# CS6241 Final Project: Doubly Weighted Graph Dynamics and Reddit

https://github.com/orichardson/util-graph-dynamics

Oliver Richardson (oer5)

## 1 Introduction

### 1.1 Related Work

When formalizing agency, there is a standard assumption that people act to satisfy preferences / utilities which do not change over time — or at least they ought not to, if the agent is rational. Micro-economists model people in situations where preference changes are not particularly important and time scales are short. At the same time, those interested in machine agency give agents explicit *reward functions* which are part of their identity and have no mechanism for change. Humans, however, often do not have well-defined preferences, and change the best-defined ones regularly. The issue of dynamic preferences has recently been an AI safetey concern [1], as an agent will likely need to to change more concrete ones as it encounters distributional shift. More directly, the issue has come into focus in certain parts of the philosophy community [5]

Meanwhile, there is a standard assumption that graphs are static objects, and often perfectly good temporal structure is thrown out in graph analysis, as recognized by [6][2]. However, these formulations of temporal graphs are not obviously continuous, and to my knowledge nobody has done explicitly done the work of thinking of graphs as continuous dynamic objects. Network interpolation based on edit distance [8] is a related problem, but the focus of this paper is not on the functional analysis of graphs as dynamic objects. So far as I can tell, there also has not been much work on discrete choice on static graphs: nodes are not assumed to have "values" per-se, but rather their role is determined purely by their connection to other nodes. In the same spirit might be something like [3], but this imposes a number of other unneeded constraints such as a spatial embedding.

### 1.2 This Project

Broadly speaking, there are four components to this project, corresponding to sections 2, 3, 5, and 6 —

**Section 2: Temporal Graphs.** Relatively easy functional analysis aimed at elucidating how we think about temporal graphs, situates them for use with standard functional analysis tools, such as convolutions, and allows you to formulate discrete choice problems in terms of derivatives rather than time steps, which results in qualitatively different behavior.

**Section 3: Discrete Choice on Graphs.** I formalize a few discrete choice models that operate on doubly weighted graphs, show that they result in stochastic matrices, and characterize fixed points.

**Section 4: Empirical Studies** The majority of the effort of the project is the empirical study with reddit data, where I fit my models to a real data set and use some light versions of the theory in the previous two sections to make a bit of sense of the network dynamics

**Section 5: A Library for Doubly Temporal Graphs.** Finally, in the course of running my experiments, I have produce an artifact of code with many nice properties and some optimized algorithms that my be useful to people in the future looking to do similar kind of work.

## 2 Temporal Graph Representations

There are many ways of thinking of a graph with temporal information. In class, we saw the formulation presented in [7], in which a temporal graph $T = (V, E)$, where $V$ is a set of vertices (as usual), and $E \subseteq V^2 \times \mathbb{R}$ is a collection of time stamped edges. Let $\mathsf{TG}$ be the set of temporal graphs of this kind.

Often people prefer to think of a "temporalized" version of a mathematical object $O$ as being a function $\mathbb{R} \to O$ which gives an object at every point in time. A 'temporal graph' of this kind, then, is a graph $G(t) = (V_t, E_t)$ for every time $t$; let $\mathfrak{T}$ be the set of temporal graphs of this kind. If we assume that the vertex set $V_t$ is constant, there are two different ways of embedding the temporal graphs in [7]. The first is to simply take the slice $G_{=t}$ of $t$ at time $t$ to be the subgraph which has exactly $t$ as its timestamp.

$$\begin{aligned}(-)_{=} : \mathsf{TG} &\to \mathfrak{T} \\ (V, E) &\mapsto \ \lambda t. \Big(\ V,\ \Big\{(u,v) \ \Big|\ (u,v,t) \in E\Big\}\ \Big)\end{aligned}$$

Assuming $V_t$ is constant, $(-)_=$ is a bijection, with inverse on edges given by $E \mapsto \{(u, v, t) \mid \exists t.(u, v) \in E(t)\}$. Of course, this structure is very discontinuous in time, and worse, the set of times $t$ such that the graph slices $E_t$ are inhabited has zero measure. It is reminiscent of a density function. To solve the latter problem, we can integrate over time to get a cumulative graph:

$$G_{\leq t} = \int_0^t (G_{=\tau})\ \mathrm{d}\tau := \left( \bigcup_{\tau:[0,t]} V(G_{=\tau}),\quad \bigcup_{\tau:[0,t]} E(G_{=\tau}) \right)$$

More explicitly, this gives the following alternate embedding of $\mathsf{TGplus.2ex}$ into $\mathfrak{T}$:

$$\begin{aligned}(-)_{\leq} : \mathsf{TG} &\to \mathfrak{T} \\ (V, E) &\mapsto \ \lambda t. \Big(\ V,\ \Big\{(u,v) \ \Big|\ \exists \tau \leq t.\ (u,v,\tau) \in E\Big\}\ \Big)\end{aligned}$$

### 2.1 Weighted Temporal Graphs and Convolution

When analyzing temporal graphs, we generally are not interested in the exact times that things happen, but rather we assume that there is some underlying stochastic process that one might like to shed light on. For this reason, we think of the actual data as samples from a stochastic process, rather than the ground truth about what the graph ought to look like.

If $G \in \mathfrak{T}$, then we have an adjacency matrix $A_G(t) : V^2 \to \{0, 1\}$ for each time $t$. Continuing our search for continuity, we will add weights to the edges so that $A_G(t) : V^2 \to \mathbb{R}$ is instead a real-valued function, which can represent partial edges. We can now add, scale, and integrate graphs (with a fixed vertex set) by adding, scaling, or integrating their adjacency matrices. Under this interpretation, the integral we wrote earlier may be clearer. We will now show that the representations discussed above can all be expressed in terms of convolutions.

Given a temporal graph $\mathfrak{T}$ and a kernel function $\phi : \mathbb{R} \to \mathbb{R}$, we can define the convolution

$$G * \phi := \lambda t.\ \int_{\mathbb{R}} \phi(t - \tau)(G(\tau))\ \mathrm{d}\tau$$

In particular,

- the cumulative graph is a convolution with the Heaviside function $\mathbb{1}(t) = \begin{cases} 0 & t < 0 \\ 1 & t \geq 0 \end{cases}$

$$G_{\leq t} = \int_{-\infty}^t (G_{=\tau})\ \mathrm{d}\tau = \int_{\mathbb{R}} \mathbb{1}(t - \tau)(G_{=\tau})\ \mathrm{d}\tau = G_= * \mathbb{1}$$

- The windowed graph is a convolution with a window $W_{\pm T}(t) = \begin{cases} 1 & t \in [-T, T] \\ 0 & \text{otherwise} \end{cases}$

$$G_{t \pm T} = \int_{t-T}^{t+T} (G_{=\tau}) \, \mathrm{d}\tau = \int_{\mathbb{R}} W(t-\tau)(G_{=\tau}) \, \mathrm{d}\tau = G_{=} * W_{\pm T}$$

- The static graph, obtained by deleting all timing information, can be thought of as a convolution with the constant function $\mathbf{1}$:

$$G[:] = \int_{-\infty}^{\infty} (G_{=\tau}) \, \mathrm{d}\tau = \int_{\mathbb{R}} 1 \cdot (G_{=\tau}) \, \mathrm{d}\tau = G_{=} * \mathbf{1}$$

We can also use Gaussian kernels for smoothing, exponential decay, and so forth. One should think of the kernel as the persistence of edges over time.

