Université de Montréal

Detection of Recurring Defects in Airline Incident Reports

David Alfonso Hermelo, Ilan Elbaz, Tianjian Gao, Fabrizio Gotti, David Kletz, Philippe Langlais, Vincent Letard, Lucas Pagès, Frédéric Piedboeuf, Helen Samara Dos Santos, Tina Yang-Zhou

Contact :
Philippe Langlais +1 514 343 61 11 ext: 47494
RALI/DIRO felipe@iro.umontreal.ca
Université de Montréal http://www.iro.umontreal.ca/~felipe/

Date : 27/09/2020

Disclaimer

This report presents the work conducted during the Tenth Montreal Problem Solving Workshop for the problem submitted by Air Canada, with the much appreciated assistance of Keith Dugas and Nicholas Popovic from Air Canada. The data provided by Air Canada is extremely complex, therefore, this report, which expresses the view of the persons involved in developing solutions, might be inconclusive in a number of ways.

1 Problem Description

This section is slightly adapted from the official description provided by Air Canada.

1.1 Context

Transport Canada mandates per the Canadian Aviation Regulation (CAR 706.05 and STD 726.05) that an Air Operator Certificate (AOC) holder must include in its maintenance control system procedures for recording and rectification of defects, including the identification of recurring defects. Defects can be classified into two distinct categories : Safety/Airworthy related defects and Non-Safety/Airworthy related defects.

1. Safety/Airworthy defects are covered under the Minimum Equipment List (MEL), a document approved by the Minister pursuant to CAR 605.07 (3) that authorizes an operator to operate an aircraft with aircraft equipment that is inoperative under the conditions specified therein ; the MEL may specify that certain equipment must be operative. Each MEL has its own unique identifier and each MEL-type defect has an Air Transport Association (ATA) technical classification.
2. Non-MEL defects are defects that are raised for items that are not Safety/Airworthy related, such as scratches or gauges on surfaces, among many more designated classes. Each defect has an Air Transport Association (ATA) technical classification.

Recurring defects are the focus of the current problem.

1.2 The Problem

The ATA defect classification is carried out manually in real time by the engineer, flight attendant, or pilot on board. The ATA classification tables have generic identifiers such as 25-00-00, which is labelled “Cabin General.” This means that any defect that occurs in the cabin can technically be classified as such, making the effort of tracking recurring non-MEL defects onerous. Since there are hundreds of different combinations in the ATA classification categories and thousands of employees reporting defects, the probability of defects being reported with the required ATA classification standard (apart

from the generic classification) is low. As a result, the ATA category numbers cannot be considered as a unique identifier for the purpose of tracking recurring non-MEL defects. Another problem is the wide presence of synonyms and acronyms while describing defects : for example, "Nose Landing Gear," "Nose Gear," or even "NLG" may refer to the same type of defect. Consequently, the classification has had to be carried out manually on the basis of the defect descriptions, which is again time consuming and arduous.

1.3 Desired Solution

Air Canada Maintenance wishes to detect recurring defects automatically in a way that meets and exceeds Transport Canada requirements for both MEL and Non-MEL defects. Defects are considered recurring if a failure mode is repeated 3 times, on an aircraft, within 15 flight segments of a previous repair made with respect to that failure mode. For this workshop, the goal was slightly reframed and we strived to automatically detect recurring intervals of 3 defects in 30 days, 4 in 40 days and 5 in 50 days. Additionally, Air Canada desires to re-label reports with ATA Chapter/Section labels in a more exact way, in an effort to sanitize the dataset. To carry out these tasks, Air Canada provided a large dataset of defect reports, including MEL, textual defect description, ATA labels, aircraft tail number, etc. Auxiliary data was also be provided, including reference tables of acronyms and synonyms used in the airline industry.

The next sections are organized as follows. We present in Section 2 the Air Canada dataset that we worked on. We describe in Section 3 the normalization techniques implemented. In Section 4, we report the experiments conducted for classifying a defect into its ATA code, while in Section 5 we relate our efforts to detect recurrent defects. Finally, we discuss in Section 6 possible continuations to the work conducted for the workshop.

2 Data

Air Canada provided the team with a corpus of logbooks of aircraft defects reported by different Air Canada employees (technicians, cabin crews, pilots) from January 2018 to December 2019. We had access to those defects as a spreadsheet with various fields (48 in total) of different data types describing each defect. Prior to the workshop, Fabrizio Gotti sanitized the data and crea-

Type	Description	Timestamp	Chapter	Section	MEL	Resolution
C	C/M POS 104 NEEDS RELAMPING FOR "HOT PLATE ON".	2018-01-04 02:13:00	25	0		RELAMPED.
E	GALLEY, WALL PANEL, LAMINATES, SCRATCHED, AT LOCATION:AFT, AT POSITION:FELL OFF THE WALL	2019-01-17 15:03:00	25	30		REPAIRED OK FOR SERVICE.
L	ON MORNING POWER UP HMU ADVISORY MSG APPEARED.	2019-12-28 13:13:00	46	0	491753	TRIAGE

FIGURE 1 – Excerpt of the dataset of defect reports provided by Air Canada over the period 2018-2019. Each defect is composed of 48 fields, among which a type indicating the logbook type of the defect, its description (a short text), the time it was reported, as well as its ATA code (a chapter and a section which together refer to a predefined node in the ATA taxonomy of defects).

ted a GitHub repository containing sample Python scripts illustrating how to load the data and perform a few simple operations. This has greatly helped the team start digging into the enormous data. Keith Dugas also provided many explanations on the data fields in the weeks leading to the workshop. These explanations led to additional—and valuable—documentation made available to the workshop participants.

