Πρόλογος

Το πρόβλημα της αξιοπιστίας των δεδομένων που εξάγονται από τον Ιστό και χρησιμοποιούνται σε εφαρμογές είναι ένα συχνό φαινόμενο. Οι εφαρμογές στηρίζονται σε δεδομένα από διαφορετικές πηγές που, πολλές φορές, φαίνεται να έχουν διαφορετική τιμή ανά πηγή. Αυτό μπορεί να συμβαίνει, για παράδειγμα, διότι οι πηγές έχουν διαφορετικό βαθμό αξιοπιστίας ή διότι οι τιμές έχουν υποστεί αλλοίωση κατά την επεξεργασία τους. Για την αντιμετώπιση προβλημάτων αυτής της φύσης έχουν αναπτυχθεί διάφορες τεχνικές «συγχώνευσης δεδομένων» (data fusion) ώστε να υπάρχει η δυνατότητα της διάκρισης των δεδομένων σε αληθή ή ψευδή για τη δημιουργία ενός καθαρότερου συνόλου δεδομένων.

Σενάριο χρήσης

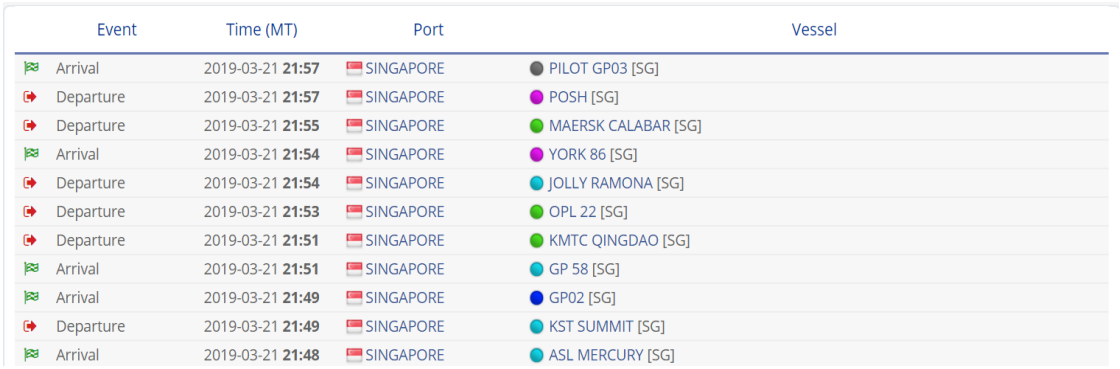
Αυτή η κατηγορία προβλημάτων εμφανίζεται σε πολλούς κλάδους, ένας εξ αυτών είναι και η ναυτιλία. Συγκεκριμένα, υπάρχουν πολλές πηγές που παρέχουν διάφορες πληροφορίες για πλοία και λιμάνια, οι οποίες αρκετές φορές παρέχονται και σε πραγματικό χρόνο. Τέτοιου είδους πληροφορίες μπορεί να είναι η τοποθεσία ενός πλοίου, η διαδρομή που ακολουθεί, ποιος είναι ο προορισμός του, πότε έφτασε σε ένα λιμάνι αλλά και πότε έφυγε από αυτό. Ενώ, οι διαφορετικές πηγές παρέχουν τις ίδιες πληροφορίες και θα έπρεπε να συμφωνούν σε κάποια κοινά σημεία (π.χ. την ακριβή ώρα που έφτασε ένα συγκεκριμένο πλοίο σε ένα συγκεκριμένο λιμάνι), αυτό δεν γίνεται. Για αυτόν τον λόγο, θα επιχειρήσουμε να συλλέξουμε τα δεδομένα από 4 διαφορετικές πηγές τέτοιου τύπου και να τις συγκρίνουμε. Ένα σημείο που πρέπει να επισημάνουμε είναι πως, αφού προέρχονται από διαφορετικές πηγές, τα δεδομένα θα έχουν διαφορετικές μορφές (π.χ. timestamp, ώρα Local Time, ώρα UTC) και για αυτόν τον λόγο θα πρέπει να γίνει κάποια επεξεργασία προκειμένου να ομοιογενοποιηθούν. Επίσης, πολύ σημαντικό βήμα στην διαδικασία αυτή είναι η αφαίρεση διπλότυπων εγγραφών. Για την αποθήκευση των δεδομένων θα χρησιμοποιηθεί μια βάση δεδομένων NoSQL για την βέλτιστη απόκριση της εφαρμογής. Αφού τα δεδομένα έχουν ομοιογενοποιηθεί και αποθηκευθεί στην βάση δεδομένων, θα τα χρησιμοποιήσουμε για την ανάπτυξη τεχνικών data fusion.

