Natural Language Processing for the identification of Human factors in aviation accidents causes: An application to the SHEL methodology

Original

Natural Language Processing for the identification of Human factors in aviation accidents causes: An application to the SHEL methodology / Perboli, Guido; Gajetti, Marco; Fedorov, Stanislav; Giudice, Simona Lo. - In: EXPERT SYSTEMS WITH APPLICATIONS. - ISSN 0957-4174. - STAMPA. - 186:(2021), p. 115694. [10.1016/j.eswa.2021.115694]

Publisher:

Elsevier

Published

DOI:10.1016/j.eswa.2021.115694

Natural Language Processing for the identification of Human Factors in aviation accidents causes: an application to the SHEL methodology

Guido Perboli[a,*], Marco Gajetti[b], Stanislav Fedorov[c], Simona Lo Giudice[d]

[a] *DIGEP and ICELab@Polito Politecnico di Torino Corso Duca degli Abruzzi, 24 10129 Torino, Italy*
[b] *Deloitte s.p.a. Galleria S. Federico, 54, 10121 Torino, Italy*
[c] *DAUIN and CARS@Polito Politecnico di Torino Corso Duca degli Abruzzi, 24 10129 Torino, Italy*
[d] *Vrije Universiteit Amsterdam 1105, 1081 HV Amsterdam, Netherlands*

---

[*] Corresponding author

Email addresses: `guido.perboli@polito.it` (Guido Perboli), `mgajetti@deloitte.it` (Marco Gajetti), `stanislav.fedorov@polito.it` (Stanislav Fedorov), `simonalogiudice93@gmail.com` (Simona Lo Giudice)

URL: `https://orcid.org/0000-0001-6900-9917` (Guido Perboli)

Preprint submitted to Expert Systems with Applications

Abstract

Accidents in aviation are rare events. From them, aviation safety management systems take fast and effective remedy actions by performing the analysis of the root causes of accidents, most of which are proved to be human factors. Since the current standard relies on the manual classification performed by trained staff, there are no technical standards already defined for automated human factors identification. This paper considers this issue, proposing machine learning techniques by leveraging on the state-of-the-art technologies of Natural Language Processing. The techniques are then adapted to the Software Hardware Environment Liveware (SHEL) standard accident causality model and tested on a set of real accidents. The computational results show the accuracy and effectiveness of the proposed methodology. Furthermore, the application of the methodology to real documents checked by experts estimates a reduction of the time needed for at least 30% compared to the standard methods of human factors identification.

Keywords: SHEL, Human Factor, Aviation Safety, Natural Language Processing.

1. Introduction

In general, accidents and incidents in aviation are rare events, for which the aviation safety management systems take fast and effective remedy actions. In 2017, there were over 36.6 millions of estimated departures in the world, and only 88 accidents, with 5 fatal events and 50 fatalities (ICAO Safety, 2018a). Starting from 2013 until 2018, the accident rate per million departures has been floating around 3%. As positive as this is, the availability of data to develop smart supporting solutions is somehow restricted. Additionally, the harmonization of standards and criteria among the organizations in the world is a relatively new topic (starting in 2010). Although, with an aggregation of global data and shared database systems available for all the organizations, these two factors are no more considered to be an obstacle in creating a supporting smart system for specific purposes like Human Factor detection. It is estimated that using such tools would drastically decrease the time spent by the investigator in re-analyzing the report, his effort in the process, and, last but not least, would automatically contribute to the ADREP (ICAO ADREP, 2019), which is the Accident/Incident Data Reporting system, globally operated and maintained by International Civil Aviation Organisation (ICAO). The ADREP system receives, stores and provides organizations with incidents' data that will assist them in validating safety.

As the technology progressed towards the reliability of the plains, attention shifted to the Human Factor (HF). The era of HFs brought the concept of Crew to the fore and focused on the actions of the individual, still not having a clear relationship between the person and the Organization. More detailed studies and analysis of statistical results led to the classification of organizational factors (as an important part of HFs) - which includes the organizational culture and operational context of a complex environment.

The job of the investigator, when analyzing an accident, is, firstly, to identify the "root" factors that caused the events leading to the accident. From these Human Factors, the investigator can proceed with drafting safety recommendations and remedy actions that can eliminate avoidable human, economical and social costs (Hawkins, 1993; Aviation Safety Improvement Task Force, 2005; ICAO, 1993). Extracting valuable information from the accident's full-text report is a critical step, that can be supported by an autonomous system able to process natural language. Currently, the level of automation in the process is low and limited to tagging each event with a standard accident causality model, called SHEL (Reason, 1992, 1990). This conceptual model is a widely used tool in aviation, allowing analysis of the interaction between multiple industrial system components, such as the ones classified in the four capital letter acronyms:

- S = Software, any procedures, document, checklists, training, computer programs e.g intangible knowledge;
- H = Hardware, machines, and equipment, including controls, tools, and interfaces;
- E = Environment, weather conditions - oxygen, pressure, temperature, but also socio-economic considerations in which the individual is living;
- L = Liveware, any person involved in the workplace - pilots, crew, Air traffic control, engineers, etc.