Due to the time constraint and the difficulty of understanding all the intricacies of the data fields, we focused on a small subset of the features :

defect_type describes the origin of the defect report. **L** : The aircraft defect logbook is used to record any technical defects of the aircraft as relates to the technical dispatch of the aircraft and/or any safety of flight items. These items reported from the flight deck are more serious. **C and E** : Cabin defect logbook used to record defects to the status of the passenger cabin. **E** indicates electronic transcript of a paper logbook. **E** description is generated automatically. These codes derive their names from the following explanation : C : cabin defect logbook, E : electronic (transcript) cabin defect logbook, L : aircraft defect log book.

L-defects are considered more accurate than C defects and type E defects are considered very reliable. Unfortunately, E-type defects are overall rare in the corpus (0.3%), the majority of defects (60.1%) being of type C.

defect_description a short textual description of the defect

ac Aircraft code (aircraft manufacturer and series, obfuscated). This uniquely designates a particular aircraft in the fleet. This code designates the particular plane the report was created for.

reported_datetime date and time of the report

chapter first level classification of the defect according to ATA code

section secondary classification according to ATA code ; chapter and section define what is referred to as the ATA code hereafter

recurrent the clustering output of a system (TRAX) deployed at Air Canada trying to detect recurrent defects

resolution_description a short description of the defect resolution written by the maintenance technician/engineer

An excerpt of the corpus is shown in Figure 1. The defect descriptions are in uppercase and contain jargon including acronyms, terms, seat numbers etc. Evidently, the descriptions are typically short, as are the resolution descriptions.

The ATA label (chapter-section) provides important information characterizing a defect, even if it is only partially reliable, as stated in the official description provided by Air Canada. Therefore, we report in Figure 2 the distribution of defect chapter labels as well as the distribution of sections within a chapter. We observe that some chapters appear much more frequently than the others ; chapter 25 which indicates cabin incidents being the most frequent. Also, the section distribution within a chapter is non-uniform. Among the chapter-25 defects, section 20 is the most frequent. The ATA code 25-20 corresponds to issues with passenger convenience items, non-essential equipment and furnishings, flight attendant seats, passenger seats, and non-essential storage equipment.[1] We also observe that an important number of defects (18.8% of the full dataset) is associated with section 0, a non-informative section used as a "catch-all" section when the maintenance person fails to identify the section specific to the problem reported.

1. https://tc.canada.ca/en/aviation/aircraft-airworthiness/master-minimum-equipment-list-mmel/ata-25-equipment-furnishing

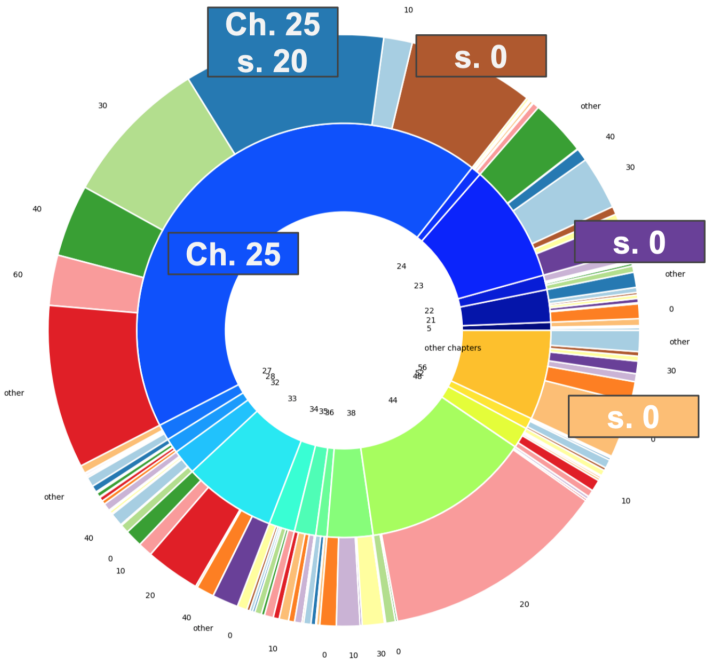

FIGURE 2 – Distribution of ATA codes (chapter-section) in the corpus provided. The inner circle of the pie chart indicates the chapter distribution, while the outer circle represents the section distribution. The most frequent ATA code is 25-20, characterizing issues with passenger convenience items, non-essential equipment and furnishings, flight attendant seats, passenger seats, and non-essential storage equipment.

2.1 Subsets

We partitioned the original data into 3 subsets in which the main characteristics are reported in Table 1 : FULL, containing 460k defects reported by various persons, and therefore containing a high level of noise ; TRAX gathering 47k defects that were concerned by the TRAX system in place at Air Canada[2]; and RELIABLE which contains 34K defects that are believed to be

2. The TRAX system identifies 14.2k clusters covering 47k of the defects in the full corpus, for an average cluster size of 3.3 defects per cluster. Initially, we thought that this was the ground truth for the clustering task, but we came to understand during the workshop that this isn't so. On the contrary, these clusters are only a rough cut, and are reviewed by humans after they are created by TRAX. This automated system clusters

the most reliable according to our understanding of the dataset.[3] We split along the aircraft boundaries each subset into a training, validation and test parts. The details of each dataset are reported in Table 1.