Συλλογή δεδομένων

Με βάση το παραπάνω σενάριο χρήσης, επιλέξαμε να συλλέξουμε τις εξής πληροφορίες: *ώρες αφίξεων και αναχωρήσεων όλων των πλοίων σε ένα συγκεκριμένο λιμάνι για την περίοδο μιας ημέρας.* Το λιμάνι που επιλέξαμε για αυτόν τον σκοπό είναι το λιμάνι της Σιγκαπούρης, ένα από τα πιο πολυσύχναστα λιμάνια του κόσμου, προκειμένου να έχουμε και το ανάλογο πλήθος αφίξεων και αναχωρήσεων. Η ημέρα που επιλέξαμε για την συλλογή των δεδομένων ήταν η 20/03/2019. Οι πηγές που επιλέξαμε για να συλλέξουμε τα δεδομένα είναι οι εξής:

- **MyShipTracking**

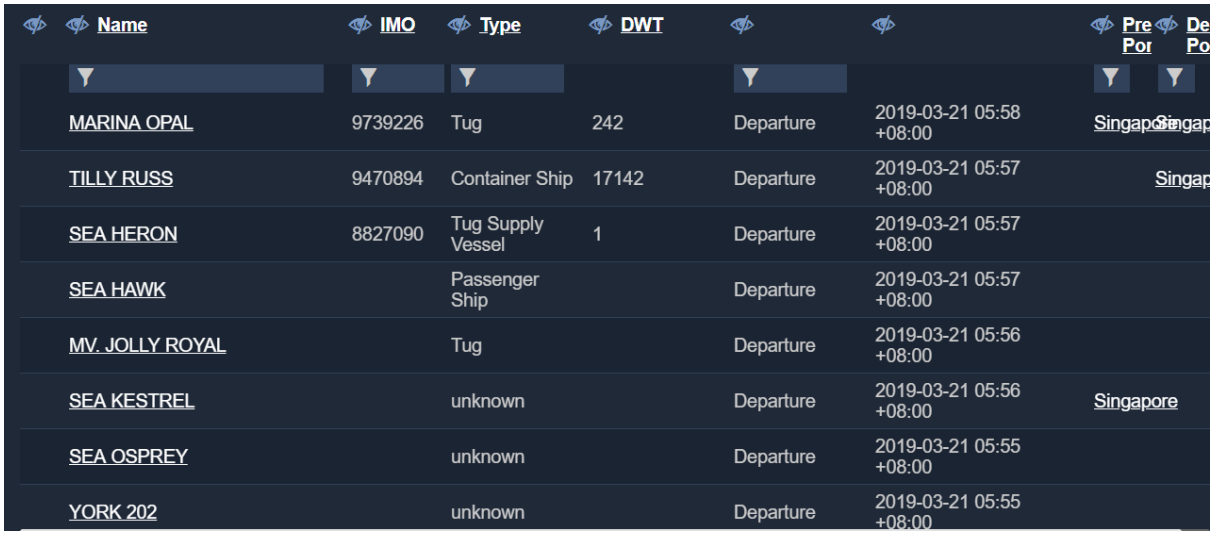
  Στην πρώτη πηγή, το άθροισμα των εγγραφών που συλλέχθηκαν ήταν **3150**. Τα πεδία της κάθε εγγραφής ήταν τα εξής: *είδος γεγονότος, ώρα και ημερομηνία, όνομα λιμανιού, όνομα πλοίου.* Παρακάτω φαίνεται μια εικόνα των εγγραφών όπως φαίνονται στην πηγή:



Event	Time (MT)	Port	Vessel
Arrival	2019-03-21 **21:57**	SINGAPORE	PILOT GP03 [SG]
Departure	2019-03-21 **21:57**	SINGAPORE	POSH [SG]
Departure	2019-03-21 **21:55**	SINGAPORE	MAERSK CALABAR [SG]
Arrival	2019-03-21 **21:54**	SINGAPORE	YORK 86 [SG]
Departure	2019-03-21 **21:54**	SINGAPORE	JOLLY RAMONA [SG]
Departure	2019-03-21 **21:53**	SINGAPORE	OPL 22 [SG]
Departure	2019-03-21 **21:51**	SINGAPORE	KMTC QINGDAO [SG]
Arrival	2019-03-21 **21:51**	SINGAPORE	GP 58 [SG]
Arrival	2019-03-21 **21:49**	SINGAPORE	GP02 [SG]
Departure	2019-03-21 **21:49**	SINGAPORE	KST SUMMIT [SG]
Arrival	2019-03-21 **21:48**	SINGAPORE	ASL MERCURY [SG]

Ο τρόπος συλλογής των δεδομένων από την συγκεκριμένη σελίδα έγινε με την υλοποίηση ενός parser σε Python με την χρήση της βιβλιοθήκης scrapy. Η βιβλιοθήκη scrapy δίνει την δυνατότητα ορισμού ενός URL, το οποίο θα γίνει parse. Επίσης, στον parser γίνεται η δήλωση των στοιχείων που θα γίνουν parse (με την χρήση κάποιων html/css selectors) από την σελίδα και τη μορφή (δημιουργία αντικειμένων με τις επιθυμητές ιδιότητες που θα ορίσουμε) την οποία θα έχουν στην έξοδο. Η έξοδος μπορεί να είναι είτε η κονσόλα είτε κάποιο αρχείο. Στην περίπτωσή μας η έξοδος ήταν ένα αρχείο .json.

