mind, we reached out to the DP staff to obtain the author positions info. And after cleaning, only 9% of articles are missing author positions. The other point to consider is the validity of the duration of each article on the home page. We realized that the aforementioned approximation method is only valid for the recent 2 years because the current layout of thedp.com home page is effective around 2018 and layouts prior to that are vastly different. As a result, we decided to revert back to use tags alone to indicate positions of articles on the home page while putting duration aside.

In summary, our data set contains 28897 articles. The feature of each article consists of the title, content, published date, authors, tags and author positions as well the page view which servers as the y.

EDA



The left plot shows how the number of articles per month varies over time. It is evidently very seasonal - with more articles written during the fall and spring semester and fewer articles written during the summer and winter breaks. Over time, the number of articles per month seems to be decreasing slightly.

The right plot shows how the average first 3-day views per article change over time. As you can see, the number of viewers generally increases over time due to the more effective outreach and online presence of the DP. The monthly average article views are also highly volatile, as individual articles can become big hits that pull up the month's average. Hence, this explains our decision why we would want to predict based on relative popularity (percentiles within the same month) rather than absolute popularity.

4 Problem Formulation

We primarily formulated this as a classification problem, but did some investigation into regression as well.

For the classification problem, we attempt to use the various features of an article to predict which bucket an article falls into: those expected to perform in the top 25 percent of articles, and those that will perform

in the bottom 75 percent. We originally considered sorting the articles into quintiles (5 buckets), but since editors do not think about articles in five different tiers, this would be less helpful. They generally only discriminate between promising articles that deserve special treatment, and all the rest. Thus, our model will be a binary classifier. Upon further reflection, we determined that the best breakdown for our classifier was the 25-75 division mentioned above. If the top tier is any bigger, it will cease to be useful. If it gets too small, it risks simply becoming a mechanism to predict gems; this is unnecessary, as it is generally obvious to the editors which articles will blow up.

For the regression problem, we use the various features of an article to predict the percentiles of page views. Initially, raw page views are used as the y, however, this might lead to bias because the articles used span more than 10 years, and more recent articles might get systematically higher page views that earlier ones because the DP website is used by more people. With this in mind, we decided to consider the percentile of the page view with respect to articles written in a similar time frame. In this case, percentiles are computed for articles in the same month. Getting a point estimate of the percentiles will also allow us to rank articles.

5 Methods

We will be using Logistic Regression in Supervised Learning as our baseline model throughout this project. This is because Logisitic is comparatively simple method compared to say Adaboost and also naturally includes all variables in its model. Logistic Regression on Title serves as the overall baseline.

Sklearn has been used for the machine learning methods for supervised and unsupervised learning.

Preprocessing
The first task was to preprocess our textual data and prepare it for machine learning methods. The following tasks were performed on the textual data

- Lower-casing: To ensure uniformity. "Student" and 'student' should be recognized as the same word as it is probably the different structure that made one capitalized.
- Remove punctuation and numbers: Numbers will tend to differ from article thus can not indicate similarities or differences between articles. Similarly with punctuation.
- Lemmatization: This ensures that our algorithms recognize different forms of a word (such as different tense, verb to adverb, noun to adjective etc) as the same word as the form of the word does not change its inherent meaning.
- Stopword removal: Words such as 'a', 'to', 'as' etc. are everpresent parts of the English language but do not actually indicate or differentiate between articles as they are merely sentence builders. We did not want our algorithms to get caught making predictions or differentiating articles based on these meaningless variables thus used NLTK's stopwords dictionary to remove them from the dataset.

Supervised Learning
After preprocessing, we start off by implementing some basic supervised learning methods before stepping into the more complicated deep learning and eventually the ensemble methods.

First, we converted texts into vectors of numeric numbers using both the *bag of words* approach and the *TD-IDF* approach. Then, we constructed feature matrices for each feature: title, content, tags and author positions. Now, with both the X and y ready, we trained a regression model (Ridge Regression) and several classification models, including Logistic Classification, Decision Trees, random forest, AdaBoost, and SVM.

The hyperparameters in each of these models were tuned using grid search to maximize test accuracy. For logistic regression, we weakened the default regularization parameter C in the scikit-learn, making C=10. For random forest, we performed a grid search to determine the ideal max depth, max features, and number

of estimators, finding values of 45, 50, and 25, respectively. For adaboost, we determined that the best max depth for our model was 200. For our SVM model, we tried out a linear kernel, a polynomial kernel, and rbf-based kernel, and determined that a linear kernel led to the highest performance.

Topic Selection
The topic of an article is an important indicator of the number of views that article will get, although it is important to realize that is not the only indicator. Therefore, we needed to get topics. Sadly, the DP does not classify its articles into a set topics so machine learning in the form of unsupervised learning needed to be done to gain these topics.

