# Company Data Analysis Summary

Antoine J.-P. Tixier, Matthew R. Hallowell

June 8, 2020

SAFETY AI



**How to cite this report?**
Antoine J.-P., Tixier and Hallowell, Matthew R. Company Data Analysis Summary. *Safety AI technical report*. June 8, 2020. URL: https://github.com/safetyAI/Company_report_public/blob/master/Company_report_public.pdf

**Information, contact and feedback**:
https://www.safetyfunction.com/safety-ai-details
matthew.hallowell@safetyfunction.com or antoine.tixier@safetyfunction.com

# Contents

## List of Figures

## List of Tables

# 1 About this report

The present document is a technical report that was delivered by Safety AI as part of a consulting project. It is made publicly available in an anonymized form with expressed consent from the client. The objective of making this report publicly available is twofold: giving future Safety AI's clients a better idea of what they can expect out of a project, and transparently sharing methods with the construction safety research community and practitioners.

**Note**: in order to get an official third-party time stamp, this report was uploaded to GitHub, a popular version control repository and hosting service. It can be found in the following repository: https://github.com/safetyAI/Company_report_public. All future versions will be uploaded to that same repository to maintain a clear version history.

## 1.1 How to cite this report?

If you use some of the ideas and methods presented in this report in your research or your own work, or if you just need to refer to this report, in a literature review for instance, please use the following reference:

Antoine J.-P., Tixier and Hallowell, Matthew R. Company Data Analysis Summary. Safety AI technical report. June 8, 2020. URL: https://github.com/safetyAI/Company_report_public/blob/master/Company_report_public.pdf

To refer specifically to the methodology used in some sections of this report, you might also consider citing the following journal papers, in addition to the present report:

› Machine Learning
• Baker, H., Hallowell, M. R., & Tixier, A. J. P. (2019). AI Predicts Independent Construction Safety Outcomes from Universal Attributes. *arXiv preprint* arXiv:1908.05972 (**accepted for publication in Automation in Construction**) https://arxiv.org/pdf/1908.05972.pdf

› Risk Analysis
• Tixier, A. J. P., Hallowell, M. R., & Rajagopalan, B. (2017). Construction safety risk modeling and simulation. *Risk analysis*, 37(10), 1917-1935. **[link]**

› Word Embeddings
• Tixier, A. J. P., Vazirgiannis, M., & Hallowell, M. R. (2016). Word embeddings for the construction domain. arXiv preprint arXiv:1610.09333. https://arxiv.org/pdf/1610.09333.pdf

› Deep Learning
There is no deep learning section in the present report, as the client was not interested in this offering. We refer the interested reader to our recent publication:
• Baker, H., Hallowell, M. R., & Tixier, A. J. P. (2019). Automatically learning construction injury precursors from text. arXiv preprint arXiv:1907.11769 (**accepted for publication in Automation in Construction**) https://arxiv.org/pdf/1907.11769.pdf

The following illustrational video is also available:

**Deep learning live video demonstration**

https://vimeo.com/404968330

### 1.2 Contact

If you are interested in collaborating with Safety AI, or if you simply want to learn more about the Safety AI research and offerings, please visit https://www.safetyfunction.com/safety-ai-details or email matthew.hallowell@safetyfunction.com or antoine.tixier@safetyfunction.com

## 2 Summary

The present report accompanies the two interactive web applications that were developed to expose the output of our analyses.

› The desktop application, shown in Fig. 1, exposes the output of all analyses that were conducted as part of this project, with many tuning parameters, visualizations, and download capabilities. The desktop application is meant to be used by people in the office, such as data scientists and analysts, and requires a screen of at least 13-14".

**Desktop application live demo video**

While the desktop application is not publicly available, live video demonstrations of its main features can be watched at https://www.safetyfunction.com/safety-ai-details. Whenever available, links to videos will be directly provided throughout this report in boxes such as the current one, featuring a icon.

› The mobile application, shown in Fig. 2, is a simpler, lighter version of the desktop application. It features only the functionalities that are the most useful onsite. It was designed to be used onsite by site managers, safety managers and crew leaders. The mobile application can be saved to the home screen of the mobile device and be used full-screen, just like a native application. However, it is still a web application, so, no installation is necessary, and it is compatible with both Android and iOS devices.

**Mobile application live demo video**

https://vimeo.com/421593720

This report follows the organization of the tabs in the desktop app, as shown below:

1. Diagnostics (section 5),
2. Field (section 6),
3. Risk (section 7),
4. Text search (section 8),
5. Word embeddings (section 9),
6. Topic modeling (section 10).

For each tab, details about (1) what it does, (2) how it can be used in practice, and (3) the underlying methodology and analyses are provided. In addition, a brief overview of the attribute framework and the NLP tool is provided in section 4.

In addition to the video link boxes, three types of boxes are used throughout this report to highlight important information. The *What questions can this part answer?* boxes (featuring the icon) provide a few concrete examples of intelligence that can be gained from using a particular functionality

of the app. The boxes featuring the ⚙ icon are used to interpret the analyses and results in simple terms. Finally, the boxes with the ☑ icon provide a high level summary of the concepts that have been covered in the preceding section or subsection.

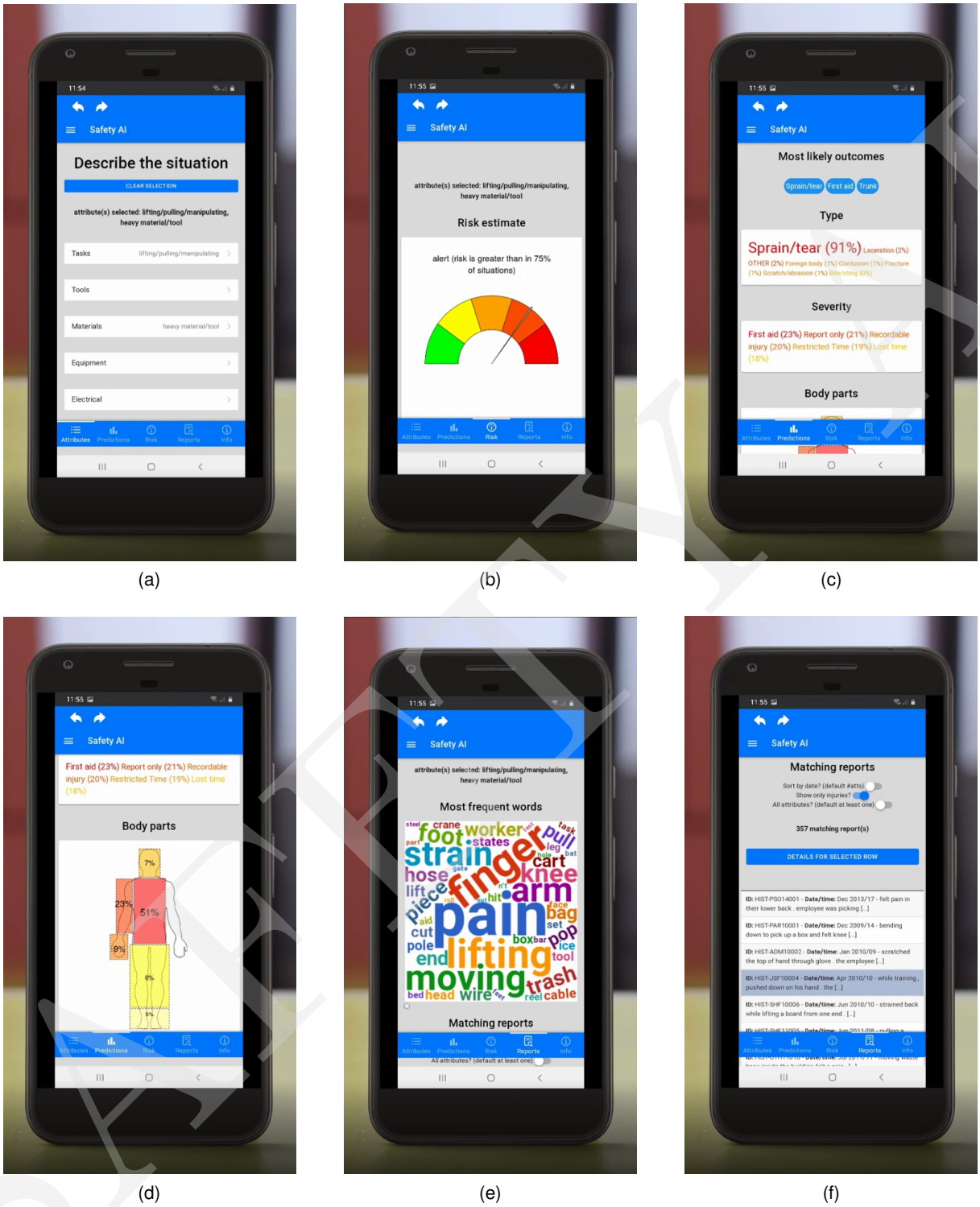
Note that in the application, directions and instructions are available to the user on almost every tab through the **i** buttons.

Together with the present report, we are also delivering, in a directory whose structure is detailed in the next section (section 3), some code and all the files generated during our analyses[1]. These files include, but are not limited to, the output of the NLP tool, the plots shown in this report, the machine learning models grid search results and final weights, the word embeddings and document embeddings models, and all the files used by the application.

