Recommending Routes in the Context of Bicycling: Algorithms, Evaluation, and the Value of Personalization

Reid Priedhorsky,[*] David Pitchford,[†] Shilad Sen,[‡] Loren Terveen[†]

[*]IBM Watson Research Center
Cambridge, MA, USA
reid@reidster.net

[‡]Macalester College
St. Paul, MN, USA
ssen@macalester.edu

[†]University of Minnesota
Minneapolis, MN, USA
{pitch006,terve002}@umn.edu

ABSTRACT

Users have come to rely on automated route finding services for driving, public transit, walking, and bicycling. Current state of the art route finding algorithms typically rely on objective factors like time and distance; they do not consider subjective preferences that also influence route quality. This paper addresses that need. We introduce a new framework for evaluating edge rating prediction techniques in transportation networks and use it to explore ten families of prediction algorithms in Cyclopath, a geographic wiki that provides route finding services for bicyclists. Overall, we find that personalized algorithms predict more accurately than non-personalized ones, and we identify two algorithms with low error and excellent coverage, one of which is simple enough to be implemented in thin clients like web browsers. These results suggest that routing systems can generate better routes by collecting and analyzing users' subjective preferences.

Author Keywords

geographic recommenders, geowikis, recommender systems, route finding

ACM Classification Keywords

H.5.3 Group and Organization Interfaces: Collaborative computing, computer supported cooperative work

General Terms

Algorithms, Human Factors

INTRODUCTION

Modern websites frequently personalize users' online experiences. For example, Amazon and Netflix depend on recommender systems to help users find books and movies, Google can display a personalized selection of news stories [8], and social networking sites use personalized algorithms to help users make new connections [7]. This paper explores whether and how personalization can improve route finding, doing so in the context of Cyclopath, a route finding service for bicyclists in the Minneapolis/St. Paul metro area of Minnesota.

Rather than returning a route that is optimal according to some objective metric like time or distance, a personalized route finding system returns a route which is optimized to meet the personal preferences of the user who requested it. For example, one person might prefer slightly longer routes (in time or distance) if they are scenic, while another might prefer the opposite tradeoff – i.e., the quality of a route can be influenced by users' *subjective preferences*. This notion is supported by prior work in the automobile and wheelchair domains [4, 15, 21]. In addition, during the design of Cyclopath, users reported a desire for personalization [24, 25].

All but the most prolific users express preferences for only a tiny fraction of the transportation network. Because route finding algorithms such as A* require information about the entire network, systems that personalize routes must *infer* a user's preferences for the bulk of the edges in the network. This paper explores algorithms for making these inferences by studying the following questions:

1. Which preference prediction algorithms have both good accuracy and good coverage?
2. Do *personalized* route finding algorithms that account for subjective user preferences yield better routes than objective algorithms that do not?

Three key characteristics of route finding domains provide a novel challenge for personalization: (a) the structure of the rating datasets are different (items dramatically outnumber users rather than the other way around, and ratings are much sparser); (b) items have structured relationships and the output is a sequence of items (a route); and (c) 100% coverage is essential (no location pairs can be unrouteable due to lack of rating data).

This paper seeks to close this gap. We introduce a three-stage framework for evaluating personalization in the context of route finding and evaluate 10 families of algorithms using this framework.

We first survey related work and evidence suggesting the need for personalized routing. We then describe our evaluation framework, present the results of the algorithms we tested under that framework, and close with a summary and directions for future work.

RELATED WORK

Recommender systems

These systems use algorithms that detect patterns in users' past preferences to predict their future preferences. *Collaborative filtering* approaches include user-based [27] and item-

CSCW'12, February 11–15, 2012, Seattle, Washington, USA.

based [30] algorithms, with later refinements such as dimensionality reduction using singular value decomposition [11] and *ensemble recommenders* which combine the output of multiple algorithms [5]. While collaborative filtering leverages patterns among users' preferences for items, some recommender systems also use *content-based* algorithms [2] that leverage patterns in the attributes of items.

More recently, researchers have begun to explore geographically aware recommender systems. Some systems recommend items near a user's current or future location, such as restaurants [23, 34] or tourist destinations [1, 28]. Others assume that a user's location contains preference information. For example, Li et al. found that similarity of users' location histories predicts social tie strength [19]. Zheng et al.'s GeoLife2 social networking system does both, drawing on location to infer user similarity and recommending social activities between potential friends who are near each other [35]. Our work resembles these types of systems in that it predicts user satisfaction for edges in the transportation network, but we go beyond prior work to explore how these individual predictions affect their assembly into a complete route.

Finally, researchers have also studied recommending collections of items. Ziegler et al. showed that user satisfaction with a set of recommended items was enhanced when diversity of items within the set was considered, rather than simply minimizing the total prediction error [36]. Felfernig et al. explored recommending product configurations, where different components required or conflicted with other components [10]. Our work builds upon these ideas by addressing recommendation of large collection of items (the sequence of edges which forms a route) and the new types of structure relevant to route finding applications.

Route finding

Modern route finding systems are supported by a large scholarly literature. For example, geographers have explored the effects of transportation on human beings: the meaning of mobility and distance, economic and political influences, and where people choose to live and travel within their communities [29]. Computer scientists have studied the technical aspects of route finding, exploring issues like algorithm performance and scalability [9], uncertain costs [16], and multimodal routing (e.g., trips by both car and bus) [6].

Researchers have also begun to explore *personalized* routing. Bederson et al. motivate personalization by pointing out the need for "subjective human experience" in route finding, proposing a system for recording such experiences [4]. Personalized algorithms that accommodate these differences in routing preferences have been shown to outperform traditional routing algorithms. The OurWay system created by Holone et al. asked nine users in wheelchairs to rate route segments in order to receive personalized route recommendations, and these outperformed traditional algorithms [15]. McGinty and Smith used case-based reasoning to find a route by drawing on data from other users with similar route preferences [21].

Our work improves upon these personalized routing ideas in two ways: we draw upon a much larger dataset from a mature production route finding system, and we apply established techniques from the machine learning and recommender systems communities.

