Using Machine Learning to Optimize Public Transit Scheduling and Efficiency

Ningsong Shen

Department of Computer Science, Western University

Scholar's Electives 2200E: Supervised Individual Research I

Professor Mike Katchabaw

April 12, 2021

Table of Contents

Abstract

An explorational study was conducted to evaluate a range of machine learning techniques in the context of bus arrival times and transit system scheduling. The public transit system in London, Ontario was used as a case study, taking advantage of the open data portal. Three models: regression, decision trees, and neural networks all had predictive power, albeit each with its advantages and drawbacks. A continuous information incorporation system was then developed successfully to improve the models, allowing for the potential to include additional data sources in the future.

Introduction and Background

To know precisely when the next bus is coming (or whether it has already left) greatly improves the passenger experience and reduces perceived waiting time. (Fan, Guthrie, & Levinson, 2016). In recent years, many approaches have been taken to predict the arrival times of buses, including single/multi-line analysis, historical data, statistical models, neural networks, and hybrid predictors (Altinkaya & Zontul, 2013). Many commercial solutions have also conducted work into predicting the next bus arrival time, including *Google Maps* and *Transit App*. Finally, other trackers involving vehicle location and passenger counters have been used to improve prediction and schedules. (Cathey & Dailey, 2003).

These advances provide great benefits to transit users and improve the perception of the entire transit system. This 'delay-side' prediction has been quite accurate, and useful when implemented worldwide, especially given the advanced technologies that can be tacked onto a vehicle and the mobile devices found in user's pockets. A virtuous cycle ensues, encouraging more transit users and thus demanding that transit providers deliver a higher level of service.

However, knowing how long a transit vehicle will be delayed is not merely good enough for transit system operators. Unlike a transit user, these operators are responsible for the many moving parts in

the overall system and will demand more information than a simple length of the delay. Besides, there may be many delays throughout the system that have ripple effects, and this information needs to be provided to transit system operators. Other operational reasons include keeping consistent personnel shifts, maintaining connections, and reducing bunching on high-frequency routes. These provide great rationale to know that transit vehicles will adhere to a schedule.

Existing Methods

Currently, many public transit systems worldwide develop schedules based on simple route timings and manual adjustments. Conventional techniques of taking averages and reusing common recurrences to determine timings are common. But the real world is quite messy, and the variance involved with timing a route often renders these 'average' schedules ineffective (Chien, Ding, & Wei, 2002).

Many factors influence arrival times, including traffic events and passenger load, many of which are data that transit agencies track. COVID-19 has also introduced rapid changes in travel patterns as another challenge to be overcome. It would be useful to have a frequently adjusting schedule based on a constant stream of information.

Given recent developments in predictive algorithms (Yu, Wang, & Shan, Prediction of Bus Travel Time Using Random Forests Based on Near Neighbors, 2018), technology can also be a large aid for such a system. As mentioned previously, some work has been done to predict arrival times using machine learning algorithms with historical schedules, passenger counting and vehicle location data. However, there has been less focus on applying a continuous stream of new information to schedule optimization. This type of system will make frequent adjustments to match the scheduling that is possible in the real world. The incorporation of real-time, relevant information combined with state-of-the-art predictive algorithms would make an effective scheduling system to improve the passenger experience.

Literature Review

The problem of predicting bus arrival times is an old one, and as such, there has been a body of research conducted by researchers around the world. Most have focused on 'delay-side' prediction and estimation, including proprietary and published work. Several diverse and interesting techniques have come to be quite popular.

To start, several data sources are used to support the various techniques. Automatic vehicle location data (often GPS data) was used frequently, given the ubiquity of such devices (Cathey & Dailey, 2003). Another source used was automatic passenger counters, which were used to help estimations of bus location and dwell times (Chen, Liu, & Xia, 2004). Others combined the two location and passenger data sources (Shalaby & Farhan, 2004). Researchers have even considered the differing implications of rural transit compared to the more conventional urban transit system (Lin & Zeng, 1999). Finally, unconventional methods have also been approached including smartphone data (Zhou, Zheng, & Li, 2014) and simulation models (Chien, Ding, & Wei, 2002).

