NPS-CS-18-001

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

PREDICTIVE PUSH LOGISTIC USING RUNTIME MONITORING OF HIDDEN AND VISIBLE DATA

by

Doron Drusinsky

February 2018

Approved for public release; distribution is unlimited

Prepared for: Naval Research Program (NRP)

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From-To)*
02/15/2018	Technical Report	10/01/2016-12/31/2017

4. TITLE AND SUBTITLE
Predictive Push Logistic Using Runtime Monitoring of Hidden and Visible Data

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Doron Drusinsky

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)
Naval Postgraduate School, Monterey CA

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-CS-18-001

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NPS Naval Research Program

10. SPONSOR/MONITOR'S ACRONYM(S)
NRP

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report described a general purpose predictive logistics software package based on three primary components:

1. A spreadsheet of ship related orders.
2. A Probabilistic Temporal Finite State-Machine (PTFSM) automatically learned from that data.
3. (Optional) Expert rules written in English.

The deliverable tool predicts orders per *Ship-ItemType p*airs. It has two main prediction modes:

a. Predict a probability of an order (for one or more *Ship-ItemType* pairs) to be required in n weeks.
b. Predict the number of weeks required for the probability of an order (for one or more *Ship-ItemType p*airs) to exceed some given value *p*.

15. SUBJECT TERMS
Predictive logistics, Hidden Markov Models

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Doron Drusinsky
Unclassified	Unclassified	Unclassified	UU	25	**19b. TELEPHONE NUMBER** ***(include area code)*** 831-656-2168

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Ronald A. Route
President

Steven R. Lerman
Provost

The report entitled "Predictive Push Logistic Using Runtime Monitoring of Hidden and Visible Data" was prepared for and funded by NPS Naval Research Program.

Further distribution of all or part of this report is authorized.

This report was prepared by:

Doron Drusinsky
Associate Professor
Computer Sciences Department

Reviewed by:

Peter Denning, Chairman
Computer Science Department

Released by:

Jeffrey D. Paduan
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This report described a general purpose predictive logistics software package based on three primary components:

1. A spreadsheet of ship related orders.
2. A Probabilistic Temporal Finite State-Machine (PTFSM) automatically learned from that data.
3. (Optional) Expert rules written in English.

The deliverable tool predicts orders per *Ship-ItemType p*airs. It has two main prediction modes:

a. Predict a probability of an order (for one or more *Ship-ItemType* pairs) to be required in n weeks.
b. Predict the number of weeks required for the probability of an order (for one or more *Ship-ItemType p*airs) to exceed some given value *p*.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

1. INTRODUCTION

Distributed lethality seeks to capitalize on the mobility and dispersed presence of naval forces through increased lethality over time looking out to 2030. Operating over an extended geographic area, particularly in the littorals creates a range of operational problems for an adversary by diluting its force to locate distributed, but lethal, Surface Action Groups (SAG) further complicating the adversary's ability to focus on the high value unit. Distributed lethality (DL) also presents a complicated logistics problem in supporting that distributed force. A key component of DL is predictive push-logistics, where suppliers use a push model rather than waiting for consumers to pull. This research focuses on predictive push-logistics.

This report described a general purpose predictive logistics software package based on three primary components:

1. A spreadsheet of ship related orders.

2. A Probabilistic Temporal Finite State-Machine (PTFSM) automatically learned from that data.

3. (Optional) Expert rules written in English.

The deliverable tool predicts orders per *Ship-ItemType p*airs. It has two main prediction modes:

a. Predict a probability of an order (for one or more *Ship-ItemType* pairs) to be required in n weeks.

b. Predict the number of weeks required for the probability of an order (for one or more *Ship-ItemType p*airs) to exceed some given value *p*.