(a) (b) (c) (d)

Figure 1: Some tabs from the desktop application: (a) word co-occurrence network visualization and keyphrase extraction, (b) deep learning (predictions, attention visualization, and document embeddings), (c) injury characteristics forecast, (d) risk analysis.

[1]Not made publicly available.




(a) (b) (c)

(d) (e) (f)

Figure 2: The mobile application is a simpler, lighter version of the desktop application. It can be saved to the home screen of the mobile device and be used full-screen, just like a native application. However, it is still a web application, so, no installation is necessary, and it runs equally well on both Android and iOS devices. The mobile app features only the functionalities that are the most useful onsite: (a) attribute selection, (b) risk estimate, (c) injury type and severity prediction, (d) body part injured prediction, (e) matching reports wordcloud, (f) matching reports list.

# 3 Files delivered

```
+---code
|   |   data_checks_normalization.py
|   |
|   \---machine_learning
|           predict.py
|
+---data
|   |   normalized_labels.csv
|   |   text_for_nlp_tool.csv
|   |   text_only.csv
|   |   text_only_cleaned.txt
|   |
|   +---deep_learning
|   |       text_only_cleaned_dl.txt
|   |       text_only_cleaned_ints.txt
|   |       token_counts.json
|   |       vocab.json
|   |
|   +---dicts_counts
|   |       body_labeldict.json
|   |       ...
|   |       SourceOfInjury_normcounts.json
|   |
|   +---IR
|   |       basic_stopwords.txt
|   |       custom_stopwords.txt
|   |       data_stats
|   |       full_stopwords.txt
|   |       inverted_index
|   |       inverted_index_text.pkl
|   |       inverted_index_text_stem.pkl
|   |       prep_cloud_with
|   |       prep_cloud_without
|   |       prep_gow_basic
|   |       prep_gow_full
|   |       prep_html
|   |       prep_html_stemmed
|   |       SMART_stopwords.txt
|   |       stopwords_text_ir.txt
|   |       suggestions.txt
|   |       time_series
|   |       time_series.json
|   |       time_series_atts
|   |       time_series_nbnas.json
|   |       time_series_stem.json
|   |       time_series_stem_nbnas.json
|   |       to_query
|   |       Company_att_names.txt
|   |       Company_att_names_risk.txt
|   |       Company_att_out_risk_names
|   |       Company_report_risk.json
|   |       Company_risk_table.csv
|   |       Company_risk_values.json
|   |       Company_var_names
|   |       Company_var_names_selectize.txt
|   |
|   +---machine_learning
|   |   |   cat_weights_Classification.json
|   |   |   cat_weights_Nature.Of.Injury.json
|   |   |   cat_weights_Part.Of.Body.json
|   |   |   ensemble_logs.txt
|   |   |   read_results_logs.txt
|   |   |   train_val_test_ids.json
|   |   |
|   |   +---grids
|   |   |       boost_Classification_confs_2019-11-04_14-12-09.json
|   |   |       ...
|   |   |       svm_Part.Of.Body_params_2019-11-04_14-12-09.csv
|   |   |
|   |   \---models
```

```
|   |           boost_Classification_2019-11-04_14-12-09
|   |           ...
|   |           svm_Part.Of.Body_testperf_2019-11-04_14-12-09.json
|   |
|   +---risk_analysis
|   |       Company_Attribute_exposure_survey.csv
|   |       Company_attribute_exposure_survey_follow-up.csv
|   |       Company_exposure_survey_att_names.txt
|   |       Company_hist
|   |       Company_ranges.txt
|   |
|   +---tool_output
|   |       Company_binary_full_2019-11-04_14-12-09.csv
|   |       Company_binary_full_2019-11-04_14-12-09_overwritten.csv
|   |       Company_names_full_2019-11-04_14-12-09.csv
|   |       Company_names_full_2019-11-04_14-12-09_overwritten.csv
|   |
|   +---topic_modeling
|   |       ldavis.json
|   |
|   \---word_doc_2vec
|           d2v
|           tsne.txt
|           w2v
|           w2v_vocab.txt
|
\---plots
    |   sent_distr.pdf
    |   word_distr_doc.pdf
    |   word_distr_sent.pdf
    |   word_embeddings.pdf
    |
    +---machine_learning
    |       boost_Classification_attimp_2019-11-04_14-12-09-crop.pdf
    |       ...
    |       svm_Part.Of.Body_attimp_2019-11-04_14-12-09.pdf
    |
    \---risk_analysis
            severity_scores.pdf
            severity_scores_cropped.pdf
            Company_risk_hist_kde.pdf
            Company_risk_hist_kde_cropped.pdf
```

**Details about the files**. Note: to manually inspect the `.txt` and `.json` files, we recommend opening them with an advanced text editor like `notepad++`[2].

- the `/code/` folder contains the script that was used for outcome normalization (see subsection 6.1.1), and a script that shows how to load the trained machine learning models and use them to generate predictions (see section 6).
- the `/data/` folder contains all output files that were generated as part of our analyses. More precisely:
  - the `/deep_learning/` subdirectory contains the cleaned reports, in a format suitable to be used for training deep learning models. Even though deep learning was not used in this project, we provide these files because they could prove useful in the future, and because we used them to build the inverted index of the text-based search engine (see section 8) and for topic modeling (in section 10).
  - the `/dicts_counts/` subdirectory contains the dictionaries mapping outcome category names to their acronyms and the counts per category, after normalization.
  - the `/IR/` subdirectory (IR stands for Information Retrieval) contains the files that are used by the Diagnostics and Text search tabs of the application.

[2] https://notepad-plus-plus.org/

  - the `/machine_learning/` subdirectory contains the results of the grid searches (parameter optimization) and the weights of the final trained models, used by the Field tab of the app.
  - the `risk_analysis` subdirectory contains the responses of the two exposure surveys that were sent out to Company, and the histogram and ranges used by the Risk tab of the app.
  - the `tool_output` subdirectory contains the output of the NLP tool.
  - the `word_doc_2vec` subdirectory contains the trained word2vec and doc2vec models, along with the vocabulary and coordinates of the words in the 2D space. These files are used by the Word embeddings tab and by the semantic search engine of the Text search tab.

- Finally, the `/plots/` folder contains various plots that are used throughout this report.

# 4 Attribute-based framework and NLP tool

**Illustrational video**

https://vimeo.com/406817894

The attribute-based framework [17, 44] allows any construction situation to be uniquely and comprehensively described by a finite number of attributes that are observable *before* any injury occurs. These basic descriptors of the jobsite are universal, independent of any context, and span construction means and methods, environmental conditions, and human factors. To illustrate, in the following excerpt of an injury report: "*employee tripped on a cable while exiting job trailer*", 4 fundamental attributes can be identified: (1) cable, (2) object on the floor, (3) exiting/transitioning, and (4) job trailer. To automatically extract attributes from text, we used the most recent version of the NLP tool introduced in [40], and expanded it to detect 11 additional attributes specific to the electrical domain and Company: clearance, LOTO/labeling, transformer, pole, vault, breaker, relay, switches, fuses, fan, and heater. In addition to the full set of attributes (shown in Table 12), the NLP tool is also capable of extracting 3 categories of outcomes, shown in Table 1.

| Energy source | Injury code | Body part |
|---|---|---|
| biological | caught in/compressed | head |
| chemical | exposure to harmful | neck |
| electricity | fall on same level | trunk |
| gravity | fall to lower level | upper extremities |
| mechanical | overexertion | lower extremities |
| motion | struck by/against | not detectable |
| pressure | transportation accident | |
| radiation | | |
| thermal | | |

Table 1: Outcomes extracted by the NLP tool.

**Key points**

- Any incident can be viewed as the resulting outcome of the joint occurrence of some **fundamental attributes** and the presence of a worker.
- The NLP tool, based on the attribute framework, allows the automatic extraction of **standardized information** from raw injury reports.

The text associated with each Company incident case (injury and non-injury) was processed by the NLP tool. That text was extracted from the following columns: `Short Description`, `What Happened`, `Why Occur`, `Immediate Actions`, `Comments`, `Activity`, `Cause`, `Description`, and `Object`. For the non-injury reports, the outcome columns were overwritten with NA (except energy source). The files for which such post-processing was applied can be found in the `/data/tool_output/` subdirectory as the `overwritten` files, along with the original files. The overwritten files are the ones that we used for all our analyses.

# 5 Diagnostics

The Diagnostics tab of the application features three subtabs: (1) Statistics (see section 5.1 below), (2) Keyphrases (section 5.2), and (3) Attribute networks (section 5.3). All three subtabs share the same filtering interface, shown in Fig. 3.

On that interface, the user can apply filters based on all Company variables (including time and date), and based on the attributes and outcomes extracted by the NLP tool (energy source, injury code, and body part). In the remainder of this report, we refer to the latter set of variables as the SafetyAI variables. Note that in the web app, for space constraints, most of the Company variables, which tend to have long names, have been abbreviated. The user can lookup any abbreviation using a dedicated interface at the bottom of the filtering interface. The acronym lookup functionality is also available on the Field tab (see section 6). We now provide details about each subtab.

