1. [ages](https://confluence.kroger.com/confluence/collector/pages.action?key=SCIEN&src=breadcrumbs-collector)
2. [84.51° Heartbeat Home](https://confluence.kroger.com/confluence/pages/viewpage.action?pageId=93459784&src=breadcrumbs-parent)

[RNN Traffic](https://confluence.kroger.com/confluence/display/SCIEN/RNN+Traffic)

* Created by [BRIAN BEBEAU](https://confluence.kroger.com/confluence/display/~B642765) on [May 15, 2020](https://confluence.kroger.com/confluence/pages/viewpreviousversions.action?pageId=141099105)

The RNN traffic code was developed to provide a more accurate traffic forecast than the existing forecast. It uses the Recurrent Neural Network (RNN) model, rather than the linear model found in the current traffic code (in the Forecast\_v4\_lm.R and associated files).

RNN code notes

The RNN traffic code consists of about 35 Python files, with a top-level directory and a few subdirectories. The top-level directory is in the Git repository under the main “src” directory and is called “rnn”. The Python files get installed in the instance as an “rnn” directory tree under the “bin” directory. The main executable is called **rnn\_traffic.py**.

If any changes need to be made to the Python code, say, for changing file paths, it will most likely need to be made in the top-level **configurations.py** file, where almost all configuration is done.

The RNN code also utilizes its own holiday file, **USHolidayDates\_1.csv**. This is installed in the ${conf} directory with the other holiday files.

Output files are put in the “rnn” directory under ${traffic\_dir} (${root}/data/traffic/rnn) and have a file name of **rnn\_traffic\_pred.${fcst\_begin\_dt}.txt** and **rnn\_traffic\_pred\_integrated.${fcst\_begin\_dt}.txt.**

The Python code makes use of the Tensorflow library. Since we don’t want to have to keep up with the Tensorflow versions and assorted dependencies, it was decided to run the RNN code in its own Docker container. The Docker options used are discussed below.

Implementation

The Heartbeat training code needs “${fcst\_num\_days}” number of days of traffic data. Currently, this is set to 35. The RNN code only produces 7 days of data. Since the RNN code already consistently takes a bit short of two hours to run just to produce the 7 days, it was decided to run the existing linear model code found in the existing R scripts, then run the new RNN code, and merge the two outputs. The resulting file would have the forecasts from RNN for the first 7 days, then use the linear model forecasts for the remaining days.

The **gen\_traffic.sh** script runs both the linear and RNN models, producing the traffic forecasts. The script sets a number of variables, then calls the Forecast\_v4\_lm.R script. This script runs the **hb\_traffic\_baskets.sh** script and uses the output. The R script uses the linear model functions in Forecast\_v4\_functions.R and creates an output CSV file. After that runs, the new **gen\_traffic\_rnn.sh** script is run. The RNN code is in a separate script, so that if an error happens, it can simply exit with a value of 1, and the **gen\_traffic.sh** script can continue on, using just the linear model output. This prevents the entire system from failing, and a usable traffic forecast is still output, even if the RNN code fails. The RNN output is also in a CSV format. The outputs from the two models are then merged using ‘awk’ into a final output format that sxvelocity can read. The final output file is also copied and compressed to the data archive directory.

The new **gen\_traffic\_rnn.sh**script is passed the date, the temporary directory it should use, and the log file it should write the RNN debugging statements to. It first checks that the “docker” command is installed. If it is not, it exits with value 1 (this is so that the **gen\_traffic.sh** script won’t fail when run on a server (like ham-lnx-ps-002) that doesn’t have Docker). It then runs the **hb\_traffic\_baskets.sh** script for all stores and saves the output of that in the temp directory (Note for future work: change the existing R scripts so that we run hb\_traffic\_baskets.sh only once, and both models read that output file). Since the RNN code *requires*a store to have at least 31 days of basket traffic, the **gen\_traffic\_rnn.sh** script finds the stores in the traffic basket output that have less than that and strips those stores from the output. The RNN code will not produce any output for those stores, but we will then just use the linear model output for them.

The “docker\_envs” file is then created with the following environment variables that the RNN code will use to find file paths:

* **DIVISION=${division}**– the division we’re working on.
* **LOG\_DIR=${clientlogs}**– the directory to put log files in.
* **CONF\_DIR=${conf}**– the config directory where the holiday file is.
* **TMP\_DIR=${tmpdir}**– the temp directory for files that will be removed later.
* **OUTPUT\_DIR=${traffic\_dir}/rnn**– the directory to save output files to.

The docker\_envs file is recreated each time, since the temp directory changes each time. The variable DOCKER\_IMG is set to "tensorflow/tensorflow:1.5.0-py3". This is the container image that provides the environment for the docker command to run. The variable DOCKER\_CMD is set to "**python3 ${bin}/rnn/rnn\_traffic.py -g \"-1\" -d$asof**". This is the Heartbeat command the docker command will run. Note that it *must* use python3, not python2. It runs the **rnn\_traffic.py** script with two arguments. The “**-g**” argument with “-1” says not to look for any GPUs. The “**-d**” argument is the date to provide traffic up to.