Using those data sources, an equally wide variety of techniques were applied to the data. Kalman Filters were quite popular, having been used successfully on automatic vehicle location data (Cathey & Dailey, 2003), passenger counts (Shalaby & Farhan, 2004), and simulation models (Chien, Ding, & Wei, 2002). State-of-the-art machine learning techniques have also been applied, including artificial neural networks and support vector machines (Yu, Lam, & Tam, Bus arrival time prediction at bus stop with multiple routes, 2011). Deep learning with convolutional neural networks was surveyed (Petersen, Rodrigues, & Pereira, 2019), in addition to link-based models (Yu, Wang, & Shan, Prediction of Bus Travel Time Using Random Forests Based on Near Neighbors, 2018). Finally, many papers used random forests and multivariate regressions to compare them with other techniques.

These papers have covered a diverse range of topics that help in the prediction of arrival times. However, some notable missing elements were common across them all. First, is the lack of analysis over a large amount of data. This is unsurprising for earlier papers that did not have access or ability to process that data, but it was surprising to see that some newer papers did not take full advantage of this power. Also, many researchers did not look at data from an entire system, rather focusing on a single line over a period. This is understandable given time constraints and the lack of data-keeping by transit agencies.

Other studies looked at simulated data, which would make modelling easier. However, this modelling is not particularly representative of the real world and such studies would be less effective. The growth of open data and the growing number of providers who release open data mean that this approach is no longer needed. Open data combined with large processing power and convenient libraries mean that now is the perfect time to attempt a more thorough analysis.

Finally, a significant number of papers focused on delay prediction for the consumer side. There was more work done with the individual bus routes and arrival times rather than schedules and systems. Although some papers focused on transit operators and the system side, these were relatively fewer, and this is where additional research could be useful.

There is room for a study to make use of a large amount of real data, which may involve implementing a data collector. The combination of an improved data collector and comparative modelling and analysis shows that there is potential to improve system scheduling for public transit efficiency.

Project Rationale and Hypothesis

Rationale

Given the lack of work in the operator and schedule side of predictions, an explorational study in this area is a sensible step to take. The rapid growth of data and analytics in the past decade makes this work easier and as seen in the literature review, provides many techniques to build on.

Hypothesis

The goal of this paper will be to evaluate a variety of predictive models and algorithms to optimize transit schedules on a more frequent basis. If it is possible to accurately predict bus arrival times with continuous streams of real-time information, this will show that existing techniques are sufficient to provide enhanced scheduling methods. Positive results will support additional work into this area by experts in the field who better understand the nuances and requirements of designing a robust system for transit agencies.

The rest of the report is organized as follows. A description of the experimental methods will be provided. Then, the results will be analyzed. Finally, a conclusion is provided with research expansion proposals.

Experimental Methods

Environment

Conducting an exploration of predictive transit schedules requires a significant amount of data. London, Ontario was used as the case study for this project given its many benefits, including research proximity, open data access, and significant vehicle and passenger volumes.

Given the limited timespan of the project, a working prototype was developed to demonstrate the viability of such a system. This system will incorporate as much data as possible, however, it would be advisable to include much more data in a production system. The prototype tested out various schedules on the models to accurately determine scheduling. Such a task was found to be challenging enough and remained the goal throughout the project.

Data Collection

The data source for the London, Ontario transit system (London Transit) was provided through the City of London Open Data program. Through this, an application programming interface was provided on the London Transit website which provided schedules and real-time data as seen in Figure 1. All data was provided in the standard General Transit Feed Specification (GTFS) format, which allowed for straightforward processing and cleaning. Real-time updates were then transferred through protocol buffers during all operational hours which would need to be collected.

trip_id	arrival_time	departure_time	stop_id	stop_sequence	pickup_type	drop_off_type	timepoint
1342560	6:13:00	6:13:00	KIPPADEL	1	0	1	1
1342560	6:13:51	6:13:51	KIPPBELF	2	0	0	0
1342560	6:14:41	6:14:41	KIPPBARK	3	0	0	0
1342560	6:15:13	6:15:13	KIPPARBO	4	0	0	0
1342560	6:16:04	6:16:04	KIPPBRIA	5	0	0	0
1342560	6:16:43	6:16:43	BRIAMEL2	6	0	0	0
1342560	6:17:17	6:17:17	BRIAMEL3	7	0	0	0
1342560	6:17:35	6:17:35	BRIAHURO	8	0	0	0
1342560	6:17:56	6:17:56	HUROBRI2	9	0	0	0
1342560	6:18:46	6:18:46	HUROBARK	10	0	0	0

Figure 1: Raw scheduled arrival time data from London Transit

A data collector was written in Python for its ease of use and integration with GTFS protocol buffer formats. Although the initial script was not particularly complex, the underlying mechanisms that allowed it to operate 24 hours a day 7 days a week were slightly more complex. A cron job was used to run the script in one-minute intervals and download live data. This cron job was run on a Linux machine hosted remotely on Amazon Web Services. Initially, manual downloads were performed to capture and archive the data. A later script was used to convert the GTFS format into an easily analyzable CSV format.