Before the SHEL, ICAO laid down the requirements for the formal Safety Management Systems (SMS) of airlines and airports. Most of these tools were based on readily available technologies (Plioutsias et al., 2018). Despite its proven efficiency, the SHEL approach is heavily expert-based and could take up to 18 months. Moreover, the effect of a human-based analysis is the difficulty in comparing the results of a SHEL analysis done by different expert teams. Operators had formal tools only and have to manually filter unnecessary information. The introduction of language processing technology opens the way to optimize the existing methods of HF identification in terms of saving resources of working time and processing costs while satisfying the high safety standards.

Our work represents the first approach to the HF identification in the accident investigation process by applying existing machine learning methods, and state-of-the-art technologies of Natural Language Processing (NLP). To the best of our knowledge, there have never been other automated systems implemented for the specific problem of the final HF identification starting from a SHEL-based tagged report. Since the current standard relies on manual intervention only, there are no technical standards already defined for automated HF Identification. That makes it hard to represent the quantitative accuracy of the system from a software point of view. Moreover, air accidents are rare events, and collecting the data in terms of reports with fully tagged HF is an expensive task for each company. Thus, the tagging is normally considered an internal asset and a limited number of fully tagged documents are available. Hence, the final objective of this paper is the introduction of a decision support system that can be used in the aviation industry to improve the performance of the root-cause analysis of the accidents and that can work in conditions of limited training data. The results on-field of our application and the conclusion of the experts in the field confirm the success of our approach.

The paper is organized as follows. Section 2 recalls the main literature. Section 3 summarizes the methodological aspects of the proposed tool: knowledge database used, word and sentence embedding aspects and similarity measure adopted. Section 4 presents the experimental results of the proposed method evaluation, while Section 5 discusses the results of an on-field test, showing the importance and industrial relevance of the proposed solution. Finally, Section 6 draws the conclusions and highlights the future axes of research.

2. Literature review

Natural Language Understanding (NLU) is a growing field, for which many companies have invested time and resources, reaching increasingly better results due to the availability of a huge amount of data. In terms of general domain language, many outstanding results were obtained in giving the machine the ability to understand the semantic meaning of documents (Semaan, 2012; Turney & Pantel, 2010; Mirończuk & Protasiewicz, 2018; Castrogiovanni et al., 2020). The limitation of these technologies is that the effectiveness of these models strictly depends on the particular task they were implemented for, due to the main issue of systems based on Neural Networks - limited generalization and abstraction capacity. Therefore, different researches were conducted in a more specific-domain field, like medicine (Soğancıoğlu et al., 2017), or law (Sugathadasa et al., 2017). The recent investigation was focused to develop a real-time safety prognosis by mining and classifying accident reports, where the expected output of the system is intended to give information if the accident is likely to happen to the first-class passengers (Srinivasan et al., 2019).

The identification of the causal chain of events leading to an accident is a very costly and time-consuming process, which takes up to 18 months, but is crucial to improve aviation safety. Hence, the reduction of the time needed by the annotation of the safety reports has a high impact on the industry both from the cost and the safety points of view. Then, any automatic system in this sector should be crucial to support the decision-maker (the investigator) in his analysis, without substituting him in the work, but heavily reducing the time for the analysis, letting the companies and the regulators quickly act on the system for improving its safety. To the best of our knowledge, currently, there is not an automatic system able to directly extract HF from accident reports. The most recent approach to analyze the accident reports is given by Hu et al. (2019), where the authors compared several machine learning algorithms in textual indicator extraction tasks and outlined the best ones. However, the high aviation safety standards do not allow such models for HF extraction. That is why the decision support system working in parallel with the expert is the required and only possible solution.

In general, until 2015, the analysis of accident reports was only manual, with time and resources invested in an avoidable and inefficient way. According to Mirończuk & Protasiewicz (2018), NLP successfully applied to several industries, but not to the air accident classification. To the best of our knowledge, the only other related work is by Mosca (2015). The authors describe how the analysis of an aircraft accident can be processed in a partially automatic way, developing a supporting system that can address the safety investigator during his analysis. Currently, the system can read accident reports and classify the events following a particular safety standard SHEL. To be able to read, process, and identify single events in the report, some Natural Language Processing methods were used, like a customized Part-Of-Speech Tagger to identify relevant words in the text. The outcome of this system is a SHEL-based tagged report - where each relevant event is tagged according to the SHEL standard. This semi-automatic system is supposed to help the investigator moving forward with the analysis in a faster way than simply a manual process. From this stage, the extraction of HFs from the accident events begins.

3. Methodology

The proposed solution follows a Semantic Text Similarity approach. The general strategy behind it is to leverage examples of events that are already tagged with the respective HF and are collected in our knowledge base. When analyzing a new event, we compare it with the tagged examples in terms of semantic meaning. If these events are enough semantically similar to the examples we have, then it is highly probable that they contain also the same HF. Based on the notions of Distributional Semantic theory, we designed a system to represent aviation-related sentences in a semantically meaningful way, and then applied it to identify a correlation between phrases containing the same HF (see Figure 1). This correlation was then used in a machine-learning algorithm to improve the recognition of the HF in new sentences, increasing the knowledge base.

