# Forecasting Wikipedia Page Views with Graph Embeddings

**Category: General Machine Learning**


Anthony Miyaguchi
acmiya@stanford.edu

Shaon Chakrabarti
shaonc09@stanford.edu

Nicolai Garcia
nicolaig@stanford.edu


December 14, 2019

## 1 Introduction

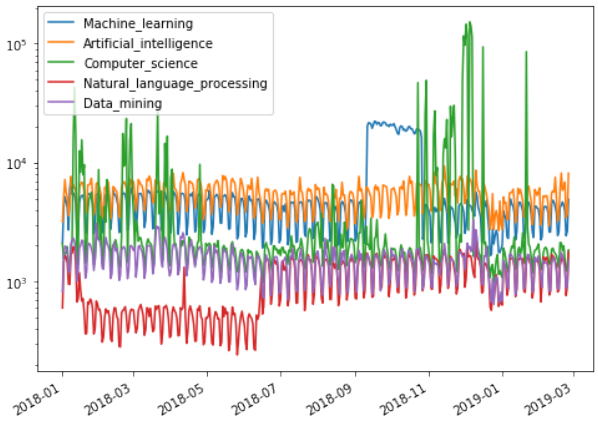
Web traffic patterns evolve in response to information demand. Models of these patterns can be used to analyze trends across pages or to forecast future demand. Can the hyperlinked structure of the web be used to improve forecasting accuracy of page-view across a set of sites?

We analyze models of Wikipedia page views and the effects of include unsupervised graph features . The models predict a 7 day window of traffic to articles using 60 weeks of historical page views. We hypothesize that embedding article hyperlinks into unsupervised features can improve the accuracy of our predictions. We quantify our results in two experiments to determine the significance of the graph structure on page traffic regression.

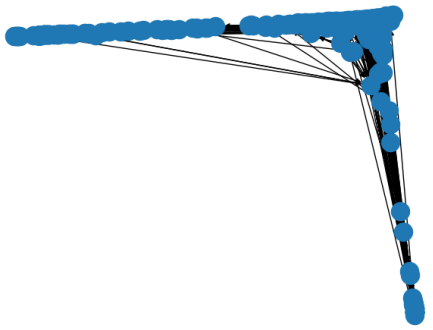
## 2 Related Work

We are motivated by a competition hosted by Kaggle for web traffic forecasting of a 1 month period over 145,000 articles [15]. The large number of correlated time-series can be modeled using latent-space methods like temporal-regularized matrix factorization for imputation and forecasting [10]. Convolutional neural networks have also been used to model the physical spatio-temporal nature of vehicle traffic networks [12].

## 3 Data




(a) The page views for top-10 articles ordered by PageRank score in log scale.



(b) A drawing of the small network using a spectral embedding into two dimensional space. This is drawn by the Spectral layout in NetworkX.[4].

We collect the Wikipedia dump from August 2019 and 21 months of page view data from January 2018 to September 2019 [16]. We contribute to `epfl-lts2/sparkwiki` and use the pre-processing tools to convert raw SQL dumps into Parquet [13]. There are $5.9 \times 10^6$ articles and $4.9 \times 10^8$ directed hyperlinks between them.

Our exploratory experiments comprise of a small topic-specific data-set, with $1.2 \times 10^3$ series connected by $3.0 \times 10^4$ edges with average degree of 48 as seen in figure 1b and figure 1a. This is sampled by inducing a graph from the neighbors of the `Machine_learning` article. Induction removes all edges that do not contain an end within the article set.

In our spectral embedding experiment, we consider a graph induced by uniformly sampling 5% of articles, then keeping the largest connected component. The trial in table 2 is $3.7 \times 10^5$ series linked by $1.66 \times 10^5$ edges with an average degree of 9.

In our spectral clustering experiment, we consider a graph formed by sampling 50% of articles with $2.9 \times 10^6$ series connected by $5.2 \times 10^7$ edges. Due to increased memory and computation requirements, we implement a recursive clustering algorithm to generate unsupervised labels and lower-dimensional embedding information.