More significant challenges lay within the data cleaning aspect of the collection process. Given that the collector ran on one-minute intervals to get up-to-date data, there tended to be a large amount of repetitive data. This is due to the wider headways between transit vehicles. Redundant updates and repetitious rows were commonplace, and this required plenty of comparison to keep only necessary rows. It also became a challenge to manage multiple files, which required more automation with Python scripts to keep the workspace tidy.

Within the files and data, there were several quality and consistency issues that were tackled. Of note are the timekeeping issues, of which there were many. London Transit provides schedules that are all localized. However, the protocol buffer and remote machines produced times that were UTC, a default time zone. This conflict meant that further conversions had to be done using Python datetime objects, which often had their own defaults. Elsewhere, empty rows, invalid data, and mixed headers were all resolved as part of a standard data engineering process (Figure 2).

In achieving all these steps, a level of automation was conducted to ensure smooth and accurate operations. A modular system was designed so that the entire process could be repeated with just one click. This saved a significant amount of researcher time and reduced the potential for human error at the same time.

id	trip_id	start_date	route_id	stop_seq	realtime_dep_time	vehicle_id	scheduled_dep_time
2735543	1331895	2020-11-07	6	16	16:48:00	3172	16:48:00
613213	1343233	2020-11-11	10	12	15:45:01	3136	15:45:01
607801	1343225	2020-11-17	10	69	15:40:24	3144	15:40:59
1924122	1339888	2020-11-18	31	4	08:00:52	3350	07:59:52
1841457	1338069	2020-11-09	20	3	07:15:18	3378	07:15:00
2636984	1343356	2020-11-21	10	26	13:00:37	3312	13:02:46
1171153	1345792	2020-11-13	15	33	19:20:41	3172	19:19:41
779022	1340516	2020-11-06	34	9	16:44:00	3335	16:43:00
639682	1345017	2020-11-09	07	3	15:49:20	3147	15:49:35
1593704	1345753	2020-11-24	15	20	23:23:26	3177	23:23:41

Figure 2: Cleaned, processed output data for analysis

To write code for these systems, several development tools were used. The integrated development environment was Visual Studio Code, which had the advantage of being able to remotely connect to a virtual machine. GitHub was used for source control and versioning, ensuring that a historical record was maintained. Various file transfer tools were used as well.

Data Analysis

A standard Python data science stack was used for analysis, consisting of the Anaconda package, Jupyter notebooks, NumPy, Pandas, and Sci-Kit Learn. This was accessed through the Visual Studio Code integrated development environment. Using a widely supported stack meant that plenty of resources were available and an extensive set of libraries were available to use for analysis.

The first technique applied was a regression. A simple linear regression that minimized the squared differences between data served as a benchmark to compare against the other techniques. It was earlier noted that the variance involved in arrival times made the regression less effective, which is why it serves as a baseline for comparison (also done in other papers). This was then improved upon by applying a multivariate regression, considering the other useful variables that are provided in the data.

Multiple subsets of the data were then compared under the linear regression. An overall assessment of the entire system was done, as was each individual route. Further comparison was done to identify advantages, drawbacks, and anomalies.

The same data was then analyzed with decision tree regression. This method had some slight differences with linear regression; however, the output was still to predict the scheduled arrival times and the methodology remained the same.

Finally, a black box algorithm was tested to compare its effectiveness. More specifically, neural networks were used on both the system and route data. The entire process was again followed, from training to backpropagation to testing. Although the manipulation of weights and biases under the cover may not be visible, it was interesting to see the results that came out of such an algorithm. There may be potential for further developments in reinforcement and deep learning along this path.

Iteration and Improvement

With the analytical and predictive techniques in place, it was worthwhile to spend time exploring the implementation of the system. One of the key additions was aggregating data to improve the regressions, decision trees, and neural networks. As time passes, more bus routes will have been travelled and more data will be generated. This can then be incorporated into the prediction system to help optimize the schedule. Such an architecture will be outlined and tested. The system would then be processing a data stream, which would require further analysis for a production system.