We considered using article's titles to generate these topics, in accordance with the idea that as we are looking at views instead of a factor such as engagement, the user would decide the topic based on title alone. However, titles have extremely few words and therefore it is possible fo multiple titles of almost the same topic to have almost no keywords in common. Therefore, we decided to instead get our topics from the content of the article instead, assuming that topic will match up with what the reader understands from the title.

Two methods for finding unsupervised topics were looked into. Latent Dirichlet allocation (LDA) and Non-negative matrix factorization (NMF).
For LDA: we used Bag-of-Words.
For NMF: we used TFIDF
As TFIDF assigns comparitively greater weightage to less frequent words than Bag-of-Words, we were aware it might focus on words that appear on a few articles and use them to incorrectly isolate those articles therefore set a minDF value to 20. Even though TFIDF manages to decrease the importance of frequently occurring words compared to Bag of Words (aka 'local stopwords') we set maxDF to 0.7 to completely remove the influence of such words.
But of course values for maxDF and minDF for Bag of Words were much different than that used in TFIDF. As frequent words can massively change the outcomes of algorithms based on Bag of Words, the maxDF used was 0.1, much less than that for TFIDF. MinDF was set to 0.001. This is because low frequency words do not have a high impact in BoW therefore a higher minDF does not impact our results but instead manages to reduce LDA runtime as we have smaller vectors.

We then select a number of topics for each model, train it, then calculate probabilities for each topic for each article and send these values for supervised learning where these probabilities will be the features.
This part is then similar to the Supervised Learning section.

Deep Learning
We also explored deep learning methods, seeking to exploit the additional predictive power of sequence models such as RNNs to improve our accuracy.

Word2Vec
As we saw in the earlier count vectorization approach, there were a massive number of words (16000), hence leading to each feature being relatively sparse, which will hinder a good model from being trained.

Word2Vec is an algorithm that takes in the corpus of text data and learns a mapping from words to vectors. In this way, the algorithm will learn the relations that words in the vocabulary have in relation to each other, and similar words will be mapped closer to each other in the vector space. Additionally, this will greatly reduce the dimensionality of the feature space (from 16000 words, to 100-dimensional vectors).

Procedure
Firstly, the raw content text is preprocessed â which involves removing punctuation, removing non-alphabetic words, and filtering out stop-words.

Then, the Word2Vec model was trained using the corpus of article content data to learn a 100-dimensional embedding. In this way, the model will be able to capture the semantic relationships specific to Pennâs con-

text, such as between buildings in Penn, or between fraternities and sororities (see images below), hopefully giving the classification model more predictive power.

```
2 model.wv.most_similar_cosmul(positive = ['fraternity', 'female'], negative = ['male'])
executed in 21ms, finished 09:17:39 2020-12-11

[('ifc', 0.9987518191337585),
 ('sorority', 0.9928473830223083),
 ('mgc', 0.9732818603515625),
 ('panhellenic', 0.9669802188873291),
 ('sororities', 0.9545149207115173),
 ('greek', 0.9544658064842224),
 ('panhel', 0.9450474381446838),
 ('chapter', 0.9426577091217041),
 ('chapters', 0.9415575265884399),
 ('recruitment', 0.9344643354415894)]
```

```
2 model.wv.most_similar('huntsman')
executed in 24ms, finished 09:19:46 2020-12-11

[('steinbergdietrich', 0.7880867123603821),
 ('houston', 0.7732806205749512),
 ('stiteler', 0.7369509935379028),
 ('tangen', 0.7138106822967529),
 ('meyerson', 0.7133698463439941),
 ('golkin', 0.7085722088813782),
 ('fagin', 0.7020806074142456),
 ('fisherbennett', 0.70000797510147l),
 ('gathwala', 0.6967608332633972),
 ('skirkanich', 0.6869189143180847)]
```

Then, we use a tokenizer to convert the raw titles and content into a sequence of token-ids, and pad the resulting tokens to ensure they all have the same length. We then use the tokenizer to create the embedding matrix (mapping a token-id to the 100-dimensional embedding corresponding to its word), which will be the first layer in the recurrent neural network.

GRU Model

To capture sequential information in an article title, we train recurrent neural networks on the tokenized input matrix, with the aim of predicting whether an article will be in top 25% (classification), or the articleâs percentile for that month (regression).

Since the vanilla RNNs suffer from short-term memory, more sophisticated improvements, such as Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRUs), are used by the NLP community, especially due to long-distance relations between different words in a sentence or article. They have gates that can regulate the flow of information, learning which data in the sequence is important to keep or forget, and all state-of-the-art results based on RNNs are achieved through LSTMs. GRUs are very similar to LSTM, just with only two gates (reset and update gate), and hence are much faster to train while retaining almost similar accuracy. Given the constraints in time and computational power, we decided to use GRUs for our project.