## 2.2 Distance Metrics on Temporal Graphs

For non-temporal weighted graphs with the same vertex set, there may be a number of reasonable distance metrics, depending on the application, but since we don't really have a good way to differentiate between edges in the general case, for now we will use the Frobenius norm on adjacency matrices, i.e.,

$$d(G, H) = \Big\| G - H \Big\|_F := \Big\| A(G) - A(H) \Big\|_F = \sum_{u,v \in V} |G_{u,v} - H_{u,v}|^2$$

While this may work well for static graphs, integrating it over time is a terrible idea for dynamic ones — to see this, let $Q$ be a static weighted graph on a vertex set $V$, $G$ be $Q$ with all edges annotated with a time $t_0$, and $G'$ be $Q$ with all edges annotated with the time $t_0 + \epsilon$, for some vanishingly small $\epsilon$. While $G$ and $G'$ are $\epsilon$-close in some sense (perhaps even the same graph differing by a small timing measurement error), the integrated distance is $2\|Q\|_F$, and in particular, larger than the distance between $G$ and the trivial 0-weighted graph.

The solution to this is to allow some bleed-over between times: implicitly, we are thinking of time as being somehow persistent. Given some kernel function $k$, we can define a time-smoothed norm on temporal graphs which we can integrate over:

$$\begin{aligned} \Big\langle -, = \Big\rangle_k &: \mathfrak{T}^2 \to \mathbb{R} \\ \Big\langle G, H \Big\rangle_k &:= \int_{\mathbb{R}} \Big\| (A(G) * k - A(H) * k)(t) \Big\|_F \, \mathrm{d}t \\ &= \int_{\mathbb{R}} \Big\| [(G - H) * k](t) \Big\|_F \, \mathrm{d}t \end{aligned}$$

Note that, while we had motivated this as a distance metric, the final form is only in terms of the difference between the two temporal graphs, making it implicitly a norm. Define $\|\cdot\|_k : \mathfrak{T} \to \mathbb{R}$ by

$$\|G\|_k = \int_{\mathbb{R}} \Big\| [G * k](t) \Big\|_F \, \mathrm{d}t$$

**Definition 1.** If $T = (V, E) \in \mathsf{TG}$ is a temporal graph and $D$ is a distribution on time offsets, let

$$\Big[ T \overset{\leftarrow}{\sim} D \Big] := \Big( V, \ \Big\{ (u, v, t + d) \ \Big| \ (u, v, t) \in E, \ d \sim D \Big\} \Big)$$

**Conjecture 1.** *If $k$ is $c$-Lipschitz, $D$ is a distribution with zero mean, and $T \in \mathsf{TG}$ is a temporal graph, then*

$$\Pr\left( \Big\| \Big[ T \overset{\leftarrow}{\sim} D \Big]_{=} - \ T_{=} \Big\|_k \geq \epsilon \right) \leq \frac{c\mathrm{Var}(D)}{\epsilon^2}$$

## 3 Utilities on Graph Nodes

Thus far, we have been considering temporal graphs with edge weights. However, in addition to link strength, we had also wanted to model utilities in the nodes themselves: in our experiments, the nodes will be subreddits, posts, or comments, all of which have some measure of quality for the node itself.

### 3.1 Simple Logit

To begin, let's note what happens in the extremely simple case where the population is modeled by a single chooser, selecting between the nodes of a graph whose edges don't matter. If this chooser has utility $U : V \to \mathbb{R}$, and we assume noisy selection drawn from a gumble distribution, then we recover a simple softmax of utilities. This distribution is determined purely by intrinsic utility functions of nodes, making choices purely based on the best utility induces a Markov process $P : V^2 \to \mathbb{R}$ whose transition probabilities are

$$P_{i,j} \propto \exp(U_i - U_j)$$

On the other extreme, if all nodes are equally good, and we just diffuse through the network according to some random walk, perhaps with a Markov process with stochastic matrix $P : V^2 \to \mathbb{R}$. These could be defined by weighted probability distribution over the weights (if non-negative).

### 3.2 Mixing Models

Imagine the following procedure: at node $u$, we draw a non-empty 'neighborhood' (which we may or may not require to include $u$ itself) $N_u \subseteq V$ from some distribution depending on the transition probabilities, which favors nodes easy to get to from $u$. From there, we choose an element of $N$ with probability proportional to the softmax of their utilities. This procedure defines a Markov process whose matrix $P$ is given by

$$P = (u, v) \mapsto \sum_{N \subseteq V} \left[ \Pr(N; u) \; \frac{e^{U_v}(\mathbb{1}_{v \in V})}{\displaystyle\sum_{s \in N} e^{U_s}(\mathbb{1}_{v \in N})} \right] \tag{1}$$

where $\mathbb{1}_\varphi$ is an indicator function for the proposition $\varphi$. I have written out the indicator in the bottom sum to more explicitly show that this is (the limit of) a mixed logit model, where the neighborhood $N$ is a latent variable, whose only role is to mask off alternatives that are not observed.

The probability $\Pr(N; u)$ on neighborhoods could be constructed in many reasonable ways — one could use random walks, even non-Markov ones, to generate paths, and sample from the probability of a given vertex appearing. To analyze the behavior in closed form, we will start by imposing the relatively minor restriction that the neighborhood probabilities depend only on some transition probabilities given by a stochastic matrix $Q$. Re-interpreting the above formula, this gives us a transformer

$$\begin{aligned} &\Psi : \mathrm{Stoch}(V) \to \mathrm{Stoch}(V) \\ &\Psi(Q) := (u, v) \mapsto \sum_{N \subseteq V} \left[ \Pr(N; u, Q) \; \frac{e^{U_v}(\mathbb{1}_{v \in N})}{\displaystyle\sum_{s \in N} e^{U_s}} \right] \end{aligned} \tag{2}$$

A natural question now becomes: when does $\Psi$ have a fixed point? Or equivalently, what are the $\Psi$-stationary stochastic matrices on $V$? In the context of the reddit analysis, such a matrix can be interpreted as an equilibrium flow of attention, where the appropriate attention (aside from some noise) is directed at the highest utility objects

To answer this question, we need to nail down $\Pr(N; u)$ a bit more; the remainder of this section will go through the details for specific neighborhood selection choices.

— **.1** ***k*-Sample Neighborhoods** Suppose that our neighborhood is given by $k$ i.i.d. samples from the transition probabilities $Q_{u,-}$ out of node $u$. In this case, $\Pr(N; u, Q) = \prod_s Q_{u,s}$ and so our transformer becomes

$$\left[\Psi^{(k)}(Q)\right]_{u,v} := \sum_{N \in V^k} \left[ \left( \prod_{s \in N} Q_{u,s} \right) \frac{e^{U_v}(\mathbb{1}_{v \in N})}{\sum\limits_{s \in N} e^{U_s}} \right] \tag{3}$$

First, a few sanity checks: in the specific case where $k = 1$, we get $\Pr(\{j\}; u, Q) = Q_{u,j}$, the transition probability, which is what we had wanted.