Experimental Results and Discussion

Ordinary Least Squares Regression

The first linear regression conducted used inputs of the scheduled arrival time compared against the dependent variable of the actual arrival time. A linear relationship was captured between the two

variables, with high R^2 scores recorded for most routes. For the overall system, this score was around 0.92 while the scores for individual routes ranged from 0.75 to 0.99. Figure 3 shows the predictive output using the complete set of system data. There is a correlation between the variables, however, there is a significant amount of variance. This means that the regression captures the correlation on average, but unfortunately, there are very few situations that consist of the average. This will reduce the predictive power of the model.

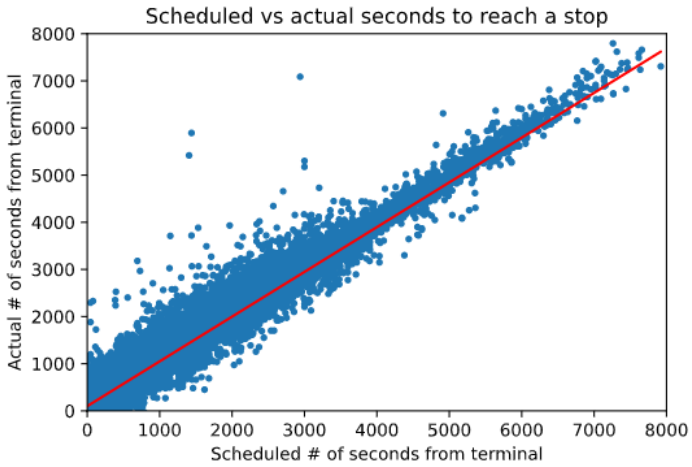

Figure 3: Linear regression using the complete set of data points

Additional regressions were conducted for each individual trip. Like the first regression, the relationship between the scheduled time and actual time was successfully captured. There are slightly less variance and a more pronounced pattern with each individual trip, so it is in many cases that the individual regression has a better fit (Figure 4). However, there were also many cases where it was not a better fit. First, there may simply be a lack of data for a particular trip, making it challenging to fit an accurate line (Figure 5). Second, the variance problem seen in the overall system data could be more pronounced in the individual trip data (Figure 6). Finally, there may be recurring anomalies and patterns in the data that do not fit a perfectly straight line (Figure 7)

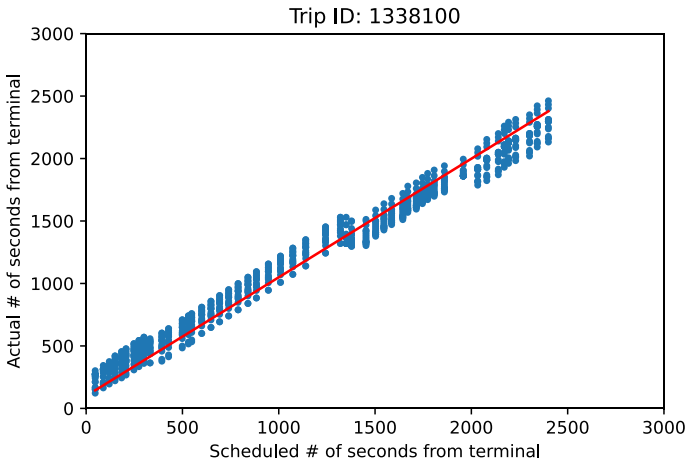

Figure 4: Linear regression with a single trip ID, good results

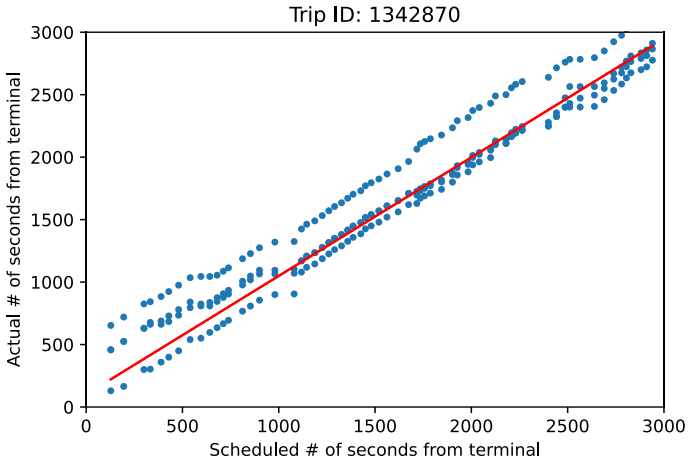

Figure 5: Linear regression with a single trip ID, insufficient data

As the last step, additional parameters were added. This improved the error scores slightly. However, due to the likely correlation between variables, this did not have a significant impact on the improvement of the results. Overall, linear regression was a useful tool that considers all data points to come up with a comprehensive prediction for trip times. It is more effective than taking simple averages at limited timepoints, however, it will not cover all cases due to variance. These factors will need to be considered when implementing a system to optimize transit schedules.