Route finding for cyclists

There are numerous websites that address various aspects of bicycle route finding. Gmaps Pedometer,[1] Bikely,[2] and others let users manually define and share routes. Wayfaring[3] enables collaborative editing of routes and places of interest. Open Cycle Map[4] is a cyclist-focused rendering of Open Street Map data. A few offer automatic route finding, such as YTV Journey Planner for Cycling[5] and Fietsrouteplanner,[6] and others offer automatic route finding based on Open Street Map user-contributed data, such as the Cambridge Cycling Campaign's "journey planner" for cyclists.[7] Biketastic[8] is a research system which uses a mobile phone application to share information about rides, collected using both built-in sensors (pavement roughness, noise level) and user contributions (photos, videos, and text descriptions) [26]. Finally, Google recently introduced bicycle route finding in Google Maps for many American cities [18]. This was a key advance for cyclists across the country, most of whom previously had no online bicycling map or automated route finding at all.

A variety of metrics for computing the bikeability of streets exist, but they are generally impractical because they require an unrealistically large number of attributes for each street; for example, Sorton's bicycle stress level [33] requires 6 to 14, and an analysis at Macalester College [20] identified 16. Cyclopath currently uses a metric developed by the Chicagoland Bicycle Federation [3], which requires 4 attributes; of these, we have moderately complete data for only 3 (speed limit, daily traffic volume, and shoulder width) and we must estimate the last one (lane width).

Many of these systems let users express preferences each time a route search is initiated. However, none support rich user profiles that capture preferences about specific streets and trails. We take advantage of such preference data in Cyclopath to explore personalized route finding algorithms, developing a systematic framework for evaluating route finding algorithms and using that framework to draw empirical conclusions about specific algorithms.

ROUTING AND PERSONALIZATION

Route finding in Cyclopath, at a high level, works as follows.[9] The streets and trails traveled by cyclists form a graph, where street intersections are nodes and the city blocks between them are edges; we call this graph the *transportation network*. Figure 1 shows a small portion of this network as it is displayed in the web application, along with the underlying graph representation.

[1] http://gmap-pedometer.com
[2] http://www.bikely.com
[3] http://wayfaring.com
[4] http://opencyclemap.org
[5] http://kevytliikenne.ytv.fi/?lang=en
[6] http://fietsersbond.nl/fietsrouteplanner/
[7] http://www.camcycle.org.uk/map/route/
[8] http://biketastic.com
[9] Technical details are available by consulting the Cyclopath source code: http://cyclopath.org/wiki/Tech:Source_Code.

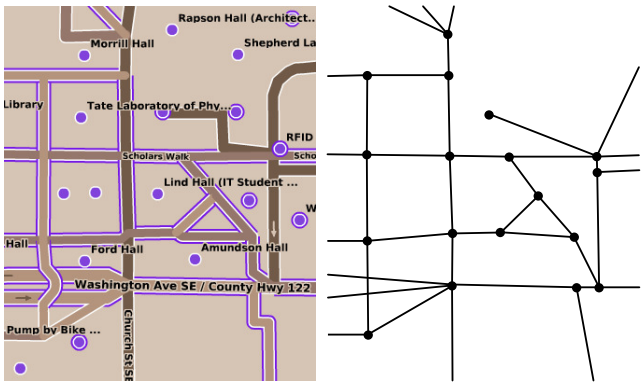

Figure 1. A portion of the Cyclopath web map covering the University of Minnesota campus (left) and its representation as a graph (right).

Cyclopath users can rate the subjective *bikeability* of the graph on an edge-by-edge basis, on a five-level scale from "impassable" through "excellent". We provide user interface support to make it easy to enter many ratings at once. For the purposes of analysis, we treat bikeability as a number ranging from 0 (impassable) to 4 (excellent).

When searching for a route, a user can also express preferences for that particular search, making both a tradeoff between higher bikeability and shorter distance as well as specifying bikeability penalties and bonuses for tags that edges might have (e.g., *bumpy* or *caution with merging*).

These notions are brought together as follows. Cyclopath uses A* search [14] to compute the minimum-cost route through the transportation network. In addition to small costs for traversing notes, the weight of an edge is a function of that edge's bikeability score and the route search preferences. The challenge is thus to accurately estimate bikeability for each edge that A* might evaluate during its search (as all but the most prolific raters have actually rated only a tiny fraction of the edges in the graph).

Evidence for personalization

Several informal analyses of the Cyclopath ratings database suggest that personalized route finding would be useful. As of April 2011, Cyclopath had 68,274 ratings by 481 users on 35,608 edges, with a mean rating of 2.78 and standard deviation of 1.13; in MovieLens, a movie recommendation site operated by our research group, these statistics are 3.51 and 1.06, respectively (as of April 2010). Also, of the 110,671 instances when two users rated the same edge, they gave the same rating 53% of the time, disagreed by one star 34% of the time, and disagreed by two or more stars 13% of the time. In other words, nearly half the time, users disagree on the appropriate rating for an edge, and about one eighth of the time, this disagreement is at least two stars (the difference between "fair" and "excellent" or "poor" and "good").

Another way to quantify disagreement is: to what degree would different users' ratings actually lead to different decisions by the graph search algorithm? We explore this by examining the preference order of edges rather than actual rating values; the intuition is that even if two users give different ratings, but graph search would make the same choices regardless, the two users' ratings are effectively the same.

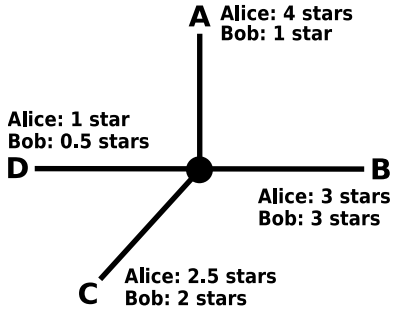

Figure 2. A hypothetical node in the transportation network. Bikeability ratings by two users are shown.

Figure 2 shows a hypothetical graph node with four adjacent edges, A through D, and bikeability ratings from two users, Alice and Bob. The predicted preference order of the four edges is ABCD according to Alice but BCAD according to Bob. For example, Edge B has the same rating (3 stars) from both users, but only for Alice is it the most preferred edge adjacent to this node.

This is relevant because graph search algorithms such as A* wander through a graph, choosing at each node the best edge to wander next. For example, if a graph search algorithm arrived at this node via Edge C, it would next choose Edge A if using Alice's bikeability ratings, but Edge B using Bob's ratings (assuming that other considerations such as edge length are the same).