	#defects	#token types	avr. desc. length	% section-0 defects
		FULL		
train	380209	736	63.4 (11.7)	18.8
valid	33465	510	63.9 (11.6)	20.2
test	46920	521	64.2 (11.8)	19.8
		TRAX		
train	28309	363	85.1 (15.4)	26.6
valid	9436	304	85.1 (15.5)	26.8
test	9437	299	84.8 (15.3)	26.5
		RELIABLE		
train	29220	116	105.7 (18.9)	0.12
valid	2897	97	109.1 (18.9)	0.38
test	2317	92	100.9 (17.7)	0.22

TABLE 1 – Main characteristics of the datasets used to benchmark solutions. The number of defects refers to the total number of defect used in the dataset, the number of token types refer to the total number of different labels (chapter-section combination), the average description length is the average number of character in the defect description, and in parentheses the number of words following a simple space splitting, and finally the number of % of section-0 defects refers to the percent of defects that have 0 as a section, which we remove for training and testing.

defects based on very simple rules, most important of all is the equality of the ATA codes between two defects. TRAX also factors in the defect timestamp in order to create clusters of various levels of recurrence (1, 2, 3 depending on the timespan encompassed by a cluster)

3. We removed C-type defects because they are less reliable. For the other types, we replaced the chapter and section of the provided ATA code thanks to a mapping from the MEL code that was explained to us by Air Canada.

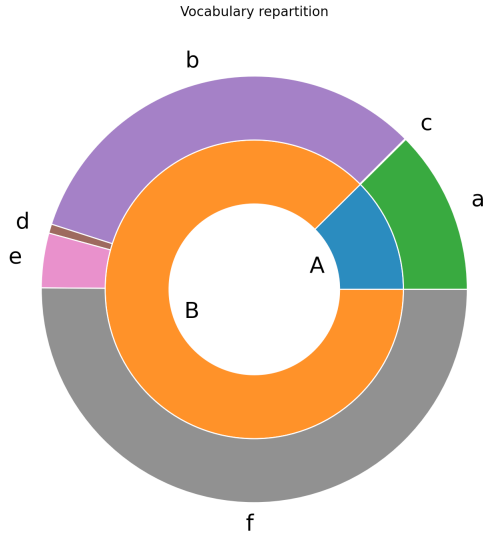

FIGURE 3 – Distribution of token types in the description field of the FULL dataset. The inner circle identifies two categories of token types : those listed in an in-house English lexicon (A, 12.5%) and others (B). The outer circle refines the distribution by distinguishing token types listed in dedicated resources as acronym (c), airport codes (d), abbreviations (e), and words with at least one digit (f). Thus, section (b) identifies token types unknown from our lists of tokens, while section (a) still represents the proportion of known token types in our English lexicon.

3 Data normalization

As is often the case with real data, we rapidly noticed a large number of words containing spelling mistakes, abbreviations, jargon or acronyms. To give a sense of the kind of noise[4], we report in the inner pie chart of Figure 3 the proportion of token types[5] that are listed in a in-house lexicon gathering

4. We call it noise with the perspective of a model, but the data has nothing wrong in it, it is simply the way it is !

5. We distinguish a token type (a word form) from its occurrences in a corpus. Token types are defined here according to the `word_tokenize` function from NLTK library.

370 107 English words (A) : only 12.5% of token types present in the defect descriptions are belonging to the lexicon, we call them known words. The vast majority (B) of token types are indeed unknown. The outer pie chart further refines the categorization of unknown words into acronyms (c), airport codes (d), abbreviations (e), and words containing at least one digit (f). For compiling those broad statistics, we had at our disposal a list of 5 328 airport codes (e.g. *SAP* for *SAN PEDRO SULA*), 12 288 abbreviations (e.g. *MONG* for *MONITORING*), and 2 188 acronyms (e.g. *ACFT* for *aircraft*).

Part of the team therefore spent some time investigating different normalization methods of such material, which we will report later on.

3.1 Acronym Detection

Even if a dedicated website has been prepared previous to the workshop, with all useful information listed — including a rather large list of acronyms — one member of the team[6] did investigate whether acronyms (e.g. *AVOD*) and their possible plain forms (e.g. *AUDIO/VIDEO ON DEMAND*) could be mined directly from the textual description of the defects.

We searched in all the descriptions the presence of a bracketed sequence of letters[7], then output the n preceding words as a context into which we searched for possible resolutions. Identifying candidate resolutions of an acronym can be done by aligning the letters in the acronyms with those of the context. Often, many alignments are possible. Therefore, we scored each alignment in order to favour those where the aligned letters in the context are the first letters of words. We output the m best scored resolutions for a given context, provided they receive a decent enough score. Since different defect descriptions use similar acronyms (possibly with different contexts), we get a distribution of acronyms and their resolutions.

This process is depicted in Figure 4 for the acronym *FAP* found 37 times in parentheses in the defect descriptions of the training material of the FULL dataset. On this dataset, with m and n set to 5 and 3 respectively, we identified 4 146 pairs of acronym/left context pairs, involving 558 different acronyms. Once resolved, this led to 665 acronym/resolution pairs, involving 202 acronyms (some acronyms may have different resolutions, as illustrated in

6. Contact person of the present report that feels ashamed not having noticing the already large acronym list available ...