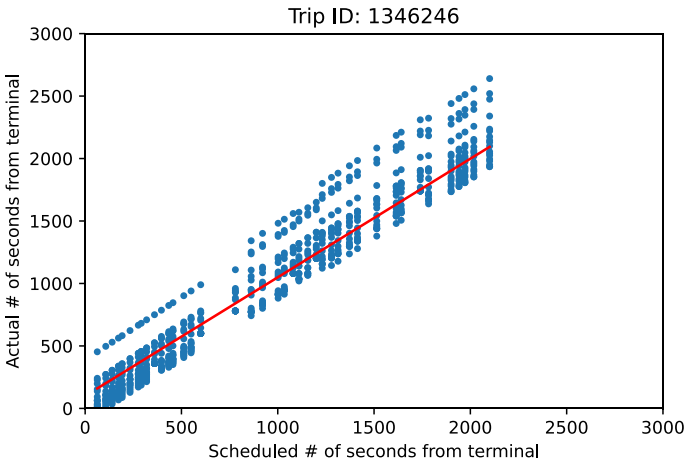

Figure 6: Linear regression on a single trip ID, too much variance

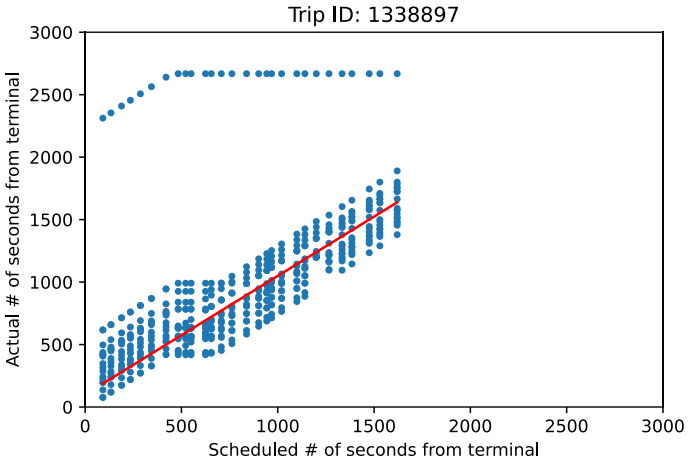

Figure 7: Linear regression on a single trip ID, too many anomalies

Decision Trees

The decision tree model, which had the same independent and dependent variables, was also able to capture the relationship between the two variables. Figure 8 shows the predictive output from the complete set of data, with an R^2 score of 0.96. This is slightly higher than for linear regression, however, the difference is not significant. With the parameters set on the decision tree model, a more fine-grained prediction was possible for the arrival times. However, this came at the risk of overfitting. Depending on the number of data points, the predicted line was often very choppy, and this would not

be particularly representative of a bus arrival time. But understanding and controlling the relevant parameters would reduce this issue and make decision trees a better choice for prediction.

Further decision trees were trained on individual trip data. Like the first decision tree, the relationship between the scheduled time and the actual time could be captured with regression. Given that there were more prominent patterns and anomalies within each individual route, decision trees were able to capture these nuances better than would be with linear regression. R^2 values remained high, with most being around or exceeding 0.9. Notable anomalies that were better represented by a decision tree include frequent/consistently longer layovers, curved timings, and more. However, the model was again far more prone to overfitting. Particularly if there was significant variance or if there was a lack of data, the decision tree often tried to exactly fit the points, jumping up and down. This was not a good sign, however, by controlling the parameters this issue could be reduced.