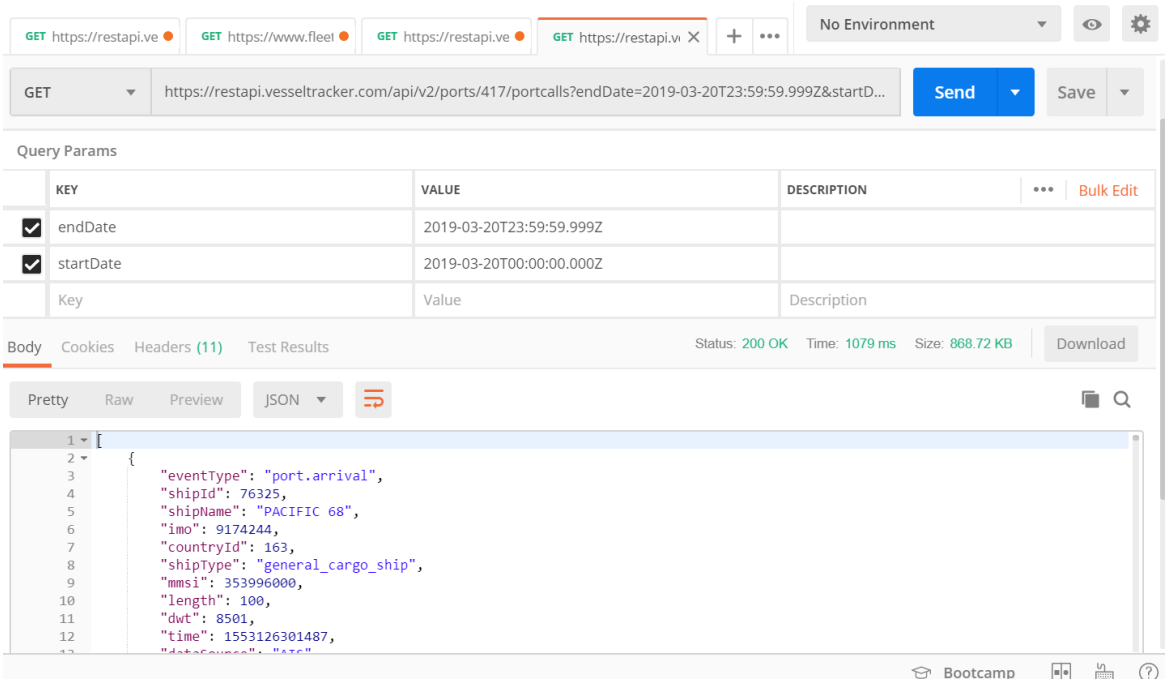
- **VesselTracker**
  Στην πηγή αυτή, το σύνολο των αφίξεων και αναχωρήσεων από το λιμάνι της Σιγκαπούρης ήταν **2902** εγγραφές. Η κάθε εγγραφή περιέχει τις εξής πληροφορίες: *είδος γεγονότος, id πλοίου, όνομα πλοίου, IMO πλοίου, id χώρας, τύπος πλοίου, MMSI πλοίου, μήκος πλοίου, DWT, timestamp γεγονότος, πηγή προέλευσης δεδομένων, id λιμανιού προορισμού, όνομα λιμανιού προορισμού, AIS προορισμός, id προηγούμενου λιμανιού, όνομα προηγούμενου λιμανιού.* Παρακάτω φαίνεται μια εικόνα των πληροφοριών όπως εμφανίζεται στην πηγή:



Name	IMO	Type	DWT			Pre Por	De Po
MARINA OPAL	9739226	Tug	242	Departure	2019-03-21 05:58 +08:00	Singapore	Singap
TILLY RUSS	9470894	Container Ship	17142	Departure	2019-03-21 05:57 +08:00		Singap
SEA HERON	8827090	Tug Supply Vessel	1	Departure	2019-03-21 05:57 +08:00		
SEA HAWK		Passenger Ship		Departure	2019-03-21 05:57 +08:00		
MV. JOLLY ROYAL		Tug		Departure	2019-03-21 05:56 +08:00		
SEA KESTREL		unknown		Departure	2019-03-21 05:56 +08:00	Singapore	
SEA OSPREY		unknown		Departure	2019-03-21 05:55 +08:00		
YORK 202		unknown		Departure	2019-03-21 05:55 +08:00		