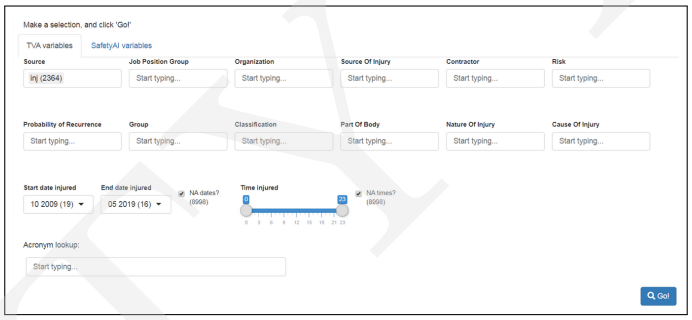



Figure 3: Shared filtering interface of the Diagnostics tab.

## 5.1 Statistics

**Live video demonstration**

https://vimeo.com/411151993

A screenshot of the statistics tab is shown in Fig. 4. The distributions of Company and SafetyAI variables along with the risk ranges computed in section 7 are displayed for the combination of filters selected by the user under the 'Company variables' and 'SafetyAI variables' subtabs, respectively.

The distributions of the number of matching cases over time (hourly and historical/monthly) are also displayed in the 'Time and date subtab' (see Fig. 5). Note that date information is only available for the 2,364 injury reports, and time information for 2,175 of the injury reports.

Finally, the matching reports themselves can be inspected in the 'Matching report(s)' subtab, as shown in Fig. 6. The user can customize the table of results and interactively navigate the rows. The user also has the ability to report any error made by the NLP tool directly from the interface, as shown in Fig. 7. Note that a similar interface can be opened from the table of results of the Text search tab (see section 8).

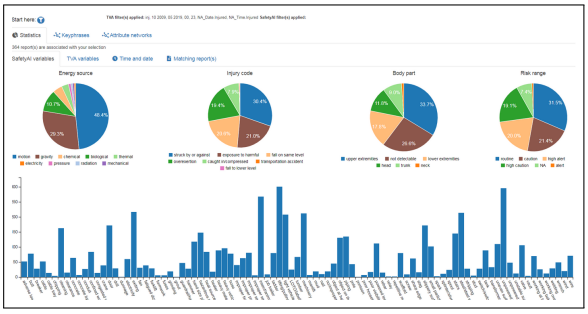


Figure 4: Statistics subtab (SafetyAI variables).

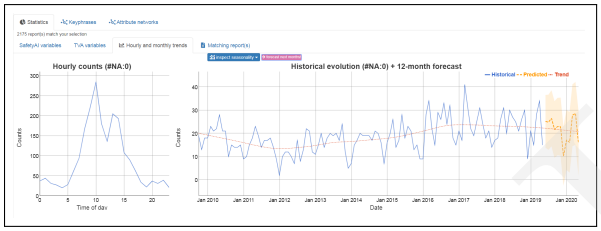


Figure 5: Statistics subtab (time and date).

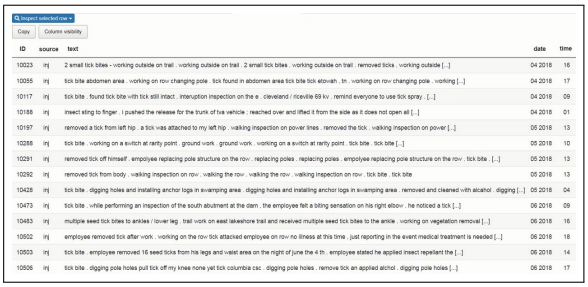


Figure 6: Statistics subtab (SafetyAI variables).

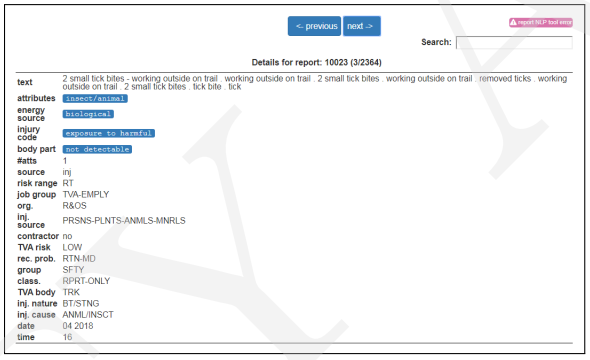


Figure 7: NLP tool error reporting interface of the Statistics subtab.

**What questions can this part answer?**

- Which attributes are the most frequent in the entire dataset?
- Which attributes prevail for *fall on same level* cases?
- How many near misses are associated with high risk?
- What are the expected injury counts in the months to come?

› The most frequent attributes at the dataset level are *unpowered tool* (1,329 occurrences), *object on the floor* (1,198), and *improper procedure/inattention* (1,179).

When filtering the dataset to retain only the 2,364 injury cases, the first and third most frequent attributes become *lifting/pulling/manipulating* and *insect/animal*, respectively with 301 and 268 counts, and *unpowered tool* is second, with 296 counts.

› We can inspect the distribution of attributes for *fall on same level* reports by entering `FALL-SME` on the `Company variables/Cause Of Injury/` search bar of the filtering interface. We can observe that 483 reports are returned, and that the top attributes for these reports are *slippery surface* (count of 130), *exiting* (97), *uneven surface* (81), *stairs* (79), *object on the floor* (69) and *light vehicle* (62).

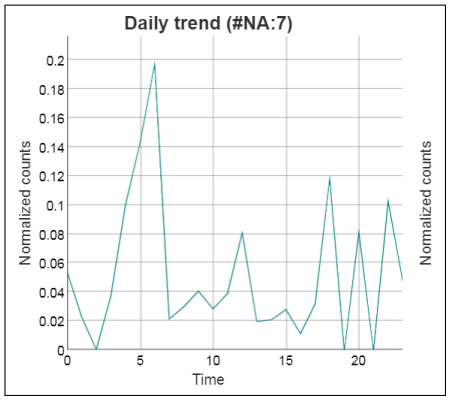



Figure 8: Daily trend for *fall on same level*, *exiting* cases.

The presence of *exiting* is interesting. We can inspect the corresponding reports by entering

`exiting` in the `SafetyAI variables/attributes/` search bar of the filtering interface and then going to the Matching report(s) subtab.

Indeed, we observe that many *fall on same level* reports involve people exiting vehicles or places or transitioning from one place to another. Some examples include 'employee felt pain when exiting boat', 'when he stepped out of the vehicle he slipped', 'individual exited ride share van and tripped', 'slipped exiting crane', 'employee was transitioning down from a set of stairs to the ground', etc. Finally, looking at the daily trend plot of the 'Time and date' subtab (see Fig. 8), we can see that most of the *fall on same* cases involving the *exiting* attribute occur early in the morning (around 6 am).

› Entering near miss (NR-MSS) in the `Company variables/Classification/` search bar of the filtering interface, and then `high caution`, `alert` and `high alert` in the `SafetyAI variables/risk range/` search bar returns 715 results. This means that out of the 1,454 near misses, almost half (49.17%) are associated with high risk, and that therefore, many close calls have the potential to create serious injuries.

› For any filter combination, the user can analyze the time series of records, with a dedicated seasonality button, an overlaid trend, and the ability to get predictions for up to 12 months into the future (powered by an ARIMA model[3]). See figures below.

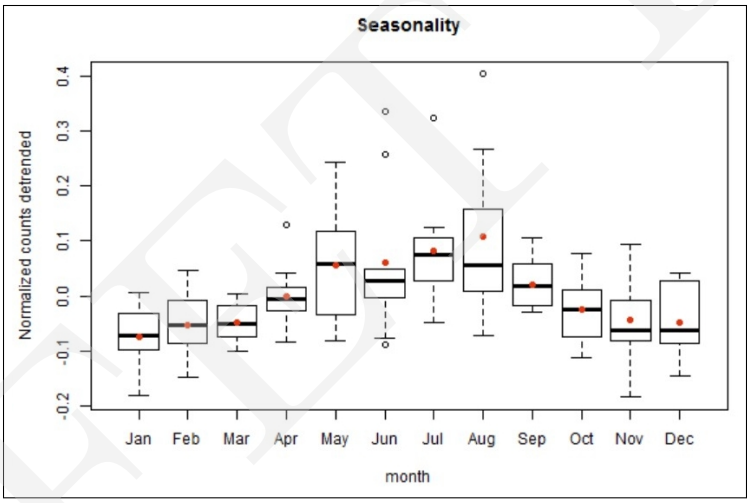



Figure 9: This functionality shows the monthly distribution of counts. It allows to identify any seasonal pattern.