# 4 Features

Our features split between historical page view and unsupervised graph features. Due to the complexity of mining graph features, we consider two related spectral algorithms at different scales.

## 4.1 Autoregression

Historical page views are the basis of the traffic regression. For each window in time, we consider a limited window of history as auto-regressive features. The equation for an auto-regressive model $AR(p)$ is given as follows:

$$x_t^{(i)} = \sum_{j=1}^{p} \theta^T x_{t-j}^{(i)} - \varepsilon_t \tag{1}$$

## 4.2 Graph Embedding

The adjacency matrix of Wikipedia is sparse and high dimensional. The matrix is unsuitable for general machine-learning applications left in this form. We compute a graph embedding to incorporate the structure of the network into our models. An embedding preserves distances between nodes in the graph spectrum and transforms the sparse adjacency matrix into dense features. In our first experiment with spectral embeddings, we compute a Laplacian eigenmap [3] by solving for the smallest-but-one eigenpairs of the modified adjacency matrix of the graph.

$$Lf = \lambda Df \tag{2}$$

L is the graph Laplacian and D is the diagonal degree matrix. We omit $f_0$ corresponding to the first eigenpair sorted by descending eigenvalues, and slice the next m eigenvectors normalized by eigenvalues to obtain a m-dimensional embedding of the graph. The similarity between two nodes is computed as the dot product of their mappings in embedding space. Figure 1b demonstrates the use of this embedding to separate nodes along two principal axes. The Laplacian eigenmap is computed using `scipy.sparse.csgraph.laplacian` and `scipy.sparse.linalg.eigsh` functions [14].

In our second experiment, we sample an induced graph with $5 \times 10^8$ edges. Due to the performance issues with the spectral embedding, we implement a spectral clustering algorithm introduced in [2] resulting in a partition vector and associated label. The partition vector is the 1-dimensional Laplacian eigenmap, utilizing the connectivity properties of the second smallest eigenvector of the Laplacian to approximate a cut that splits the graph in two equally-sized partitions. We exploit the even volume of partitions in our bi-partitioning routine, implemented using Apache Spark[11] and the GraphFrames[8] library. We cut the graph with 8 splits, generating 256 partitions that each contain 10,000 articles. The sign features are the sign of the partition vector, which are used to determine the partition id for each article.

| | partition_id | title | degree | inDegree | outDegree | sign_0 | fiedler_0 | 2018-01-01 | 2018-01-02 | 2018-01-03 |
|---|---|---|---|---|---|---|---|---|---|---|
| 242143 | ooxooxox | Chrome_Cats | 90.0 | 41.0 | 49.0 | False | -2.425485e-11 | NaN | 105.0 | NaN |
| 269854 | xxxxxxoo | Theta_Apodis | 81.0 | 30.0 | 51.0 | True | 6.379909e-10 | NaN | NaN | NaN |
| 208110 | oooooxxo | Baking | 972.0 | 745.0 | 227.0 | False | -1.833444e-06 | 347.0 | 555.0 | 565.0 |
| 59530 | xxxoxxxx | Formia | 484.0 | 203.0 | 281.0 | True | 1.780937e-10 | NaN | NaN | 118.0 |
| 226987 | ooxxooxo | Muscat | 556.0 | 326.0 | 230.0 | False | -1.835822e-06 | 796.0 | 1027.0 | 907.0 |

Figure 2: A sample record from the design matrix of our second experiment. The Fiedler vector is the 1-dimensional Laplacian embedding that determines the algebraic connectivity of the graph [1]. There are 8 sign and vector features and 608 page view features. PageRank and degree features for experiments are computed using NetworkX and GraphFrames respectively.[4][8]

### 4.3 Models

We report the performance of the following regression models for page view traffic. The first is a persistence model, which requires no learning and exploits the weekly seasonality of the time-series. The next baseline is the mean of each series in the train set. Our baseline machine learning model utilizes linear regression on a week of data, corresponding to an $AR(7)$ model.