THIS PAGE INTENTIONALLY LEFT BLANK

2. IMPLEMENTATION

The tool-set described in this report consists of two components:

1. *LearningPhase.* This component consists of two packages: *split* and *ptfsm*, as follows.
   - The *split* program which splits the large Excel spreadsheet *paul-bt-07181601_v2.csv* received from the Navy, into CSV spreadsheets containing data per Ship-ItemType pairs. For example, the file *smallCSV704096740.csv* contains information pertaining to the Ship-ItemType pair: USS CHANCELLORSVILLE CG 62, "HELMET (27). Further details are provided in section 2.1.1.
   - The *ptfsm* program which generates a PTFSM per *Ship-Item* pair. The generated PTFSM is serialized in JSON format with a corresponding file name, such as *smallCSV704096740.hmm.json* being the PTFSM for *smallCSV704096740.csv.* Further details are provided in section 2.1.2.
2. *Runtime*. This component consists of a program named *Alpha*. Alpha either predicts the probability of a human placing an order (per Ship-ItemType pair) in *n* weeks (e.g., 10 weeks), or calculates the number of weeks after which the probability of an order (by Ship-ItemType pair) is greater than some given probability value. Further details are provided in section 2.2.

2.1. LEARNING PHASE PROGRAMS

2.2.1. The Split (*splitcsv*) program.

2.1.1.1 Overview and extracting relevant information

The *split* program splits the large Excel spreadsheet *paul-bt-07181601_v2.csv* received from the Navy into CSV spreadsheets containing data per Ship-ItemType pairs. For example, the file *smallCSV704096740.csv* contains information pertaining to the pair: USS CHANCELLORSVILLE CG 62, "HELMET (27). Files that contain an insufficient number of rows to be usable for prediction (less than 10) are not generated. Those *Ship-Item* pair files that are generated are in the *meaningfulShipItemtypeFiles* subdirectory of the *Data* folder. The file *mapping.txt* contains the mapping between the file name (e.g., *smallCSV704096740.csv)* and the *Ship-Item* pair (e.g., USS CHANCELLORSVILLE CG 62$"HELMET (27)).

Generated *Ship-Item* pair files (e.g., *smallCSV704096740.csv*) do not contain all columns of the original file. Rather, they contain only data for the columns specified in *Data/ HmmColumnNames.txt.* Currently, this file contains the columns: *DATE_ORDERED, IS_REORDER, IS_CANCELLED, IS_COMPLETE_BACKORDER, PRIORITY, QTY.* We call these columns ***relevant columns*** *or* ***relevant items***.

Table 1 depicts *smallCSV704096740.csv*, where columns are, by order from left to right*:*

QTY

DATE_ORDERED

IS_REORDER

IS_CANCELLED

IS_COMPLETE_BACKORDER

PRIORITY

10	6/19/14	0	0	0	13
3	6/19/14	0	0	1	13
1	6/19/14	0	0	1	13
2	6/26/14	0	0	1	13
1	6/26/14	0	0	1	13
2	9/9/14	0	0	1	13
4	9/17/14	0	0	0	6
4	10/10/14	0	0	0	3
4	10/10/14	0	0	0	3
2	6/28/15	0	0	1	5
50	8/24/15	0	0	0	5
50	8/24/15	0	0	0	5
12	8/24/15	0	0	0	5
1	8/24/15	0	0	0	5
37	8/24/15	0	0	1	5
50	8/24/15	0	0	0	5
50	8/24/15	0	0	0	5
50	8/24/15	0	0	0	5
50	8/24/15	0	0	0	5
50	8/24/15	0	0	0	5
10	9/1/15	0	0	1	5
10	9/1/15	0	0	1	5
10	9/1/15	0	0	0	5
2	2/7/16	0	1	0	12
1	2/12/16	0	1	0	5
1	2/12/16	0	0	0	5
50	4/27/16	0	0	0	12

Table 1. *smallCSV704096740.csv*

2.1.1.2 Executing the *splitcsv* program.

To execute the *splitcsv* program, as well as other programs associated with this project, you need to set up a directory named *Data* that contains: (i) the orders csv file provided by the Navy, and (ii) the three jar files: splitcsv.jar, ptfsm.jar and alpha.jar, (iii) the *HmmColumnNames.txt* file discussed above.

Open a command-line window and CD (change directory) to the Data directory.

The command-line for executing this program is:

java -jar splitcsv.jar

The *splitcsv* program generates a translation table, named *mapping.txt* (resides in *Data/ meaningfulShipItemtypeFiles*), which maps every Ship-ItemType pair name to the corresponding csv file. For example, it contains the following line-pair:

File: smallCSV704096740

An original Ship$item-type:USS CHANCELLORSVILLE CG 62$"HELMET (27)

Meaning: The Ship-ItemType pair USS CHANCELLORSVILLE CG 62$"HELMET (27) is represented by smallCSV704096740.csv.