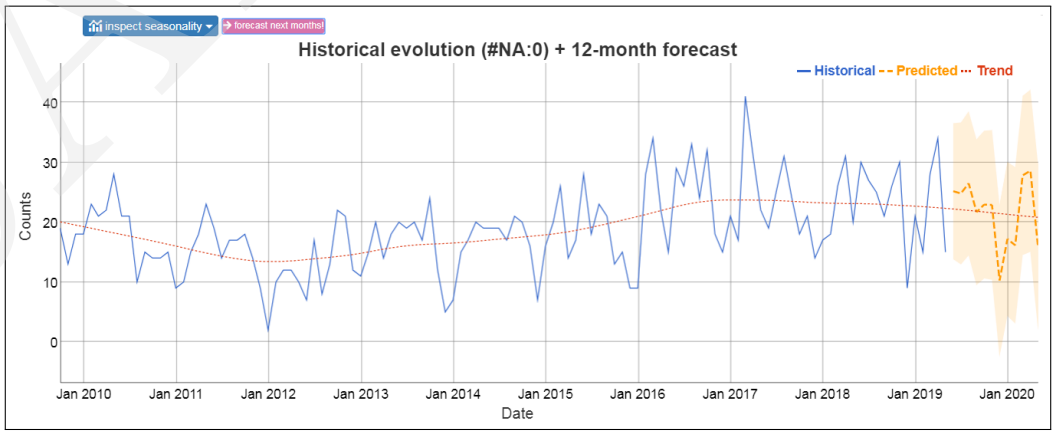



Figure 10: Time series injury count modeling and forecasting.

[3] https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

## 5.2 Keyphrases

**Live video demonstration**

https://vimeo.com/411775022

In this subtab, a word co-occurrence network is first built from the text of the matching reports. We rely on the word co-occurrence network of [25] as our representation, for its simplicity, high historical success, and above all because it was recently used in several approaches that reached very good keyword extraction performance [36, 37].

As shown in Fig. 11, such a network represents a piece of text as an undirected graph where there is an edge between two nodes if the terms they represent co-occur within a fixed size sliding window applied over text from start to finish. Edge weights are equal to co-occurrence counts. In the application, we use a window of size 3 that does not overspan sentences nor reports[4].

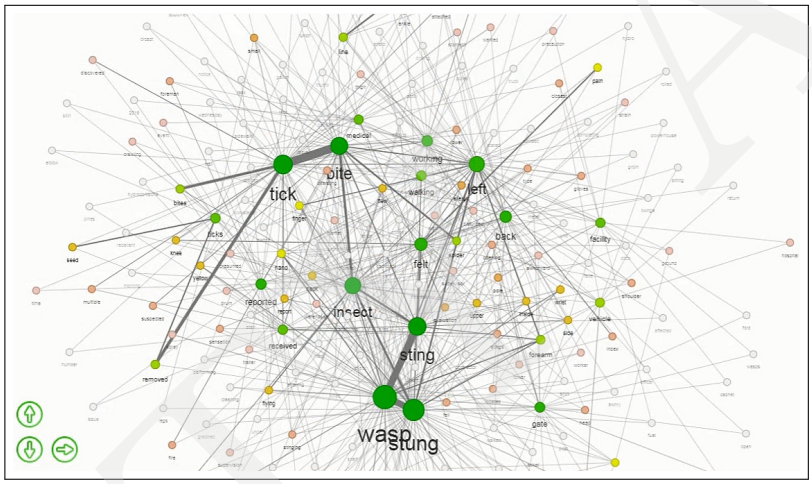



Figure 11: Word co-occurrence network example from the app.

Google's PageRank algorithm [29] is then applied to the graph to assign a score to each node. The top nodes (keywords) are used as seeds from which keyphrases are reconstructed: the words with highest scores that appear together in the text are pasted together. The scores of the keyphrases correspond to the sum of the scores of the keywords they contain. Finally, as can be seen in Fig. 12, reports are ranked according to the quantity and diversity of the keywords they contain. Reports that are too short or that do not contain any keyword are not considered. Duplicate reports are also removed. The most relevant reports (at most 100) are then displayed.

Some basic statistics (wordclouds of most frequent words, results and collection size distribution comparisons) about the reports corresponding to the filter selection, including the short and keywordless ones, are also displayed.

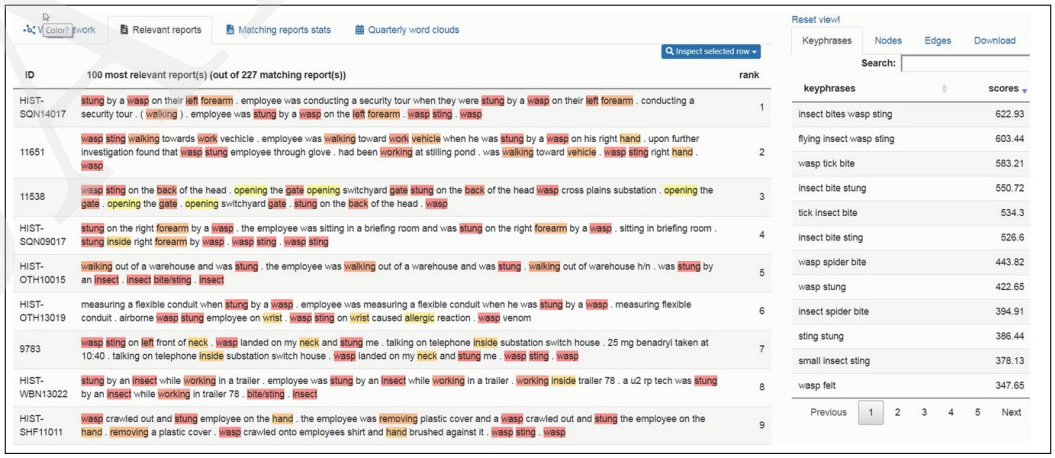



Figure 12: Reports are ranked according to the quantity and diversity of the keywords they contain.

[4] To avoid counting spurrious co-occurrences between words at the end/beginning of consecutive sentences/reports.

## 5.3 Attribute co-occurrence networks

**Live video demonstration**

https://vimeo.com/412484002

### 5.3.1 Overview

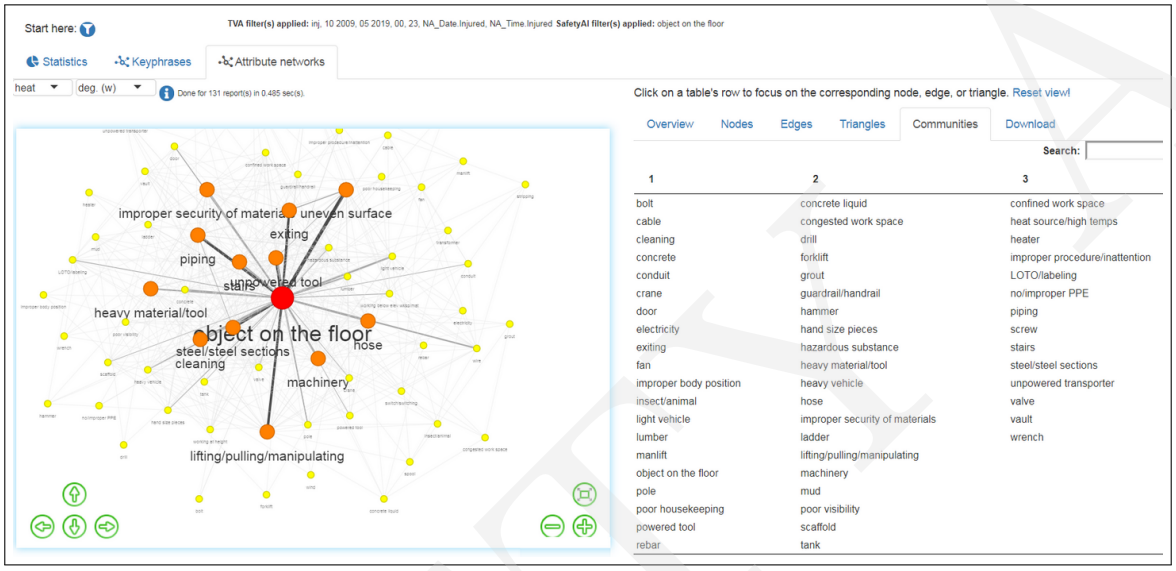



Figure 13: Attribute networks subtab.

**What questions can this part answer?**

- Which attributes frequently appear together in Company's dataset?
- What are the reports associated with a specific combination of attributes? How are words distributed among those reports?
- Which attributes, or groups of 2, 3, or more attributes play a key role for a particular outcome?

› The first question can be answered by applying a community detection algorithm to a given attribute network. Let us consider the network constructed from the full Company dataset (no filter entered). Communites can be inspected by going to the Attribute networks subtab and the Communities tab on the right pane. We can observe meaningful community structure. For instance, in the fifth community, the attributes *breaker*, *clearance*, *fuses*, *LOTO/labeling*, *relay*, *switch/switching*, and *transformer* are all clustered together. In the third community, we also find *object on the floor* and *poor housekeeping* together. To identify smaller groups of frequently co-occurring attributes, the user can also look at the Edges and Triangles tabs: the edges and triangles with the highest weights correspond to the pairs and triplets of attributes that co-occur the most. Using the same network as above, the edges with greatest weights are *electricity/cable* and *object on the floor/cable*, while the top 2 triangles are *object on the floor/electricity/cable* and *unpowered tool/object on the floor/steel sections*.

› When an attribute selection has been made in the filtering interface (e.g., *adverse low temperatures*), the user can go to the Statistics/Matching report(s) subtab to inspect the reports correspond-

ing to the attribute selection[5]. The user can also visualize a wordcloud (see Fig. 14) displaying the counts of the words appearing in the retrieved reports by visiting the Keyphrases/Matching reports statistics subtab. Finally, note that to refine the results, the user can also go to the Text search tab, click the 'Apply Diagnostics tab filters' button, and enter some textual queries.