7. We used a simple regular expression for this, insuring the sequence was at least 2 character long, and at most 5, two metaparameters that were not investigated.

input

▷ L2 DOOR LOW PRESSURE. (CHECK DOOR PRESSURE MESSAGE ON THE FLIGHT ATTENDANT PANEL **(FAP)**)
▷ (CABIN DOOR CHECK SLIDE PRESSURE MESSAGE ON FLIGHT ATTENDANT PANEL **(FAP)**)
▷ FWD. F/A PANEL **(FAP)** CIDS "CAUTION" LIGHT WENT ON.
▷ SCREEN FOR CABIN LIGHT CONTROL **(FAP)** R5 AND L2 WITH BLACK IMAGE
. . .

acronym / left context

FAP	ON THE **F**LIGHT **A**TTENDANT **P**ANEL
FAP	MESSAGE ON **F**LIGHT **A**TTENDANT **P**ANEL
FAP	FWD. **F**/ **A** **P**ANEL
FAP	. SCREEN FOR CABIN LIGHT CONTROL
FAP	TANK INFO ON CIDS PANEL

resolutions

2	FLIGHT ATTENDANT PANEL
1	F/A PANEL

FIGURE 4 – Illustration of detection of candidate resolutions for the (potentiel) acronym *FAP*. 37 defect descriptions in the training part of the FULL dataset contain the mention *(FAP)* — we show 4 of them in the top box — leading to 5 different left contexts (middle box), which resolution leads to 2 candidates (bottom box).

Figure 4). Table 2 shows the 5 most frequent candidates, as well as the 5 less frequent ones. Some candidate resolutions are clearly wrong, such as the last one. Filtering is of course possible, but we did not explore this.

Since our dataset also contains a column `resolution_description`, we also applied our procedure to this material from which we could extract other acronyms and their resolutions. We identified 61 new pairs (33 new acronyms). The description in the resolution column is much more standardized, and often, acronyms are used without being mentioned in parentheses.

Because we have access to a list of 2 871 acronyms/resolution pairs (2 144 acronyms) we can use this reference to evaluate our process. We can also

freq.	acronym	candidate resolution
915	AVOD⋆	**A**UDIO/**V**IDEO **O**N **D**EMAND
306	EFB	**E**LECTRONIC **F**LIGHT **B**AG
223	IFE	**I**N-**F**LIGHT **E**NTERTAINMENT
182	ICS⋆	**I**NTEGRATED **C**OOLING **S**YSTEM
170	APU	**A**UXILIARY **P**OWER **U**NIT
1	TA	**T**RAFFIC **A**LERT
1	TAT	**T**OTAL **A**IR **T**EMPERATURE
1	TCP ⋆	**T**UNING AND **C**ONTROL **P**ANELS
1	VFSG⋆	**V**ARIABLE **F**REQUENCY **S**TARTER **G**ENERATOR
1	WALL⋆	RO**W**45 **A**ND 46 **L**IGHT PN**L**

TABLE 2 – The 5 most frequent acronym/resolution pairs in the defect description field of the FULL corpus, as well as the 5 less frequent ones. The letter alignment (the best scored one) is indicated in bold. Acronyms marked by a star are not listed in our reference list.

check whether the automatic process finds acronyms that were not previously listed. Out of the 235 acronyms we found, 85 where already listed as acronyms in the reference list, 150 were not. While we are not able to judge the validity of the new acronyms being found, a random inspection of them seem to indicate that they are mostly good acronyms. The ones marked by a start in Table 2 are actually new acronyms. Again, the last one is an error of our extraction procedure (that would be easy to filter). It is of course tempting to evaluate the 85 acronyms we identified that are already in our reference list, but this turned out to require human intervention because the reference list often contains some annotations that we should remove before comparing the lists. Suffice it to say that most acronyms we found are actually correctly resolved, sometimes with minor variations. We found cases where the resolution automatically identified is pretty much different from the reference one, as for *CAM* in Table 3.

3.2 Spell Checking

Without much surprise, the description of defects are fraught with many typos. In order to identify some of them, we gathered a lexicon of 307k words (plain words and their inflected forms, including conjugations) we collected

ADF	CAN	AUTOMATIC DIRECTION FINDING
	REF	Australian Defence Force
	REF	Automatic Direction Finding (equipment)
	REF	Automatic direction finder
AIP	CAN	ATTENDANT INDICATION PANEL
	CAN	ATTENDANT INDICATION PANELS
	REF	Aeronautical Information Publication
CAM	CAN	CABIN ASSIGNMENT MODULE
	REF	Cockpit area microphone (part of the cockpit voice recorder)

TABLE 3 – Excerpt of acronyms and resolutions automatically identified (CAN), and their corresponding resolution in our reference list (REF).

from a github repository[8] to which we added the 55k most frequent words in English Wikipedia, as well as an in-domain lexicon built by listing all the alphabetical words found in the description and resolution columns of the dataset. Then for every word of every defect description and defect resolution fields, we computed the list of closest words from our lexicons, according to the Levenshtein distance (LEVENSHTEIN 1966). Actually, we used the so-called Damerau-Levensthein distance[9] which accounts for the transposition of 2 adjacent symbols as one operation (eg. *glucometer* / *gulcometer*) while it would cost 2 operations with the plain Levenshtein distance. We conservatively kept the words with a distance of at most 1 edit per 5 characters, yielding a list of 15 297 typo/correction pairs involving 5 631 different correct forms, the 5 most frequently misspelled word being reported in Figure 5. It is rather surprising that some words got so many faulty variants. For instance the word *intermittently* has no less than 54 variants according to our procedure. While some might be due to segmentation issues (e.g. *outintermittently*, *upaircraft*), most variants we inspected seem to be just typos. This clearly militates in favour of a unified application for typing defect reports.