**Proposition 1.** $\prod_{s \in N} Q_{u,s}$ *is a pdf over* $N \subseteq V$ *of size* $k$ *, i.e., it is positive and integrates to one.*

*Proof.*

$$\begin{aligned}
\sum_{N \in V^k} \prod_{s \in N} Q_{u,s} &= \sum_{n_1 \in V} \sum_{n_2 \in V} \cdots \sum_{n_k \in v} \Big( Q_{u,n_1} Q_{u,n_2} \cdots Q_{u,n_k} \Big) \\
&= \sum_{n_1 \in V} Q_{u,n_1} \Big( \sum_{n_2 \in V} Q_{u,n_2} \Big( \cdots \sum_{n_k \in v} \Big( Q_{u,n_k} \Big) \cdots \Big) \Big) \\
&= \Big( \sum_{n_2 \in V} Q_{u,n_2} \Big( \cdots \sum_{n_k \in v} \Big( Q_{u,n_k} \Big) \cdots \Big) \Big) \Big( \sum_{n_1 \in V} Q_{u,n_1} \Big) \\
&\vdots \\
&= \prod_{i=1}^{k} \Big( \sum_{n_2 \in V} Q_{u,n_i} \Big) \quad = \quad \prod_{i=1}^{k} (1) \quad = \quad 1
\end{aligned}$$

□

Next, note that when $k = 1$, we aren't actually making any choices, so the decision cannot be influenced by the node utilities; in this case, we recover exactly the distribution $Q$ no matter what the utilities are. More precisely, theorem

**Proposition 2.** $\Psi^{(1)}(Q) = Q$

*Proof.*

$$\Psi(Q)_{u,v} = \sum_{n \in V} \left[ Q_{u,n} \, \frac{e^{U_v}(\mathbb{1}_{v=n})}{e^{U_n}} \right] = \sum_{n \in V} Q_{u,n} \mathbb{1}_{v=n} = Q_{u,v}$$

□

**Proposition 3.** *In general,* $\Psi^{(k)} Q = Q$ *if and only if, for all* $u, v \in V$,

$$\mathop{\mathbb{E}}_{n_1, \cdots n_k \sim (Q_u)^{k-1}} \left[ \text{softmax}(U_{\{v, n_1, \cdots n_{k-1}\}}) \right] = \frac{1}{k}$$

*Proof.*

$$
\begin{aligned}
& Q_{u,v} = \left[\Psi^{(k)}(Q)\right]_{u,v} = \sum_{N \in V^k} \left[ \left( \prod_{s \in N} Q_{u,s} \right) \frac{e^{U_v}(\mathbb{1}_{v \in N})}{\sum_{s \in N} e^{U_s}} \right] \\
\iff \quad & Q_{u,v} = k \sum_{N \in V^{k-1}} \left[ \left( Q_{u,v} \prod_{s \in N} Q_{u,s} \right) \frac{e^{U_v}}{e^{U_v} + \sum_{s \in N} e^{U_s}} \right] \\
\iff \quad & \frac{1}{k} = \sum_{N \in V^{k-1}} \left[ \left( \prod_{s \in N} Q_{u,s} \right) \frac{e^{U_v}}{e^{U_v} + \sum_{s \in N} e^{U_s}} \right] \\
& \quad = \mathop{\mathbb{E}}_{n_1, \cdots n_k \sim (Q_u)^{k-1}} \left[ \text{softmax}(U_{\{v, n_1, \cdots n_{k-1}\}}) \right]
\end{aligned}
$$

□

— **.2 Independent Inclusion Neighborhoods** While the above formulation works out nicely and the probability takes a convenient product form, it also has a hyperparameter $k$. Rather than restricting the size of $N$, we can select each node independently to be part of the choice set with probability $Q_{us}$. This may seem at first glance to be different, but is actually equivalent to a mixture of the above process with binomial coefficients. As a result, really we have not removed any extrinsic choices, but rather marginalized them out with an implicit choice of distribution over the length $k$. In general, for a general distribution $\lambda$ over $\mathbb{N}$, this means we have a transformer defined as

$$
\left[\Psi^{\lambda}(Q)\right]_{u,v} := \sum_{k \in \mathbb{N}} \lambda(k) \sum_{N \in V^k} \left[ \left( \prod_{s \in N} Q_{u,s} \right) \frac{e^{U_v}(\mathbb{1}_{v \in N})}{\sum_{s \in N} e^{U_s}} \right]
$$

— **.3 Binary Choice** The complexity of calculating both of the above models is exponential in the highest value of $k$ that contributes a non-negligible probability mass. Therefore, as a final suggestion, we propose the following:

$$
\Psi(Q;U)_{u,v} = \begin{cases} Q_{u,v} \dfrac{1}{1 + e^{U_u - U_v}} & i \neq j \\ Q_{u,u} + \sum_{s:V} \dfrac{Q_{u,s}}{1 + e^{U_s - U_u}} & i = j \end{cases}
$$

## 4 Doubly Weighted Temporal Graphs

We would ultimately like to develop theory for graphs which have both vertex and edge weightings which vary over time. If $G \in \mathfrak{T}$ is a temporal graph, we would like to use utilities $U : V \to E$ to explain changes in the graph weights over time. This corresponds to finding a solution to the optimization problem

$$
\Xi^* = \arg\min_{\Xi} \left\| \frac{\partial G}{\partial t} - \Xi(G;U) \right\|_{\phi}
$$

for some persistence kernel $\phi$, where $\Xi : \mathfrak{T} \times R^{V(\mathfrak{T})}$ ranges over graph transformers such as the ones in section 3.

## 5 Experiments

The second, and primary branch of this project is an empirical one: we focus on the task of predicting the future connectivity of a temporal graph based on its current configuration, including both vertex and edge weights. Rather than predicting the raw weights from these temporal graphs, we first column-normalize them, for several reasons:

1. As with with many real world data-sets distribution of most connectivity metrics (number of links included) across sub-reddits is heavy-tailed and unbounded. Moreover, the problem is worse for temporal graphs: there may not be temporal consistency in the weightings. For instance, Reddit as a whole is still growing, and so any method trained on the past would be off-distribution for the future in terms of raw magnitude.
2. Many of the simple methods at our disposal are very scale sensitive, and often explicitly expect zero-centered and unit-variance. SVM's are notorious for requiring a great deal of pre-processing to get good results.
3. It is still enough information to understand the dynamics of a graph: the column-normalized adjacency matrix $Q$ of a graph $G$ is enough to totally determine a linear markov process on the weights, which means that $Q$ is sufficient to construct most centrality and role-labeling methods, such as page-rank vectors, node2vec embeddings, and so forth.

The net effect of this is to give all of the methods below a boost, by allowing them to predict an easier problem, focused on the relative connectivities of the nodes, rather than the absolute scale of the values, which is less interesting from our perspective. With this in mind, our precise research question we target is as follows:

**Research Question:** *Given a stochastic transition matrix $Q_t$, arising from the column-normalized weight matrix of a the graph a time $t$, can we use the discrete choice models discussed above to predict the transition matrix at time $Q_{t+\Delta T}$ at some future time $t + \Delta T$?*

### 5.1 Data

For the experiments which follow, we view Reddit as a doubly weighted temporal graph, in which the nodes are subreddits, and the edges are numbers of links between them posted at a given time, i.e., in subreddit `r/A` a user might post a comment containing the string `r/B`. To someone who hasn't interacted with the site much, such links may not seem like a very strong signal, but in fact posting one is an incredibly common contribution, and can carry a great deal of information: it is often used as a concise way of pointing to a particular, well-known trope. For instance, people who fail to recognize a joke might be referred to `r/wooosh`, an arrogant submission might be referred to `r/iamverysmart`, and a mean-spirited one to are`r/trashy`.

The fact that people often don't know about these communities (some of which are better than others) and follow them, or re-discover them by seeing references, rather than simply scrolling through an algorithmically determined feed, makes Reddit a place particularly suited to this kind of network.

**— .1 Formal Graph Description** Because Reddit contains a lot of data across a giant period of time in a lot of different places, my data does not consist just of a single temporal graph, but rather many subgraphs of different numbers of nodes, across different time windows, and with varying temporal precision.

For a time interval $T$, a threshold paramter $\theta$, and size parameter $N$, I form a temporal subgraph with a fixed number of nodes, by identifying the $N$-highest utility nodes, call them $V_N$, and setting, for each $u, v \in V_N$ and $t \in T$,

$$\begin{aligned}(W_{u,v})_t &= \#\text{ links u}\rightarrow\text{v posted at time } t \\ (U_u)_t &= \#\text{ of posts with } \theta \text{ or more comments in u posted at time } t\end{aligned}$$

Because it is faster and more efficient to do this with a smaller number of discrete items in $T$, we model time as a number of discrete windows. We arbitrarily choose $\theta = 9$ to exclude posts with very few comments.