At the core of our algorithm there is the semantic meaning of sentences, and thus, of words. It requires an effective representation of the words, carrying all the semantic information that the word has. For this purpose, we choose the Distributional Semantic approach to be the fundamental

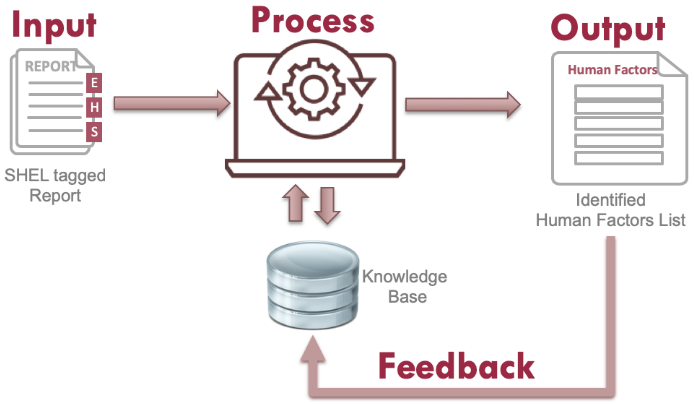

Figure 1: Schematic representation of the constructed system

base, and Vector Space Model (VSM) is a very effective system to represent tokens accordingly to their context (White et al., 2015). Starting from the representations of single words composing a given sentence, the system would then extract the representation of the sentence itself, using aggregation methods. In particular, we explored three different methods of text representation:

1. A model over document (or sentence) embeddings, the $d2v\_model$
2. A model for word embeddings first and sentence representation then, with a relatively small corpus, the $Genw2v\_model$
3. A model implemented to verify the effectiveness of the algorithms of the second model when increasing significantly the corpus dimensions (Mahoney, 2006), the $TFw2v\_model$

All of the three models were trained including also the integration of the specific- domain corpus, built from the aviation-related text. Moreover, the VSM allows us to use an exact numerical measure to assess the element's similarity: it is related to the concept of distance between vectors, which is estimated through the so-called *cosine distance*, and it's related to the angle between these vectors.

The main steps of our solution are:

1. Select an adequate Corpus for embedding models and integrate the specific- domain full text.
2. Pre-processing over the Corpus to improve the effectiveness of embeddings.
3. Build and train a machine learning vector representation model, leveraging the available data.
4. Use the model to represent sentences and tokens from the report that is processed.
5. Compute the semantic similarity between new sentences and old tagged sentences and get the HF with the highest value.
6. Register the new sentence and similarity score as tagged under the relative HF.

Creating and Cleaning the Corpus. The general corpora used for the three models were mainly of two different dimensions: a generic corpus and a domain corpus.

The natural language text containing words of all inflected forms was taken from the literature. In details, we used the *Brown Corpus* and the *Text8* corpus (Francis & Kucera, 1979; Mahoney, 2011). The *Brown Corpus* is the oldest available corpus, compiled in 1960s at Brown University. This corpus is relatively small, about 1 million words, and considered a bit dated, but still widely used in the NLP field. The *Text8* corpus was created as a result of the compression projects by Matt Mahoney (2011), and it considers $253,885$ unique words, while the total number of words (considering the repetitions) is 100 billion. What we used, given our resource availability and the need for a light and portable system, was a share of this corpus, which is about 17 million of words. The share of the specific-domain part over the total corpus would be 8%, which is good enough for our solution.

The domain corpus was extracted from books and air accident documents. However, training a neural network over obtained raw text would result in each of the inflected forms of a single word represented by a separate embedding vector. This in turn leads to many drawbacks and inefficiencies. Maintaining a separate vector for each inflected form of each word makes the model bloat up and increases unnecessarily the memory usage. For this reason, we made a deep work of cleaning of the domain corpus by Python-implemented modules: these tasks include deleting punctuation (symbol digits, paragraph spaces, tabs), lowering case, deleting or changing the stopwords (Sebleier, 2010), tokenizing, Part-of-speech tagging and lemmatizing. The outcome of the specific-domain corpus was about 1.5 million lemmas.

Selecting and Training the models. The corpora created were then used to train the two models selected for the vector representation of words and sentences. Among the possible paradigms applicable for the final purpose of semantic similarity, we chose the first model to be *Word2vec*, and consequently, the second model to be *Doc2vec* (Simmons & Estes, 2006; Mikolov et al., 2013; Le & Mikolov, 2014; Cuzzocrea et al., 2020). The idea behind developing more than one model using different paradigms comes from the fact, that in the solution design there are many possible decisional factors to consider at different stages of the implementation and not enough information on the selection of the adequate parameters or options.