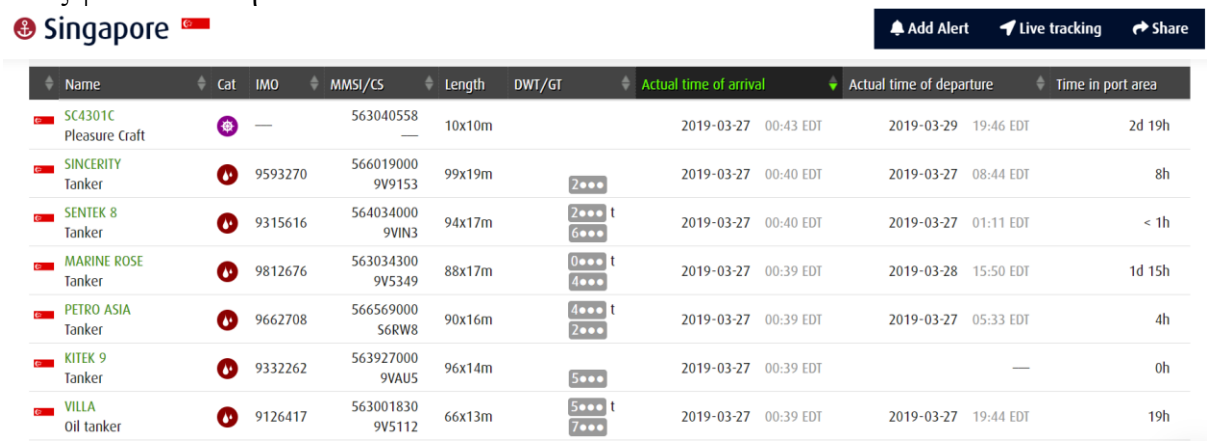
Ο τρόπος που επιλέξαμε για να συλλέξουμε τα δεδομένα από την συγκεκριμένη πηγή ήταν να αναπαραστήσουμε το get αίτημα προς το API της σελίδας μέσω της εφαρμογής Postman. Το Postman είναι ένα εργαλείο που χρησιμοποιείται πολύ συχνά από web developers για να αναπαραστήσουν τις κλήσεις προς το API τους και, στη συνέχεια, να επαληθεύσουν τα αποτελέσματα που επιστρέφει η κάθε κλήση. Εμείς το χρησιμοποιήσαμε για την κλήση προς το API του VesselTracker και αποθηκεύσαμε τα

δεδομένα σε ένα αρχείο .json. Προτιμήσαμε αυτόν τον τρόπο για λόγους ευκολίας καθώς δεν χρειαζόταν η ανάπτυξη ενός parser από την στιγμή που τα δεδομένα είναι διαθέσιμα με αυτόν τον τρόπο. Επίσης, το authentication στη σελίδα καθιστά ελαφρώς πιο δύσκολη την υλοποίηση ενός parser σε σχέση με άλλες πηγές. Παρακάτω φαίνεται μια εικόνα από την εφαρμογή Postman με το αίτημα get, τις παραμέτρους αλλά και την μορφή των αποτελεσμάτων:



- **FleetMon**

  Στην συγκεκριμένη πηγή το άθροισμα των εγγραφών έφτασε τις **4045** εγγραφές για το ίδιο λιμάνι (Σιγκαπούρη) την ίδια ημερομηνία (20/03/2019), γεγονός που επαληθεύει την υπόθεση πως πηγές που παρέχουν τα ίδια δεδομένα διαφέρουν μεταξύ τους όχι μόνο στις τιμές των διάφορων εγγραφών αλλά και στο πλήθος των εγγραφών αυτών. Η κάθε εγγραφή αποτελούνταν από τα εξής πεδία: *σημαία πλοίου, τον χρόνο που πέρασε το πλοίο στο λιμάνι, πότε εμφανίστηκε για πρώτη φορά στο λιμάνι, όνομα του πλοίου, πότε εμφανίστηκε για τελευταία φορά το πλοίο στο λιμάνι, IMO πλοίου, μήκος πλοίου, τύπος πλοίου, MMSI πλοίου, CS πλοίου.* Παρακάτω φαίνεται μια εικόνα των εγγραφών όπως φαίνονται στην σελίδα:



Ο τρόπος που επιλέξαμε για να συλλέξουμε τα δεδομένα μοιάζει πολύ με τον προηγούμενο, με την διαφορά πως η συγκεκριμένη πηγή μας περιόριζε στα 50 αποτελέσματα ανά αίτημα get. Προκειμένου, λοιπόν, να αποθηκεύσουμε τα 4045 αποτελέσματα χωρισμένα σε ομάδες των 50 εγγραφών έπρεπε να αυτοματοποιήσουμε την διαδικασία. Για αυτόν τον λόγο υλοποιήσαμε ένα script σε Python που υλοποιεί ένα αίτημα get για την κάθε 50άδα των αποτελεσμάτων με τις παραμέτρους που χρειάζεται το API (π.χ. λιμάνι, ημερομηνία κλπ.) και τα αντίστοιχα απαιτούμενα headers. Στη συνέχεια, το script αποθηκεύει τα αποτελέσματα του API σε ένα αρχείο .json. Σε δεύτερη φάση, έπρεπε να καθαρίσουμε και να επεξεργαστούμε τα δεδομένα ώστε να τους δώσουμε μια μορφή πιο κοντά στα δεδομένα των υπόλοιπων πηγών. Αυτό έγινε με την υλοποίηση ενός ακόμα script σε Python και την χρησιμοποίηση της βιβλιοθήκης BeautifulSoup για τον καθαρισμό των δεδομένων από html tags.