Next, we add an $L_2$ regularization parameter to the linear model, which increases the bias of the algorithm and prevents the parameters from becoming too large. To tune the regularization parameter, we search over the reciprocal distribution between $1 \times 10^{-9}$ to $1 \times 10^{9}$ using a random search. We train a non-linear model using a fully connected neural network. Rectified linear units (ReLU) provide non-linearity, while varying layer sizes and depths are tuned for the size of the data-set using cross-validation. We hypothesize that non-linear nature of time-series and the hierarchical nature of the graph embeddings can be exploited by a dense neural network.

Finally, we compare our methods with the reference implementation of the temporal-regularized matrix factorization (TRMF) [10] library to compute a 7 day forecast. Like the collaborative filtering algorithm in recommendation systems, TRMF factorizes a latent temporal space that can be used for imputation and forecasting by utilizing the co-occurrences between different series.

| Model | Description |
|---|---|
| Persistence | $\hat{y}_{t_0}^{(i)} = x_{(t_0-T)}^{(i)}$ |
| Mean | $\hat{y}_{t_0}^{(i)} = \frac{1}{T}\sum_t^T x_{(t_0-T+t)}^{(i)}$ |
| Linear Regression | $\hat{y}_{t_0}^{(i)} = \sum_{t=1}^{T} \theta^T x_{(t_0-T+t)}^{(i)} - \varepsilon_{t_0}$ |
| Ridge Regression | Linear model with $L_2$ regularization |
| Neural Network | Fully connected, feed-forward, with $L_2$ regularization |
| TRMF[10] | Temporal latent space model using matrix methods |

Table 1: The models that are reported for the spectral embedding and clustering experiments.

In addition to the models reported in our main experiments, we include results in our exploratory experimentation. We include a weighted linear regression model that utilizes PageRank and the $L_2$ norm of the 10 nearest-neighbors in embedding space. We use a decision-tree regressor as another class of non-linear models. We also model the time-series using Poisson Regression.

Our models are primarily trained using the implementations provided by `scikit-learn` [6]. We also use the generalized linear model for Poisson regression in `statsmodels`. [5][11].

## 5 Experiments

### 5.1 Training, Validation, and Testing

We divide 14 weeks of historical page views into three parts: 58 weeks for modeling, 1 week for validation, and 1 week for testing. We choose a window to forecast, train the model on 58 weeks data and use that model to make a prediction using the concatenation of shifted training data and validation data. The prediction is used to calculate the error between predicted and actual value.

### 5.2 Experiment Design

As mentioned in section 4.2, we run two separate experiments to determine the effect of graph embeddings at different scales in forecasting traffic. In the first experiment on a smaller graph, we can compute expensive centrality and embedding features like PageRank and the Laplacian eigenmap. In our second experiment, we opt to compute degree measures and a recursively-mined partition vector as a scalable analogue. These unsupervised graph features are concatenated to the page history. For each model, we tune our parameters using 5-fold cross-validation. We then run 10 trials for each embedding type across all reported models. We perform an ablation analysis by removing features in a backward search. We compute the difference in our error metrics when removing the feature from our best non-linear model.

### 5.3 Error Metrics

We consider two metrics to determine whether our prediction $Z$ fit well to the observed data $Y$. We consider measurements over a predicted time window. Our measurements are consistent with TRMF [10]. The Mean

Absolute Precision Error (MAPE) has the form:

$$MAPE = \left[ \frac{1}{nT} \sum_{i=1}^{n} \sum_{t=1}^{T} \left| \frac{Y_t^{(i)} - Z_t^{(i)}}{Y_t^{(i)}} \right| \right] * 100\% \tag{3}$$

The Normalized Root Mean Square Error (NRMSE) has the form:

$$NRMSE = \frac{1}{\sum_i^n \sum_t^T Y_t^{(i)}} \sqrt{\frac{1}{nT} \sum_{i=1}^{n} \sum_{t=1}^{T} (Y_t^{(i)} - Z_t^{(i)})^2} \tag{4}$$

# 6 Results

| model | NRMSE | MAPE |
|---|---|---|
| persistence | 3.27 | 8.3 |
| mean | 3.25 | 13.6 |
| linear reg | 3.65 | 20.0 |
| poisson | 5.10 | 48.7 |
| weighted linear reg (PageRank) | 3.45 | 25.6 |
| weighted linear reg (average embedding) | 3.39 | 21.4 |
| decision tree | 3.27 | 13.6 |
| neural network | 2.88 | 11.4 |

Table 2: A table of results from our exploratory experiments.