A wordcloud is also available on the Text search tab, via the cloud button on top of the left pane (☁).


Figure 14: Wordcloud example.

› Once a filter selection has been made, rankings of the nodes and edges in the resulting attribute network can be inspected. E.g., among the network built from the 377 cases associated with the Company HD/NCK outcome (head/neck), the top nodes in terms of weighted degree are *small particle*, *steel/steel sections*, *hazardous substance*, and *exiting*, and the top 2 edges in terms of weight are *hazardous substance/small particle* and *door/exiting*.

Moreover, different node and edge metrics are available. Each one uses its own definition of importance and thus provides a different ranking. For instance, the attributes that co-occur with many other attributes (the hubs) can be identified via their degrees or PageRank numbers.

Safety-critical combinations of 2 or 3 attributes can be found by inspecting the edges and triangles with high weights or high betweenness scores. The edges and triangles high on betweenness are important because they link attributes or groups of attributes that rarely co-occur and that otherwise would not be connected (e.g., because they belong to different well-separated communities).

### 5.3.2 Graphs

A graph $G(V, E)$, or network, is defined as a set of vertices (or nodes) $V$ and a set of edges (or links) $E$. Graphs are rich, flexible, and universal structures that can accurately represent the interaction among the components of many natural and human-made complex systems. Graphs have been notably used to describe and analyze the interplay among proteins within cells [4], the organization of the brain [11], the World Wide Web [29], textual documents [25], and information propagation through a population [23]. In construction more specifically, graphs have been previously used to model people interaction during projects [13], or safety communication among workers [1]. Due to the ubiquitous nature of networks, learning on graphs is a very active area of research in machine learning and artificial intelligence, as well as in many application domains.

### 5.3.3 Attribute co-occurrence networks

As defined in [41] and as shown in Fig. 13, an attribute co-occurrence network is an undirected, weighted graph where each node is an attribute and there is an edge between two nodes if the two attributes they represent are found together in at least one report. Edge weights represent co-occurrence counts. This representation is powerful, as it enables any tool from graph theory to be used in analyzing a set of injury reports. In what follows, we detail the graph mining metrics and algorithms that are available in the application.

### 5.3.4 Basic graph statistics

Graph stats are available on the right pane, in the Overview tab. The **diameter** of a graph is the length of the longest shortest path[6] between any two vertices in the graph. The **density** of a graph

[5]Note that this procedure can be followed for any selection of filters, not only attributes.
[6]A path is a sequence of non-repeating nodes.

is the ratio of its number of edges to the maximum number of edges it could have. For an undirected graph, density is computed as:

$$\frac{2|E|}{|V|(|V|-1)} \tag{1}$$

where $|V|$ is the number of nodes and $|E|$ the number of edges in the graph. A graph is said to be dense when its density approaches 1 and sparse when it approaches 0. The global **clustering coefficient** (or transitivity) of a graph is the ratio of its closed triplets to its connected triplets[7]. It measures how tightly interconnected are the nodes in the graph.

Finally, the **average distance** of a graph is the average shortest path distance between any two nodes in the graph.

**Graph statistics interpretation**

Let $G(V, E)$ be an attribute co-occurrence graph, and $R$ the set of reports that were used to construct $G$.

- The number of nodes $|V|$ of $G$ corresponds to the number of unique attributes found in $R$.
- The number of edges $|E|$ of $G$ corresponds to the number of unique co-occurrences between any two attributes in $R$.
- $G$ has a small **diameter** if many attributes co-occur in the same reports.
- $G$ has a low **average distance** if it is dense, or if a small set of attributes appear in many reports.
- $G$ is **dense** if many attributes co-occur with many other attributes in $R$. Conversely, $G$ is **sparse** if only a few attributes are found together in $R$. Density is in general inversely proportional to diameter and average distance.

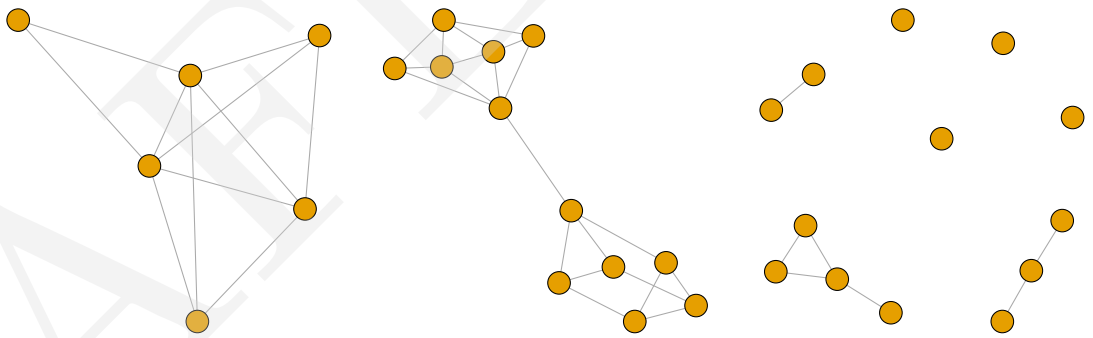


Figure 15: Left: many attributes co-occur with many other attributes. $G$ is dense (73.33%) with a diameter of 2 and an average distance of 1.3. Middle: two attributes appear in many reports. $G$ is quite sparse (32%), and has a large diameter (5) but a low average distance (2.3). Right: only a few attributes co-occur. $G$ is very sparse (9%).

### 5.3.5 Node centrality metrics

Node scores for each metric are available on the right pane, in the Nodes tab. There are many ways to define the importance of a vertex in a network. The node centrality metrics presented in what follows, which are implemented in the application, are among the most popular. As shown in Fig. 16, each metric ranks nodes according to a different criterion.

[7] A triplet is defined as 3 connected nodes.

**(Un)weighted degree**. The degree of a node is the count of its connections. The weighted degree (or strength) of a node is the sum of the weights of its incident edges. Both variants of the degree are local measures, in that they only use information from the immediate neighborhood of a node to compute its score.

**PageRank**. PageRank [29] is a well-known algorithm that was initially created to measure the importance of webpages. Unlike degree, it is a global measure of node importance. The score of a given node can be seen as the probability of ending up at that node after having started a random walk from any vertex in the graph. In undirected networks, PageRank scores are strongly correlated with degrees.

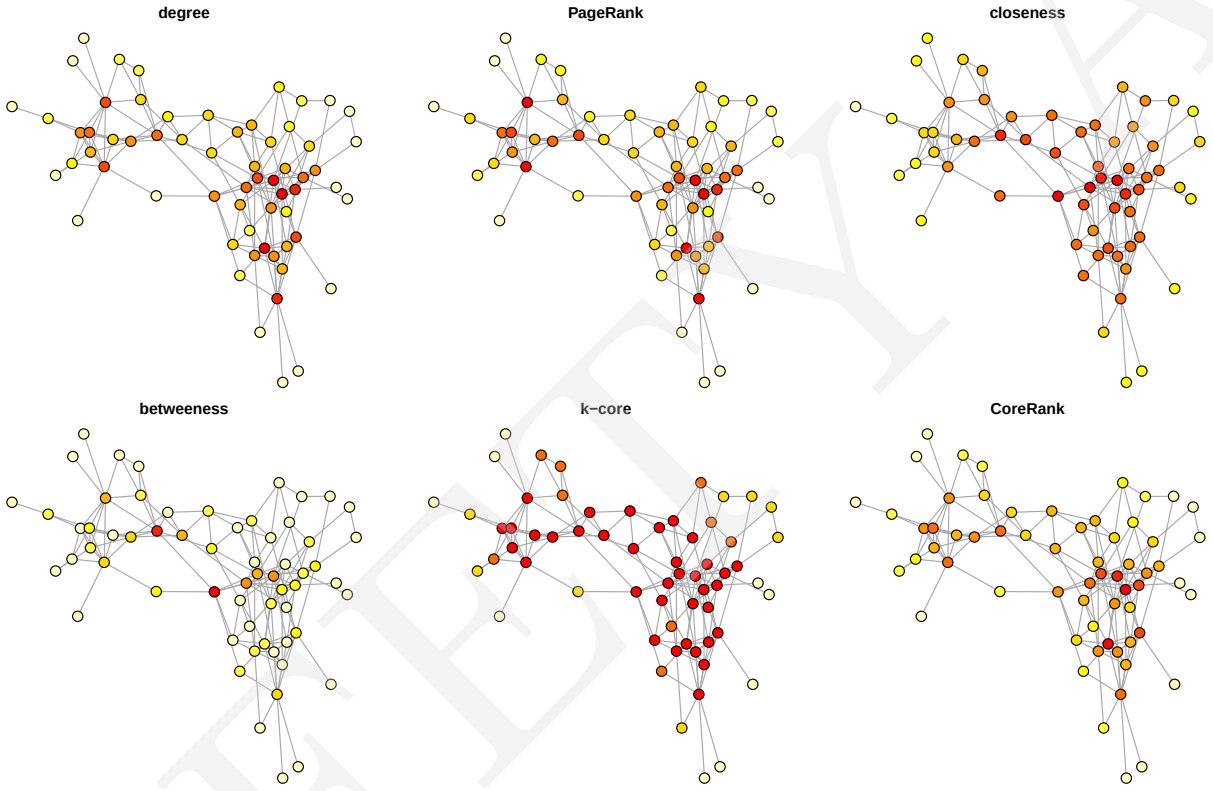



Figure 16: Comparison between the different node centrality metrics implemented in the application. Node scores are shown using a color scale from light yellow (low) to red (high).