### 5.2 Predictors

In addition to our own models, we run a battery of additional baselines.

— **.1 Constant Function.** The easiest thing to do (which turns out to be very difficult to beat), is to predict that the transition matrix does not change, i.e., predict $Q_{t+\Delta T} = Q$. The effectiveness of this model points to the difficulty of the task.

— **.2 Random Noise.** An even worse baseline is a random matrix (elements drawn iid from a uniform distribution), scaled to be stochastic (each column summing to 1). While there is no shame in failing to beat the constant function (after all, conservatism is very safe and quite effective!), this baseline is the performance floor.

— **.3 The Identity Matrix.** The prediction $Q_{t+\Delta T} = I$ corresponds to users only referencing the sub that they're currently posting in, oblivious to the rest of the website.

— **.4 Matrix Powers.** Here we predict $Q_{t+\Delta T} = Q^n$ for some $n \in \mathbb{N}$. This corresponds to paths of exactly $n$ hops along a random walk given by $Q$. If Reddit users were ignoring content and merely clicking on every sub link, then posting a submission or comment, we would expect the evolution of the graph to look like this, and for $Q$ to be a fixed point of itself if the dynamics were stable.

Note that this subsumes the constant function and identity matrix baselines, with $(n = 1)$ and $(n = 0)$, respectively. Because of our interpolation methods (below), we will also see data for fractional powers of matrices, so we will get the entire spectrum between the two.

— **.5 SVMs.** We now use some more sophisticated baselines. We use support vector machines (SVMs) with linear and rbf kernels as regressors from the entire transition matrix, to each individual matrix element. If it were the case that this prediction problem was totally different for each subreddit identity, we would expect a method like this (although perhaps with a more sophisticated, kernel, such as one corresponding to a deep net) to outperform everything else, given enough training data.

— **.6 Linear Models.** We also use linear least squares in two ways: first, in the same way that we train the SVM's above: predicting each matrix element of $Q_{t+\Delta T}$ independently, and a second, we estimate $A \in \mathbb{R}^{|Q|^2 \times |Q|^2}$ directly using a single application of linear least squares on the entire dataset.

— **.7 Inversion Averages** We can also naively average the linear transformations seen in the training set to estimate what a matrix linear transform might look like:

$$Q \mapsto \left( \frac{1}{T} \sum_{t=1}^{T} Q_{i+1} (Q_i)^{-1} \right) Q$$

In practice, some matrices are only barely invertable, so as written, this method generates a matrix with entries of giant magnitude. To prevent this from hugely skewing averages, we clamp the values of the inverted matrix to $[-1, 1]$,

— **.8 Discrete Transformers** Finally, we arrive at the methods developed in the previous section. We make some use of $\Psi^{(k)}$ for $k = 2, 3$, and mostly focus on the non-subscripecause Reddit contains a lot of data across a giant period of time in a lot of different places, my data does not consist just of a single temported one $\Psi$, because it performs similarly (or better) empirically, and is much faster to compute, as $\Psi^{(k)}$ takes time exponential in $k$ to calculate without approximation.

## 5.3 Covariates and Parameters

— **.1 Interpolation** Because our methods are discrete in nature, we need a way of making their impact smaller when less time has elapsed. To this end, we form an $\alpha$-mixture with the constant function to obtain our final result. More explicitly, given a predictor $f$, which takes as arguments a matrix $Q$ and perhaps other arguments such as utilities $(-)$, we instead use one of the functions below as our predictor for $Q_{t+\Delta T}$:

$$\text{Linear: } (Q, -) \mapsto (1 - \alpha) Q + \alpha f(Q, -) \qquad \text{Multiplicative: } (Q, -) \mapsto \Big( f(Q, -) Q^{-1} \Big)^{\alpha} Q$$

— **.2 Temporal Offset** In principle, we could also fix the above problem by explicitly setting up the learning task by tailoring the temporal offset $\Delta T$ explicitly to our discrete choice methods. This is in some sense the dual to changing the interpolation parameter, except it also impacts the performance of the other predictors, which are actually fit to data. The way that this plays out (especially with respect to the mixture coefficient $\alpha$) is one of the more interesting results below.

— **.3 Graph Characteristics** As described in section 5.1.1, I have many different temporal subgraphs of Reddit data that I have trained on, whith different paramters, such as numbers of nodes, time windows, and temporal precision. Some of these characteristics, in particular, the number of nodes in the graph, have an important, non-trivial role to play in the the difficulty of the learning problem.

— **.4 Utility Scaling** Finally, because logit models are not scale-free in utility, the scaling of the utility metric matters. Because my surrogate for "utility" has numbers on the order of $10^7$ for some nodes and $10^2$ for others, taking the exponential of the difference between these numbers is rather ridiculous. I have three modes of dealing with this problem:

1. Take $U = \log U$, which undoes this exponential distribution, and effectively turns the logit model into a normal weighted coin. This is a reasonable thing to do if you think that the number of posts in a subreddit is already the exponential of some internal utility
2. Scale $U$ so that its maximum value is a constant, $\mathsf{U}_{\max}$. This preserves the logit flavor of the model. As $\mathsf{U}_{\max}$ goes to zero, the discrete choice models approach diffusion, as everything has the same utility. Accounting for this implicitly gives us one more baseline: the laplacian matrix $LQ$
3. Simply divide utilities by a big constant so it's possible to compute with them. This leaves the absolute magnitude in the equation somehow, but there is no principled way of setting factor you divide by.

## 6 Results and Discussion

Each heading below is a non-trivial result. As this is an exploratory study, I have looked at my data from many different angles. Combined with the fact that there is not a clear way to measure the number of degrees of freedom for many of the quantities I measure, it would be misleading to report $p$-values. Nonetheless, each finding I report persists across different slices of the reddit graph, and is stable, as far as I can tell—I have explicitly tried to falsify all of them. In some cases, I illustrate this with multiple figures, tables, or averages, but in general there is not enough space to include all of the evidence.

Although I believe all of these trends hold, a more targetted study would be required in order to fully establish any of the claims I make here. In an effort to make sense of the data, I also include some post-facto speculation in which I retroactively guess the causes for trends; this, too (and perhaps especially), should be taken with a grain of salt.

On that note, let's start with a depressingly negative result:

### 6.1 Without Interpolation, The Constant Function Outperforms Everything

This is true across all graph slices. Consider figure 1. On the left, we have $Q^2$, and the right, $Q$, used as predictors for $Q$ at a different time step. The $Q_t$ used to produce the prediction is on the right, and the prediction target time is given by the x-axis. This visualization is cluttered but lets us verify that there is nothing special going on temporally that would cause an average to hide things (for instance, if the graph didn't change throughout the year of 2018, causing the constant predictor to be better on average).

While our binary choice method, $\Psi$, outperforms $Q^2$, it does not even come close to the constant predictor, as seen in figure 2, where they are overlaid. These data are all from the 100-node temporal graph of most active subreddits from 2017-2019, but this is definitely the most robust feature of the data-set. There does not exist a time span $\Delta T$ for which $\|\Psi Q_t - Q_{t+\Delta T}\| > \|Q_t - Q_{t+\Delta T}\|$.