8. https://github.com/dwyl/english-words/
9. https://fr.wikipedia.org/wiki/Distance_de_Damerau-Levenshtein

word	# typos	5 randomly picked typos
intermittently	54	intrmittently, inttermittently, ntermittently, outintermittently, intermittentally
illuminated	51	illuuminated, iluminated, ilumminated, immuminated, innluminated
glucometer	44	gluscmeter, glvcometer, glycometer, glyvometer, gucometer
aircraft	43	aircrqaft, aircrtaft, upaircraft, aircvraft, airfcraft
working	41	workign, workiing, workimg, worlking, worrking

FIGURE 5 – 5 most frequent words that got misspelled according to the automatic procedure described, the number of different typos identified, as well as 5 randomly picked ones. Keep in mind that some typos are due to a tokenization issue, and that we may wrongly associate a typo to a given form.

4 Classification of defects

This section is concerned with the automatic classification of a report into its ATA code (chapter and section). We tried a number of typical approaches to classification that we applied to our datasets, focussing only on representing the defect description column, while some other columns may improve performance. The metric we report is the standard F1 score (the harmonic mean of precision and recall).[10]

4.1 Bag-of-word models

A strong baseline consists in representing the input (in our case the defect description) as a bag of words (bow), and then train a classifier on top of it. We ran a number of variants of a support vector (SVM) classifier[11]. More precisely, the defect description is represented into a huge sparse vector which dimension equals the number of different units in the descriptions of the

10. https://en.wikipedia.org/wiki/F1_score. Other metrics, such as accuracy were not much different here.
11. We used the SVC implementation of scikit-learn.

training part. The coefficient associated to each dimension is the so-called tf-idf score which favours frequent units, while downgrading those that are present in too many descriptions.[12] We considered different types of units, among which words, ngrams of words and ngrams of characters. Since this kind of representation can be quite large, we also considered variants where the most frequent units are kept in the bow representation, but filtering typically comes at a price in performance.[13]

4.2 Deep learning models

We also tested a number of deep learning approaches. The approach consisting in fine-tuning a pre-trained BERT model (Devlin et al. 2019) on the training material available is nowadays ubiquitous in NLP, since the authors reported impressive results in doing so in a number of challenging benchmarks. It is worth noticing that this is a much heavier approach : fine-tuning BERT requires 30 minutes per epoch on the Reliable dataset and 15 hours on Full for a computer equipped with a GTX 1070 GPU, while an SVC model is typically trained within a few minutes of a laptop CPU, if not less depending on the variants.

Unfortunately, fine-tuning BERT was not successful in our case[14]. While we did not have time to perform in depth analysis of the reasons why BERT failed to learn, we believe that it is mostly due to the label distribution. It is a known problem that performing backpropagation with unbalanced dataset tend to perform poorly, since backpropagation favors the majority classes and tend to ignore the uncommon labels. Since in the distribution of the dataset the majority of labels are uncommon, it is logical that a regular neural network would not be able to efficiently learn to classify the dataset.

We developed two variants to try and deal with label imbalance : over-sampling and weighting the samples. Both however diminished the performances, since the network was now over-predicting uncommon classes. Time constraint prevented us from exploring more solutions, such as the focal loss.

We also tried a number of canonical deep learning approaches including

12. See `https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction` for more.

13. For instance, on the Trax benchmark, the SVC computed on bow of all the unigram and bigrams performs an F1 score of 81.1, while keeping the most 5k (resp. 1k) frequent ngrams yields 79.2 (resp. 75.7).

14. It is to note also that the text had to be normalized before being used by BERT

a recurrent neural network (with a GRU (Chung et al. 2014) cell) for classification, as well as simpler variants where a defect description representation is obtained by averaging pre-trained word embeddings, then fed into an SVC classifier. We considered pre-trained GloVe embeddings (Pennington et al. 2014).[15] We also trained our own embeddings on the descriptions of the Full dataset, with the hope that they would capture specificities of the data (acronyms, typos, etc.). We used for this a Skip-gram model (Mikolov et al. 2013).[16] We also trained fastText word embeddings (Bojanowski et al. 2017). For some reasons however, we did not tested them for classification,[17] but used them as a sanity check that they capture useful information. Figure 6 lists the most similar words of some randomly picked words according to fastText : We observe that word embeddings behave as expected, that is, they capture words that share related meaning (synonyms, antonyms, etc.), as well as words that share similar spellings (typos, morphological variant).

screen	screeen ptv blackscreen black creen ptc sreen screeb
water	potable waterspigot nowater faucets faucett hotwater flowing
missing	broken mising missising retaining boken brokened brken
sink	clogged draining drain draing drains unclogged sinks glogged
open	close closed closing opening reopen unlatch latch unlatched

Figure 6 – Most similar words (right) of some randomly picked words (left), according to a fastText word-embedding model trained on the descriptions of the dataset.

4.3 Results

We report in Table 4 the results of some of those variants we tested in-domain. By this, we mean that the models are trained (or fine-tuned) on the training part of a given benchmark, and tested on the testing part of the same benchmark. Note that due to time and memory issues, some variants were not tested on all the datasets. Clearly more investigations are required to give a clear picture of the task. Overall, the SVC variants are the best

15. Some normalization was applied (such as error detection and acronym resolution) in order to better fit the model's vocabulary.