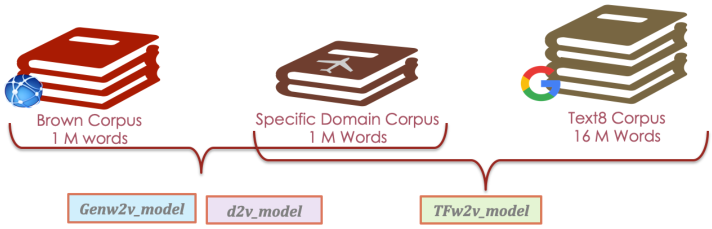

Figure 2: Schematic representation of the corpora usage for training the tree considered models

The *Word2vec* model was trained over the smaller corpus (Brown + Domain-specific Corpus) to create the *Genw2v_model*, and over the bigger corpus (Text8 + Domain-specific Corpus) to create *TFw2v_model*. The *d2v_model* was created by training a Doc2vec network over the Brown + Domain-specific Corpus (see Figure 2). Let us explain how the two basic models (*Word2vec* and *Doc2vec*) are different and what's the outcome expected from both.

Word2vec model belongs to the set of predictive approaches to generate dense embeddings. It learns embeddings by training a neural network to predict neighbor words. The approach, designed by Google, was born to transfer the semantic meaning of words into the embeddings created, so it is particularly useful when it comes to evaluating similarity. The advantage of this set of methods is that they are fast and efficient and easy to train. There are two possible implementations included in the paradigm: the the *Skip − Gram* and the *Continuous Bag of Words* (CBOW) methods (Mikolov et al., 2013). While CBOW architecture predicts the current word based on the context, the Skip-gram works in reverse, predicting surrounding words given the current word. For the implementation of the prototype, the *Skip−gram* model is used, as it is proven to be more efficient in smaller corpora and it is said to be accurate for rare words, while CBOW is faster by a factor of window size, which has positive impacts with larger text corpora. In particular, we used the *Skip−Gram with Negative Sampling* (SGNS), which is a more effective version of the skip-gram, as it adds to the maximizing objective function in the learning algorithm, a minimization component, over the negative examples. The outcome of a trained *Word2vec* network is a system able to process each word of a document and represent it as word embedding. Thereby an additional phase is required for the *Word2vec*-based models implemented: starting from the vector representations of words composing a given sentence, we need to obtain a comprehensive dense vector for the entire sentence.

Doc2vec differentiates from *Word2vec* since it directly returns the sentence vectors. The paradigm lets us build directly sentence vectors, without first developing the embeddings of the composing words. It was developed by the same creators of Word2vec and it is sort of an extension of its model, designed to represent a whole document of any length, starting from its words' semantic representation. Having a fixed-length vector representing sentences and not only a single word is the objective of this comprehensive model, which is based on the *Paragraph Vector* algorithm. This algorithm is an unsupervised model that learns continuous distributed vector representations for variable-length pieces of text. As in Word2vec, the outcome of the model is that semantically similar sentences have similar vector representations, that we can call paragraph vectors. The model trained over Doc2vec will be able to process a whole sentence and give ad output a dense embedding representing that sentence, so there is no need for additional steps before the similarity comparison phase.

As explained, while the *Doc2vec* network gives directly a sentence vector, the *Word2vec*-based networks need an additional phase to get the sentence embedding, starting from the word vectors composing the sentence itself. This additional phase was called Sentence Embedding. Additionally, we decided to try two different approaches for the two different models we had (*Genw2v_model* and *TFw2v_model*). The first approach is the average method, for the *Genw2v_model*, and it is based on the easiest idea: simply computing the average vector of the word embeddings v_w composing the sentence $s = \frac{1}{|S|} \sum_{w \in S} v_w$, considering that every sentence processed is first lemmatized and cleaned. The second method for the *TFw2v_model* is called the *Smooth Inverse Frequency* (SIF) method (Sidorov et al., 2014; Pagliardini et al., 2017): it computes the sentence vector $s = \frac{1}{|S|} \sum_{w \in S} a_w v_w$ as the average of the word embeddings v_w, weighted over a factor related to the inverse frequency a_w of each word appearing in a document (in our case the corpus used). The principle of this method is that frequent words are usually the least relevant, regardless of the discourse. Therefore, such frequent words should have less impact on the final sentence vector.

Semantic Similarity Computation. When reading a new accident report, the developed prototype first collects the events (sentences) in the document and then compares each one of them with the HF-tagged sentences belonging to the knowledge base we created initially. If the similarity is high enough, there are chances that the relative HF is present in the new event as well. In the Vector Space Model (VSM), the traditional cosine measure (Sidorov et al., 2014) is commonly used to assess the similarity between two vectors, which represent the objects we want to compare. The length of vectors is usually related to the frequency of a word in the documents, and this aspect is not relevant when it compares the semantic meaning of words. Having a similar semantic meaning tend to have the same direction in the N-dimensional vector space, where N is the length of the embeddings. With this in mind, we can simply analyze the angle between the two vectors. In particular, the cosine of the angle gives us an idea of the relative direction of the vectors and it is computed through the dot product $\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}|| \, ||\vec{b}||}$ between the embeddings, as consequence of geometric and mathematical interpretation. In general, more sophisticated similarity measures exist, that form a so-called kernel. However, the study of different similarity measures in VSM is not related to the content of the case study and out of the scope of this paper.