Results of this analysis over Cyclopath ratings[10] are as follows. Given adjacent edges A and B and users U and W who have each rated both edges, in 87.3% of the 81,903 observations, U and W agree: either all four ratings were the same (84.7% of observations) or both users prefer A to B (2.6%). On the other hand, 12.7% of the time, they disagree; either U prefers A to B and W rates them equally (12.0%) or U prefers A to B but W prefers B to A (0.7%). At first glance, this result is cause for skepticism – users give a different preference order only one eighth of the time. However, the average computed route contains 76 edges, and the graph search to produce this average route evaluates thousands of edge pairs; together these results imply that each set of 23 edge pairs is 95% likely to contain a disagreement ($0.873^{23} = 0.044$). Thus, even this fairly low level of disagreement will result in graph searches that encounter many points of disagreement where the computed paths could diverge.

These three informal analyses suggest that route personalization is valuable. The remainder of this paper is a systematic evaluation of this hypothesis.

Challenges for personalization

As noted earlier, there are three core challenges in applying personalization techniques to route finding systems. First, the structure of the user/item rating matrix is less favorable. As noted in Table 1, when considering all edges (not just those that are rated), as is needed to address the first challenge, the Cyclopath dataset is 10-42 times more lopsided than active MovieLens or the Netflix Prize dataset [13], and in the opposite direction; for example, MovieLens has 7.6 users per

[10]This analysis uses ratings data as of April 2010.

Dataset		Users	Items	Aspect	Ratings	Sparsity
MovieLens		102,757	13,608	7.6 u/i	17,488,416	0.013
Netflix Prize		480,189	17,770	27 u/i	100,480,507	0.012
Cyclopath	rated edges	481	35,608	74 i/u	68,274	0.0040
	all edges		154,242	321 i/u		0.00092

Table 1. Descriptive statistics for MovieLens, Netflix Prize, and Cyclopath ratings datasets. We include only Cyclopath users who have made at least one rating, in order to permit comparison with other systems; for the same reason, we present the number of rated edges as well as the total number of edges in Cyclopath. The latter is of greater interest in our analysis due to the 100% coverage requirement.

item, but Cyclopath has 321 items per user. Furthermore, the Cyclopath dataset is 13 times sparser.

Second, both items and the output of the system have significant constraints. Items (edges) have highly structured relationships defined by the topology of the transportation network, and rather than outputting one or a few individual items, recommenders for route finding must produce a structured sequence of items – a route.

Finally, universal coverage is required. In contrast to systems that recommend products (e.g., 17% of movies in MovieLens are unrecommendable), graph search algorithms need edge weights for every edge in the graph in order to produce a path for arbitrary starting and ending points.[11]

EVALUATION FRAMEWORK

We propose the following basic structure for the design and evaluation of personalized routing systems. *Prediction algorithms* predict bikeability ratings for individual edges, which are then used by a graph search algorithm to compute the complete route. Our framework tests performance at three stages – individual edge rating predictions, node-level graph search decisions, and the final route – enabling analysis at each stage to see if differences between algorithms remain meaningful.

Stage 1: Predicting edge ratings

The goal of this stage is to assess the accuracy of edge rating predictions when compared against actual user ratings; it corresponds to the "predict" task in traditional recommender systems [31]. Here, items are the edges in the transportation network: given an edge and a user, predict the user's rating of that edge. The procedure is straightforward: (a) define and implement different prediction algorithms, (b) evaluate them using standard k-fold cross-validation, and (c) report coverage and error for each algorithm, using standard metrics like mean absolute error (MAE) and root mean squared error (RMSE).

Stage 2: Comparing node-level decisions

This stage considers whether edge weights created by different prediction algorithms would lead to different decisions by a graph search algorithm. Similarly to the discussion of user ratings above, the intuition is that even if Algorithm A has better accuracy than Algorithm B, but graph search would make the same decisions regardless, there is no effective difference between the two. It corresponds to the "recommend" task in traditional recommender systems [31], where a system tries to choose the best available item (e.g., "find me the movie I would most like"); the key difference is that in recommending routes, this decision is made thousands of times per high-level operation (find a route), rather than a few times (find a movie to watch).

We introduce a metric called *best edge agreement* (BEA) for this analysis: given two algorithms and a set of nodes, the BEA is the fraction of nodes where the two algorithms select (from the edges adjacent to the node) the same edge as best. This metric is hard to operationalize, however, because it requires both algorithms to predict *all* edges adjacent to a particular node in order to assess their BEA at that node; as we show below, many algorithms have poor coverage, so this requirement is often not met. Thus, we take a pairwise approach, introducing a variant metric, *pairwise best edge agreement* (PBEA): given two algorithms and a set of pairs of edges such that both edges (a) are adjacent to the same node and (b) have predictions from both algorithms, the PBEA is the percentage of pairs where both algorithms select the same edge as best.

PBEA can be used to estimate BEA. For example, suppose that Algorithm X's predictions for four edges adjacent to a node give preference order ABCD; in particular, Edge A has a higher predicted rating than B, C, and D. The probability that Algorithm Y chooses the same best edge (A) is the probability that its own rating prediction for Edge A is also higher than for the other three edges, i.e., Y must agree with X on the predicted preference order of three edge pairs: AB, AC, and AD. Assuming that edge predictions are independent,[12] the probability of the algorithms agreeing three times is the probability that they agree once raised to the third power: we estimate BEA on this four-edge node as PBEA cubed. Cyclopath nodes have on average 3.2 edges (excluding dead ends – nodes with degree 1 – which don't affect graph search decisions), so we estimate BEA $\approx$ PBEA$^{2.2}$ for this context.

Stage 3: Comparing paths

This stage evaluates how much *paths* produced with different prediction algorithms differ. The motivation is similar to

[11] To be pedantic, this is not strictly true. One need not predict *all* edge weights; rather, one must predict edge weights for all "important" edges – those which have a realistic chance of being considered for routes (not just included in the result). However, 100% coverage is achievable with good performance, as we elaborate below, and it is a simple way to meet this stricter criterion.