16. We used a window size to 5.

17. Yet another future work.

classifier	Rel.	Trax	Full
SVC variants			
word 1-5 ngrams, no normalisation	97.1		
word 1-3 ngrams, no normalisation	97.5	80.6	
word 1-2 ngrams, no normalisation	97.7	80.9	59.9
word 1-2 ngrams, spelling replacement	97.8	79.8	59.9
word 1-2 ngrams, acronym and spelling replacement	97.9	79.9	60.1
word 1-2 ngrams, nltk porter/snowball stemming	97.6	81.1	
char 2-5 ngrams, no normalisation	97.5	81.8	
dummy : majority class	36.2	6.7	6.7
BERT fine-tuning, acronym and spelling replacement, number replaced	18.4		
GRU, acronym and spelling replacement, number replaced	11.3		
GloVe		50.2	
Skip-gram		57.3	

Table 4 – In-domain classification results (F1 scores) on our 3 benchmarks : Reliable (Rel.), Full and Trax. Due to time constraints, not all variants were tested over all benchmarks.

performing ones across all benchmarks. Normalizing the defect descriptions has no to little impact on performance, and considering ngrams of characters (which avoids text normalisation) instead of ngrams of words typically results in similar or better performance, while being more much memory efficient.

The augmentation of the perfomance while comparing the datasets of various reliability shows that indeed the full corpus is very noisy. While the trax dataset is not manually verified (as is the Reliable dataset), we consider it more reliable since it contains only reccurent defects, which are themselves classified as reccurent based on their ATA label. That means that in order to be in the Trax dataset, an erroneous defect would have to have been erroneously labelled 3+ times, which reduced the number of erroneous labels that makes it into the trax dataset. The Reliable dataset, which has been manually verified, is 100% reliable and the fact that we obtain such high

results shows that the task itself is not very complicated but that the noise makes it very hard to correctly classify.

ATA code	ATA label
▷ most correlated features	
11-32	placard :missing
▷ placard, placards, placcard, belongs, theres, damage, sticking, sure	
21-20	distribution, distribution :inoperative
▷ recirculation, fans, fan, gasper, recirc, installed, smell, present, recir	
21-30	pressurization control, pressurization control :inoperative
▷ auto, alt, outflow, cabin, tcn, pressure, indicator, rate, altitude, auto2	
21-40	heating, heating :inoperative
▷ heating, heater, heaters, heat, ovht, cargo, iii, duct, forward, vent	
21-50	cooling
▷ pack, cooling, conditioning, deflector, ball, exhaust, packs, fcvs, bypass	
21-60	temperature control, temp control :too cold
▷ temp, zone, compt, modulating, overboard, trim, control, temperature	

FIGURE 7 – Most useful words for identifying randomly picked ATA code (chapter and section) according to a tf-idf bow logistic regression model trained the defect descriptions of the FULL dataset.

We were rather surprised by the overall good performance of the bow approach on our benchmarks. We investigated why this was so by training a logistic regression (LR) model on the same tf-idf bow representation. This model slightly underperforms the SVC classifier, but it is easier to investigate which feature was found important. We report the 10 most important words according to the LR model for some randomly picked AT codes. We observe important words often come with either morphological (e.g. *heating* / *heaters*) or typographical (e.g. *placard* / *placcard*) variants. This suggests that the model is capable of some data normalisation, further explaining why the normalization we conducted was not very rewarding. Also, we observe (al-

though it would deserve a real analysis) that words are topically distributed, and globally correspond to words we would expect based on the ATA label.

5 Detecting Recurrent Defects

Figure 8 shows the time span of a RD, that is, the difference in days between the first reported day to the last reported one for each genuine RD. The majority of recurrent defects are emitted during the same day and very few span over more than 11 days.

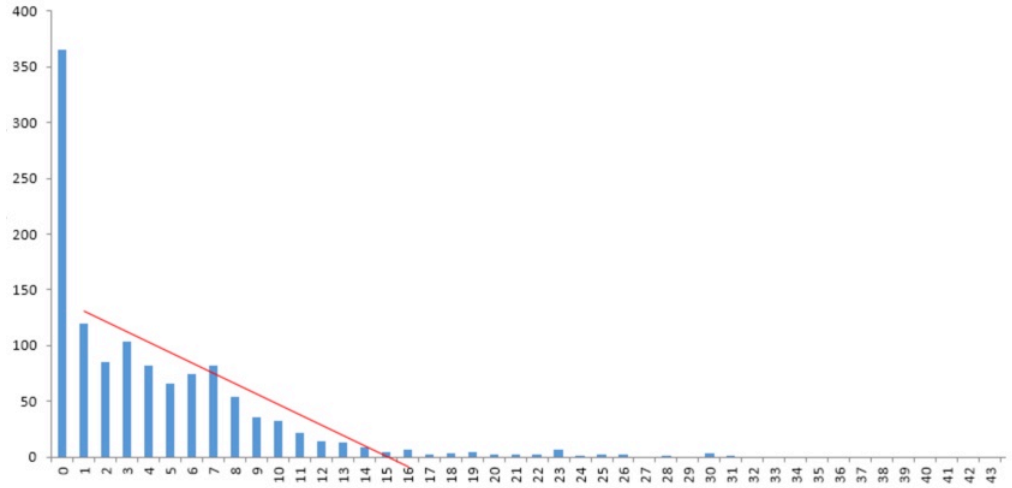

FIGURE 8 – Frequency (y-axis) of the number of days (x-axis) between the first reported day to the last reported one of each manually attested recurrent event in the dataset. The majority of RDs are emitted during the same day.

Figure 9 shows the defect descriptions associated to RD cluster 88805 (ATA code : 33-10, flight compartment). Descriptions can be rather different for the inexperienced.