Learning from the outcome. After computing the cosine similarity between the new processed event in the report and each Human Factor-tagged sentence in our knowledge base, the obtained highest score is registered and linked to the related HF. As a result, every HF will have a similarity score with the processed sentence; the HFs and their similarity score are stored in a dictionary that sorts them based on the highest score. The first n HFs of the dictionary are then shown to the user (the investigator), who will evaluate the outcome and decide among the n HFs which one is contained in the processed event. This is done for every event in the report.

The comparative summary of the hyper-parameters chosen for all the three models outlined in Table 1. In the following we shortly recall the most important parameters:

- *Learning Model* selects the most appropriate ML algorithm for the specific problem we are solving and designs the appropriate neural network architecture.
- *Training Algorithm* represents the selection of the activation function. Since our problem is a multi-class logistic regression we choose the Hierarchical Softmax function, which will allow the system to decrease its computational complexity, through the use of a binary tree structure. When using SGNS though, the loss function is applied to a binary problem, so a Softmax with negative sampling can be used.
- *Embedding Size* (dimension) is the length of the vector representations of words and sentences. To compare the models, the initial value was set to 128 for all the models, based on online community advice related to the objective of the work.
- The *window size* is related to the context window L, the maximum distance from our target word when considering the context neighborhood words.
- *Min_count* is the minimum frequency for a word to be considered during the training process.

- The *number of epochs* are the training steps, that correspond to an entire processed dataset forward and backward. This parameter affects three factors: the training accuracy (percentage of correctly labeled examples during the training), the validation accuracy (the precision on the testing set), and the cross-entropy (it is a loss function that shows the cross-validation accuracy). In general, with a small number of epochs, there is a risk of not having a sufficiently trained model, which results in underfitting the model. On the other side, validation accuracy typically starts to decrease after some number of epochs, because of the overfitting. It is typical to use more than one iteration, but as input data gets larger, the benefits of extra iterations decrease.

Table 1: Summary of the chosen parameters

Parameters	**d2v_model**	**GenW2V_model**	**TFw2v_model**
Corpus	Specific domain + Brown Corpus	Specific domain + Brown Corpus	Specific domain + 1/10 Text8 Corpus
Corpus Length	66,575 sents	2,421,344 words	17,005,207 words
Learning model	PV-DBOW/PV-DM	SGNS	SGNS
Training Algorithm	Hierarchical Softmax	Softmax with Negative Samples	NCE loss
Learning Rate	0.025	0.025	0.025
Embedding Size	128	128	128
Window size	L=5 wndow_size=11	L=2 wndow_size=5	L=2 wndow_size=5
Min_count	1	2	5
Bath_size	sentence based	32	adaptive
Epochs	20	15	2

4. Computational results

The evaluation of our solution was performed along two axes. First, it was necessary to assess the effect of embedding the developed methods in the existing overall process. Second, the actual precision of the models in identifying the right HF was considered, by comparing the punctual identifications with the ones performed by the experts. During this process, we used the reports provided and checked by the aviation experts from Deloitte and the support of their experts (Deloitte & Touche, 2020). We considered tagged 24 documents provided by the company. The documents were split into a training (20 documents) and a test set (4 documents). A Monte Carlo Cross-Validation was performed by repeating 10 times the process and randomly choosing the training and the test documents (Xu & Liang, 2001).

This amount of data along with the basic material with the definitions and explanations of ones was enough to make the system sufficiently reliable. The output of the automated annotations are then compared with the annotations done by real investigators are used for result validation. For both the evaluations, we identified a model that performs better with respect to the others. Although, it is possible to notice that the overall strategy was reasonably acceptable.

The first axis of evaluation is represented by the embed training results, which are outlined in Table 2. The benchmark values were taken from Google's pre-trained *Word2vec* model, which is a 300-dimensionality model trained over a 100 Billion corpus. This is considered to be the state-of-the-art model for the general purposes of word embedding. It is a good evaluation parameter for general text classification, even not relevant for our purpose of this paper, which is the prediction of the entire sentence meaning. However, this work-in-progress outcome is useful to have a broad overview of how our models behave in general.

Table 2: The embed training results

Parameters	**d2v_model**	**GenW2V_model**	**TFw2v_model**	**Benchmark**
Time for Model Creation	00.00.03	00.01.31	00.00.01	-
Time to load the corpus	00.02.01	00.02.01	00.16.38	-
Time for Model Training	00.13.44	00.08.26	00.28.46	1-3 days
Number of vocabs learnt	34184	34184	211080	3000000
Loss Drop	43%	21%	90%	98.5%
Accuracy	19%	23%	76%	96%

As expected, the simplest model, *Genw2v_model*, is the fastest in building, but also less effective than *TFw2v_model*. This does not mean that it cannot be anyway exploited and give useful results further on, considering that our final task is sentence similarity and not word embed. In fact, as will be shown later, the difference between the two models reduces drastically when we deal with the real goal of our prediction, i.e., the tagging of entire sentences of a document. For all the models the standard deviation due to the Monte Carlo Cross Validation was less than 0.025.

A few considerations over the similarity values computed with the three models need to be said:

- all the sentences are processed using our enhanced Lemmatizer system. This increases the chances of finding similarities because words with the same root are considered to be identical.
- it is important to identify more than one possible HF because we cannot assume that a machine would be able to replace completely the role of the investigator. This is why the system is meant to give a list of *n* HFs with the highest similarity scores.
- looking at the whole set of sentences, the value of the cosine similarity was pretty high on average. This may give a multiple HF prediction, each one critical, bringing later to a manual check to improve the overall prediction system.