Hyperparameter Tuning Model Architecture

Tuning the hyperparameters in the RNN model is of utmost importance because of neural networksâ propensity to overfit the data. This was evident from the first time the RNN model was trained non this dataset â attaining up to 90% accuracy for the train dataset but only 60% for the validation set. To solve this, we implemented a grid search over 3 regularization hyperparameters to determine the best model architecture. Each model architecture was fitted on the training data and validation accuracy is the criteria that we based our model selection on.

The final model architecture and hyperparameters are shown below, as well as the information on the training process.

Neural Network Architecture	
Layers (with final hyperparameters)	**Hyperparameters Considered**
Embedding Layer (100-dimensional)	
Dropout p = 0.5	**1**: Dropout p: 0.1, 0.2, 0.3, 0.4, 0.5
Gated Recurrent Unit (32 recurrent units, dropout p = 0.5, Kernel L2 weight decay of 0.001)	**1**: Dropout p: 0.1, 0.2, 0.3, 0.4, 0.5, **2**: Kernel L2 Weight Decay: 0.0001, 0.001, 0.01
Batch Normalization	**3**: Include Batch Normalization: True or False
Dense (32, Kernel L2 weight decay of 0.001)	**2**: Kernel L2 Weight Decay: 0.0001, 0.001, 0.01
Output (1-dimensional)	
Training Information Trained initially with embedding layer frozen for 20 epochs, early-stopping with patience of 5 Optimizer: Adam, Learning Rate 0.001 Then embedding layer unfrozen, trained for further 50 epochs, early-stopping with patience of 10 Optimizer: Adam, Learning Rate 0.0005	

Model Explainability with ELI5

A commonly-perceived limitation of deep learning models, despite their superior accuracy, is that the models are black-boxes and not interpretable. In recent years, however, much progress has been made in developing methods to make deep learning models more interpretable, such as Local Interpretable Model-Agnostic Explanations (LIME). We used the Python package ELI5 to derive the contributions of each word in to the overall title prediction, hence offering a much deeper level of insight into how the trained model learns to classify article titles.

BERT

While the Word2Vec learns word embeddings, the words are not context-specific and face the problem of polysemy (same words having different meanings based on context). In recent years, NLP researchers have developed more sophisticated approaches to solve this problem, such as training language models using complex bi-directional LSTM architectures, such as ELMo and ULMFiT. However, the current state-of-the-art language modelling method is based on a transformer architecture, and does not require recurrent units. For the purposes of our classification, we use BERT â or Bi-directional Encoder Representations from Transformers. BERT contains stacked blocks of encoders, attention heads and multiple layers of embeddings to capture deep semantic elements of language. BERTâs pre-training steps â Masked Language Modelling and Next Sentence Prediction â allow it to learn a language model from massive amounts of unsupervised text data on the web.

We can thus use transfer learning to fit the pre-trained BERT language model to train further on our text classification task. A straightforward implementation of BERT classification models that abstracts away all the pre-processing steps is provided in the simple transformers package which we used. The BERT model was trained for 20 epochs on article titles, as well as article content, separately.

BERT Classification Results

Classification Results (Test Data) – Article Titles

Accuracy 68.22%
Precision 77.56%
Recall 51.28%

	Predicted	
Actual	**0**	**1**
0	2657	463
1	1520	1600

Classification Results (Test Data) – Article Content

Accuracy 67.26%
Precision 80.65%
Recall 45.42%

	Predicted	
Actual	**0**	**1**
0	2780	340
1	1703	1417

Final Ensemble

In this process we take predictions of each and every trained model on a dataset neither of them have been trained upon. After doing the usual train, test split on these predictions, we then use these 23 predictions as a feature for a new supervised learning against the percentiles.

6 Experiments and Results

We will be using Logisitic Regression as our Baseline throughout this project. This is because logisitic is a comparitively simple method compared to say Adaboost and also naturally includes all variables in its model. Logistic Regression on Title serves as the overall baseline.

In this section, we will present a summary of the results obtained from all models: LDA, supervised learning, deep learning, and ensemble.

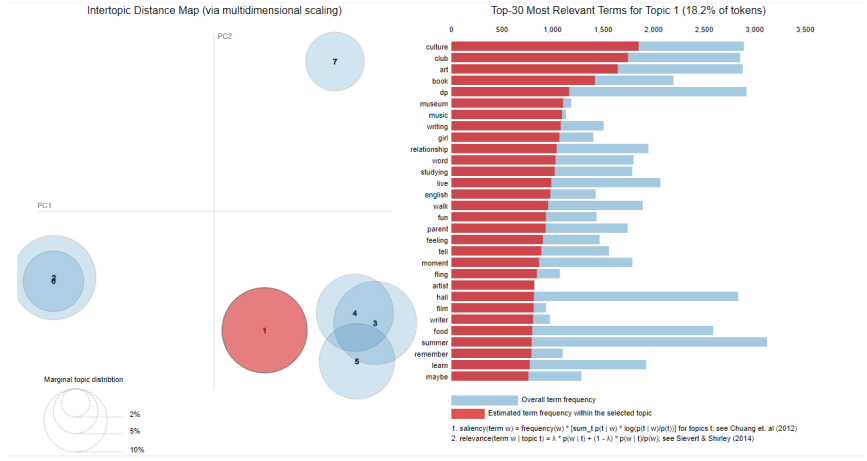
Note that any scoring metric mentioned will always refer to scoring metric when the model is applied on a testing set, i.e. its scores for data it has not been trained on.