- **MarineTraffic**
  Το πρόβλημα με την τελευταία πηγή δεδομένων μας είναι πως προσφέρει μόνο 500 εγγραφές αφίξεων/αναχωρήσεων από το σύνολο των εγγραφών. Ο τρόπος που προσπαθήσαμε να έχουμε πρόσβαση στα δεδομένα είναι με την αποστολή ενός email για την παροχή δεδομένων του συγκεκριμένου λιμανιού την συγκεκριμένη ημερομηνία που εξετάζουμε. Τελικώς, δεν μπορέσαμε να έχουμε πρόσβαση σε αυτά τα δεδομένα καθώς το κόστος που μας έθεσε η εταιρεία ήταν απαγορευτικό. Δυστυχώς, δεν βρήκαμε κάποια εναλλακτική λύση για την πρόσβαση στα δεδομένα της πηγής αυτής και τα 500 αποτελέσματα έγιναν μη προσβάσιμα μετά απο 3 ημέρες (εάν δεν έχεις δικαιώματα premium χρήσης τα ιστορικά δεδομένα γίνονται μη προσβάσιμα μετά από 3 ημέρες).

Επεξεργασία και κανονικοποίηση δεδομένων

Στο πρώτο μέρος είχαμε αναφέρει πως ένα πολύ σημαντικό βήμα στην ανάπτυξη της εργασίας είναι η επεξεργασία και η κανονικοποίηση των δεδομένων πριν από την χρησιμοποίησή τους. Η μεγάλη ανομοιογένεια μεταξύ τους τριών διαφορετικών πηγών ήταν ένα πρόβλημα που κληθήκαμε να λύσουμε. Μερικά από τα βήματα επεξεργασίας και κανονικοποίησης των δεδομένων που υλοποιήσαμε κατά την διάρκεια της εργασίας ήταν τα εξής:

- Τα ονόματα των πεδίων που περιείχαν τις ίδιες πληροφορίες μετονομάστηκαν σε ένα κοινό όνομα (και στα 3 σετ δεδομένων). Για παράδειγμα, το ένα dataset ονόμαζε το λιμάνι άφιξης «destination_port_name» ενώ κάποιο άλλο «destPortName», οπότε και τα δύο μετονομάστηκαν σε «destination_port_name» καθώς είναι το πιο προσδιοριστικό όνομα από τα δύο.
- Αφαιρέθηκαν τα πεδία από τα σετ δεδομένων που είχαν μόνο κενές τιμές επειδή σε αυτές τις πληροφορίες είχαν πρόσβαση μόνο premium χρήστες. Πιο συγκεκριμένα, στο dataset του FleetMon, η πληροφορία DWT ήταν παντού κενή για τον λόγο που προαναφέρθηκε, οπότε αφαιρέθηκε εξ ολοκλήρου από το dataset.
- Οι τιμές των πεδίων που περιείχαν την ίδια πληροφορία αλλά την απεικόνιζαν με λίγο διαφορετική μορφή, ομογενοποιήθηκαν σε μια κοινή μορφή σε όλα τα datasets. Μια τέτοιου είδους αλλαγή ήταν το πεδίο «event» στο dataset του VesselTracker που αναφερόταν ως «port.arrival» ενώ στα δεδομένα του MyShipTracking ως «Arrival». Η απόφαση που πήραμε ήταν να μετονομαστούν οι τιμές του VesselTracker σε «Arrival» και «Departure» από «port.arrival» και «port.departure» αντίστοιχα.