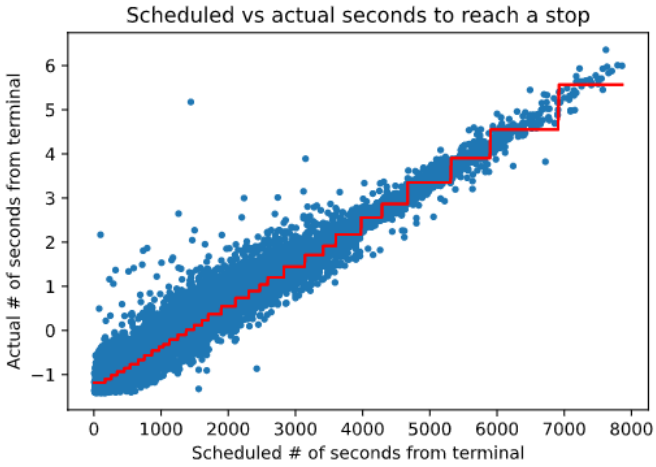

Figure 8: Decision tree on overall system data

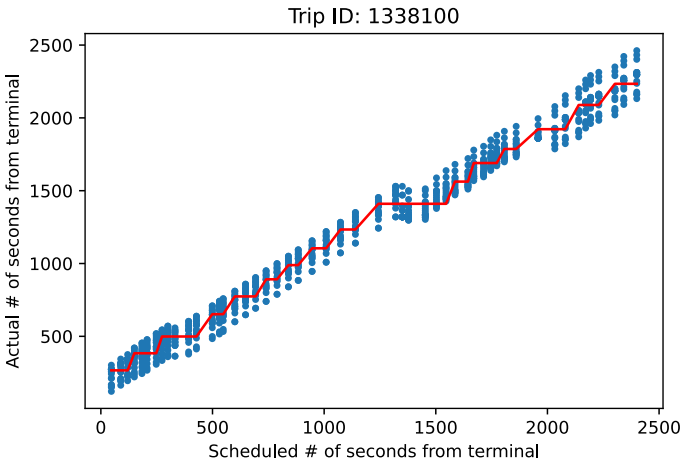

Figure 9: Decision tree on a single trip ID

Finally, additional parameters were added as in linear regression. This improved the error scores but not significantly. Given the more detailed analysis, the impact was slightly larger than with linear regression. The decision trees performed better than the regression overall, with the ability to capture more nuances in each line. Overall, it is important to beware of overfitting so that the models generalize well to future predictions.

Black Box Algorithms

Finally, a simple neural network model was also able to capture the relationship between two variables. Figure 10 shows the predictive output from the complete set of data. Decent results were obtained from using it on the entire dataset, achieving an R^2 score of 0.97. However, this type of model had a more significant risk of overfitting. The fine-tuning of neural networks combined with very specific data meant that the predictive power could be reduced on new data. Luckily, the train-test split of data meant that the black box algorithm still performed decently.

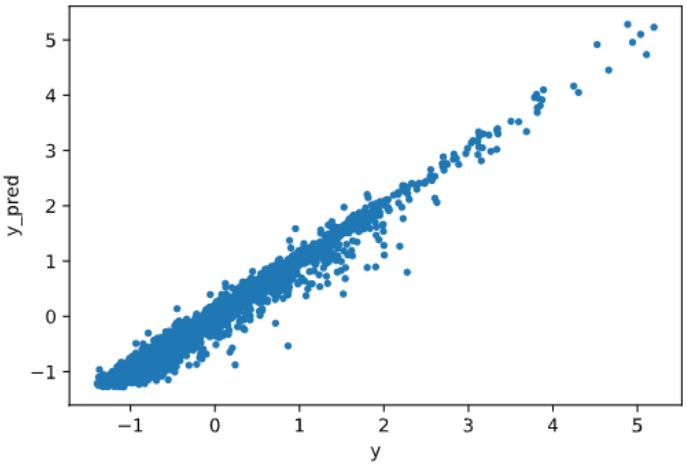

Figure 10: Neural network using entire system data

Within individual trips, a train-test split was also conducted, and the simple neural networks performed similarly. They worked best on trips without large anomalies, and with large amounts of data. Under those conditions, R^2 scores remained above 0.93, dropping to 0.34 for lines with large anomalies. Given the smaller dataset, the model would be improved over longer periods of data collection. Neural networks would benefit from a much larger dataset. Still, the current predictive output was quite useful, often more detailed than the decision trees, although the issue of overfitting was always present.