Table 3 considers the second axis of evaluation concerning the specific application. The table shows the performances of the three models over the accidents processed with a specific corpus extracted from a repository of existing documents. The sentence embedding refers to the method used to embed sentences out of word vectors, which - for the *d2v_model* - is automatically done in the training phase. The table reports the precision and recall measures. We do not report the standard deviation due to the Monte Carlo Cross Validation, being in all the cases very low (between

0.01 and 0.02). To this end, we outline the percentage of correctly identified HFs in all the test set as the precision, and the recall is the missing HFs. Moreover, the correctly identified HFs in the top-five list are grouped by the SHEL tags, namely: Software, Hardware, Environmental, Liveware Pilots and Operators. In this way, it is possible to show the model's sensitivity to different HFs. However, Precision and recall are not sufficient to qualify the methods due to the specific setting. Indeed, the system relies on the knowledge of the specialist in the field, which should explore the highlighted points in the report. We also provide the Cosine Similarity Threshold, i.e., the minimum level of similarity in the document part structure that it has to reach to be considered relevant. Among the three different models, the one performing best is the $TFw2v\_model$, with a total precision of $88,89\%$. Although, the second-best model, $Genw2v\_model$, gives a great precision as well ($86,67\%$), an average rank for the correct HF in the top-five list of 2.08 against the slightly worse 2.27 of the $TFw2v\_model$, and a distance of the correct HF score with respect to the first position is of only 0.018. If we consider the time required to create and train the $Genw2v\_model$, the smaller corpus used, and the fact that the precision is only worse by 2.22%, we can consider the $Genw2v\_model$ as the most successful one for our purpose.

Not only it is relatively simpler and faster, but, if we imagine training it over a larger corpus like the $Text8$ used for the $TFw2v\_model$, we can predict that the performances would improve even more, and eventually out-stand the $TFw2v\_model$'s outcome. The drawback is that by increasing the training corpus' size, the time for building the model would increase as well. Among the three models, the one performing the worst is $d2v\_model$. This is explained by the fact that Doc2vec is a general model for document embedding, which returns the document vector, independently from the actual document's size. Additionally, the sentences composing the corpus over which the model was trained are not particularly similar to the ones that are being processed, and that negatively influences the similarity measure.

5. On-field test of the solution

To prove the efficacy of the proposed method we did an on-field test, integrating our solution in the proprietary tagging system of Deloitte. The test was performed considering 5 additional documents already tagged by the experts of Deloitte and then asking our system to tag them according to the SHEL.

Concerning the accuracy of the system, the results showed precision in tagging the document of 86%. As a second-level evaluation, we considered the punctual classification of the sentences. We compared the results with the manually processed by the investigator and therefore with the correct HF already identified. Knowing the correct HF belonging to each processed sentence, it was possible to compare the system outcome with the expected results, for each model implemented. The experts by Deloitte checked the punctual classification given by our solution (in some cases several classifications are possible) and evaluated the accuracy of the second-level tagging in 67%. Table 4 presents the outcome for a subgroup of evaluated events. For each event, we first checked if the correct HF had been identified by the system in the n-length list of potential HF. Since the list is ordered based on the similarity score, the rank of the correct HF in the list was also captured (position). Additionally, for those correct HFs which were included in the n-length list but were not in the top position, we registered the error as the distance between the correct HF score and the score of the HF ranked at the top position by the system. For example, for those events in which the correct HF was identified and ranked at the top position by the system, the distance was set to 0.

We then asked some experts of Deloitte to evaluate the results of the tagging and the impact of the proposed solution inside the entire process (tagging, identification of the root causes, and proposal of recommendations) from three points of view: cost saving, air safety, process automation, and standardization.

Concerning the pure cost-saving, they evaluated a save of about 30% of the expert time (including the check of the tagging done by them). Being this cost reduction mainly due to the salaries of well-trained investigators, this has a double side effect. First, the obvious decrease in the total costs for the company, making Deloitte more competitive. Second, normally each investigator has specific expertise. Reducing the effort needed to analyze a document can free human resources for a deeper and comparative analysis of the reports to find similarities of the series of root causes, which are not obvious at a first sight.

And this brings us to the second axis of industrial impact: aviation safety. The contraction of the time needed for the annotation of the reports might have a direct impact on the total time for the identification of the causal chain of events leading to the root of the problem up to 18 months. Moreover, by letting the experts focus on a portion of the document first (the tagged part), they might be able to analyze some of the causes immediately, speeding up the process of the root causes identification. Their evaluation in the contraction of the time between the tagging and first hypotheses on the root causes was estimated to be up the 20%.

These are just the direct impact of the application of our automated methodology. Indeed, an indirect effect is the standardization of the annotation procedure, with the consequent possibility to use the output of the standardized tagging to train more complex Artificial Intelligence systems able to highlights the more probable sequence of causes beneath a series of accidents.