[12] This assumption is probably untrue – i.e., ratings of adjacent edges are probably correlated – so it's likely that this technique underestimates BEA. However, if we assume that the ratings of adjacent edges are positively (and not negatively) correlated, this underestimation is bounded, as the true BEA must then lie between PBEA and the estimated BEA.

Stage 2: even if different algorithms produce different low-level graph search decisions, if the resulting paths are the same or essentially the same, the algorithms are effectively equivalent. However, it differs from Stage 2 in two important ways. First, it requires a corpus of route searches to recompute. Second, only algorithms with 100% coverage can be tested, because now rating predictor and graph search algorithms are combined. As our results below show, often an ensemble predictor is needed, because there is a tradeoff between accuracy and coverage.

We next turn to our application of this framework, using data from Cyclopath to test several bikeability prediction algorithms and explore the value of personalized route finding.

ALGORITHMS TESTED

We tried 10 families of algorithms, which we divide into four classes: objective algorithms which do not consider user ratings, algorithms which average user ratings in simple ways, collaborative filtering algorithms, and machine learning algorithms.

"Objective" algorithms

These two algorithms, which make use of only edge attributes rather than ratings (i.e., objective rather than subjective data), represent the general approach of current state-of-the-art route finding.

We first consider *Objective-CBF*, a bikeability metric developed by the Chicagoland Bicycle Federation which takes into account speed limit, daily traffic volume, lane width, and shoulder width [3]. While simpler than other metrics created by the bicycling community, it is the only one we could find whose input attributes had sufficient coverage in Cyclopath, which is based on the best available data (and data availability for our area is typical for American cities). Our implementation has minor modifications to make it usable in Cyclopath, motivated by data availability, the first author's expertise as a bicyclist, and feedback from Cyclopath users (again, the technical details are available by consulting the Cyclopath source code).

We created *Objective-Simple* to achieve 100% coverage. This ad-hoc metric predicts a base rating depending solely on the edges' type (e.g., major road, local road, bike path) and then applies a few of the same adjustments as in *Objective-CBF*.

Simple averaging algorithms

We next turn to algorithms that average ratings in a simple ways. First, we tested two very simple baseline methods. *Global-Mean* predicts the mean of all ratings in the system, while *User-Mean* predicts the mean rating by the user for whom the prediction is being made.

Second, we tried a family of algorithms which predict the mean of other users' ratings on a particular edge; for example, if an edge is rated 2 stars by User A and 3 by User B, the prediction for other users would be 2.5 stars. We varied the threshold of ratings required for prediction, only considering a prediction valid if there are more than n ratings on an edge. We tried $n = 1, 2, 3, 5, 10$, and we refer to these algorithms as *Edge-Global-Mean.*n.

Finally, we tried algorithms which group edges into clusters and then predict using average edge ratings within each cluster. We took advantage of existing structure in the data, creating clusters based on the 9 edge types as well as the presence or absence of a bike lane. For example, major roads with a bike lane were one cluster, major roads without a bike lane were another, bike paths were a third (as bike lanes on bike paths do not make sense), and so on.

This yielded two algorithms. *Type-Global-Mean* predicts the mean of all users' ratings on edges of the same type, while *Type-User-Mean* predicts the mean of the searching user's ratings on edges of the same type.

Collaborative filtering

Collaborative filtering (CF) is one of the most powerful recommendation techniques in other contexts [5]. We expected CF to give accurate ratings but have problems with coverage, given our extremely sparse dataset. We implemented collaborative filtering according to [31] and [32], with several variations: user-based vs. item-based, different distance metrics (Pearson correlation, Euclidean distance, cosine similarity), and adjusting or not adjusting ratings by the user average. We used a similarity threshold of just one co-rating in an effort for more coverage (except for the Pearson distance metric, which is not defined for one co-rating).

Machine-learning algorithms

Finally, we tried machine learning techniques similar to linear regression. The regressions take as input a training set of $\langle r(u,e), f_1(u,e), f_2(u,e), ...\rangle$ examples, where $r(u,e)$ is a rating by user u for edge e, and each $f_i(u,e)$ is a numeric or categorical feature giving information about the user or edge associated with a rating (e.g., speed limit). They then analyze the relationships between features and ratings and generate predictions for unknown ratings based on these relationships. Machine learning regression algorithms are well-suited to our task because they are fast and regularized to avoid over-fitting. We used the Vowpal Wabbit conjugate gradient descent software [17] because it efficiently handles large datasets with many features.

We tried two algorithms based on these ideas. First, *ML-User* creates a separate regression model for each user. We included seven features in total: mean rating by all users on that edge, mean rating by the user in question on all edges, and 5 edge attributes (speed limit, lane width, log-transformed average daily traffic, tags, and lane count).

Because this algorithm builds an independent model for each user, it cannot analyze rating patterns across different users. We addressed this with *ML-Global*, which builds one shared regression model for all users, with 6 additional features (for 13 total) to support personalization in the shared model. First, we added user ID as an indicator variable. Second, we added a user profile vector that captured how users felt about the five edge attributes; a value in the profile was either the Pearson correlation between the attribute and the user's ratings (for numeric attributes) or the fraction of of the user's rated blocks with that feature (for categorical features). Finally, as personalized algorithms must capture how a specific user (or type of user) feels about a specific edge, we added in-

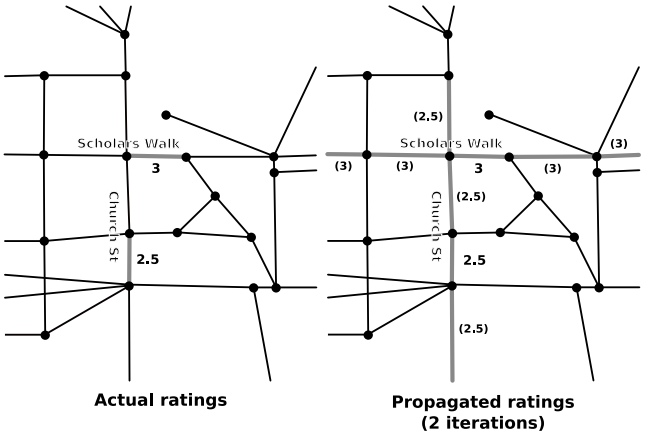

Figure 3. Ratings propagation. This illustration shows our (largely failed) technique to address the sparsity of ratings.

teractions between the edge attributes and both user ID and the profile vector. (Vowpal Wabbit optimizes interactions directly, eliminating the need to add them as separate features.)