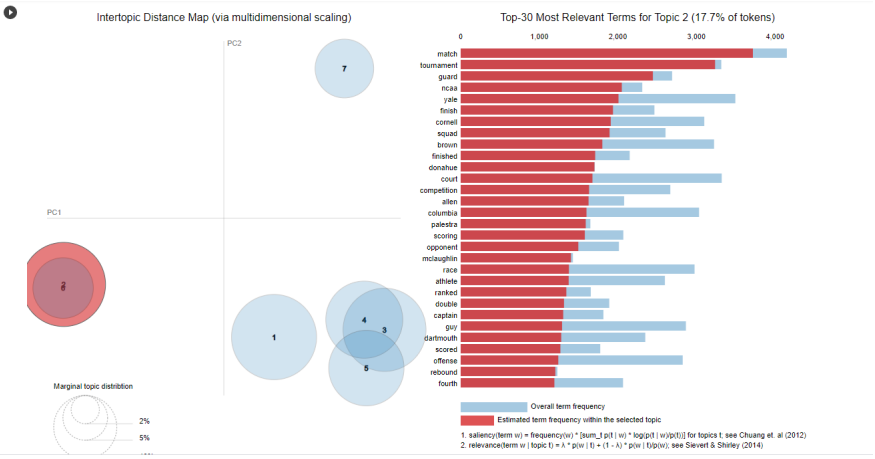
Topic Clustering

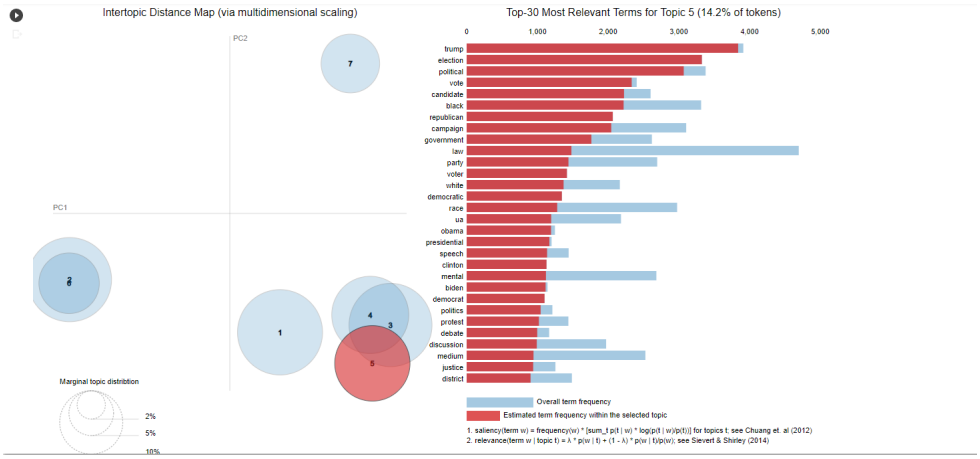
LDA

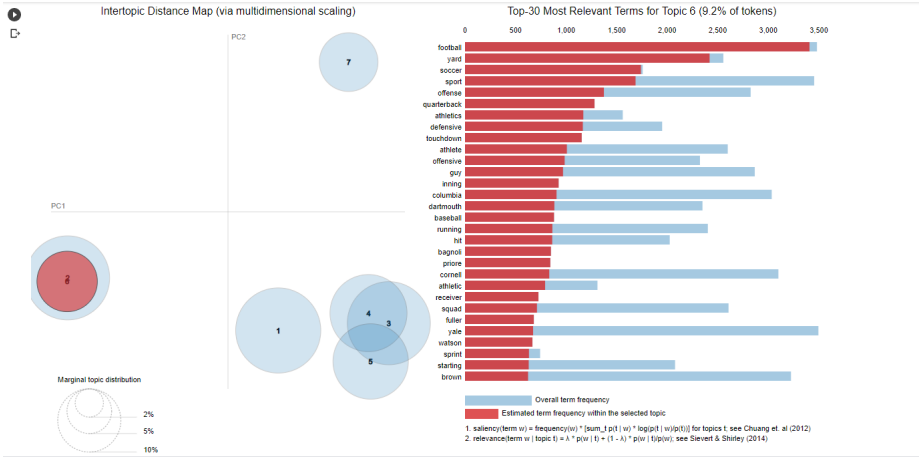
Sklearn's LDA was used on our count vectorizers. The main hyperparamter to look for was the number of topics. Divided data into test and train and trained an LDA on train for different number of topics. Would then check the log likelihood score for each LDA on the test data to select the best number of topics as 7. Below is a PyLDAvis graph describing each of the 7 topics, their keywords, and a chart to show how much they differ from each other.