**Betweenness**. The betweenness of a node $v$ is the number of shortest paths between any pair of nodes in the graph that pass through $v$. Nodes with high betweenness scores control information flow in the network. Betweenness is a global metric.

**Closeness**. The closeness of a node is the inverse of the average shortest path distance from the node to any other node in the graph. Vertices high on closeness have more direct access to all other vertices in the graph. Like for betweenness, we use the weighted version of the metric here (with inverse edge weights). Closeness is a global metric.

**Graph degeneracy ($k$-core)**. A core of order $k$ (or $k$-core) of a graph $G$ is a maximal subgraph of $G$ in which every vertex $v$ has at least (weighted) degree $k$ [33]. $k$-core is thus a local metric. The $k$-core decomposition of $G$ forms a hierarchy of subgraphs that are recursively included in one another and whose cohesiveness/size increases/decreases with $k$. The core number of a node is the highest order of a core that contains this node. Nodes with high core numbers have the desirable property of not only being central (like nodes with high degree or PageRank centrality) but also part of cohesive subgraphs with other central nodes. They are *influential spreaders* [23].

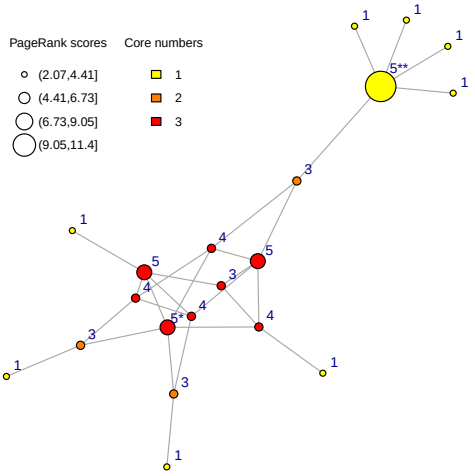



Figure 17: Degrees vs PageRank scores vs core numbers. Node labels indicate degrees. Nodes $*$ and $**$ both have same degree (5) and high PageRank numbers (resp. in $(6.73, 9.05]$ and $(9.05, 11.4]$). However, node $*$ lies in a much more central location, which is captured by its higher core number (3 vs 1) but not by degree or PageRank.

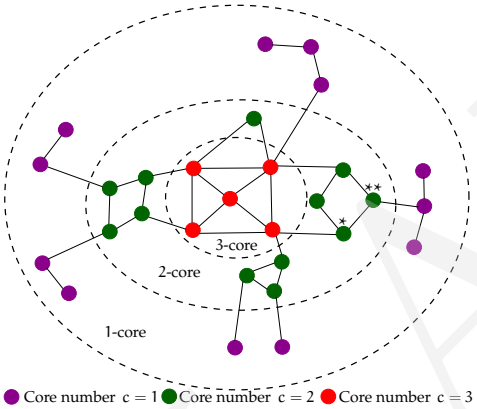



Figure 18: $k$-core decomposition

**Node centrality metrics interpretation**

- Select **degree** or **PageRank** to highlight the hubs in the network. These nodes correspond to the attributes that co-occur with many other attributes, or with the same small set of attributes many times.
- Use **betweenness** to identify the attributes that act as intermediaries between communities of attributes. These attributes are not necessarily very frequent but connect rarely co-occurring attributes.
- Pick **closeness** to focus on the attributes that (in)directly co-occur with many other attributes, i.e., the attributes that are at short distance from all the other attributes.
- Use $k$**-core** to find cohesive groups of frequent attributes.

### 5.3.6 Edge/triangle centrality metrics

The weight of an **edge** represents the number of times the two attributes linked by the edge were found together in a report. Edge betweenness is the number of shortest paths going through the edge divided by the total number of weighted shortest paths in the network [20]. Since edge weights indicate distance here, we used the inverse of the edge weights. Edges with high betweenness control flow in the network and connect communities, as shown in Fig. 19. The weight of a **triangle**[8] is the sum of the weights of the edges it contains.

**Edge/triangle centrality metrics interpretation**

- Edges/triangles with high weights correspond to pairs/triplets of attributes that are frequently found together in reports.
- Edges with high betweenness correspond to pairs of attributes that are not necessarily very frequent and that rarely co-occur (e.g., edges overspanning multiple communities).

[8] A triangle is a closed connected triplet of nodes.

### 5.3.7 Community detection

As illustrated in Fig. 19, communities are disjoint or overlapping groups of nodes within which connections are dense and between which they are sparse [18]. Many natural and human-produced networks exhibit community structure. Studying communities is valuable as nodes belonging to the same groups often share similar properties.
The application uses one of the most famous non-overlapping community detection algorithms, the *fast greedy* algorithm [14]. It merges at each step the pair of nodes that yields the largest gain in modularity until a single community remains. The best partition is the one associated with the greatest modularity value. The modularity function measures the strength of the partition of a graph by comparing the number of within-group edges to the expected such number in a null model [28].

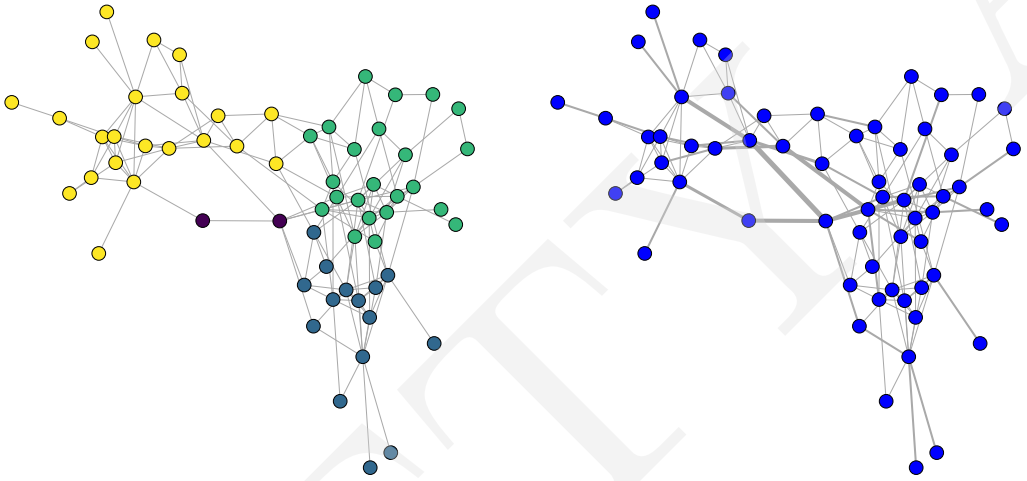


Figure 19: Left: communities as identified by the *fast greedy* algorithm. Right: edge betweenness (the thicker the edge, the greater its betweenness score).

**Key points**

- Communities are groups of frequently co-occurring attributes.
- Different node centrality metrics can be used to identify important attributes from different perspectives.
- Interesting combinations of attributes can be found among the edges and triangles with large weights, or scoring high on betweenness.

## 6 Field

**Live video demonstration**

https://vimeo.com/408891625

**Relevant journal paper**

Baker, H., Hallowell, M. R., & Tixier, A. J. P. (2019). AI Predicts Independent Construction Safety Outcomes from Universal Attributes. *arXiv preprint* arXiv:1908.05972 (**accepted for publication in Automation in Construction**) https://arxiv.org/pdf/1908.05972.pdf

This tab exposes the trained machine learning models[9]. The user can select attributes and get a prediction of the most likely outcomes, should an accident occur. The Company outcomes that are predicted are `Classification` (injury severity), `Part Of Body`, and `Nature Of Injury` (injury type). In addition to the predictions, the user is also provided with the risk gauge of the Risk tab, which provides a risk estimate in terms of percentage of the maximum simulated risk, and attribute trends over time (daily and historical/monthly).
Finally, by clicking a button, the attribute selection of the user is applied to the filtering interface of the Diagnostics tab and the user is taken to the Statistics/Matching report(s) subtab of that tab. This enables inspecting historical reports containing the combination of attributes selected, which can be useful in guiding the toolbox talk or job hazard analysis (JHA).

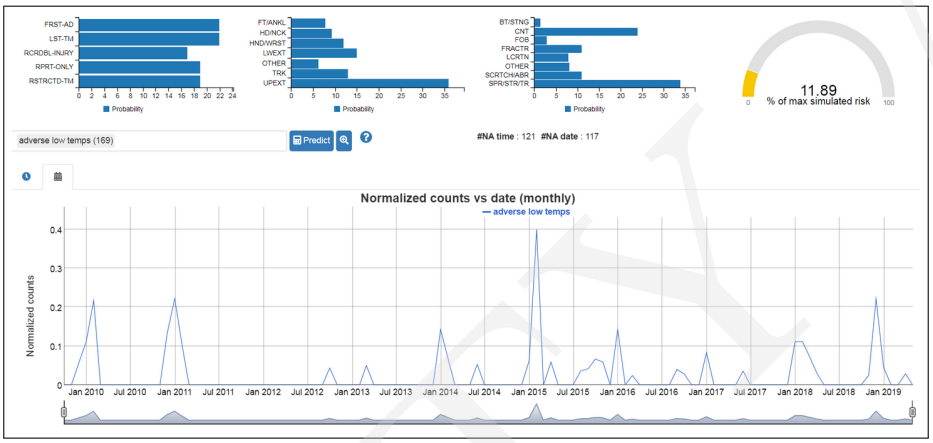



Figure 20: Machine learning tab.