For both of these algorithms, we normalized all feature and rating values to the interval [0,1], as recommended by the Vowpal Wabbit documentation.

STAGE 1: PREDICTING RATINGS

We begin by analyzing the coverage and accuracy of these rating prediction algorithms, to find which are promising in the context of route finding.

Method

We implemented the algorithms in Python and evaluated their accuracy with 10-fold cross-validation, using Cyclopath ratings as of April 2011. This dataset contains 68,274 ratings by 481 users on 35,608 of 154,242 edges.[13]

We report accuracy using MAE and RMSE. Coverage is computed statistically: we took a random sample of 5,000 user,edge pairs and tried to predict each, counting how many times we succeeded. We use the full dataset for this estimate, i.e., without cross-validation, which is not needed because we are not using prediction values.

We report coverage for users who have made at least one rating. Cyclopath also computes routes for users who have made no ratings, but we report as we do for two reasons. First, it allows for comparison with other recommender systems. Second, it is impossible to compute coverage including users who have rated no edges, because this set contains an unknown and unbounded number of anonymous users. However, some algorithms can make predictions for a user without any ratings from that user: in these cases, coverage for both classes of users is equal. We point out such algorithms in our discussion below. In particular, when we claim an algorithm has *universal coverage*, that means it can predict a rating for any user,edge pair even if that user made no ratings.

[13]We removed 1,400 ratings on expressways made before the system was complete, when rating was the only way to prevent certain expressways from being offered in routes, as working around this missing feature was unrepresentative of typical rating behavior.

We were concerned about the sparsity of our dataset (recall that only 0.092% of user,edge pairs actually had a rating, 13 times sparser than MovieLens or Netflix), and we tried to address this by inferring some ratings. Our approach is similar to Good et al., who inferred rating profiles based on movie attributes such as genre and keywords [12]. The intuition is that unrated edges "near" rated edges are likely to have similar ratings (i.e., edges show *spatial autocorrelation*). Figure 3 gives an example; two ratings are shown, one on Scholars Walk and one on Church St. We can reasonably infer that topologically adjacent edges of the same street have the same rating, as do edges adjacent to those, and so on. The example shows two iterations of this propagation – we have inferred 7 ratings, increasing the number of rated edges from 2 to 9.

We did this for 8 iterations, creating 58,066 inferred ratings for a total of 126,340; 49,277 edges then had at least one rating, up from 35,608. This reduced sparsity from 0.00092 to 0.0017, still 7 times more sparse than MovieLens or Netflix and much less improvement than we expected. We speculate that this is for two reasons. First, some ratings were near the "end" of a street, so it was not possible to propagate for the full 8 iterations. Second, rated edges are not uniformly distributed. For example, if a user has rated 20 edges, they are frequently in an adjacent sequence already (e.g., Hennepin Ave. between 20th and 40th St.), so propagation only expands the first and last ratings in the sequence – yielding closer to 20 inferred ratings than 400. Unsurprisingly in light of this, the technique turned out to be not very helpful: coverage was improved only modestly, and accuracy was damaged (perhaps due to the obvious risk that inferred ratings are necessarily guesses), for all algorithms except the *ML-* pair. Details are given below as algorithm results are summarized.

Results

Table 2 lists our full results, divided into sections according to algorithm class. Below, we discuss key results and interpretations for each algorithm.

Objective algorithms. *Objective-CBF* has coverage of 88% and MAE of 0.709, while *Objective-Simple* has universal coverage (its motivation) and MAE 0.770. While both of these algorithms have excellent coverage, their accuracy is uncompetitive compared with other algorithms.

Simple averaging algorithms. *Global-Mean* has universal coverage and MAE of 0.882, while *User-Mean* has 99.8% coverage (the gap from universal arising from a few users having rated only edges which were later deleted from the system) and MAE of 0.802.

Accuracy of *Edge-Global-Mean.*n is surprisingly good; even copying a single other user's rating (*Edge-Global-Mean.1*) is slightly more accurate (MAE of 0.684) than a moderately sophisticated objective algorithm (*Objective-CBF* with MAE 0.709). *Edge-Global-Mean.2* and higher are competitive with the more complex algorithms below, with MAE from 0.633 to 0.440. However, coverage is poor. *Edge-Global-Mean.1* can be stretched to 31% using ratings propagation with essentially no penalty in accuracy (MAE declines by 0.004), but other algorithms are superior in both coverage and accuracy, and a higher threshold quickly leads to very

Algorithm	Without propagation MAE	RMSE	Cov.	With propagation MAE	RMSE	Cov.
Objective algorithms						
Objective-Simple	0.770	1.035	1.000 ⋆		no effect	
Objective-CBF	0.709	1.031	0.882 ⋆		no effect	
Simple averaging algorithms						
Global-Mean	0.882	1.122	1.000 ⋆	0.903	1.123	1.000 ⋆
User-Mean	0.802	1.015	0.998	0.818	1.021	0.998
Edge-Global-Mean.1	0.684	0.975	0.212 ⋆	0.688	0.963	0.307 ⋆
Edge-Global-Mean.2	0.633	0.871	0.078 ⋆	0.650	0.879	0.153 ⋆
Edge-Global-Mean.3	0.601	0.830	0.041 ⋆	0.627	0.846	0.096 ⋆
Edge-Global-Mean.5	0.556	0.778	0.015 ⋆	0.604	0.821	0.043 ⋆
Edge-Global-Mean.10	0.440	0.653	0.005 ⋆	0.563	0.776	0.014 ⋆
Type-Global-Mean	0.674	0.887	1.000 ⋆	0.704	0.908	1.000 ⋆
Type-User-Mean	0.482	0.698	0.609	0.520	0.736	0.637
Collaborative filtering						
CF-User.pearson	0.614	0.895	0.029	0.636	0.907	0.059
CF-User.euclid,unadj	0.553	0.845	0.076	0.502	0.777	0.116
CF-User.euclid,adj	0.621	0.911	0.075	0.624	0.911	0.119
CF-User.cosine,unadj	0.675	0.961	0.080	0.677	0.947	0.140
CF-User.cosine,adj	0.610	0.885	0.049	0.623	0.886	0.087
CF-Item.pearson	0.410	0.637	0.007	0.461	0.664	0.021
CF-Item.euclid,unadj	0.582	0.797	0.082	0.614	0.807	0.145
CF-Item.euclid,adj	0.582	0.797	0.082	0.614	0.807	0.145
CF-Item.cosine,unadj	0.638	0.862	0.072	0.674	0.881	0.130
CF-Item.cosine,adj	0.566	0.785	0.067	0.590	0.790	0.118
Machine learning algorithms						
ML-User	0.558	0.799	0.650	0.514	0.723	0.664
ML-Global	0.495	0.719	1.000 ⋆	0.477	0.677	1.000 ⋆