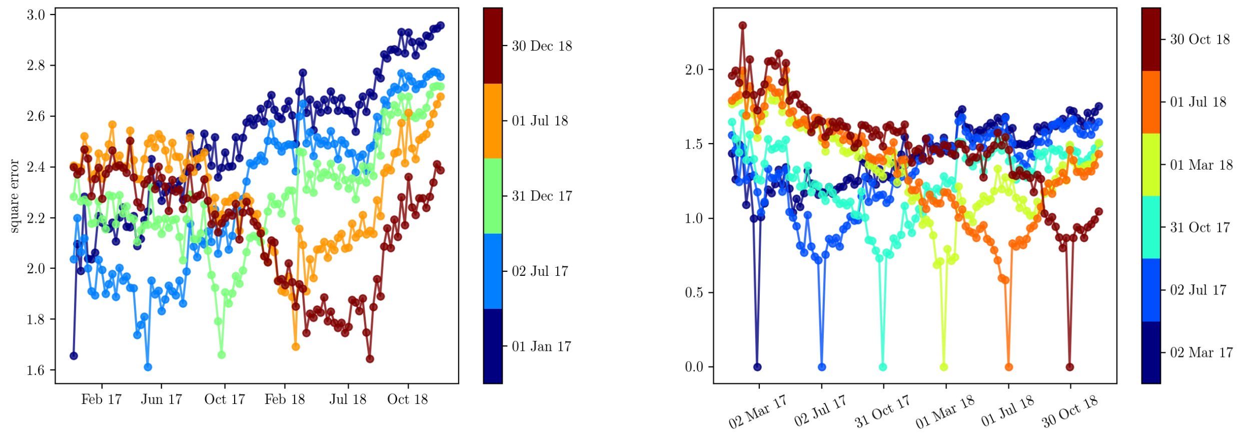


Figure 1: Errors for $Q_n^2$ (left) and $Q_n$ (right) as predictors for $Q_t$. $n$ is chosen based on the bar to the right, and $t$ is the x-axis

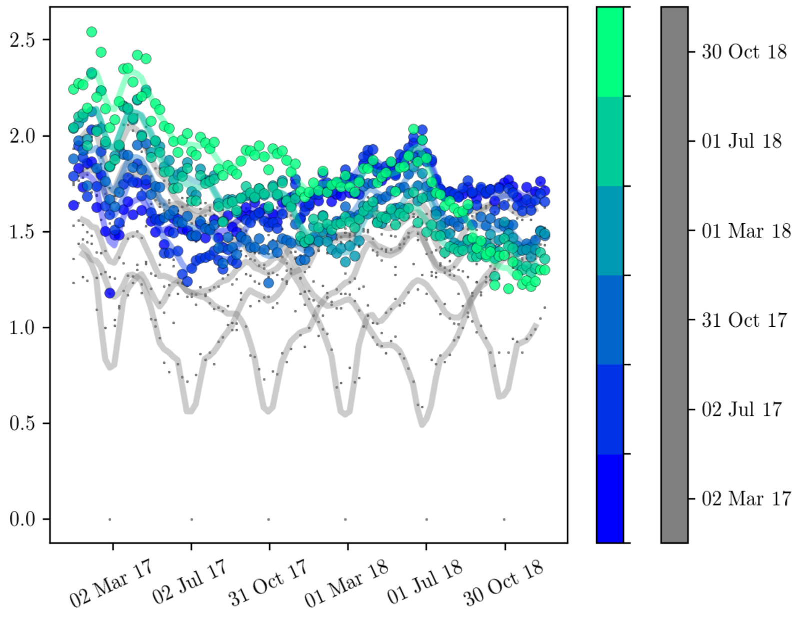
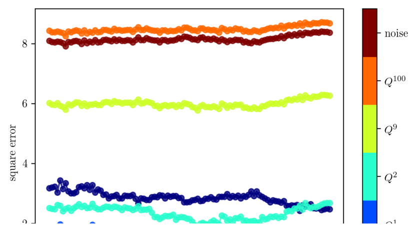


Figure 2: Errors for $\Psi$, in color, and the constant predictor $Q$, in gray. The different trends lines correspond to samples of $Q_t$ used to make the prediction, taken from different times.

## 6.2 Higher Matrix Powers are Really Bad



Because the visualization before is so cluttered, we'll throw out everything but the middle point, and look only at the matrix powers in figure 3. Note that each matrix power is strictly worse than the previous one, independent of what it is being used to predict. This goes against the prediction we made in section 5.2.4, where we speculated that $Q$ might be its own fixed point due to local stability. The limit point is approached by $Q^{100}$, on the right graph, which is even worse than everything else.

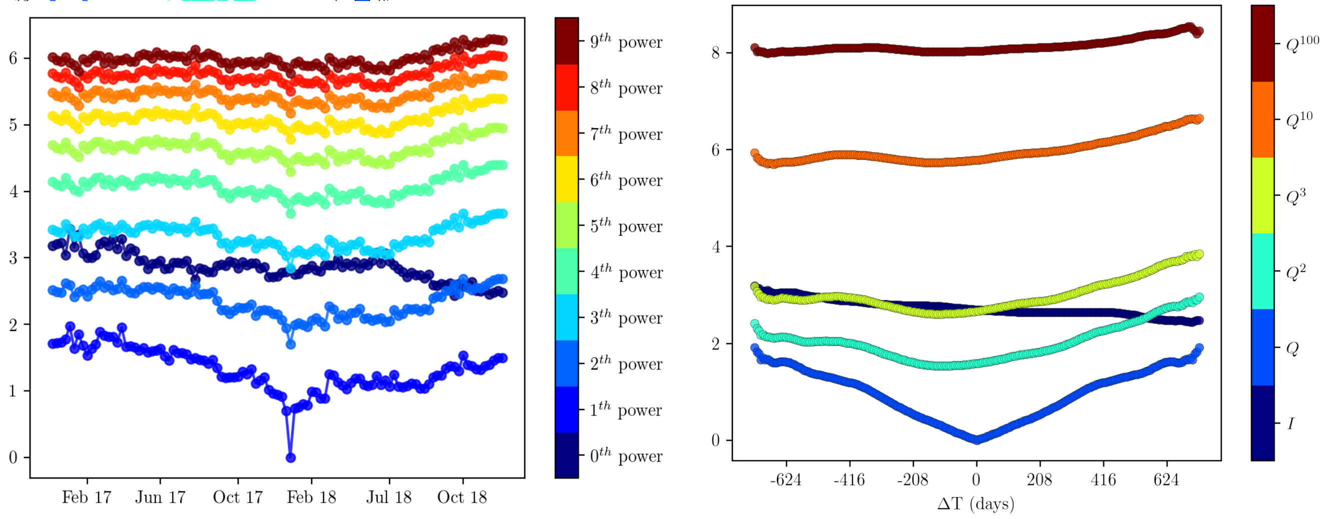


Figure 3: Matrix Power Errors. Left: for 100 nodes, single sample. Right: for a 30-node graph section, averaged across time deltas for stability

By the time that the matrix exponents get above 7, in this case, matrix powers are even worse than normalized noise, as seen on the left.

Another strange feature of both of these is that the identity matrix becomes a better predictor over time. This lends credibility to our next finding:

## 6.3 Reddit is Slowly Moving Towards Subreddit Isolation

Look again at figure 3. Note that, even averaged across all time steps, the identity function shows a very clear negative slope. This must mean that, no matter where you anchor your prediction, predicting the identity function is worse if you look at previous times than if you look at future ones — and therefore, as time goes on, more and more references to a subreddit are posted inside itself.

## 6.4 With the Right Interpolation, Ψ Outperforms Everything

Now, for the good news: by adding interpolation, Ψ outperforms every other model, including the fancier trained ones, despite having only a single parameter ($\alpha$), rather than a huge collection of support vectors or giant weight matrices. Now, this difference is very small, but but definitely significant, as it persists not only across graphs, but also across utility scalings. In table 1, you can see the results for most of the baselines that we have not yet seen on one particular graph.