The entire docker command with all arguments is:

**docker run -u $(id -u):$(id -g) -v ${root}:${root} --env-file ${env\_file} -w ${root} -it --rm  ${DOCKER\_IMG} ${DOCKER\_CMD} > ${log} 2>&1**

The arguments to the docker “run” command mean the following:

* “**-u**” -- run the docker command as the specified user/group, in this case, svc\_hb.
* “**-v**” -- tells the docker command to mount the Heartbeat instance file system with the same path in the docker environment. This allows the command to access the Heartbeat files.
* “**--env-file**” -- passes the above-mentioned environment variable file. The docker command will read it and set those variables in the running environment.
* “**-w**” -- sets the working directory to the instance root, so that the Heartbeat command will be run from that directory.
* “**--rm**” command removes the docker container when it’s done.

After the docker command is done, the **gen\_traffic\_rnn.sh** script prints out error or success messages and returns to the **gen\_traffic.sh** script.

Merged output

The **gen\_traffic.sh** script merges the output of both the linear model and the RNN model. Since sxvelocity only reads the first five fields of the file, that’s all we need to provide. The final output fields should then be:

Store ForecastDate BenchmarkTraffic ForecastTraffic ActualTraffic

The BenchmarkTraffic and ForecastTraffic fields should be “NA”, and the forecast should be in ActualTraffic. We then need to pass sxvelocity in **hb\_forecast.sh** the “+A” argument to have it use the ActualTraffic column data.

This is the header for the linear model output in file **traffic.lm.20190801.35.csv**:

"Store","ForecastDate","ForecastTraffic","BenchmarkTraffic","ActualTraffic","Holiday","UnadjustedForecastTraffic"

As an example, this is the traffic for the first 12 days for store 014\_00305:

"014\_00305",2019-08-01,14941.9154673694,14321,NA,NA,14941.9154673694  
"014\_00305",2019-08-02,16629.6996453095,15599,NA,NA,16629.6996453095  
"014\_00305",2019-08-03,18994.4643673602,18583,NA,NA,18994.4643673602  
"014\_00305",2019-08-04,22399.4850747754,22404,NA,NA,22399.4850747754  
"014\_00305",2019-08-05,16516.1245678775,15121,NA,NA,16516.1245678775  
"014\_00305",2019-08-06,14714.9693578478,13761,NA,NA,14714.9693578478  
"014\_00305",2019-08-07,14383.2337343034,13755,NA,NA,14383.2337343034  
"014\_00305",2019-08-08,14381.5010260197,13866,NA,NA,14381.5010260197  
"014\_00305",2019-08-09,16069.602958795,16514,NA,NA,16069.602958795  
"014\_00305",2019-08-10,19040.0420876074,18403,NA,NA,19040.0420876074  
"014\_00305",2019-08-11,22241.9312350453,21105,NA,NA,22241.9312350453  
"014\_00305",2019-08-12,16222.40570537,15286,NA,NA,16222.40570537  
  
This is the header for the RNN model output in file **rnn\_traffic\_pred\_integrated.20190801.csv**:

Store,window,Date,integrated\_pred,preds

The traffic for store 014\_003005 is (only 7 days are output):

014\_00305,2019-08-01,2019-08-01,14083.919287153378,13964.688954223548  
014\_00305,2019-08-01,2019-08-02,16765.017018814022,16670.746131877026  
014\_00305,2019-08-01,2019-08-03,18509.939751926933,18411.218569676723  
014\_00305,2019-08-01,2019-08-04,22004.011270939824,21890.97227444764  
014\_00305,2019-08-01,2019-08-05,16079.848924181986,16064.544063154699  
014\_00305,2019-08-01,2019-08-06,14325.566017637328,14275.112728476523  
014\_00305,2019-08-01,2019-08-07,13863.398096801773,13863.398096801773

The merged output in the final file, **traffic.20190801.txt**, for the first 12 days for store 014\_003005 is:

Store   ForecastDate    BenchmarkTraffic        ForecastTraffic ActualTraffic  
014\_00305       20190801        NA      NA      14083.919287153378  
014\_00305       20190802        NA      NA      16765.017018814022  
014\_00305       20190803        NA      NA      18509.939751926933  
014\_00305       20190804        NA      NA      22004.011270939824  
014\_00305       20190805        NA      NA      16079.848924181986  
014\_00305       20190806        NA      NA      14325.566017637328  
014\_00305       20190807        NA      NA      13863.398096801773  
014\_00305       20190808        NA      NA      14381.5010260197  
014\_00305       20190809        NA      NA      16069.602958795  
014\_00305       20190810        NA      NA      19040.0420876074  
014\_00305       20190811        NA      NA      22241.9312350453  
014\_00305       20190812        NA      NA      16222.40570537

We can see that the forecast for the first seven days is from the RNN output, and the forecasts after that are from the linear model output.