We can see from the images below that Topic 1 appears to be Art Culture. Topic 2 appears to be related to general sports. Topic 5 appears to be about politics. The Graph also shows how different topics are similar to each other, for e.g. Topic 6 seems to overlap with Topic 2 which makes sense because Topic 6 is some particular types of sport such as soccer and football.









NMF

Similar was done for NMF. However, we noticed that NMF manages to compute 0 probabilities for most topics for any articles compared to LDA which still had a small probabilities for any topic for any article. As such it can be seen as a form of regularization thus a higher number of topics could be selected. So in order to create variance in the supervised models to be trained later, we choose NMF with 15 topics. Attached below are some topics, its keywords, and its top 5 articles.

Topic 0: Keywords: ivy team season league game year princeton tournament player conference harvard coach title quaker win play yale championship ncaa basketball

Jacobson — Ivy Tournament was fun â but was it best for the league?
After 26 seasons, Ivy League baseball abandons divisional format
Penn women's basketball attempts to climb closer to the Ivy League mountain top
Photo Essay — Men's Lacrosse Historic 2019 Season
Women's Hoops Issue: Penn women's basketball faces tough 2014-15 schedule

Topic 2: Keywords: people like time friend life thing class year think really know make want college experience feel day way work world
Carlos Arias Vivas — Navigating Penn with your best friend
Emily Hoeven — Our invisible legacy
Emily Hoeven — The 'best four years of our lives'
Senior Column by Emily Hoeven — The columnist manifesto
Tyler Larkworthy — Your four years at Penn don't have to be the best of your life

Topic 4: Keywords: trump election republican candidate vote clinton voter political state democratic president democrat biden presidential party campaign pennsylvania donald obama hillary
Louis Capozzi — A tale of two Trumps
Clinton-supporting Republicans are also shocked, upset over Trump victory
Louis Capozzi — Another brick in the wall
There is voter fraud in the U.S., but will not change the outcome of the election, according to Penn Prof.
A roadmap of Trump's candidacy and its effects at Penn 1718 Editorial — Making change in Trump's America 1747 Students political groups upset, shocked over Trump's victory 149 Trump to make first 2020 appearance in Pa. in a town hall on Thursday 2119 How college towns voted on Super Tuesday 2149 Cameron Dichter — âSanders, Trump and the threat of Democracy

Topic 8: Keywords: sexual assault fraternity campus violence university greek woman student group sorority member event chapter organization policy community black rape issue
Editorial — Expanding our vision
One year after the results of the AAU sexual assault survey, a look at what's changed
Penn to revamp NSO sexual violence training
Three changes coming to campus following the AAU sexual assault survey
Guest column by Isabella Auchus — A letter to President Gutmann

Topic 11: Keywords:ua class board representative election student vote assembly sac president body undergraduate committee member group year event running chair budget
UA presidential hopefuls discuss experience, visibility in first of three debates
Guest column by Gabe Delaney and Julie Bittar — It's about leadership, not salesmanship
Editorial — UA: Undeniably absurd
UA Recap: SAC Moratorium and UA parliamentary procedure
Third and final UA debate tackles efficiency, housing and minority representation

Topic 12: Keyword:health mental cap student care patient wellness service medicine suicide medical resource counseling issue task illness psychological campus help hospital
Guest column by Penn Democrats — Health care sanity
Penn launches mental health hotline
The year in mental health
Editorial — Donât let the CAPS stigma discourage you from seeking help
The Penn mental health resources every freshman needs to know

Supervised Learning

- **Feature: Article Titles**

Mean Squared Error	Mean Absolute Error
23.23%	19.26%

Table 1: performance of Ridge Regression using *article titles*

Model	Hyper-parameter Tuning	Accuracy	Precision	Recall	F1 Score
Logistic Regression	$C = 10$	66.47%	69.35%	59.04%	63.78%
Random Forest	max depth = 45, max features = 50, number of estimators = 25	65.85%	61.63%	83.97%	71.09%
AdaBoost	max depth = 200	61.28%	68.16%	42.34%	52.23%
SVM	Linear	66.89%	69.86%	59.42%	64.22%