- Παρόμοιας φύσης προβλήματα ήταν και τα πεζά/κεφαλαία γράμματα σε ονόματα και τιμές πεδίων. Σε πολλά ονόματα και τιμές πεδίων διέφεραν τα πεζά και τα κεφαλαία γράμματα και για αυτόν τον λόγο τα μετονομάσαμε με μια κοινή πολιτική και στα 3 σετ δεδομένων.
- Η μετατροπή των τιμών ημερομηνιών από τύπο timestamp σε τύπο datetime. Συγκεκριμένα, το σετ δεδομένων που είχαμε εξωρύξει από το VesselTracker παρείχε την άφιξη/αναχώρηση ενός πλοίου σε τύπο timestamp (unix time). Στο συγκεκριμένο dataset κάναμε τις απαραίτητες μετατροπές προκειμένου να μετατρέψουμε τις τιμές του πεδίου αυτού σε τύπο Datetime ώστε να έχει την ίδια μορφή με τα υπόλοιπα δεδομένα.
- Αφού υλοποιήθηκαν οι παραπάνω μετατροπές και τα σετ δεδομένων ήρθαν σε ένα επιθυμητό σημείο ομογενοποίησης, εισάχθηκαν σε μια βάση δεδομένων NoSQL (MongoDB). Αφού δημιουργήσαμε μια βάση δεδομένων (localdb) και μια συλλογή (data_fusion), εισάγαμε τα δεδομένα που είχαμε διαθέσιμα με την προσθήκη δύο επιπλέον πεδίων σε κάθε dataset. Τα πεδία αυτά (source_id και source_name) προστέθηκαν για την εύκολη αναζήτηση και υλοποίηση queries που έχουν ως στόχο συγκεκριμένο σετ δεδομένων.
- Καθώς προχωρούσε η εργασία, μας γεννήθηκε η ανάγκη για μια επιπλέον μετατροπή η οποία, εξ αρχής, δεν φάνταζε τόσο απαγορευτική στην ανάπτυξη της εργασίας. Το dataset του FleetMon δεν είχε ένα πεδίο αλλά δύο για την άφιξη/αναχώρηση του κάθε πλοίου. Στο ένα αναφερόταν η ώρα άφιξης και στο άλλο η ώρα αναχώρησης χωρίς να αναφέρεται το είδος του event (άφιξη/αναχώρηση). Για αυτόν τον λόγο, δημιουργήσαμε ένα νέο dataset με βάση το παλιό. Για κάθε εγγραφή του παλιού dataset δημιουργήσαμε δύο νέες καινούργιες:
    1. Η μία είχε στο (νέο) πεδίο datetime την ημερομηνία άφιξης του πλοίου και στο (νέο) πεδίο event την τιμή «Arrival».
    2. Η άλλη είχε στο (νέο) πεδίο datetime την ημερομηνία αναχώρησης του πλοίου και στο (νέο) πεδίο event την τιμή «Departure».
- Με την διαδικασία αυτή, οι εγγραφές πολλαπλασιάστηκαν και πολλές είχαν ημερομηνίες πριν ή μετά την 20/03/2019 που ήταν η ημέρα που εξετάζαμε. Για αυτό επιλέξαμε να αφαιρέσουμε όσες εγγραφές είχαν στο πεδίο «datetime» ημερομηνία διαφορετική της 20/03/2019.
- Το νέο σύνολο δεδομένων αποθηκεύτηκε σε μια καινούργια συλλογή της MongoDB (data_fusion_alt).

Εξαγωγή gold standard

Σύμφωνα με την βιβλιογραφία, το gold standard είναι ένα σετ δεδομένων όπου οι εγγραφές του είναι κοινά αποδεκτές ως αληθείς. Συνήθως, αυτό το gold standard εξάγεται από επίσημες πηγές δεδομένων (π.χ. πρακτικά από αεροδρόμια, λιμάνια ή τις επίσημες ιστοσελίδες εταιρειών) για την σύγκριση των αποτελεσμάτων από τις τεχνικές συγχώνευσης. Δυστυχώς, στη δική μας περίπτωση δεν υπάρχει κάποια επίσημη αρχή που να μας παρέχει ένα τέτοιο gold standard. Για παράδειγμα, η επίσημη ιστοσελίδα του λιμανιού της Σιγκαπούρης δεν προσέφερε τέτοιου είδους δεδομένα. Η διαδικασία εξαγωγής του gold standard με την χρήση των δεδομένων που είχαμε στα χέρια μας περιγράφεται παρακάτω:

- Αφού φέρουμε τα δεδομένα δύο πηγών από την βάση δεδομένων, για κάθε εγγραφή της πρώτης πηγής ψάχνουμε για υποψήφιες κοινές εγγραφές στην δεύτερη πηγή. Οι κοινές εγγραφές προσδιορίζονται από τον ίδιο τύπο event και από το ίδιο όνομα πλοίου. Στην συνέχεια, οι υποψήφιες κοινές εγγραφές φιλτράρονται με βάση το πεδίο datetime. Οποιαδήποτε εγγραφή με διαφορά μισής ώρας (παίρνουμε υπόψιν το περιθώριο λάθους), απορρίπτεται. Εάν η διαφορά ώρας διαφέρει, αποθηκεύεται η μέση τιμή αυτής. Οι κοινές εγγραφές πρώτης και δεύτερης πηγής δεδομένων κρατώνται για την συνέχεια.
- Οι εγγραφές που κρατήθηκαν από το προηγούμενο βήμα, συγκρίνονται με τις εγγραφές της τρίτης πηγής. Χρησιμοποιώντας τις ίδιες παραμέτρους (ίδιο τύπο event, ίδιο όνομα πλοίου, περιθώριο λάθους 30 λεπτών), εξάγουμε τις τελικές κοινές εγγραφές. Με τον ίδιο τρόπο, αποθηκεύουμε την μέση τιμή των διαφορών ώρας των δύο κοινών εγγραφών σε περίπτωση διαφοροποίησης.
- Οι τελικές κοινές εγγραφές των τριών πηγών αποθηκεύονται στη βάση δεδομένων σε νέα συλλογή (gold_standard) και αποτελείται από 109 εγγραφές.