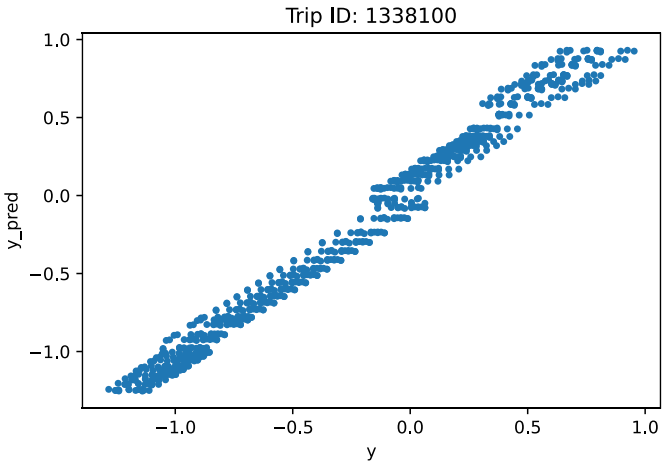

Figure 11: Simple neural network on a single trip ID

All available data were used as input to the black-box algorithms. A large number of features meant that even more nuances could be captured in the analysis compared to decision trees and especially linear

regression. But the conclusion here is that more data is required for better analysis, and continuously incorporating new information would benefit this model the most compared to the other ones. Finally, better neural network design is also something that could also be investigated.

Continuous Information Incorporation

In testing the systems, it was extremely useful to be able to incorporate new data into the models. The architecture for a continuous information incorporation system is described in Figure 12.

Note that there will always be a balance between using the most updated data and keep schedules constant and predictable. These trade-offs would have to be decided by the transit agency. The implementation of such a system would not be overly complex and would serve as a good next step for the project.

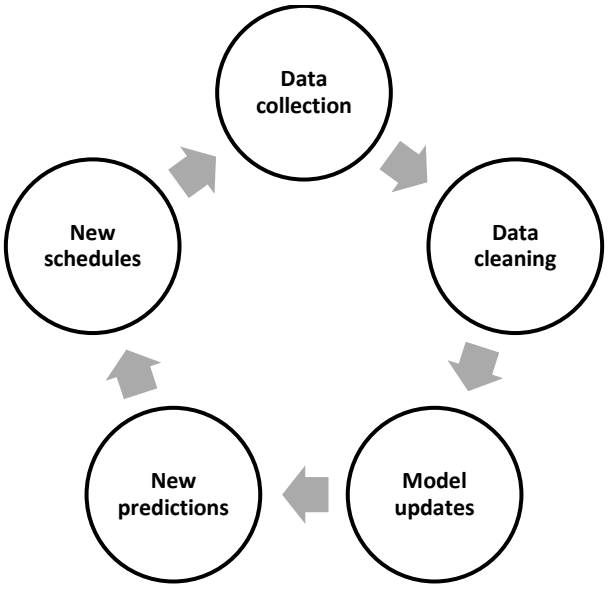

Figure 12: Continuous information incorporation pipeline model

Conclusion

Future Research Expansion

Additional work in this area can make the predictive system more sophisticated and accurate. One area of improvement is in better estimating the nuances with travel times. This may involve placing more weight on recent bus arrival times or looking at varying 'pull' destinations. Other geographical nuances may play a factor and can be integrated into the techniques being used.

Other ideas include incorporating external data sources. Transit arrival times are often affected more than just previous arrival times. Data including weather, traffic, and Western University timetables could play a role in making better predictions. It must be noted that it will be a challenge to import and clean the data so that it will mesh well with the existing system. However, a more comprehensive model could be set up such that it is more precise.

As with any predictive model, more sophisticated algorithms could be used. This explorational study only looked at a few of the many available algorithms and there are many more that could be used. The usual concerns of overfitting and data availability must continue to be monitored so that the best results can be obtained.

Finally, potential partnerships with London Transit may be pursued to iterate over the prototype, to use a greater amount of data to improve the system and to test in the real world. By testing a live system, operational nuances may be better reflected, and the entire data flow will be updated in a timelier manner. Organizational resources and administrative support will bolster such an effort.

References

Altinkaya, M., & Zontul, M. (2013). Urban Bus Arrival Time Prediction: A Review of Computational Models. *International Journal of Recent Technology and Engineering (IJRTE)*, 164-169.

Cathey, F., & Dailey, D. (2003). A prescription for transit arrival/departure prediction using automatic vehicle location data. *Transportation Research Part C*, 241-264.