2.2.2. The PTFSM program

The *ptfsm* program *(*code named *hmm)* generates a PTFSM per *Ship-Item* pair. The generated PTFSM is serialized in JSON format with a corresponding file name, such as *smallCSV704096740.hmm.json* being the PTFSM for *smallCSV704096740.csv.*

As with all finite state machines, the PTFSM consists of states and transitions. Transitions are annotated with a probability measure. Some states are associated with a delay measure.

To explain the structure of the PTFSM we first identify relevant columns. They are the columns specified in *Data/ HmmColumnNames*.txt as described in section 2.1.1. However, of those columns, IS_CANCELLED and IS_COMPLETE_BACKORDER columns are recorded for future use by the runtime program (see section 2.2), but are not otherwise used here.

The structure of the PTFSM is as follows.

a. PTFSM State set.
   - There are two types of states:
     (i) X$Y states, where X is a Cartesian product of relevant items from row *i* (*i* being any row index) and Y is a Cartesian product of relevant items from row *i*+1. See section 2.1.1.1 for a discussion of *"relevant items"*. The Cartesian product is in the format C1_C2_..._C*n*, where Ci relates to one of the relevant columns. For a Boolean column (e.g. "IS_REORDER") Ci is either that column (if the corresponding cell value is 1) name or "NORMAL". For a quantitative column (PRIORITY and QTY) Ci is ether HIGH or LOW (e.g., quantity being high or low means it is above or below quantity average for that Ship-ItemType pair).
     (ii) PREDICT_Y states, for the abovementioned Y's.
   - PREDICT_Y is, as its name suggests, a prediction for an order of this ItemType for this ship (by "this" it is meant the one associated with the PTFSM, each PTFSM is associated with such a pair). For example, IS_REORDER_HIGH_LOW is a prediction that a row with

IS_REORDER=1, QTY being above average for that Ship-ItemType pair, and PRIORITY being LOW for that pair.

- The PTFSM state set for a Ship-ItemType pair is constructed from rows of the corresponding Ship-ItemType pair spreadsheet. For example, consider the following row

  50 8/24/15 0 0 0 5

  The first column is QTY, the value of 50 is HIGH (above average for this file). The third column is IS_REORDER, which contains 0 – i.e., "NORMAL" is used to represent that value. The last column is PRIORITY which is LOW. Hence the row induces a state NORMAL_LOW_HIGH.

b. PTFSM transition set.
   The PTFSM contains two type of transitions:
   (i) From an X$Y state to a PREDICT_Y state (note the same Y in both states).
   (ii) From a PREDICT_Y state to a Y$Z state (note the same Y in both states).

   No other transitions exist. Transitions of the first type are assigned probability 1. Transitions of the second type are assigned proportional probability as follows. Suppose there are three Y$Z type states with the same Y: YA, YB, and Y$C. Remember that each such state Y$Z is created because there exists a row *i* such that: Y is the Cartesian product induced by row *i*,, and Z is Y is the Cartesian product induced by row *i*+1. Suppose Y$A is induced twice (e.g., by row pairs 5,6 and 9,10), Y$B is induced twice (e.g., row pairs 2,3 and 7,8), Y$C once (e.g., by row pair 13,14). In this case there will be three corresponding outgoing transitions from PREDICT_Y: the transitions to YA, YB with probability 0.4 each, and the transition to Y$C with probability 0.2.

c. Delays in the PTFSM.
   Recall that every X$Y state of a Ship-ItemType pair spreadsheet file is created from pairs of successive rows in the spreadsheet, where X is a Cartesian product of relevant items from row *i*, and Y is a Cartesian product of relevant items from row *i*+1. X$Y is assigned a delay parameter which is the average delay between all such pairs of rows that induce the state X$Y. For example, suppose rows 10, 11 induce state X$Y with date fields that are 10 days apart, and rows 15, 16 also induce state X$Y with date fields that are 20 days apart; the delay associated with X$Y is then 15 days. PREDICT_Y states have no delay attribute.
   The run-time usage of this delay attribute is explained in section 2.2.1.