There are a number of things to notice here: first, the margin by which our model beats the constant predictor is very slim. However, the difference cannot be attributed to noise, as we will see in later on (for instance, in fig 6, you can clearly see how much better Ψ for future predictions than past ones) —- not only clearly persists across a number of graphs.

|  | Square Error | Name | $U$-scale |
|---|---|---|---|
| █ | 0.000361 | $\Psi_{\alpha=0.06+}$ | `norm10` |
| █ | 0.000355 | $\Psi Q_{\alpha=0.13+}$ | `norm10` |
| 🟩 | 0.000354 | $\Psi Q_{\alpha=0.2+}$ | `norm10` |
| █ | 0.000355 | $\Psi Q_{\alpha=0.25+}$ | `norm10` |
| █ | 0.000363 | $\Psi Q_{\alpha=0.35+}$ | `norm10` |
| █ | 0.000373 | $\Psi^{(2)} Q_{\alpha=0.1+}$ | `norm10` |
| █ | 0.000379 | $\Psi^{(2)} Q_{\alpha=0.2+}$ | `norm10` |
| █ | 0.000373 | $\Psi^{(3)} Q_{\alpha=0.1+}$ | `norm10` |
| █ | 0.000368 | $Q$ | — |
| █████ | 0.001791 | AvgInv | — |
| █ | 0.000426 | SGDRegressor | `norm10` |
| █ | 0.000445 | SGDRegressor | No $U$ provided |
| ██ | 0.000775 | LinearSVR | `norm10` |
| █ | 0.000509 | LinearSVR | No $U$ provided |
| ████ | 0.001453 | rbf SVR | `norm10` |
| ████ | 0.001500 | rbf SVR | No $U$ provided |

Table 1: Errors across predictors, $\Delta T = 4$ days, $|G| = 30$ Nodes, 2013-2015

More things to note: linear SGD and the rbf SVR both benefit from the utility information. This is true for the other graph I ran as well. However, Linear SVR does not benefit from this, and it's not clear why this is the case. More experiments would be necessary to identify why this is happening, but unfortunately, the linear SVR actually takes the longest to train of any of these models. In the end, this probably does not matter, because all of the support vector machines are actually quite bad predictors. The average inversion is also a terrible predictor, which surprised me, but makes some sense because we're averaging in the wrong space.

More extensive tabular results are pre-computed and availble in raw text form at `https://github.com/orichardson/util-graph-dynamics/blob/master/writing/raw_results.txt`, which validate the the claims I'm making further, but to save space, and save myself the trouble of typsetting tables, I will leave the rest of them there.

## 6.5 Linear and Multiplicative Interpolation are Very Similar

This is most visually convincing from a heat map: see figure 4 on the right. There are two stacked heatmaps: in each, the color represents error (brighter is smaller), going right represents a higher value of $\alpha$ (which ranges from 0 to 1, so in particular, the left side of both diagrams is just the constant predictor $Q$), and going down represents a higher value of $\Delta T$. To get a clearer visualization of similar error colors, we are actually plotting

$$\frac{1}{10^{-15} + E}$$

The upper heat map is multiplicative interpolation, and the lower one is additive interpolation. Not only are they visually indiscernable, we also have numbers, in table 2.

The things to take from this table are: (1) linear and multiplicative interpolation result in very similar errors, (2) multiplicative is consistently slightly worse than additive, and (3) all of these again beat the constant predictor, again by a very small margin.

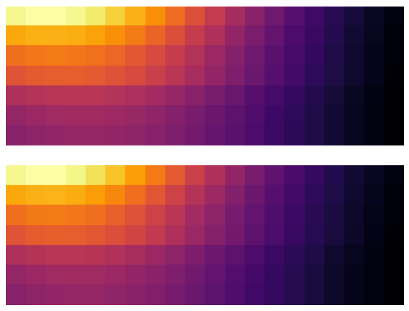


Figure 4: Linear and Multiplicative Mixtures, error on $\alpha \times \Delta T$

## 6.6 Fractional Matrix Powers Don't Help

While interpolation helps for our methods, they still do not help the matrix powers. In figure 5 are heatmaps of the same kind as before. The bottom one is the same as in figure 4, and the top one is the equivalent heatmap for the $Q^2$ predictor.

| Square Error | $\alpha$ | Interpolation Method |
|---|---|---|
| 0.00006581 | 0.06 | Linear |
| 0.00006625 | 0.06 | Multiplicative |
| 0.00006608 | 0.04 | Linear |
| 0.00006641 | 0.04 | Multiplicative |
| 0.00006577 | 0.11 | Linear |
| 0.00006629 | 0.11 | Multiplicative |
| 0.00006703 | 0 | – |

Table 2: Errors across predictors, $\Delta T = 4$ days, $|G| = 100$ Nodes, 2017-2018

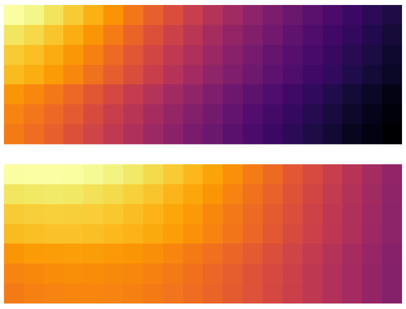


Figure 5: Fractional powers of $Q^2$ (top), and of $\Psi Q$ (bottom).

The feature of importance is that the $Q^2$ predictor's heatmap is monotonically increasing in error, both in $\alpha$ and in $\Delta T$. Multiplicative interpolation from $Q$ to $Q^2$ corresponds to fractional powers of $Q$, so this means that at no fractional point does a mixture of $Q^2$ and $Q$ outperform $Q$ alone. This is also anecdotally true for linear mixtures, but not shown.

This monotonicity is notably NOT shared by $\Psi Q$, whose brightest squares are not directly on the left edge, but rather a few further right. Once again, we remark that this is another instance of $\Psi Q$ achieving lower error than $Q$.

## 6.7 Number of Nodes Dictates Forward/Backward Bias of $\Psi$

In my view, the following is the most interesting result: for temporal graph slices that include a large number of nodes (and therefore, some with lower post activity), the accuracy of $\Psi Q$ looks like a future shifted version of $Q$[1], plus some additional error. See, for instance, the bottom row of figure 6. This is a really pretty result. Combined with our negative result in section 6.1, this suggests that we are indeed approximating a partial derivative

$$\Psi Q \approxeq \frac{\partial Q}{\partial T}$$

as opposed to merely sampling the function at some point in the future.

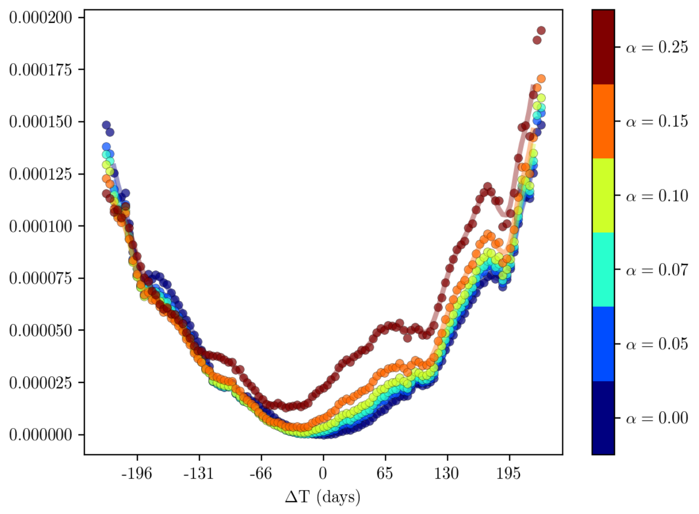
Now for the weirder bit: now take a look at the top line of figure 6. Here we see a similar (but less pronounced) shift backwards in time. Evidently, for $N = 9$, $\Psi Q$ predicts the past better than it predicts the future.