Although not entirely intuitive, we can conceive the problem of identifying RDs as clustering defects into their respective groups based on their descriptions. Defects in one group are elected recurrent. Under this view, it seems natural to evaluate the task by comparing the manual partition of defects to the one automatically found. This is the way we are evaluating our approaches here. At the same time, most defects are not part of genuine

▷ FLIGHT DECK "LT OVRD" SWITCH IS DIFFICULT TO TURN ON / OFF.
▷ DURING PDC, FOUND CAPTAIN'S DOMW LIGHT INOP.
▷ LEFT ENGINE FLOW BAR LIGHT IS U/S.
▷ LEFT ENG PRIMARY HYD. PUMP SWITCH "ON" LIGHT U/S.
▷ "L NAV " UPPER IDENTIFICATION LT. U/S.
▷ VNAV SELECTOR SWITCH ON MCP RIGHT BUTTON OF THE SWITCH THE LIGHT BULB IS U/S.
▷ TRIM AIR SWITCH "ON" LIGHT BULB IN U/S.
▷ LT. OVERIDE SWITCH "ON" BULB U/S.

FIGURE 9 – Recurrent defect 88805 which encompassed 8 defects which description is reported here.

recurrent defects, which suggest that the detection of RD clusters might as well be evaluated as an information retrieval task (with precision and recall measures). We leave this for future work. For each approach, we compute 4 metrics that are used for comparing two clusters :

homogeneity A decimal score in $[0, 1]$ representing to what extent elements of found clusters belong to the same RDs.

completeness A decimal score in $[0, 1]$ representing to what extent elements from the same RDs are assigned to the same found clusters.

v-measure The harmonic mean of both previous scores.

ari The adjusted Rand index is a similarity measure between both the reference and the computed clusterings. The **ari** has values in $[-1, 1]$, 0 meaning random assignments.

5.1 Clustering the test material with DBScan

A straightforward approach to solving the problem at hand is clustering the defects without revisiting their original ATA classification. An appropriate algorithm for this is DBSCAN (Density-Based Spatial Clustering of Applications with Noise),[18] which attempts to find, in an arbitrary vector

18. We used this implementation : `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html`

System	ARI	Homog.	Compl.	V-meas.
db-desc-tfidf-eps0.5 100-dimension tfidf with LSA	0.003	0.22	0.02	0.04
db-desc-tfidf-days-eps1.0 Same as above, + δdays	0.045	0.06	0.06	0.06
db-desc-tfidf-days-ch-eps1.0 Same as above, + δATA chapter	0.042	0.05	0.06	0.06
KMeans unigrams, 800 clusters description+resolution	0.076	0.14	0.07	0.09
DBSCAN tfidf eps 0.5 min samples 3 ; resolution only	0.074	0.29	0.06	0.11
SVC classifier 1-3 word ngrams + time constraint	0.10	0.04	0.05	0.028

TABLE 5 – Result of recurring defects detection.

space, core samples of high density then grows clusters centered on them. This method is quite interesting in our case, as it offers a natural way to find a few clusters containing only a subset of the complete dataset's defects : One simply has to set a hyperparameter `eps` to limit the expansion of clusters. We experimented with different vectorial representations of defects, including a tf-idf with latent semantic analysis (LSA) fit over the train corpus, as well as a dimension reserved for the difference in days between reported dates. We report the results below, on the test set, after exploration of the eps value on the dev set. A handy way of measuring the expansion of clusters is to measure the number of predicted clusters and their average size.

These results are disappointing, since an ARI of 0 basically means no better than chance (1 means perfect predictions). Nevertheless the score also has to do with the low reference quality. The text representation (tf-idf) does surprisingly little, which either suggests that it is an invalid representation, or that common textual defect descriptions are not a good indicator of their recurrence. Days elapsed between defects are much more important. A natural way to further explore this algorithm would be to add additional dimensions corresponding to additional features derived from defect metadata.

5.2 K-mean clustering of the full dataset

While the previous approach was only making use of the test material at test time, the present approach has been thought as a mean to exploit regularities in the full dataset, and therefore needs all the existing data (training and test) to operate. This is definitely less handy than the previous approach, since the full dataset has to be clustered. In a nutshell, we encode each defect description (or the resolution column or both) into a bow representation. We then apply the KMeans clustering algorithm[19] on those representations. Given a partition of the entire dataset, we group together defects that got clustered into the same group, provided they concern the same aircraft and they obey the time constraint given in the definition of the problem. We conducted some tuning on the train part of the FULL dataset, varying the number of clusters from 120 to 480, considering the defect_descriptions column, the resolution one, or both. This tuning was done in order to optimize completeness instead of the V-measure or ARI because the TRAX data, which is our reference for recurrent defects clustering, is very accurate (human review) but likely incomplete. The best performance we obtained was by considering 430 clusters, using unigrams for computing the tf-idf bag of word representation. This is the variant reported in Table 5 and which (although not very strong) delivers the best performance overall.

5.3 Detecting by classifying defects

This approach is straightforward and is intended to serve as a baseline. We apply a classifier trained to label a defect description into its ATA code (see Section 4) to each defect of the test set. All defects that receive the same class label are elected recurrent defects, regardless of time contraints. Except for the ARI metric, results are very low, mainly because this approach is producing very large clusters of descriptions, which are not considered recurring defects in the reference.

5.4 Universal Sentence Encoder

Google (CER et al. 2018) has released the Universal Sentence Encoder, a method intended to compute semantic similarity between any given pairs of

19. We used the implementation described at `https://scikit-learn.org/stable/modules/clustering.html#homogeneity-completeness-and-v-measure`

sentences. For example, the sentence "apple is a healthy fruit" is semantically similar to "John loves eating bananas" and is dissimilar to "Honda Accord is the best family sedan". Technically, a language model encodes and converts sentences into semantically-meaningful dense real-valued vectors. We tried to use such a model to automatically regroup defect descriptions based on semantic features in sentences such as "audio jack" and "bird strike".