Table 2. Results of predicting edge ratings. Coverage includes only users who have made at least one rating. Algorithms which can make predictions for a user without any ratings from that user are marked with ⋆. For these algorithms, coverage for the (unbounded) set of no-rating users is equal to what is listed; for the others, coverage is undefined for no-rating users.

poor coverage (*Edge-Global-Mean.2* has just 15% coverage even with ratings propagation). Furthermore, *Edge-Global-Mean.1* is vulnerable to trivial attacks – for most of the edges in the system, one user can destroy the ability of the system to predict ratings for that edge by entering a bogus rating (which then becomes the predicted rating for everyone).

Finally, both *Type-* algorithms performed very well. *Type-User-Mean* has excellent accuracy (MAE 0.482) and reasonable coverage (61%), while *Type-Global-Mean* achieves universal coverage and has reasonable accuracy (MAE 0.674). Additionally, they have several further advantages. They are amenable to an easy cold start process – simply rate one edge in each cluster – and users might not even need to rate any actual items at all (e.g., "what rating do you typically give to bike paths?"). They are difficult to attack because users' ratings do not influence other users in *Type-User-Mean* and have weak influence in *Type-Global-Mean*. Finally, the models built by these algorithms are very simple and can be easily communicated to lightweight client software like web applications; in Cyclopath, this would be useful as ratings could be predicted even for newly created edges that have not yet been saved, without a round-trip to the server.

Collaborative filtering. Our hypotheses for CF algorithms we tried were confirmed, more dramatically than we expected – these algorithms are not useful in this context. One variant (*CF-Item.pearson* without ratings propagation) is the most accurate of anything we tried (MAE of 0.410), but coverage is extremely poor, just 0.7%. Other CF variants have better, though still poor, coverage (at most 15%), but other algorithms have much better coverage and better accuracy.

Machine Learning algorithms. These algorithms also both performed very well. *ML-User* had coverage of 65% and MAE 0.558, improving to 66% and 0.514 under ratings propagation. However, *ML-Global* was superior on both dimensions, with universal coverage and MAE of 0.495; this algorithm too improved under ratings propagation, though modestly, decreasing MAE by 0.02. (Because this improvement was small, our further analyses consider the non-propagated variant for simplicity.)

Summary. Figure 4 summarizes the performance of the algorithms we tried. Three of the four groups (objective, simple averaging, and machine learning) produced algorithms with sufficient coverage and accuracy to be of plausible use. For further evaluation, we analyze the best algorithm from each:

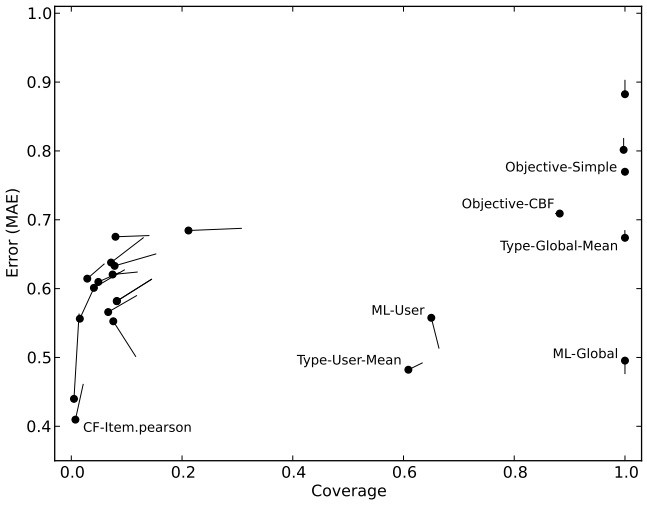

Figure 4. Scatter plot of prediction algorithm coverage vs. error. Algorithms lower and more to the right are better. Dots are results without ratings propagation, while the flags show the effect of propagation. Key algorithms are labeled.

	Objective-CBF	*Type-User-Mean*	*ML-Global*
Objective-CBF		0.75	0.44
Type-User-Mean	0.75		0.48
ML-Global	0.44	0.48	

Table 3. Pairwise edge order agreement (PBEA) between "plausible" rating prediction algorithms.

Objective-CBF, *Type-User-Mean*, and *ML-Global*. We next explore the question of whether these accuracy differences are meaningful at Stage 2.

STAGE 2: COMPARING NODE-LEVEL DECISIONS

Prediction accuracy is irrelevant if different algorithms do not lead to different graph search decisions. This section describes our evaluation of the agreement of the three plausible algorithms in this context, using the metrics and ideas explained above.

Stage 1 produced a set of user,edge,value predictions for each fold in the 10-fold cross-validation. For every pair of algorithms in each fold, we counted the number of pairs of adjacent edges where the two algorithms gave the same predicted rating ordering and the number where the orderings were different. This yielded the pairwise best edge agreement (PBEA) for the two algorithms.

Our results are summarized in Table 3. PBEA is at most 75% (an estimated BEA of 53%, suggesting that graph search would make the same low-level decisions at least 53% of the time), itself a large difference. Further, *ML-Global* has an even larger difference with both other algorithms (PBEA at most 48%), despite its modest MAE difference with *Type-User-Mean* (0.495 vs. 0.482). We speculate that this is due in part to the granularity of the predictions made by the different algorithms. For example, *Type-User-Mean* can make only 16 distinct predictions for any given user, because it returns the mean rating of one of 16 clusters; on the other hand, *ML-Global* yields continuously varying predictions in the appropriate range. The result is that *Type-User-Mean* frequently makes exactly the same prediction for a given pair of edges, while *ML-Global* rarely does, lowering the likelihood that the two algorithms will agree. In other words, the differences evident in Stage 1 do not collapse, and in fact other factors may dominate the agreement level.