Here is my speculation about why this is the case: for small $N$, we are looking only at the very most popular subs, by activity. In these subs, the increased "utility" we're measuring really is just increased posting activity. But in really large communities that already receive a great deal of posts, having more posts makes the actual post quality go down: the biggest subreddits become famously riddled with reposts and are almost incredibly difficult to moderate, until almost nobody stays there for very long. The issue with our model is not algorithmic, but rather that our surrogate for utility points in the wrong direction after a certain point. Of course, for big $N$, there will also be quite a few smaller subreddits, so our mean squared error will be brought down by cases where increased activity actually increases the utility of a community.

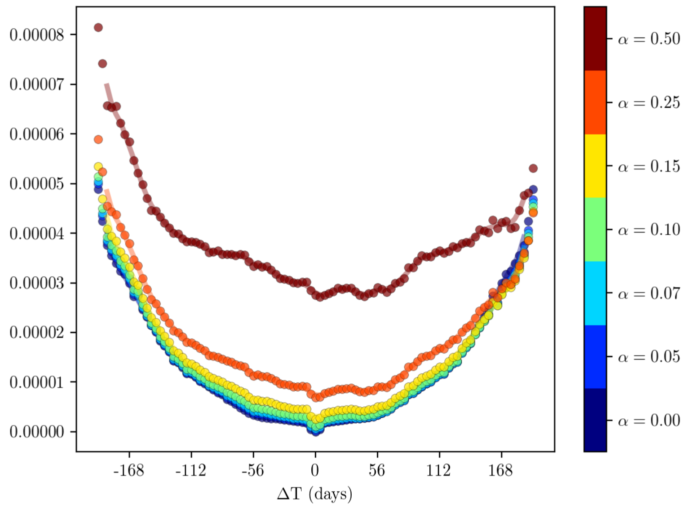
## 6.8 Utilites are Only Helpful for Near Future

We still have yet to mention the impact of the utility normalization on the error. In figure 7, we we are plotting the error against $\Delta T$, for utilities scaled by the ratio $r$ (Utilites which are by default normalized to 15), for two different temporal subgraphs.

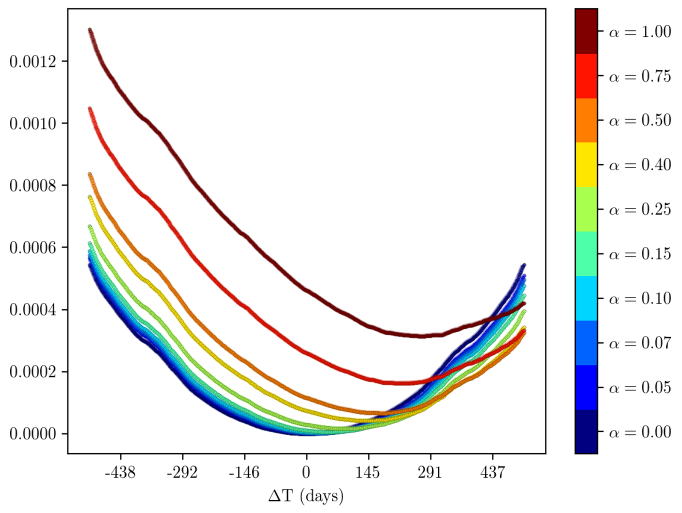
[1]Remember that $\alpha = 0$ corresponds to the constant function $Q$, in dark blue



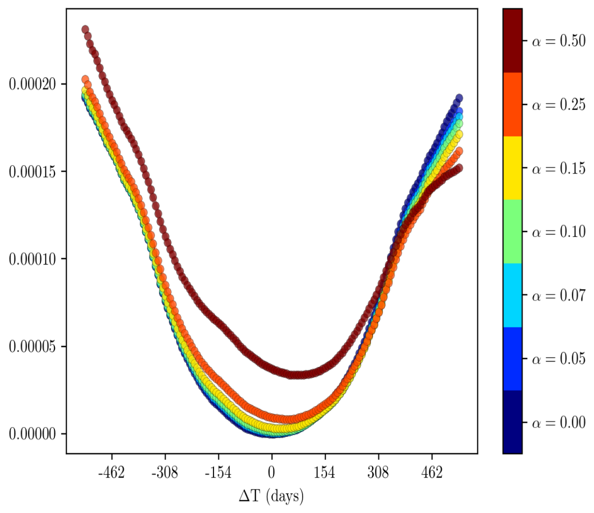
$(N = 9)$



$(N = 15)$



$(N = 30)$



$(N = 100)$

Figure 6: Tradeoff between $\alpha$ and $\Delta T$. For small $N$, the curve shifts backwards in time. In the mid-range, $Q$ is better than $\Psi Q$, and for high $N$, the error is future shifted

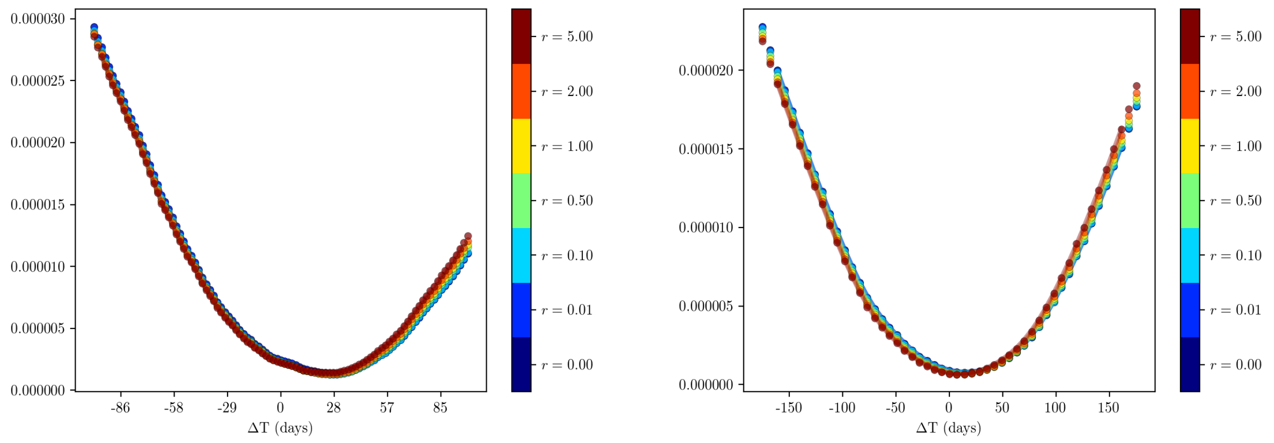


Figure 7: Big picture: with zero'd out utilities, seems to outperform the version that takes utility into account. left: $N = 30$, right: $N = 100$. Both: $\alpha = 0.07$. Utilities are base scaled to a maximum of 15, and then multiplied by the factor $r$ (colorbar)

The worrying bit is that the dark blue curve, in which we set all utilities to zero, seems to be more future-shifted than the ones in which we actually use utilities. Clearly, the curves are extremely close, at any rate. However, upon zooming in on timescales closer to zero, as in figure 8, we can see that adding utilities only makes predictions better, on the timescale of 25 days. This is true across values of $\alpha$, as shown in the figure.

The interpretation here is that in the far future, we cannot really predict the utilities very well, and a simple coin flip model is a better way to predict the connectivity than taking this very flawed measure of utility into account. If, as we suspect, our measure of utility is upside-down for large subreddits, then this makes some sense: we can't predict the long-term dynamics because the information

## 7 Implementation

A large (often overlooked) chunk of the work for projects like these is software engineering, and in my view a third contribution of this project is a collection of utilities for scraping, storing, manipulating, and plotting doubly weighted temporal graphs. I want to quickly use some space to provide a quick overview, because it's actually rather reusable. It is available at `https://github.com/orichardson/util-graph-dynamics`.