**</> Code**

Code to load the trained machine learning models and get predictions can be found in the `/Company_directory/code/machine_learning/predict.py` script.

**What questions can this part answer?**

- Given a description of a construction situation in terms of attributes, what are the most likely outcomes, should an incident occur?
- What attributes are the most predictive of each outcome category?
- What historical cases match the attribute selection?
- Which attributes are trending?

› The answer to the first question is provided by the predictions of the models.

› In subsection 6.5, we explain how to derive attribute importance scores from the trained models, and interpret them.

› By clicking a button, the attribute selection of the user is applied to the Diagnostics tab and the user is taken to the Statistics/Matching report(s) subtab of that tab[10]. Inspecting historical accident reports associated with the same selection of attributes can help illustrating the JHA or toolbox talk

[9] We improved over [39] at every level, from the models used, to the experimental setup and evaluation metrics.
[10] The user is then free to apply any other filter, to retrieve relevant historical cases associated with, e.g., a specific date, severity level, or body part injured.

with concrete examples. The retrieved cases can also be used to focus the discussion towards specific aspects which might have been overlooked otherwise. Such qualitative feedback complements well the quantitative predictions and risk assessment.

› The normalized counts over time of each attribute selected by the user are overlaid on the bottom pane. Moreover, the user can focus the visualization on specific time and count ranges.
The attribute trend visualization tool is useful to identify any time pattern. For instance, in Fig. 20, we clearly see that the attribute *adverse low temperatures* is peaking every year around January.

In what follows, we explain in detail how the data were prepared, and how the machine learning models were trained and evaluated.

## 6.1 Preprocessing

### 6.1.1 Outcome columns and label normalization

We decided to consider the following columns of Company's dataset as outcomes to be predicted:

- `Classification`: injury severity,
- `Part Of Body`: body part injured,
- `Nature Of Injury`: injury type.

Each outcome column was associated with a certain number of unique values, that we will denote in the remainder of this report interchangeably as categories, levels, or labels. Given the relatively small size of the injury part of Company's dataset (2,364 reports), reducing the number of unique levels per category and increasing the number of examples per level was necessary to guarantee the success of machine learning. To achieve this, we merged some levels together. E.g., for `Part Of Body`, the levels containing 'finger', 'hand', 'wrist', and 'thumb' in their names were merged into the hand/wrist level (HND/WRST). For `Nature Of Injury`, 'Strain or Tear' was merged with 'Sprain or Tear' into the level 'sprain/strain/tear' (SPR/STR/TR).

By merging some rare categories into bigger categories, label normalization also helped addressing class imbalance, which is a very challenging problem for machine learning algorithms.

</> Code

The exact label normalization procedure that was followed can be found in the `data_checks_normalization.py` script.

Level counts for each normalized outcome are shown in Table 2, and all abbreviation keys can be found in the `Company_directory/data/dicts_counts/*_labeldict.json` files. For each outcome, we finally retained for prediction the levels that were associated with at least 100 reports. These categories are shown in bold in Table 2.

### 6.1.2 Train/val/test splits and class imbalance

For each outcome, 90% of the reports were used for training, and 10% were left-out for testing. 10% of the training set was also left-out as a validation set (to optimize the parameters of the models). For reproducibility, the case numbers of the cases in each split were saved in the `Company_directory/data/machine_learning/train_val_test_ids.json` file.

To address the problem of class imbalance, weights inversely proportional to the frequency of each category in the training set were computed. These weights were used during training only, to make sure that the models paid equal attention to all levels, and in particular, did not neglect the

| nature of injury | | classification | | part of body | |
|---|---|---|---|---|---|
| SPR/STR/TR | **716** | FRST-AD | **1110** | HND/WRST | **549** |
| LCRTN | **323** | RCRDBL-INJRY | **518** | UPEXT | **446** |
| CNT | **307** | RPRT-ONLY | **428** | HD/NCK | **377** |
| BT/STNG | **255** | LST-TM | **153** | LWEXT | **371** |
| SCRTCH/ABR | **160** | RSTRCTD-TM | **147** | TRK | **356** |
| FRACTR | **119** | OTHER | 8 | FT/ANKL | **133** |
| FOB | **107** | | | OTHER | **121** |
| OTHER | **102** | | | NA | 11 |
| NOINJ | 85 | | | | |
| BURN | 66 | | | | |
| HTEXHST | 35 | | | | |
| INFLMMTN | 32 | | | | |
| CRSH | 24 | | | | |
| PNCT | 22 | | | | |
| NA | 11 | | | | |

Table 2: Category counts for each outcome after normalization, in the injury part of Company's dataset (2,364 reports). Categories associated with more than 100 observations were used for prediction.

minority categories. These weights were passed as category weights to the random forest and support vector machine models (RF and SVM), and as sample weights to the XGBoost model (XGB). See subsection 6.2 for full details about the models.

Statistics about the splits and weights are shown for each outcome in Tables 3 to 5.

| | train | | val | test |
|---|---|---|---|---|
| | counts | weights | counts | counts |
| FRST-AD | 887 | 1.00 | 105 | 118 |
| LST-TM | 127 | 7.00 | 12 | 14 |
| RCRDBL-INJRY | 429 | 2.10 | 46 | 43 |
| RPRT-ONLY | 350 | 2.50 | 37 | 41 |
| RSTRCTD-TM | 115 | 7.70 | 13 | 19 |
| sum | 1908 | - | 213 | 235 |

Table 3: Train/val/test splits statistics for `Classification`.

| | train | | val | test |
|---|---|---|---|---|
| | counts | weights | counts | counts |
| FT/ANKL | 111 | 4.00 | 9 | 13 |
| HD/NCK | 316 | 1.40 | 37 | 24 |
| HND/WRST | 447 | 1.00 | 46 | 56 |
| LWEXT | 290 | 1.50 | 38 | 43 |
| OTHER | 102 | 4.40 | 8 | 11 |
| TRK | 287 | 1.60 | 33 | 36 |
| UPEXT | 352 | 1.30 | 41 | 53 |
| sum | 1905 | - | 212 | 236 |

Table 4: Train/val/test splits statistics for `Part Of Body`.

## 6.2 Models

Machine learning involves a two-step process: first, a set of features has to be defined (*feature engineering*) and extracted from the input documents (*feature extraction/recognition*), second, a model is trained to predict the outcomes from the features. Here, our features were the set of 92 fundamental attributes shown in Table 12. As already explained in section 4, they were extracted by applying the NLP tool to the text associated with each report, provided by the `Short Description`, `What`

| | train | | val | test |
|---|---|---|---|---|
| | counts | weights | counts | counts |
| BT/STNG | 206 | 2.80 | 19 | 30 |
| CNT | 247 | 2.30 | 28 | 32 |
| FOB | 90 | 6.40 | 13 | 4 |
| FRACTR | 93 | 6.20 | 11 | 15 |
| LCRTN | 274 | 2.10 | 20 | 29 |
| OTHER | 82 | 7.00 | 14 | 6 |
| SCRTCH/ABR | 131 | 4.40 | 19 | 10 |
| SPR/STR/TR | 575 | 1.00 | 62 | 79 |
| sum | 1698 | - | 186 | 205 |

Table 5: Train/val/test splits statistics for `Nature Of Injury`.

`Happened`, `Why Occur`, `Immediate Actions`, `Comments`, `Activity`, `Cause`, `Description`, and `Object` columns of Company's dataset. Our models are described in detail next.

We experimented with the Random Forest [9], Extreme Gradient Boosting (XGBoost) [12], and linear Support Vector Machine (SVM) [5] models, which all are widely used in practice and are among the most successful machine learning algorithms to date. We used the `scikit-learn` implementations of Random Forest[11] and linear SVM[12], and for XGBoost, the original library[13].

In what follows, we provide details about each algorithm. While they differ in the way they combine trees, RF and XGB are both ensemble learning techniques and are both made of large numbers of decision trees grown with the CART algorithm [10]. Therefore, we first introduce CART before presenting Random Forest and XGBoost. Then, we present the SVM algorithm. Also, note that we present each model within the context of *classification*, which is the task of interest in this study.

### 6.2.1 CART

As shown in Algorithm 1 and Fig. 21, CART is a greedy algorithm whose goal is to learn a set of binary recursive rules that partition observations living in a multidimensional feature space. In our case, the dimensions of the space are the fundamental construction attributes.

More precisely, the final partitioning should be such that most of the observations in the same subspace (or leaf) belong to the same category. The most frequent category of the observations in a given leaf is then used as the prediction for these observations, as shown in Fig. 21. As opposed to *global* models such as linear regression, where the same equation holds over the entire space, decision trees are *local* models.