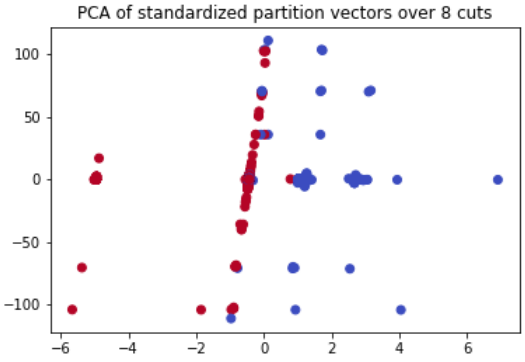



Figure 3: A projection of the zero-mean, unit variance partition vectors onto the first two principal axes. The labels are generated by the signs of the vector, corresponding to the approximate ratio-cut.

| class | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.42 | 0.58 | 174190 |
| 1 | 0.55 | 0.96 | 0.7 | 128158 |
| avg / total | 0.77 | 0.65 | 0.63 | |

Table 3: The precision, recall, and f1-scores for the labels generated by the first cut in the second experiment, corresponding to figure 3. The accuracy is reported to be 0.65.

| | Experiment 1: | Embedding | Experiment 2: | Clustering |
|---|---|---|---|---|
| model | NRMSE | MAPE | NRMSE | MAPE |
| persistence | $7.2 \pm 1$ | $7.0 \pm 0$ | $7.5 \pm 0$ | $8.4 \pm 0$ |
| mean | $5.7 \pm 0.6$ | $7.2 \pm 0$ | $6.1 \pm 0$ | $8.9 \pm 0$ |
| linear reg. | $6.1 \pm 0.6$ | $7.6 \pm 0.3$ | $5.9 \pm 0$ | $7.7 \pm 0$ |
| ridge reg. | $6.0 \pm 0.6$ | $7.5 \pm 0.3$ | $6.2 \pm 0$ | $8.5 \pm 0$ |
| neural network | $5.9 \pm 0.7$ | $6.9 \pm 0.3$ | $5.9 \pm 0$ | $9.1 \pm 0.3$ |
| trmf | - | - | $6.5 \pm 0$ | $8.6 \pm 0.1$ |

Table 4: The results for the experiments with the spectral embedding and clustering features.

| Experiment 1 | NRMSE | MAPE |
|---|---|---|
| neural network | $5.9 \pm 1.0$ | $6.9 \pm 0.9$ |
| without history | $0.2 \pm 0.6$ | $-0.4 \pm 0.4$ |
| without pagerank | $0.1 \pm 0.4$ | $0.0 \pm 0.4$ |
| without embedding | $0.0 \pm 0.4$ | $0.0 \pm 0.4$ |

| Experiment 2 | NRMSE | MAPE |
|---|---|---|
| neural network | $5.9 \pm 0.1$ | $9.1 \pm 1.0$ |
| without history | $0.1 \pm 0.0$ | $0.9 \pm 0.4$ |
| without degree | $0.0 \pm 0.1$ | $0.0 \pm 0.6$ |
| without sign | $0.1 \pm 0.0$ | $0.7 \pm 0.4$ |
| without vector | $0.1 \pm 0.1$ | $0.6 \pm 0.4$ |

Table 5: Ablation analysis performed by removing features one at a time. The first row gives the score of the model with all features, and each subsequent row displays the difference from the best model. Positive deltas represent better model performance without the feature-set.

# 7 Discussion

For our unsupervised method of partitioning, we ran a logistic regression model on the PCA transformed partition vectors to determine the relationship between the eigenvector and the assigned partition of the node. In an arbitrarily chosen trial, the learned decision boundary of a logistic regression model shows high recall for one class and high precision for the other class, with an accuracy that is better than random. In figure 3, we can visually draw a decision boundary and note that the labels act as noisy, unsupervised labels. The clustering algorithm was intended for generating a local embedding for each partition, but we observed that these induced graphs had less edges than the number of nodes in the group. Instead, we rely on equivalency of the 1-dimensional embedding and the partition vector to generalize our methods to larger graphs. The trade-off between computation and the size of the graph was one of the challenges in engineering unsupervised features for the model.