The command-line options for PTFSM are:

1. No arguments. PTFSM builds a PTFSM file per each Ship-ItemType csv file created by *splitcsv*. These input files are expected to reside in *Data/meaningfulShipItemtypeFiles.*
2. One argument: a csv file. PTFSM builds a PTFSM file for the Ship-ItemType corresponding to the input csv file. The input file files should be located in *Data/meaningfulShipItemtypeFile.*

3. Two arguments: (i) an *existing* csv file *A*, (ii) an "*increment"* csv file *I*. PTFSM builds a PTFSM file for the Ship-ItemType corresponding to the csv file resulting from appending the two input csv files: *A+I* (i.e., new version appended to old version). These input files are expected to reside in *Data/meaningfulShipItemtypeFiles*

Some command-line examples for *ptfsm* are:

1. java -jar ptfsm.jar

   Creates PTFSM files (files with extension .hmm.json) for all Ship-ItemType pair csv files in *Data/meaningfulShipItemtypeFiles.*

2. java -jar ptfsm.jar smallCSV-1002242591.csv

   Creates a PTFSM file named smallCSV-1002242591.hmm.json for the Ship-ItemType pair file: *Data/meaningfulShipItemtypeFiles/*smallCSV-1002242591.csv.

3. java -jar ptfsm.jar smallCSV-1002242591.csv smallCSV-1002242591appendTest.csv

   Creates a PTFSM file named smallCSV-1002242591.csv.appended.hmm.json for the Ship-ItemType pair csv file resulting from appending: *Data/meaningfulShipItemtypeFiles/*smallCSV-1002242591.csv with smallCSV-1002242591appendTest.csv.

2.2. THE RUNTIME PROGRAM

2.2.1. Overview

The runtime program, called *Alpha*, executes a PTFSM for one or more Ship-ItemType pairs; see command-line options below for a description of execution modes.

Alpha either predicts the probability of a human placing an order (by Ship-ItemType pair) in *n* weeks (e.g., 10 weeks), or calculates the number of weeks after which the probability of an order (by Ship-ItemType pair) is greater than some given probability value.

Alpha's operation is based on the PTFSM for the corresponding Ship-ItemType pair, generated by the *ptfsm* program of section 2.1.

Alpha's cycle by cycle operation is based on Rabiner's "α" dynamic programming algorithm formula for Hidden Markov Models [*L. W. Rabiner, A Tutorial on Hidden Markov models and Selected Applications in Speech Recognition, Proc. of the IEEE, Vol 77, No. 2, 1989*]:

$\alpha_i(t) = \Sigma_j \alpha_j(t-1)*a(j,i)*d(j)$

i.e., the probability of reaching state *i* on cycle *t* is the sum (over all states *j*) of the product of the following three values: (i) the probability of reaching state *j* in *cycle t*-1, (ii) probability of the transition from state *j* to state *i*, (iii) delay probability of state *j*.

Recall that a PTFSM contains two types of states: X$Y and PREDICT_Y. A transition X$Y→ PREDICT_Y has probability 1 but uses the learned delay (measured in days); see section 2.1.2 for details about the delay attribute. The delay d(*j*) is calculated using an

inverse exponential delay model; it effectively increases the probability of transitioning to PREDICT_Y as time passes. Transitions PREDICT_Y→ Y$A, PREDICT_Y→ Y$B,… each have a probability that is potentially less than 1, but no delay, i.e., d(PREDICT_Y) = 1.

2.2.2. Output

Alpha generates a file named *orders.txt* that contains probabilities of orders. See the examples below to understand this file. The file will reside in Data/*meaningfulShipItemtypeFile*.

2.2.3. Command-line options

The command-line options for Alpha are:

1. One argument: *n*. Alpha predicts order(s) *n* weeks into the future. This is done per PTFSM (i.e., *hmm.json*) file in the *Data/meaningfulShipItemtypeFile* directory.
2. Two arguments: (i) *n*, (ii) ship name (wrapped in quotes). Alpha predicts order(s) *n* weeks into the future for the given ship. Alpha works of all PTFSM generated files (i.e., *hmm.json* files) for the given ship that exist in the *Data/meaningfulShipItemtypeFile* directory.
3. Three arguments: (i) *n*, (ii) ship name (wrapped in quotes), (iii) item type name (wrapped in quotes). Alpha predicts an order *n* week into the future for the given Ship-ItemType pair. Alpha works of the PTFSM generated file (i.e., *hmm.json* file) for the given Ship-ItemType pair that exist in the *Data/meaningfulShipItemtypeFile* directory.