6. Conclusions

As stated in the introduction, automatizing the operation of tagging air accidents documents is a crucial task for two reasons: it reduces the time needed to analyze the document itself, reducing the time before the accident and the identification of the root causes of the accident itself, while reducing the cost due to this operation.

The results of our methodology fully meet the goals of the research. We presented an alternative way to deal with the identification of human factors within unstructured text, by proposing different approaches to the basic task. Basing the solution on unstructured text processing proved to be a viable option and promising one for the future, thanks to the huge amount of available data (big data) and the collaboration of open source communities democratizing machine learning/deep learning algorithms. In summary, the system has been tested over case studies entailing safety events with different severity, including near-miss incidents, minor incidents, and serious incidents involving loss of human lives, whether the usual time to investigate those events and annotate the documents spans from some weeks up to 18 months and over. The tests gave a precision over the 86% and the Deloitte experts estimated in a practical manner cost and time reduction of 30% for the whole investigation process.

Table 3: Performance comparison of the three models over the different type of cause

Comparison parameters	**d2v_model**	**GenW2V_model**	**TFw2v_model**
Sentence embedding	Automatic Concatenation	Average method	SIF method
Cosine Similarity Treshold	0.05	0.5	0.5
Software (S)	**100%**	**100%**	**100%**
Hardware (H)	53.36%	84.62%	**92.30%**
Environment (E)	62.50%	75.00%	**87.50%**
Liveware Pilots (LP)	71.43%	**85.71%**	**85.71%**
Liveware Operators (LO)	61.54%	**92.30%**	84.61%
Precision	**64.45%**	**86.67%**	**88.89%**
Recall	**36.55%**	**14.32%**	**12.1%**
AVG score found	0.1952867	0.865041	**0.894419**
AVG rank	2.89	**2.08**	2.27
Distance wrt first position	0.033	**0.018**	0.029

One of the possible improvements of the work is the study of different similarity measures influence of the proposed models. Another future enhancement of this work would be to add, during the pre-processing phase, a parser system that gives important grammatical information over the structure of a sentence, by organizing it in a logical tree. This additional task would increase the accuracy of the representation of the words, allowing a more reliable system. Finally, after an application of the methodology to a larger dataset, an automated root causes sequence identification might be done, similarly to the one done in Cantamessa et al. 2018. When talking about semantic similarity over sentences, it is never easy to get a starting reliable dataset. While for single word comparison it might be more obvious, giving a good measure on how the sentences are similar to each other is a task that many researchers are trying to solve. The problem is that to train effectively neural networks, the amount of data needed is “big” (clearly a big data problem), and currently there is not a suitable dimension of available data over a specific domain such as Aviation Safety Management. This problem can be overcome by the use of the implemented solution: the knowledge base currently used is increasing for every new report processed, and, when becoming “big enough”, it could potentially be leveraged as a new training dataset, more structured than just raw text, to train a neural network-based model for automatic sentence semantic similarity. The technology to train a model over such a dataset is already available (Mueller & Thyagarajan, 2016; Neculoiu et al., 2016). This idea is applicable not only in aviation; one of the most important fields where such a solution can be relevant is healthcare, where a system that helps to compare unstructured documents like case studies and diagnosis would have positive consequences on human beings’ lives. An industrialization process of the methodology is presently under development.

Table 4: Genw2v_model outcome for a subgroup of events evaluated

Target Sentences	**Manual HF identification**	**Score**	**Position**	**Distance wrt the first position**
The malfunction of the engine was due to intermittent contact cable W450-P4	Equipment failure	0.90691	1	0.00000
There were not reports of MASTER CAUTION signals neither beeps on the Head up display of the Ground Station	Worspace: Communication Equipment	0.81637	4	0.0453
The aircraft was inappropriate for long range travel, because it is not provided with APU	Equipment failure	0.90885	1	0.00000
The operation room had no air conditioner	Equipment failure	0.81513	2	0.0345
The pilot noticed the OVERTORQUE warning light had illuminated	Instrument design and illumination	Not found	Not found	Not found
The flag on the torque of the transmission confirmed the warning	Instrument design and illumination	Not found	Not found	Not found
The helicopter entered an uncontrolled descend and impacted water	Equipment failure	0.83243	4	0.0546
The rotor lost speed because of the over-torque	Equipment failure	0.77647	3	0.0579
The aircraft type has limitations in lateral and frontal view	Workspace: visibility restrictions	0.83321	3	0.0570
The colour of the Fire Extinguisher does not allow visibility in bad weather conditions	Workspace: layout	0.82913	5	0.0558
The fire extinguisher was stuck in the belly of the aircraft and the wheel bars	Workspace: layout	0.87973	1	0.0000
Absence of specific rain clothes	Equipment failure	0.85605	2	0.0402
There was a failure in the aircraft avionics	Equipment failure	0.83368	3	0.0188

Acknowledgments

While working on this paper, Guido Perboli was the head of the Urban Mobility and Logistics Systems (UMLS) initiative of the Interdepartmental Center for Automotive Research and Sustainable mobility (CARS) at Politecnico di Torino, Italy and R&D Director of ARISK, a Spin-off of Politecnico di Torino.