This relatively low agreement between the algorithms at the node level suggests that optimal routes from different algorithms may diverge at many nodes. We next explore the degree to which this is true.

STAGE 3: COMPARING PATHS

In this stage, we examine whether differences in the desirability of edges when examining nodes in isolation translate into meaningful differences between entire routes.

Our methodology allows us to report differences between predictors in this stage, but not their quality directly, for two reasons. First, our ratings data is extremely sparse – the tiny number of edge ratings a user is likely to have made on a given route may not reflect that route's overall quality. Second, based on our experience working with cyclists, we believe that users evaluate routes holistically rather than as a collection of edges.

However, we believe that our findings in this stage do allow us to form plausible hypotheses about quality. Stage 1 let us establish low-level quality differences: for any given edge, certain predictors yield better predictions than others. Stage 3 (via Stage 2) then lets us test whether a better predictor yields different routes. Thus, it seems plausible that if a predictor is *better* at Stage 1 and *different* at all stages, it is better overall.

In this stage, we continue discussion of the three "plausible" prediction algorithms: *Objective-CBF*, *Type-User-Mean*, and *ML-Global*. However, recall that in order to be used in graph search, an algorithm must have 100% coverage. *Objective-CBF* and *Type-User-Mean* do not have this property and thus must be augmented. Accordingly, we tested the following three prediction algorithms:

- *Type-Ensemble*, an ensemble predictor [5] which first attempts *Type-User-Mean*, and if that does not yield a prediction, *Type-Global-Mean*. Recall that *Type-User-Mean* has 61% coverage; i.e., on average, 61% of prediction attempts will use *Type-User-Mean* and the remainder *Type-Global-Mean*.
- *Objective-Ensemble* is similarly a fall-through ensemble predictor which attempts *Objective-CBF* (88% coverage) and then *Objective-Simple*.
- *ML-Global* in isolation.

We randomly selected 5,000 of the 57,116 route searches made by Cyclopath users since the site went live in May 2008. For each of these, we recomputed the search (i.e., we reissued the same source, destination, and route preferences as the user who originally issued the search) using ratings

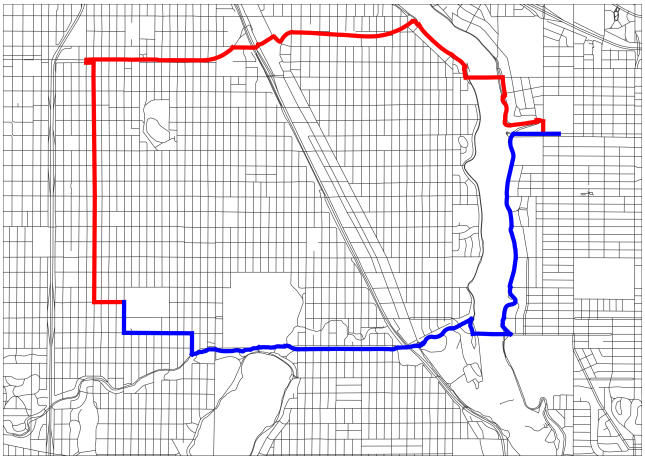

Figure 5. An example route search from Steele Flats (bottom left) to St. Thomas University (top right). The *Objective-Ensemble* predictor yielded the red upper route, while *Type-Ensemble* yielded the (almost completely different) blue lower route.

	Objective-Ensemble	*Type-Ensemble*	*ML-Global*
Objective-Ensemble		0.62	0.38
Type-Ensemble	0.62		0.44
ML-Global	0.38	0.44	

Table 4. Mean fraction of edges shared between route searches using each pair of ensemble prediction algorithms.

predicted by each of the three algorithms. We then compared the routes produced by each pair of predictors on each route. For example, Figure 5 shows an example route search which yielded two very different routes using different predictors.

Our results are summarized in Table 4. For each pair, we report the mean fraction of edges shared by the two predictors over the set of routes; for example, if predictor A produces a route of 12 edges and predictor B a route of 10 edges, and 5 edges are found in both routes, then the fraction of edges shared is $(5 \times 2)/(10 + 12) \approx 0.45$. (We also computed the mean fraction of distance shared, but it differed by no more than 2% from the mean fraction of edges shared, so we omit it for brevity.)

Different prediction algorithms do indeed produce different routes. While *Objective-Ensemble* and *Type-Ensemble* agree most often, they only share 62% of edges. This agreement indicates that attribute-based methods, such as *Objective-CBF*, do capture typical user preferences to some degree. Overlap between *ML-Global* and the other algorithms are quite low (38-44%). The low agreement with *Type-Ensemble* (44%) is particularly surprising, as both algorithms are personalized and draw upon edge ratings. This divergence suggests that the two algorithms achieve their Stage 1 accuracy through independent signals of user preference.

CONCLUSIONS

These results represent a step forward in two distinct ways. First, we have shown that personalized route finding algorithms that use edge ratings can be valuable in a production system. This is of additional interest because in many contexts, particularly open-content ones, qualitative data such as subjective ratings are easier to obtain than the quantitative data needed for objective rating predictors.

Second, our new framework and our algorithm evaluations are an advance for personalization research. Specifically, we identified two edge rating prediction algorithms that work excellently, one of which (*Type-Ensemble*) leverages a tiny model and whose algorithm to make predictions given the model is trivially simple. This makes it easy to pass the model to a thin client such as a browser and make predictions for edges of which the server has no knowledge.

There is much future work. First, the meaning for users of the differences identified in Stage 3 should be explored with a user study. Second, a deeper understanding of the relationship between Stages 2 and 3 would be useful; such an understanding may make it possible to eliminate Stage 3 and its more complex prerequisites. Third, other algorithms to predict edge ratings should be tested using our analysis framework. Fourth, rating predictions based on sensing could be fruitful: for example, cyclists could carry accelerometers to measure bumpiness, power sensors to measure effort, and/or physiological sensors to measure emotional stress.