Notes:

1. In all three cases, if the first argument is *+n*, i.e., *n* has a "+" prefix (*n* is an integer number of weeks), then the algorithm advances time by the mean delay in all X$Y states, until *n*-weeks are consumed. However, if no such prefix exists, then time will advance by a single lump *n*-weeks value.
2. In all three cases, if the first argument is *Pnn*, where *nn* is a number between 0 and 1, then *n* will be computed as smallest delay (measured in weeks) that induces an order (a PREDICTDATEENTERED_Y state) with probability *nn* or greater");
3. In all three cases, an optional Data/*rules.txt* contains 0 more more rules that affect the outcome. Rules are documented in section 2.2.4. See the sample rule.txt file for rule samples.

Some command-line examples for Alpha are:

1. java -jar alpha.jar 10
   Predicts orders 10 weeks into the future. This is done per Ship-ItemType pair whose PTFSM generated hmm.json file is in the *Data/meaningfulShipItemtypeFile* directory.
2. java -jar alpha.jar P0.55

   Predicts the number of weeks until the probability of an order is greater than 0.55. This is done on a Ship-ItemType pair basis, for all pairs whose generated PTFSM (*hmm.json*) file exists in the *Data/meaningfulShipItemtypeFile* directory.

Listing 1 below is the generated orders.txt:

```
For Ship: USS SPRUANCE DDG 111; Item: "DIAPHRAGM (12):
Number of weeks until probability of "predict order" is: P0.6, is: 8
  Most probable Priority is HIGH;  Most probable Qty is HIGH
```

Listing 1. The orders.txt file generated for the command: java -jar alpha.jar P0.6 "USS SPRUANCE DDG 111" "DIAPHRAGM"

3. java -jar alpha.jar +10
   Predicts orders 10 weeks into the future, like example #1, but uses an algorithm where time advances by the mean delay in all X$Y states, until *n*-weeks are consumed, rather than advancing time by a single *n*-week value.
4. java -jar alpha.jar 10 "USS SPRUANCE DDG 111"
   Alpha predicts order(s) *10* weeks into the future, for the ship "USS SPRUANCE DDG 111".
   Listing 2 below depicts a snippet from the generated orders.txt:

```
For Ship: USS SPRUANCE DDG 111; Item: "INSULATION SLEEVING
    (32):
Prediction for an order date 10 weeks out is P=0.6412500579989833;
    after applying rules: 0.6412500579989833
  Most probable Priority is LOW;  Most probable Qty is HIGH

For Ship: USS SPRUANCE DDG 111; Item: "PENCIL (39):
Prediction for an order date 10 weeks out is P=0.5739897161523984;
    after applying rules: 0.5452902303447784
  Most probable Priority is HIGH;  Most probable Qty is HIGH

For Ship: USS SPRUANCE DDG 111; Item: "FLANGE (12):
Prediction for an order date 10 weeks out is P=0.4716773036264559;
    after applying rules: 0.4716773036264559
  Most probable Priority is LOW;  Most probable Qty is HIGH
```

Listing 2. a snippet from the orders.txt file generated for the command: java -jar alpha.jar 10 "USS SPRUANCE DDG 111"

5. java -jar alpha.jar 10 "USS SPRUANCE DDG 111" "DIAPHRAGM"
   Alpha predicts the order for, "DIAPHRAGM" *10* weeks into the future, for the ship "USS SPRUANCE DDG 111"
   Listing 3 below is the generated orders.txt:

```
For Ship: USS SPRUANCE DDG 111; Item: "DIAPHRAGM (12):
Prediction for an order date 10 weeks out is P=0.6145287764165144;
    after applying rules: 0.6145287764165144
  Most probable Priority is HIGH;  Most probable Qty is HIGH
```

Listing 3. The orders.txt file generated for the command: java -jar alpha.jar 10 "USS SPRUANCE DDG 111" "DIAPHRAGM"

2.2.4. Adding expert rules

The runtime program integrates human created rules into the prediction process. To do so, a file named rules.txt must exist in the Data directory.
At present, there are two types of generic rules one can add to rules.txt:

1. *reduce probability by n% if there are more than m IS_CANCELLED for the affected Ship,Item pair*.
2. *reduce probability by n% if there are more than m IS_COMPLETE_BACKORDER for the affected Ship,Item pair*.