We were not able to build a model that consistently outperformed simple persistence and linear models. The short forecasting window, sparsity of historical page view data, and daily resolution may contribute to the performance. This is evidenced by lackluster performance of TRMF, which has empirically worked well in electricity and traffic data-sets with hourly resolution over a week.

In our Ridge Regression model, we noticed that the best $L_2$ regularization parameter took on a large value of $1.8 \times 10^8$, which indicates a bias and low variance model. As we varied the number of time-series in our training set, the best regularization parameter for the best cross-validation score increased as well. This may be correlated with the overall volume of traffic across all article sets and the spikiness in the data. We tried per-series scaling with standard scaling and min-max scaling to reduce the differences in scale. However, we did not see any changes in our error metrics when running linear regression and opted to leave history alone.

Training the neural neural network was a laborious process. For the first experiment, our best network was 3 hidden layers of size $(100, 50, 50)$ with regularization between $1 \times 10^{-1}$ to $1 \times 10^{-2}$. We ran an exhaustive search over cross-products up to order 7 (e.g. permutations of size 7). Then from the best 4 of these, we randomly searched for the best regularization terms. For the second experiment, our best model had layer sizes $(64, 32, 64, 16)$ with regularization 0.002. We observed that the layers chosen for the best cross-validation did not necessarily translate to full training set.

Based on our experimental design, we determined that the embedding features do not improve performance at daily granularity. In the second experiment, we note some reliance on graph features by observing the greater increase in MAPE by dropping historical page views than the sign or vector features. This observation may be related with the high regularization term needed for ridge regression to perform well during cross validation. The results of our ablation analysis on these trials prove that our hypothesis is inconclusive on a 7 day window with 60 weeks of history at a daily granularity.

# 8 Future Work

We would proceed next by building a convolutional neural network to exploit symmetries in the time and graph signals, as well as forecasting over a longer window (months) or higher frequency (hours).[12] Our model would involve a GPU-accelerated deep learning library to experiment with alternative architectures suitable for time-series like RNNs, LSTMs, and seq2seq.[7]. Depending on scale, a graph convolutional network could replace graph embeddings.[9] Additionally, we would attempt to increase the scale our clustering implementation to increase our training set for non-linear models.

# 9 Code

All code is open source and can be located at https://github.com/acmiyaguchi/cs229-f19-wiki-forecast

# 10 Contributions

Anthony helped curate data and set direction. Shaon helped model and run the experiments. Nicolai helped model and collect data.

# References


[1] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, vol. 23, no. 2, pp. 298–305, 1973.

[2] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 11, no. 9, pp. 1074–1085, 1992.

[3] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[4] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[5] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[6] 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*, vol. 12, pp. 2825–2830, 2011.

[7] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[8] A. Dave, A. Jindal, L. E. Li, R. Xin, J. Gonzalez, and M. Zaharia, "Graphframes: An integrated api for mixing graph and relational queries," in *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems*, ACM, 2016, p. 2.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[10] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Advances in neural information processing systems*, 2016, pp. 847–855.

[11] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, *et al.*, "Apache spark: A unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[12] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[13] N. Aspert, V. Miz, B. Ricaud, and P. Vandergheynst, "A graph-structured dataset for wikipedia research," in *Companion Proceedings of The 2019 World Wide Web Conference*, ACM, 2019, pp. 1188–1193.

[14] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. 1. 0. Contributors, "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python," *arXiv e-prints*, arXiv:1907.10121, arXiv:1907.10121, Jul. 2019. arXiv: `1907.10121 [cs.MS]`.

[15] *Web traffic time series forecasting*. [Online]. Available: `https://www.kaggle.com/c/web-traffic-time-series-forecasting/overview`.

[16] *Wikimedia downloads*. [Online]. Available: `https://dumps.wikimedia.org/`.