Table 2: performance of various classification models using *titles*

- **Feature: Tags**

Mean Squared Error	Mean Absolute Error
24.47%	20.74%

Table 3: performance of Ridge Regression using *tags*

Model	Hyper-parameter Tuning	Accuracy	Precision	Recall	F1 Score
Logistic Regression	$C = 0.001$	66.00%	61.83%	83.59%	71.08%
Random Forest	max depth = 40, max features = 50, number of estimators = 25	68.00%	65.74%	75.16%	70.14%
AdaBoost	max depth = 200	62.85%	58.54%	88.08%	70.34%
SVM	Linear	67.95%	67.01%	70.71%	68.81%

Table 4: performance of various classification models using *tags*

- **Feature: Author Positions**

Mean Squared Error	Mean Absolute Error
28.57%	25.07%

Table 5: performance of Ridge Regression using *author posititons*

Model	Hyper-parameter Tuning	Accuracy	Precision	Recall	F1 Score
Logistic Regression	$C = 0.001$	62.40%	60.83%	69.65%	64.94%
Random Forest	max depth = 40, max features = 50, number of estimators = 25	62.21%	63.84%	56.31%	59.84%
AdaBoost	max depth = 200	61.74%	57.70%	88.04%	69.71%
SVM	Linear	62.75%	58.63%	86.70%	69.95%

Table 6: performance of various classification models using *author positions*

- **Feature: LDA Topics Probability**

Model	Hyper-parameter Tuning	Accuracy	Precision	Recall	F1 Score
Logistic Regression	$C = 0.001$	66.12%	63.79%	74.58%	68.76%
Random Forest	max depth = 7, max features = 5, number of estimators = 25	69.03%	66.05%	78.36%	71.67%
AdaBoost	5	68.58%	65.54%	78.36%	71.38%
SVM	RBF	67.16%	63.49%	80.76%	71.09%

Table 7: performance of various classification models using *LDA topic probabilities*

- **Feature: NMF Topic Probabilities**

Model	Hyper-parameter Tuning	Accuracy	Precision	Recall	F1 Score
Logistic Regression	$C = 0.001$	64.26%	59.89%	86.31%	70.72%
Random Forest	max depth = 15, max features = 10, number of estimators = 50	66.08%	70.88%	54.61%	61.69%
AdaBoost	4	66.09%	62.70%	79.42%	70.78%
SVM	rbf	67.32%	63.56%	81.21%	71.30%

Table 8: performance of various classification models using *NMF topic probabilities*

Deep Learning

RNN (GRU) Classification Results

Test Data
Accuracy 70.27%
Precision 76.16%
Recall 60.01%

	Predicted	
Actual	**0**	**1**
0	2491	593
1	1262	1894

BERT Classification Results

Test Data – Article Titles
Accuracy 68.22%
Precision 77.56%
Recall 51.28%

	Predicted	
Actual	**0**	**1**
0	2657	463
1	1520	1600

Test Data – Article Content
Accuracy 67.26%
Precision 80.65%
Recall 45.42%

	Predicted	
Actual	**0**	**1**
0	2780	340
1	1703	1417

The results show that BERT does not give any additional accuracy over the optimized GRU model. Additionally, using the article content performs worse than the article titles. However, no hyperparameter tuning was done for BERT due to time constraints, and it is possible that a few percentage points of increased accuracy can be achieved with tuning, though it is unlikely to improve over the optimized GRU model.

Final Ensemble
The features for this are the prediction of all our models: 23 in all.

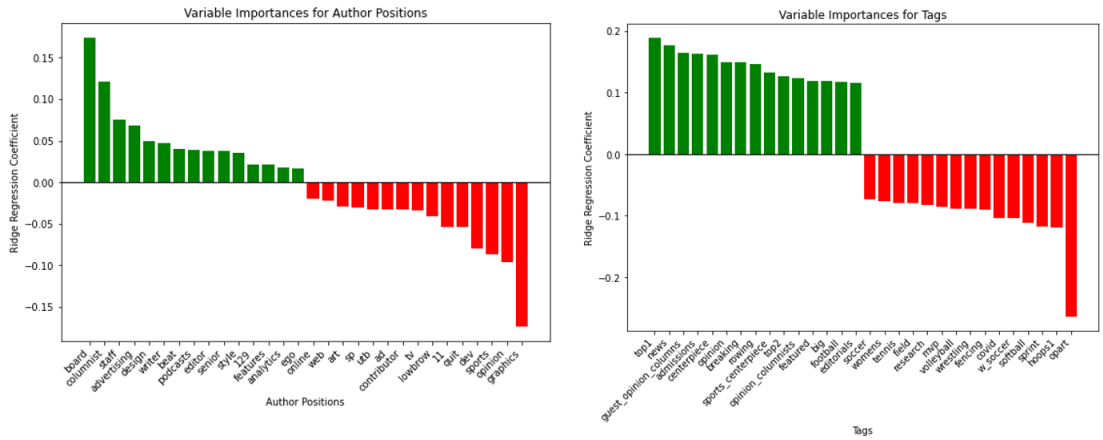
Model	Hyper-parameter Tuning	Accuracy	Precision	Recall	F1 Score
Logistic Regression	$C = 1$	87.58%	82.56%	94.41%	88.09%
Random Forest	max depth = 20, max features = 7, number of estimators = 50	85.82%	80.56%	93.42%	86.51%
AdaBoost	max depth = 7	79.66%	76.11%	84.84%	80.24%
SVM	Poly	75.72%	73.35%	78.76%	75.96%