References

Aviation Safety Improvement Task Force (2005). Department of defense human factors analysis and classification system: a mishap investigation and data analysis tool. *Kirtland AFB: Air Force Safety Center*, .

Cantamessa, M., Gatteschi, V., Perboli, G., & Rosano, M. (2018). Startups' Roads to Failure. *Sustainability*, *10*, 2346. doi:`10.3390/su10072346`.

Castrogiovanni, P., Fadda, E., Perboli, G., & Rizzo, A. (2020). Smartphone data classification technique for detecting the usage of public or private transportation modes. *IEEE Access*, *8*, 58377–58391. doi:`10.1109/ACCESS.2020.2982218`.

Cuzzocrea, A., Pilato, G., & Fadda, E. (2020). User emotion detection via taxonomy management: An innovative system. In *CEUR Workshop Proceedings* (pp. 334–342). volume 2646.

Deloitte, & Touche (2020). Deloitte s.p.a. official site. URL: `https://www2.deloitte.com/global/en.html?icid=site_selector_global` Last access: 16/08/2020.

Francis, W. N., & Kucera, H. (1979). *Brown Corpus Manual*. Technical Report Department of Linguistics, Brown University, Providence, Rhode Island, US. URL: `http://icame.uib.no/brown/bcm.html`.

Hawkins, F. (1993). *Human Factors in Flight*. Ashgate Publishing Co. Ltd, Aldershot, England.

Hu, X., Wu, J., & He, J. (2019). Textual indicator extraction from aviation accident reports. In *AIAA Aviation 2019 Forum* (p. 2939).

ICAO (1993). Circular 240-an/144. *Human Factors Digest No. 7, Investigation of Human Factors in Accidents and Incidents*, .

ICAO ADREP (2019). ADREP. URL: `https://www.skybrary.aero/index.php/ICAO_ADREP` last access: 16/08/2020.

ICAO Safety (2018a). ICAO Safety Report. URL: `https://www.skybrary.aero/bookshelf/books/4431.pdf` last access: 16/08/2020.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196).

Mahoney, M. (2006). Rationale for a large text compression benchmark. Retrieved (Aug. 20th, 2006) from: https://cs.fit.edu/ mmahoney/compression/rationale.html.

Mahoney, M. (2011). About the test data. `http://mattmahoney.net/dc/textdata.html`. Last access 16/08/2020.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, .

Mirończuk, M. M., & Protasiewicz, J. (2018). A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, *106*, 36 – 54. doi:`https://doi.org/10.1016/j.eswa.2018.03.058`.

Mosca, F. (2015). *A support model to draft safety recommendations as follow up over investigation on an aviation safety occurrence*. Master's thesis Polytechnic University of Turin.

Mueller, J., & Thyagarajan, A. (2016). Siamese recurrent architectures for learning sentence similarity. In *thirtieth AAAI conference on artificial intelligence* (pp. 2786–2792).

Neculoiu, P., Versteegh, M., & Rotaru, M. (2016). Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP* (pp. 148–157).

Pagliardini, M., Gupta, P., & Jaggi, M. (2017). Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, .

Plioutsias, A., Karanikas, N., & Tselios, D. (2018). Decreasing the distance between international standards from different domains: The case of project management and aviation safety investigations. *AUP Advances*, *1*, 7–39.

Reason, J. (1990). *Human error*. Cambridge university press.

Reason, J. T. (1992). Cognitive underspecification. In *Experimental Slips and human error* (pp. 71–91). Springer.

Sebleier (2010). NLTK's list of english stopwords. URL: `https://gist.github.com/sebleier/554280` Last access: 16/08/2020.

Semaan, P. (2012). Natural language generation: An overview. *Journal of Computer Science & Research*, (pp. 50–57).

Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D. (2014). Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, *18*, 491–504.

Simmons, S., & Estes, Z. (2006). Using latent semantic analysis to estimate similarity. In *Proceedings of the Cognitive Science Society* (pp. 2169–2173). volume 5.

Soğancıoğlu, G., Öztürk, H., & Özgür, A. (2017). Biosses: a semantic sentence similarity estimation system for the biomedical domain. *Bioinformatics*, *33*, i49–i58.

Srinivasan, P., Nagarajan, V., & Mahadevan, S. (2019). Mining and classifying aviation accident reports. In *AIAA Aviation 2019 Forum* (p. 2938).

Sugathadasa, K., Ayesha, B., de Silva, N., Perera, A. S., Jayawardana, V., Lakmal, D., & Perera, M. (2017). Synergistic union of word2vec and lexicon for domain specific semantic similarity. In *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)* (pp. 1–6). IEEE.

Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, *37*, 141–188.

White, L., Togneri, R., Liu, W., & Bennamoun, M. (2015). How well sentence embeddings capture meaning. In *Proceedings of the 20th Australasian Document Computing Symposium* (p. 9). ACM.

Xu, Q.-S., & Liang, Y.-Z. (2001). Monte carlo cross validation. *Chemometrics and Intelligent Laboratory Systems*, *56*, 1–11. doi:`https://doi.org/10.1016/S0169-7439(00)00122-2`.