Furthermore, route finding presents a number of additional complications. For example, we suspect that the subjective cost of a path is a nonlinear function of the (also subjective) weights of its edges. We already exclude some routes if one or more edge ratings fail to meet a minimum threshold. Another possibility is that each additional unit of poorly-rated edge distance has exponential additional cost; this notion is similar to the finding of Ziegler et al. that the desirability of a set of books differed from the average desirability of the individual books in the set [36]. Alternately, we might want to compute the path with with the smallest distance on poorly-rated edges or the highest worst rating ("least misery" [22]).

Finally, users have asked us numerous times for loop routes; e.g., "I am at Point A and I want a one-hour loop that I would enjoy". In general, we believe that there is much richness in route finding, particularly bicycle and other human-powered route finding, which we have not yet explored. Future research should do so, and new graph search algorithms may be required.

ACKNOWLEDGEMENTS

We thank the Cyclopath team at GroupLens Research and the Cyclopath user community. Ryan Kerwin did the analysis work in "Evidence for Personalization". This work was supported in part by the National Science Foundation, grants IIS 05-34692, IIS-0808692, IIS-0936043, and IIS-0964697.

REFERENCES

1. Liliana Ardissono et al. Intrigue: Personalized recommendation of tourist attractions for desktop and hand held devices. *Applied Artificial Intelligence*, 17(8):687–714, 2003.

2. Marko Balbanovic and Yoav Shoham. Combining content-based and collaborative recommendation. *CACM*, 40:66–72, 1997.
3. Ed Barsotti and Gin Kilgore. The road network is the bicycle network: Bicycle suitability measures for roadways and sidepaths. In *Proc. Transport Chicago*, 2001. http://bikelib.org/roads/roadnet.htm.
4. Benjamin B. Bederson et al. Enhancing in-car navigation systems with personal experience. In *Proc. Transportation Research Board*, pages 1–11, 2008.
5. Robert M. Bell and Yehuda Koren. Lessons from the Netflix Prize challenge. *SIGKDD Explorations*, 9(2):75–79, 2007.
6. Joel Booth et al. A data model for trip planning in multimodal transportation systems. In *Proc. Extending Database Technology*, pages 994–1005, 2009.
7. Jilin Chen et al. Make new friends, but keep the old: Recommending people on social networking sites. In *Proc. CHI*, pages 201–210, 2009.
8. Abhinandan S. Das et al. Google News personalization: Scalable online collaborative filtering. In *Proc. WWW*, pages 271–280, 2007.
9. Daniel Delling et al. Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks*, pages 117–139. Springer-Verlag, 2009.
10. Alexander Felfernig et al. Personalized user interfaces for product configuration. In *Proc. IUI*, 2010.
11. Simon Funk. Netflix update: Try this at home, 2006. http://sifter.org/simon/journal/20061211.html.
12. Nathan Good et al. Combining collaborative filtering with personal agents for better recommendations. In *Proc. AAAI*, 1999.
13. Ilya Grigorik. Dissecting the Netflix dataset, Oct 2006. http://www.igvita.com/2006/10/29/dissecting-the-netflix-dataset/.
14. Peter E. Hart et al. A formal basis for the heuristic determination of minimum cost paths. *TOSSC*, 4(2):100–107, 1968.
15. Harald Holone et al. Aspects of personal navigation with collaborative user feedback. In *Proc. NordiCHI*, pages 182–191, 2008.
16. William H.K. Lam and G. Xu. A traffic flow simulator for network reliability assessment. *Advanced Transportation*, 33(2):159–182, 1999.
17. John Langford et al. Sparse online learning via truncated gradient. *Machine Learning Research*, 10:777–801, 2009.
18. John Leen. It's time to bike, 2010. http://google-latlong.blogspot.com/2010/03/its-time-to-bike.html.
19. Quannan Li et al. Mining user similarity based on location history. In *Proc. ACMGIS*, pages 1–10, 2008.
20. Macalester College. Collecting bikeways data in Minnesota, 2006. http://www.macalester.edu/geography/civic/BikewaysProject_Geog364_Spring2006.pdf.
21. Lorraine McGinty and Barry Smyth. Shared experiences in personalized route planning. In *Proc. FLAIRS*, pages 111–115, 2002.
22. Mark O'Connor et al. Polylens: A recommender system for groups of users. In *Proc. ECSCW*, pages 199–218, 2001.
23. Moon-Hee Park et al. Location-based recommendation system using Bayesian user's preference model in mobile devices. In *Proc. Ubiquitous Intelligence and Computing*, pages 1130–1139, 2007.
24. Reid Priedhorsky et al. How a personalized geowiki can help bicyclists share information more effectively. In *Proc. WikiSym*, pages 93–98, 2007.
25. Reid Priedhorsky and Loren Terveen. The computational geowiki: What, why, and how. In *Proc. CSCW*, 2008.
26. Sasank Reddy et al. Biketastic: Sensing and mapping for better biking. In *Proc. CHI*, pages 1817–1820, 2010.
27. Paul Resnick et al. GroupLens: An open architecture for collaborative filtering of netnews. In *Proc. CSCW*, 1994.
28. Francesco Ricci. Travel recommender systems. *IEEE Intelligent Systems*, 17(6):55–57, 2002.
29. Jean-Paul Rodrigue et al. *The Geography of Transport Systems*. Routledge, 2006.
30. Badrul Sarwar. Item-based collaborative filtering reccomendation algorithms. In *Proc. WWW*, pages 285–295, 2001.
31. J. Ben Schafer et al. Collaborative filtering recommender systems. In *The Adaptive Web*, chapter 9, pages 291–324. Springer-Verlag, 2007.
32. Toby Segaran. *Programming Collective Intelligence*. O'Reilly, 2007.
33. Alex Sorton and Thomas Walsh. Urban and suburban bicycle compatibility street evaluation using bicycle stress level. In *Proc. Transportation Research Board*, 1994.
34. Fan Yang and Zhi-Mei Wang. A mobile location-based information recommendation system based on GPS and WEB2.0 services. *WSEAS Transactions on Computers*, 8(4):725–734, 2009.
35. Yu Zheng et al. GeoLife 2.0: A location-based social networking service. In *Proc. Mobile Data Management*, pages 357–358, 2009. Demo.
36. Cai-Nicolas Ziegler et al. Improving recommendation lists through topic diversification. In *Proc. WWW*, pages 22–32, 2005.