Table 9: performance of various classification models using *model predictions*

We notice that most of our models give accuracies similar to that of our baseline. However the final ensemble gives a much better result. From the results of the final ensemble we would choose Logistic Regression with $C = 0.1$ to be applied on predictions from our 23 models as all four metrics for it are higher than any other found throughout this project. Random Forest with the best hyper-parameters also has satisfactory performance quite near to Logistic in Final ensemble but all four of its score metrics are less than that. The accuracy of 87.58% is especially remarkable considering none of the individual models had one greater than 70%.

7 Conclusion and Discussion

As evidenced above, the overall performances by the various models are quite satisfactory, especially the final ensemble model. To provide insights into why this is the case, we decided to examine the variable importance of different words in each feature and see if the words that have the highest correlations actually match our intuition and experience when working with news staff at the DP.



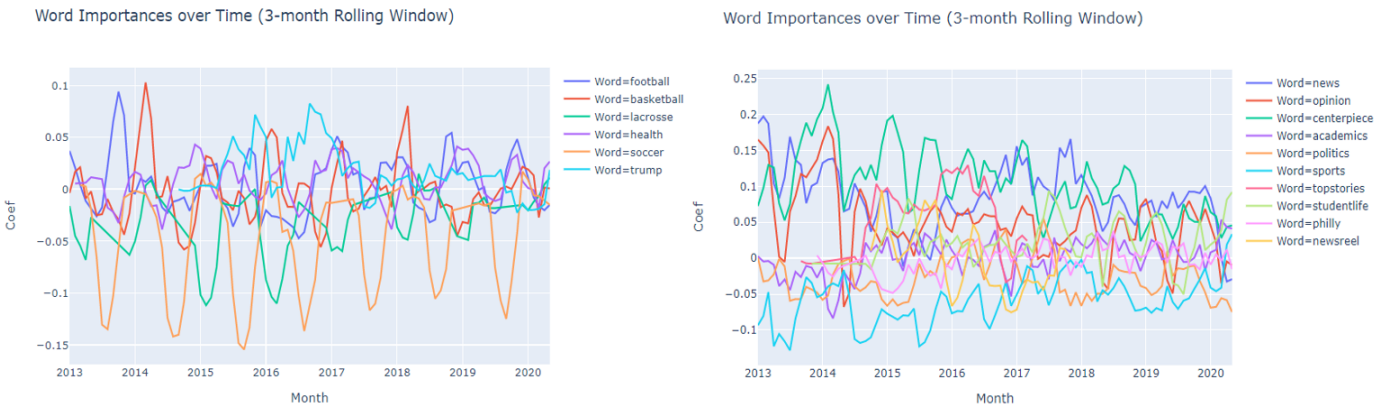
Variable Importance for author positions & tags

A good example of variable importance confirming our intuitive knowledge of how the DP works is the relatively low average performance of sports articles. While many sports articles achieve readership comparable to quality news articles, the sports department puts out a large number of articles providing live updates and recaps to sports games. These articles, though essential to good sports coverage, receive relatively few page-views and thus lower the average of articles tagged as sports articles.

At first glance, it may seem confusing that certain tech-centric roles have influence in our model given that those departments have never published articles. Their inclusion is due to staff members that have worked in multiple departments and thus are classified both as a reporter of some sort and as a tech department staffer. Analytics is a good example of this; the importance of this role remained low for many years before jumping up in importance in 2017 and 2018. We actually traced this increase in importance of analytics staffers to two members of this team, one of whom was a full time reporter in 2017 and another who was a corresponding reporter in 2018. In addition to validating that our model accurately reflects the past, this observation also has a cool circular aspect to it: our model has been influenced by work its creators completed years ago.

Besides author positions, most tags also have correlation values that match our intuition. As mentioned briefly in the section on data set, the tag of an article defines where it appears on *thedp.com* home page. So it is reasonable to see how tags like *centerpiece*, *breaking*, *featured* have pretty positive correlations compared to other tags because an article tagged *centerpiece* is the most dominant (the first) article on the home page and articles tagged *breaking* will appear on the navbar of the home page for around 48 hours. Similar explanation applies to tags such as *top1* and *top2*. Although these tags are no longer used today, they serve the same purpose as *centerpiece* and *featured* back when they are being used.