Chen, M., Liu, X., & Xia, J. (2004). A Dynamic Bus-Arrival Time Prediction Model Basedon APC Data. *Computer-Aided Civil and Infrastructure Engineering, 19*(5), 364-376. doi:https://doi.org/10.1111/j.1467-8667.2004.00363.x

Chien, S. I.-J., Ding, Y., & Wei, C. (2002). Dynamic Bus Arrival Time Prediction with Artificial Neural Networks. *Journal of Transportation Engineering*, 429-438.

Fan, Y., Guthrie, A., & Levinson, D. (2016). Waiting time perceptions at transit stops and stations: Effects of basic amenities, gender, and security. *Transportation Research Part A*, 251-264.

Lin, W.-H., & Zeng, J. (1999). Experimental Study of Real-Time Bus Arrival Time Prediction with GPS Data. *Journal of the Transportation Research Board, 1666*(1), 101-109. doi:https://doi.org/10.3141%2F1666-12

Petersen, N. C., Rodrigues, F., & Pereira, F. C. (2019). Multi-Output Deep Learning for Bus Arrival Time Predictions. *Transportation Research Procedia, 41*, 138-145. doi:https://doi.org/10.1016/j.trpro.2019.09.025

Shalaby, A., & Farhan, A. (2004). Prediction Model of Bus Arrival and Departure Times Using AVL and APC Data. *Journal of Public Transportation, 7*(1), 41-61. doi:http://doi.org/10.5038/2375-0901.7.1.3

Yu, B., Lam, W. H., & Tam, M. L. (2011, December). Bus arrival time prediction at bus stop with multiple routes. *Transportation Research Part C: Emerging Technologies, 19*(6), 1157-1170. doi:https://doi.org/10.1016/j.trc.2011.01.003

Yu, B., Wang, H., & Shan, W. (2018). Prediction of Bus Travel Time Using Random Forests Based on Near Neighbors. *Computer-Aided Civil and Infrastructure Engineering*, 333-350.

Zhou, P., Zheng, Y., & Li, M. (2014, June). How Long to Wait? Predicting Bus Arrival Time With Mobile Phone Based Participatory Sensing. *IEEE Transactions on Mobile Computing, 13*(6), 1228-1241. doi:https://doi.org/10.1109/TMC.2013.136

Appendix

A1: Data Collector Main Program (Python)

```
start = time.time()
print(f'Start: {time.strftime("%H:%M:%S", time.localtime())}')

# Remove empty files from download
print(f'Removing empty files from {LOCAL_DIRECTORY}...')
total_removed = remove_empty_files(LOCAL_DIRECTORY)
print(f'Done. {total_removed} files removed. {time.time() - start} seconds elapsed.')

# Convert all files to CSV
print('Converting files to CSV...')
total_converted = convert_folder(LOCAL_DIRECTORY, OUTPUT_DIRECTORY)
print(f'Done. {total_converted} files converted. {time.time() - start} seconds elapsed.')

# NOT NEEDED RIGHT NOW: Remove header from the files
# remove_header(LOCAL_DIRECTORY, OUTPUT_DIRECTORY)

# Keep only the latest timestamps
print('Cleaning up data...')
total_processed = keep_latest(OUTPUT_DIRECTORY)
print(f'Done. {total_processed} files processed. {time.time() - start} seconds elapsed.')
# DON'T DELETE THE REMAINING FILE, It's needed for the next pipeline download

# Remove empty files from cleaned
print(f'Removing empty files from {CLEAN_DIRECTORY}...')
total_removed = remove_empty_files(CLEAN_DIRECTORY)
print(f'Done. {total_removed} files removed. {time.time() - start} seconds elapsed.')

# Combine all files into one big file
print(f'Combining files...')
combine(CLEAN_DIRECTORY, RESULT_DIRECTORY)
print(f'All processing done. Total {time.time() - start} seconds elapsed.')
```

A2: Libraries for Prediction (Python)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

A3: Get GTFS feeds via protocol buffers (Python)

```
def get_realtime_feed_data():
    """Pull data from GTFS realtime feed and write update to text file"""
    feed = gtfs_realtime_pb2.FeedMessage()
    response = urllib.request.urlopen('http://gtfs.ltconline.ca/TripUpdate/TripUpdates.pb')
    feed.ParseFromString(response.read())

    x = datetime.datetime.now()
    dt_string = x.strftime("%Y-%m-%d-%H-%M-%S")
    f = open(f'data/{dt_string}.txt', 'a+')
    for entity in feed.entity:
        if entity.HasField('trip_update'):
          f.write(repr(entity.trip_update))

    f.close()
```