Υλοποίηση τεχνικών συγχώνευσης

Σύμφωνα με την βιβλιογραφία, δεν υπάρχει ένας μοναδικός τρόπος για την συγχώνευση δεδομένων. Οι αλγόριθμοι που έχουν υλοποιηθεί είναι πολλοί και διαφέρουν ως προς τον τρόπο που αναζητούν την αλήθεια μεταξύ των δεδομένων. Παρακάτω φαίνεται ένας πίνακας με κάποιους από τους αλγόριθμους ανά κατηγορία:

Category	Method
Baseline	Vote
Web-link based	HUB AvgLog Invest PooledInvest
IR based	2-Estimates 3-Estimates Cosine
Bayesian based	TruthFinder AccuPr PopAccu AccuSim AccuFormat
Copying affected	AccuCopy

Οι δύο αλγόριθμοι που υλοποιήσαμε για την εύρεση αλήθειας στα δικά μας σετ δεδομένων αναλύονται παρακάτω:

- **Vote**

Η συγκεκριμένη υλοποίηση είναι αρκετά απλή και αποτελεσματική. Βασίζεται στην ιδέα πως αν μια εγγραφή παρέχεται από την πλειοψηφία των πηγών δεδομένων, τότε μπορεί να θεωρηθεί και αληθής. Το πρόβλημα στο δικό μας σετ δεδομένων είναι πως οι εγγραφές είναι (σχεδόν) πάντα διαφορετικές μεταξύ τους, καθώς όταν μιλάμε και για την ίδια ακριβώς εγγραφή, η ώρα που συνέβη το γεγονός (αναχώρηση/άφιξη) πλοίου είναι ποικίλλει ανά πηγή δεδομένων. Η ποικιλία αυτή μας περιορίζει στο να μην μπορούμε να υλοποιήσουμε ακριβώς

αυτόν τον αλγόριθμο αλλά να τον τροποποιήσουμε με κάποια περιθώρια λάθους. Με την χρησιμοποίηση του περιθώριου λάθους των 30 λεπτών υλοποιήσαμε την εξαγωγή του gold standard όπως περιγράφηκε νωρίτερα. Στην συγκεκριμένη περίπτωση υλοποιήσαμε μια λύση με διαφορετική οπτική. Κατά την αναζήτηση εργαλείων για μια τέτοια λύση, βρήκαμε μια βιβλιοθήκη της python που ονομάζεται dedupe (https://github.com/dedupeio/dedupe). Η συγκεκριμένη βιβλιοθήκη χρησιμοποιείται για την αφαίρεση διπλότυπων εγγραφών και εντοπισμό οντοτήτων. Η υλοποίησή μας βασίστηκε στην βιβλιοθήκη dedupe και περιγράφεται παρακάτω:

- Αρχικά, φέρνουμε όλες τις εγγραφές του dataset από την βάση δεδομένων μας.
- Αφού κάνουμε κάποιον στοιχειώδη καθαρισμό των δεδομένων (π.χ. encoding-decoding utf-8, αφαίρεση σημείων στίξης), αφαιρούμε την ημέρα από το πεδίο της ημερομηνίας. Η συγκεκριμένη αφαίρεση έγινε διότι η ημέρα είναι ίδια σε όλες τις εγγραφές του dataset, οπότε δεν παρέχει κάποια πληροφορία μεγάλης αξίας. Επίσης, κατά τα πειράματα παρατηρήσαμε πως ο αλγόριθμος αντιστοίχιζε εγγραφές που δεν έμοιαζαν σε βασικά πεδία (όπως το όνομα πλοίου και το είδος γεγονότος) λόγω της ομοιότητας στο πεδίο ημερομηνίας.
- Στη συνέχεια, το dedupe χρειάζεται την δήλωση των πεδίων των εγγραφών και τις παραμέτρους που θα λάβει υπόψιν κατά την σύγκριση. Η δήλωσή μας περιείχε τα πεδία με τις παραμέτρους: *event: Exact, vessel_name: Exact, datetime: String*. Η παράμετρος Exact σημαίνει πως για την αντιστοίχιση δύο εγγραφών πρέπει το πεδίο event να είναι ακριβώς το ίδιο. Το πεδίο datetime το δηλώσαμε ως String για την σύγκρισή του ως String και όχι ως datetime γιατί πλέον δεν περιείχε ολόκληρη την ημερομηνία αλλά μόνο την ώρα του γεγονότος.
- Επίσης, ένα πολύ δυνατό σημείο του dedupe είναι πως καθώς εκπαιδεύεται από τα δεδομένα, εμφανίζει στο command line αντιστοιχίσεις με χαμηλή βεβαιότητα για να τις κατηγοριοποιήσουμε ως αληθείς ή ψευδείς. Με την διαδικασία αυτή ενισχύεται η εκμάθησή του και έχει καλύτερα αποτελέσματα. Τα παραδείγματα αυτά αποθηκεύονται για μελλοντική χρήση.
- Η επόμενη φάση του dedupe είναι το clustering. Δημιουργεί ένα cluster για κάθε αντιστοίχισή του. Στη δική μας περίπτωση το κάθε cluster περιείχε από μία έως τρεις εγγραφές που μοιάζουν μεταξύ τους (όσα και τα σετ δεδομένων). Για την κάθε πρόβλεψή του, παρέχει και ένα ποσοστό βεβαιότητας.
- Τέλος, δημιουργούμε ένα αρχείο csv με τα clusters που έχουν δύο ή περισσότερες εγγραφές και τα εμφανίζουμε ταξινομημένα ως προς το cluster id.
- Καταλήγοντας, αφού τα clusters περιέχουν δύο ή περισσότερες εγγραφές που αναφέρονται στην ίδια εγγραφή ανά σετ δεδομένων, πετυχαίνουμε το ποσοστό της πλειοψηφίας ανάμεσα στις πηγές δεδομένων για να θεωρήσουμε μια εγγραφή ως αληθή. Ως αποτέλεσμα του σκεπτικού αυτού, το αρχείο csv που παράχθηκε περιέχει όλες τις αληθείς εγγραφές των dataset με το αντίστοιχο ποσοστό βεβαιότητας.

- **TruthFinder**

Ο αλγόριθμος αυτός βασίζεται στην ιδέα πως ένα γεγονός είναι πιο πιθανό να είναι αληθές όταν παρέχεται από μια αξιόπιστη πηγή δεδομένων. Αν και αυτή η υπόθεση περιέχει αλληλεξάρτηση, ο αλγόριθμος κάνει εκτίμηση για την αξιοπιστία και την εμπιστοσύνη της πηγής. Κατά την έρευνα για την υλοποίηση του αλγορίθμου, βρήκαμε μια βιβλιοθήκη της python που κάνει αυτή την υλοποίηση (https://github.com/IshitaTakeshi/TruthFinder). Τα βήματα της υλοποίησης με βάση την παραπάνω βιβλιοθήκη φαίνονται παρακάτω:

  - Αρχικά, φέρνουμε τα δεδομένα από όλες τις πηγές δεδομένων από την βάση δεδομένων.
  - Αφαιρούμε από το πεδίο datetime την ημέρα και αφήνουμε την ώρα για τους ίδιους λόγους που αναφέραμε στην προηγούμενη υλοποίηση.
  - Χρησιμοποιούμε έναν TfidfVectorizer πάνω στο πεδίο datetime (εναλλακτικά το πεδίο vessel_name)
  - Εκπαιδεύουμε το μοντέλο με τον αλγόριθμο TruthFinder με τις default παραμέτρους (dampening_factor = 0.3, influence_related = 0.5) όπως αναφέρονται στην αντίστοιχη βιβλιογραφία (Yin, Xiaoxin, Jiawei Han, and S. Yu Philip. "Truth discovery with multiple conflicting information providers on the web." IEEE Transactions on Knowledge and Data Engineering 20.6 (2008): 796-808) .
  - Τα αποτελέσματα αποθηκεύονται σε ένα αρχείο csv με το ποσοστό βεβαιότητας της κάθε εγγραφής να είναι αληθής, αλλά και το ποσοστό εμπιστοσύνης της κάθε πηγής δεδομένων.

Μελλοντικές βελτιώσεις

Από την περιήγησή μας στον χώρο των τεχνικών συγχώνευσης, το πεδίο έρευνας μοιάζει μεγάλο και αρκετά ενδιαφέρον. Ως εκ τούτου, πολλές βελτιώσεις θα μπορούσαν να επεκτείνουν την εργασία μας. Για παράδειγμα, η βιβλιοθήκη scitkit-fusion μας φάνηκε πολύ ενδιαφέρουσα, αλλά προκειμένου να την χρησιμοποιήσουμε θέλει αρκετά διαφορετική δομή στα datasets. Επίσης, η υλοποίηση επιπλέον αλγορίθμων συγχώνευσης δεδομένων θα ήταν μια σημαντική βελτίωση της παρούσας εργασίας.