We can also notice that *opart* has the most negative correlation. Opart represents opinion art, such as comics and cartoon insertions. However, these types of media, being relatively informal, are less likely to be be viewed on *thedp.com* because it contradicts with the impression of *thedp.com* being a website that provides factual information. Therefore, the negative correlation makes sense. Also quite a number of sports tags receive negative correlation, such as *tennis*, *field*, *mvp* and *volleyball*. This also matches the negative correlations of sports in the variable importance of author positions.



Variable Importance Over Time

Another novel method we explored is to use a 3-month rolling window of training data to train the ridge regression model. This allows us to chart and analyse the varying nature of variable coefficients over time, and uncover some interesting trends.

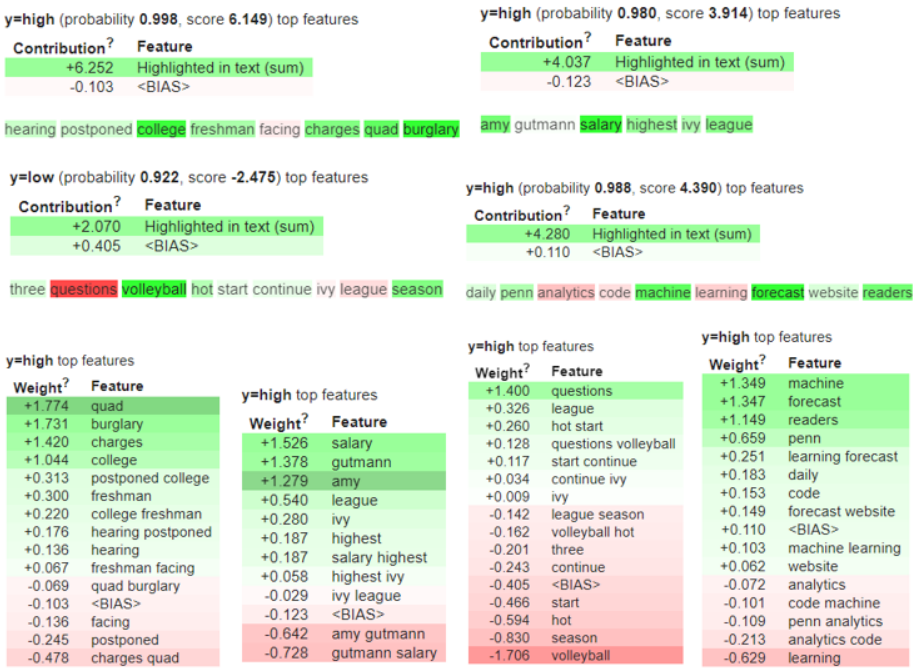
Running this method on the Bag-of-words features (left), we can tease out the most important words. The highly seasonal nature of sports is evident, with basketball peaking in March and soccer peaking in December to February, compared to lacrosse which peaks in the summer. We also see the importance of Trump in 2016, which matches our intuition because of his initial presidential campaign attracted a lot of fanfare.

Next, looking at the time plot of Tags (right), we can see that the centerpiece and news tags unsurprisingly contribute most positively to views. As we discussed in an earlier section, sports articles consistently rank the most negative contribution to views. An interesting seasonal trend is the dip in opinion and centerpiece in summers, while politics picks up in the summer, possibly because opinion articles and Penn-related

important news taper off in the summer vacation.

RNN Explainability using ELI5

Some examples of the output of the ELI5 package for explainable RNN model predictions are shown below (not all articles are real!): Note: For the articles displayed in title form, green corresponds with the words that contribute the most to the prediction. For the words displayed in table form, green corresponds with words that contribute the most to the positive class (high).



These reinforce the trends that we have discovered earlier - sports-related are associated with lower readership, while UPenn related words such as "Penn", "Amy" (Gutmann), "Ivy League", as well as common college vocabulary (college, freshman), attract higher viewers. High-polarity words such as "burglary" and "postpone" that normally appear in news headlines are also positive predictors of popularity.

Conclusion

We set out to predict the popularity of articles from Penn's student newspaper, The Daily Pennsylvanian (The DP), and we succeeded with 87.58% accuracy. Our project serves to provide The DP with a reliable way to predict whether or not an article will perform in the top 25% of articles that month, aiding the company in its website and social media strategy. We trained supervised classification models using text and metadata features, and used unsupervised topic modelling methods such as LDA and NMF. We also perform deep learning methods, such as recurrent neural networks and BERT. Finally, we combined predictions from all individual classifiers as features in a final ensemble model that in all likelihood can outperform even the most instintual of editors.

Acknowledgments

We acknowledge the DP professional staff who provided us with the staff position CSV files and news staffers for exchanging their insights and experiences on page views. Without their help, this project would not have been completed. We would also like to thank the data analytics manager for providing us with the data we needed and for being supportive of our project.