- **Experimental Utilities** [`experimental.py`]. A lot of code for calculating different kinds of regressions and heatmaps on temporal graphs. Wrappers for discrete choice functions so that they behave like `scikit learn` regressors, and can therefore be re-used by built-in sklearn gridsearch, etc. Finally, a class called `rsltdict`, whose keys are frozen parameter dictionaries, and can be queried with keywords and constraints to quickly find results for certain parameters.
- **Reddit Scraping Code** [`reddit_graph_builders.py`]. Code to build these temporal graphs from `pushshift.io` data, collecting any node or edge data, query the Reddit API to retrieve more recent karma information, and efficiently do batches under certain conditions. (It also has a compact console progress bar)
- **Data Structures** [`temporal.py`]. A python class with operations for for adding, convolving, saving, loading, and building weight tensors from, temporal graphs.
- **Choice Transformer Construction** [`choice.py`]. Perhaps less generally useful but contains things like interpolation, faster algorithmic implementations of the transformers defined in section 3, logit modes, and column normalization.

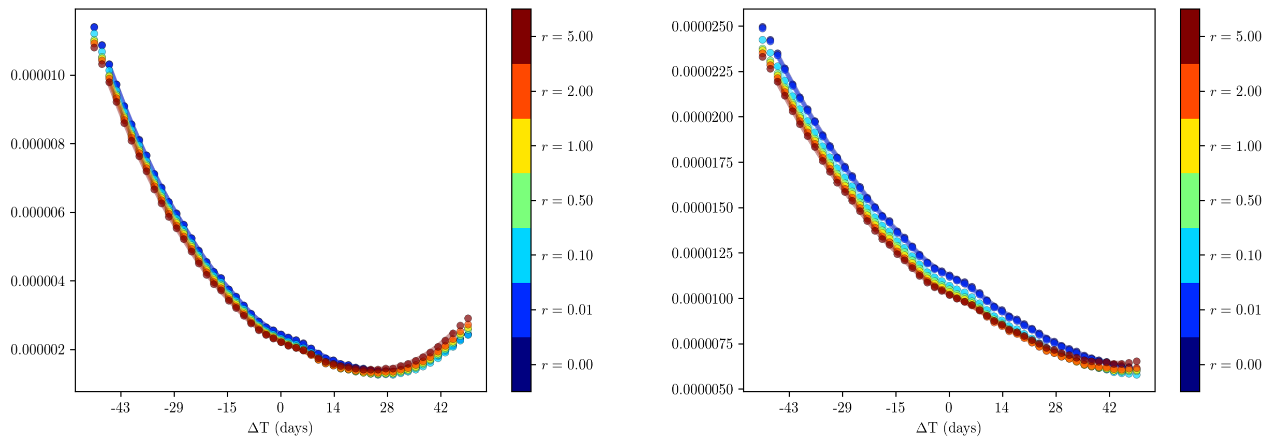


Figure 8: A zoomed in view of error for smaller positive values of $\Delta T$. left: $\alpha = 0.07$, right: $\alpha = 0.15$. Both: $N = 30$. Utilities are base scaled to a maximum of 15, and then multiplied by the factor $r$ (colorbar)

- **Plotting for Discrete Graphs** [`plotting.py`]**.** Three plotting functions built on top of `matplotlib`, which build LaTeXrendered error plots, temporal animations, and heatmaps with date formatting, color bars, and automatically selects a subset of the information to display if too much is requested at once. They directly take temporal graphs objects without requiring explicit extraction of the relevant information.

## 8 Limitations and Future Work

### 8.1 Better Utility Approximation

The primary issue with the current framework, as mentioned in sections 6.7 and 6.8, is the way we are calculating utilities. Fortunately, it should be quite easy to swap out code that does something else, in this framework. However, anything else is likely to be slower, as it cannot be done through an aggregated `pushshift.io` query. For instance, aggregating the karma submissions is straightforward to do, but difficult to do fast for arbitrary time winndows, since `pushshift.io` does not keep any up-to-date information. Another useful variable to consider while crafting a synthetic utility measure, is the number of unique posters in a sub.

### 8.2 Better Exploration of Neighborhood Distribution

There are many opportunities explore alternate discrete choice models more thouroughly. While we propose several, we stuck with just one for the majority of the simulations because it was much faster to simulate, and seemed to have similar or better performance. However, sampling neighborhoods is also a well-studied problem, and we can use something like `node2vec` [4] random walks is more likely to give us sampled neighborhoods which reveal graph structure, than our own custom, easy to integrate formulas.

### 8.3 More Principled Ways of Generalizing Parameters across Temporal Subgraphs

The way that I conducted the majority of this project was to simply select a few subgraphs that I had downloaded (they take a while to construct) and play around with visualizing things until I could understand what was going on, more-or-less. For parameters like the mixture coefficient $\alpha$, offset $\Delta T$, and the utility scaling $r$, I was able to automate grid searches over parameter spaces, resulting in satisfying plots like the ones shown. However, over variables like the number of node $N$ (which is confounded with node popularity due to the way I selected graphs), there was no good way to do this without worrying about the variables having confounding dependencies.

Future work along these lines might aim to improve the selection of graphs by expanding to less popular subs in different ways. Perhaps it even makes sense to use the neighborhood sampling techniques one might implement from `node2vec` here too.

## 8.4 Related Experiments

**— .1 Utility Prediction** Rather than predicting $Q$, we could also predict $U$, which would be easier for networks and SVMs to handle, as there are fewer outputs that need to be learned. This is interesting, as it corresponds to the question of how the utilities change in the first place

**— .2 Better Predictors: Neural or Structured** By engineering better ML baselines systems, ether by exposing better features (we definitely want some shared weights in the predictor; one possibility is exposing only the relevant rows and columns and training a single regression which we permute to get all the entries of the output matrix), or by throwing a lot more data and better network architecture at the problem, we can probably do a lot better at this task.

## 8.5 Individual Variation

In our formulation, we have assumed that there is a static utility function $U$ on the vertices of the graph, governing choice. For many applications, this one included, some people will prefer some places to others. In many ways, this is one of the distinguishing features that causes there to be places at all. The utility data we have gathered is not from an individual, but rather the aggregate "will of the people". However, it is well-known that aggregate wills are not particularly agentive (e.g., the Condorcet paradox), and in general, we get another mixture paramter, for a deeper mixed logit model, rather than altered utilities for our current one.

One interesting direction might be to model this more carefully, and try to discern the nature and number of archetypes needed to better approximate the dynamic utilities of the graphs.

## References

[1] Dario Amodei et al. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016).
[2] Philip Bramsen et al. "Inducing temporal graphs". In: *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2006, pp. 189–198.
[3] Elenna R Dugundji and Joan L Walker. "Discrete choice with social and spatial network interdependencies: an empirical example using mixed generalized extreme value models with field and panel effects". In: *Transportation Research Record* 1921.1 (2005), pp. 70–78.
[4] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2016, pp. 855–864.
[5] Till Grüne-Yanoff and Sven Ove Hansson. *Preference change: Approaches from philosophy, economics and psychology*. Vol. 42. Springer Science & Business Media, 2009.
[6] Vassilis Kostakos. "Temporal graphs". In: *Physica A: Statistical Mechanics and its Applications* 388.6 (2009), pp. 1007–1023.
[7] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. "Motifs in temporal networks". In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM. 2017, pp. 601–610.
[8] Thomas Reeves, Anil Damle, and Austin R. Benson. *Network interpolation*. 2019. eprint: `arXiv:1905.01253`.