The end-user can customize these generic rules by specifying Integer values for *n* and *m*. For example:
reduce probability by 10% if there are more than 2 IS_CANCELLED for the affected Ship,Item pair. This rule will reduce the probability of an order from 0.5 to 0.45 for every Ship,Item pair that has are more than 2 rows with IS_CANCELLED.

2.2.5. Runtime Results

After building the collection of Ship-ItemType pair PTFSM's we executed the runtime Alpha program on all those models. Some examples of the outputs, taken from Data/*meaningfulShipItemtypeFiles*/*orders.txt*, are:

1. *orders_1.txt*. This is the *orders.txt* file generated when using the command-line argument: P0.55; i.e., it contains an estimated of the number of weeks required for each Ship-ItemType pair that induces an order with probability 0.55 or higher. For each pair, it reports whether the order is more probable being Priority LOW vs. HIGH, and likewise for QTY (quantity). The following is a sample entry from *orders_1.txt*.
   For Ship: USS MOMSEN DDG 92; Item: "DISPENSER (12):
   Number of weeks until probability of "predict order" is: P0.55, is: 6
   Most probable Priority is LOW; Most probable Qty is LOW
2. *orders_2.txt*. This is the *orders.txt* file obtained when using the command-line argument: 10; i.e., it contains, for each Ship-ItemType pair, the probability of making an order within 10 weeks. For each Ship-ItemType pair, it reports whether the order is more probable being Priority LOW vs. HIGH, and likewise for QTY (quantity).
   Prediction for an order date 10 weeks out is P=0.5966288664632318; after applying rules: 0.5667974231400702
   Most probable Priority is LOW; Most probable Qty is LOW
3. *orders_3.txt*. Same as number 2 above, except the algorithm advances time repeatedly by the mean delay of all X$Y states, until the 10-week budget is consumed.

All three versions of the *orders,txt* file are provided in the deliverables package, described in section4.

3. UPDATING FILES AFTER A NEW ORDER IS CREATED

If an actual order is made in some cycle for a certain each Ship-ItemType pair, whether based on this program or not, then do the following to keep the system up-to-date:

d. Append that push-information row to the associated csv file for the Ship-ItemType pair.
e. Rerun the PTFSM builder for that the Ship-ItemType using command-line option No. 3 of the *ptfsm* program (see section 2.1.2).
f. Rerun Alpha for the the Ship-ItemType pair (i.e., command-line option No. 3, section 2.2.3).

Alternatively, you can append the push-information row to the original spread sheet:

a. Append that push-information row to the original spreadsheet: *Da*ta/*paul-bt-07181601_v2.csv*.
b. Rerun the *splitcsv* program (see section 2.1.1.2).
c. Rerun the PTFSM builder for all Ship-ItemType pairs using command-line option No. 1 of the *ptfsm* program (see section 2.1.2).
d. Rerun Alpha for all Ship-ItemType pairs (i.e., command-line option No. 1, section 2.2.3).

THIS PAGE INTENTIONALLY LEFT BLANK

4. DELIVERABLES

In addition to this report, the Deliverables/Data folder contains:

1. Executables, in the form of Java jar files: splicsv.jar, ptfsm.jar, and alpha.jar.
2. A sample *rules.txt* (see section 2.2.4).
3. The files *orders_1.txt*, *orders_3.txt*, and *orders_3.txt* discussed in section 2.2.5.
4. The file *HmmColumnNames.txt*, discussed in section 2.1.1.1.
5. The original spreadsheet, *paul-bt-07181601_v2.csv* is not not provided, because of its size

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Research Sponsored Programs Office, Code 41
   Naval Postgraduate School
   Monterey, CA 93943

4. NPS Naval Research Program
   Naval Postgraduate School
   Monterey, CA 93943