We developed a simple demo of Universal Sentence Encoder and found the results are promising. The language model is sensitive to semantic differences, can associate similar concepts such as "audio jack" and "headphone jack" and is immune to simple typos (e.g. "screen" and "screeb"). We were unfortunately not able to quantify the performance (clustering accuracy) of our model, which we leave for future investigations.

	recurrent_created_date	acft	defect_description	rec_id	predicted_cluster
194	2019-05-16 08:36:00	AA-2012	fuel "locator (see lm, lo)" outer transfer ope...	99566	0
195	2019-05-16 08:36:00	AA-2012	after engine start on 2 legs, right (runway sy...	99566	0
196	2019-06-11 04:01:00	AA-2012	cockpit door fault caution illuminated with cl...	100034	1
197	2019-06-11 04:01:00	AA-2012	cockpit door cloud top striker fault	100034	1
198	2019-06-11 04:01:00	AA-2012	cockpit door cloud top striker fault.	100034	1
199	2019-06-23 10:23:00	AA-2012	invoked reference l5707675 engine #1 thrust re...	100285	2
200	2019-06-23 10:23:00	AA-2012	during inspection of engine #1 thrust reverser...	100285	2

FIGURE 10 – Sample clustering result using Universal Sentence Encoder.

6 Conclusions and Future Works

Our journey with Air Canada was very pleasant, generating a lot of enthusiasm from the participants as well as a few disappointments. Among them, we must recognize our inability to conduct conclusive experiments on the main problem which was to identify recurring defects within a given time frame. We believe that the task, although clear at a conceptual level, requires a much better understanding of the maintenance workflow, from the moment the defect is noticed to the moment the last recurrent instance of the defect is considered closed.

We were nevertheless more lucky with our classification results, reporting very good figures with simple approaches on a subset of clean data gathe-

red. Again, we feel that the data contain too many various sources of noise, and that refinements of the task (or the reference) must encompass a better understanding of the data. With that being said, we do not feel anything particularly unmanageable within our task : most NLP tasks of interest encompass intricacies that challenge the way the data-set is built, or the way we evaluate solutions.

Some further investigations are required to investigate a few variants we devised. In particular, we found good clustering ability of the Universal Sentence Encoder that could eventually lead to good recurring defect detection. We also have to understand why some deep learning approaches performed so badly on the classification tasks we considered.

Given the very significant orthographic corruptions of the defect descriptions and other textual elements, it could be very beneficial to Air Canada to look into leveraging spell-checking technologies, such as those already present in most computer platforms (Android, iOS, etc.). This may very well prove very cheap and would offer an invaluable return when the time comes to perform data mining and NLP manipulations on the data at hand.

At the very heart of the problem submitted by Air Canada lies the issue of label reliability. Indeed, had the ATA codes been reliably and properly attributed to defects, clustering would have been trivial, the only difficulty being one of properly taking the time frame into account when creating clusters. There are a number of remarks that can be made regarding reliability in this context. Firstly, it does seem that the ATA ontology is difficult to apply consistently. This could surely be mitigated by better formation (for instance, instructing personnel to avoid catch-all clauses), but also by using automated tools. For instance, a labeling tool not unlike those presented in this report could present the user with a list of probable labels from which they could pick the best code. Secondly, it may very well be that the ontology is improperly designed in the first place, leaving the maintenance personnel at a loss when labeling defects. This should be looked into, particularly for ATA combinations that are seldom used. Thirdly, the labeling task presented in this report produces an interesting by-product, in the form of a confusion matrix, i.e. a report of labels for which the human opinion and the machine output differ. This could be an interesting starting point for an investigation into improper labeling on the part of humans : The machine could very well be wrong when producing an ATA label, but if it is not, then there is a systematic problem with human labeling. Lastly, there should be a way to identify a subset of “elite maintenance personnel” whose labeling could

form the base for an ultra-reliable subset of the original data. This way, an algorithm trained on these labels would benefit from the supervision of the most seasoned experts Air Canada has within its ranks, and therefore learn from the best.

References

Bojanowski, Piotr et al. (2017). "Enriching Word Vectors with Subword Information". In : *Transactions of the Association for Computational Linguistics* 5, pages 135–146. DOI : 10.1162/tacl_a_00051. URL : https://www.aclweb.org/anthology/Q17-1010.

Cer, Daniel et al. (2018). "Universal Sentence Encoder for English". In : *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*. Brussels, Belgium : Association for Computational Linguistics, pages 169–174. DOI : 10.18653/v1/D18-2029. URL : https://www.aclweb.org/anthology/D18-2029.

Chung, Junyoung et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In : *NIPS 2014 Workshop on Deep Learning, December 2014*.

Devlin, Jacob et al. (2019). "BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding". In : *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota : Association for Computational Linguistics, pages 4171–4186. URL : https://www.aclweb.org/anthology/N19-1423.

Levenshtein, V. I. (1966). "Binary codes capable of correcting deletions, insertions and reversals". In : *Sov. Phys. Dokl.* 6, pages 707–710.

Mikolov, Tomas et al. (2013). *Efficient Estimation of Word Representations in Vector Space*. URL : http://arxiv.org/abs/1301.3781.

Pennington, Jeffrey et al. (2014). "GloVe : Global Vectors for Word Representation". In : *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar : Association for Computational Linguistics, pages 1532–1543. DOI : 10.3115/v1/D14-1162. URL : https://www.aclweb.org/anthology/D14-1162.