**Algorithm 1** CART

**Input:** dataset of observations and features, target variable
**Output:** set of partitioning rules in the form of recursive binary splits
1: start with a root region that includes all observations
2: for each predictor, compute a 'goodness of split' value based on some criterion
3: pick the best predictor (if multiple predictors are best, select the first one)
4: split the region in two based on the values taken by the observations on the predictor selected
5: repeat steps 2-4 for each new region until the stopping criterion is reached

For classification, the criterion used at step 2 in Algorithm 1 was the decrease in Gini diversity index, and the predictor that was selected at step 3 was the one that was maximizing the decrease in Gini index. The Gini index measures the heterogeneity, or impurity, of a partition. For instance, it

[11] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[12] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
[13] https://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn

considers a partition 100% pure if it contains only observations belonging to the same class. The stopping criterion at step 5 was when a certain depth had been reached.

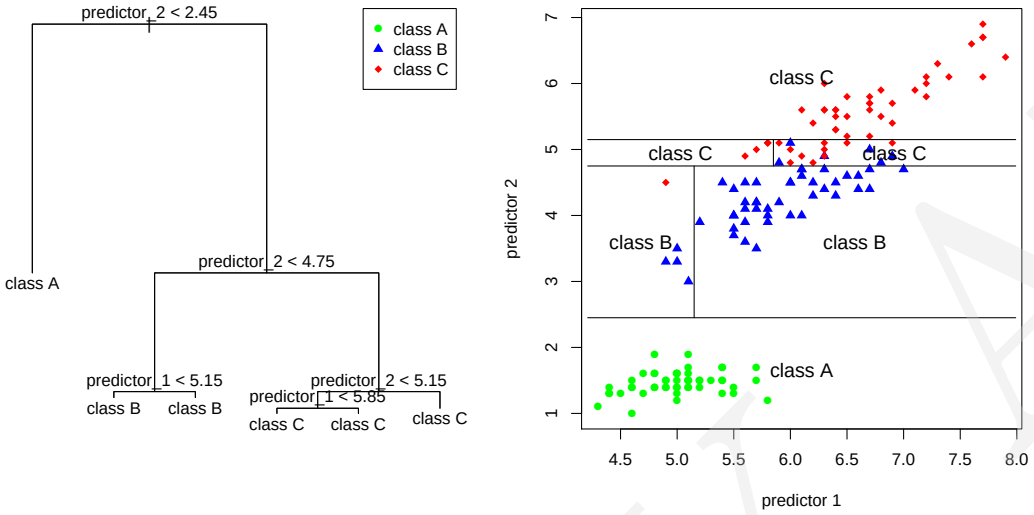



Figure 21: Decision tree toy example. For visualization purposes, the two predictors are continuous.

CART decision trees are able to capture complex nonlinear high-order interactions among predictors, scale well with the number of predictors and observations, and are relatively robust to outliers and irrelevant predictors. However, they often need to be grown very large to accurately represent the training data. This has two negative side effects: (1) poor generalization to unseen observations (overfitting) and (2) high variance, as the lower parts of the trees are very sensitive to changes in the training data. From the perspective of the bias-variance framework, where $error = bias + variance$, deep decision trees are ***low bias-high variance*** models.

### 6.2.2 Random Forest

**Bagging**. The **B**ootstrap **agg**regat**ing** (Bagging) method [7] was introduced as a way to take advantage of the low bias of deep decision trees while reducing their high variance. The Bagging procedure consists in training many deep trees in parallel on bootstrap samples of the data. A bootstrap sample is obtained by randomly selecting observations with replacement from the original training set until a data set of the same size is obtained. Approximately one third of the observations are not expected to be present in each bootstrap sample, as the probability of not selecting a given observation with replacement from a sample of size $n$ is $(1 - 1/n)^n$, which tends to $\exp(-1) \approx 1/3$ when $n$ tends to infinity. These observations compose what is called the 'out-of-bag' (OOB) sample [8]. Since the bootstrap sample is of same size as the original data set, it follows that for a large number of observations, each bootstrap sample is expected to contain about two thirds of unique examples, the rest being duplicates. This causes each tree in the ensemble to become an expert on some specific domains of the training set. Bagging thus creates an ***ensemble of local experts***. Consequently, at prediction time, there will be a significant amount of beneficial disagreement among trees, as can be seen in Fig. 22. By aggregating the predictions from all trees in the ensemble (via majority voting), one obtains a model with significantly less variance than a single tree. Such a model generalizes much better, while still having (almost) the same low bias. This approach is known as ***perturb and combine***.

Despite being a significant improvement over CART, bagged ensembles are less interpretable. Also, by definition of CART, only those variables yielding the greatest decrease in node impurity are selected at each split. Consequently, all the trees in the bagged ensemble have quite similar upper structures, and tend to generate correlated forecasts, which reduces the disagreement among trees and prevents the maximal reduction in variance from being achieved.

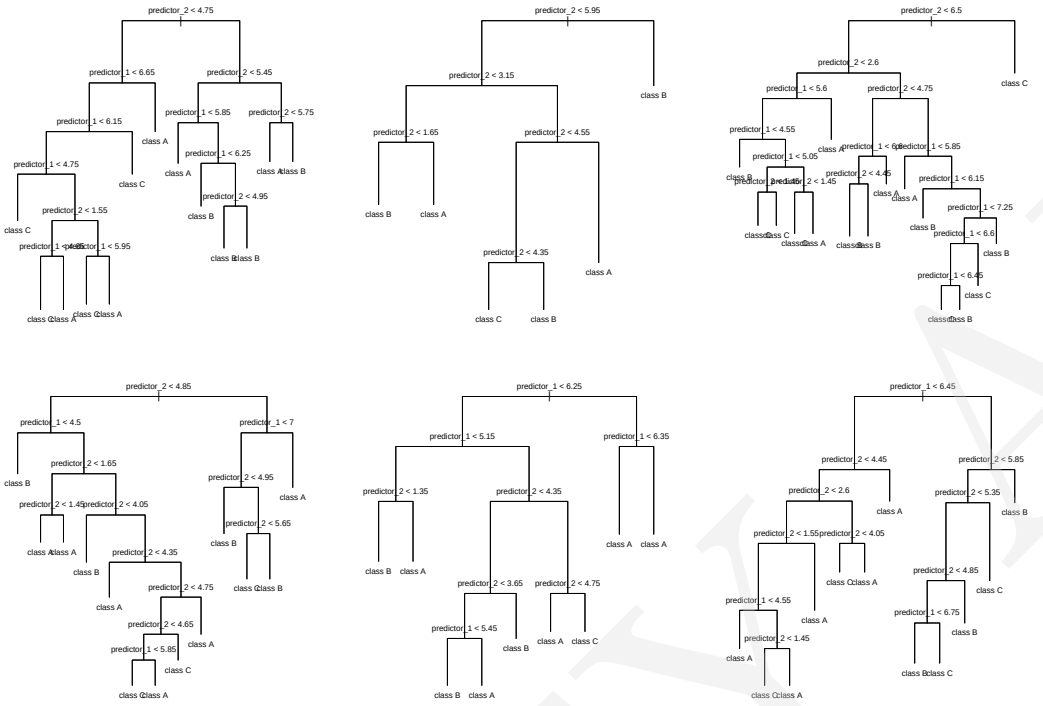


Figure 22: Example of a bagged ensemble of decision trees, using the same data as in Fig. 21. Each tree in the ensemble is grown on a boostrap sample of the original data. The large differences in the tree structures highlight well the high-variance nature of decision trees.

**Random Forest**. Random Forest [9] was designed to address the correlation problem previously described. In Random Forest, trees are still grown on bootstrap samples of the training set, like in Bagging. However, a simple modification of the CART algorithm is used: instead of trying *all* predictors at each split, only a *random subset* of them is considered. In practice, this extra randomization gives all predictors a chance to play a role in determining the upper structure of trees, which introduces a lot of beneficial variety in the ensemble and results in greater variance reduction, smaller error rates, and more accurate predictions than with Bagging.

### 6.2.3 XGBoost

Like Random Forest, the Boosting algorithm [19] is an ensemble approach that combines many base models and let them vote to generate forecasts. However, this apparent similarity is misleading, since Random Forest and Boosting tackle the task of error reduction in opposite ways. Indeed, while Random Forest seeks to reduce error by *decreasing the variance* of complex low bias-high variance base models (deep decision trees) built in *parallel*, Boosting achieves the same goal by adding weak high bias-low variance base models (shallow decision trees) in *sequence*, repeatedly *reducing the bias* of the entire sequence.

Each shallow tree is a weak learner, i.e., only slightly better than random guessing, but by adding them such that each successive tree is trained to predict the observations that were missed by the preceding one, Boosting creates a strong learner. More precisely, at each step, each new tree is fed the pseudo-residuals given by the *gradient* of some differentiable loss function with respect to the predictions of the current model, that is, the entire sequence of trees thus far. This approach is known as **Gradient Boosting**.

Moreover, it was empirically shown that training each tree on a *random* subsample of the training set (instead of the full training set) was very beneficial. This method is named **Stochastic Gradient Boosting**, to emphasize the instillation of randomness into the procedure.

Finally, **Extreme Gradient Boosting** (XGBoost) [12] adds a regularization term to the loss function of SGB in order to penalize the complexity of the model, and implements a number of optimization tricks to speed-up training.

### 6.2.4 Linear SVM

The linear Support Vector Machine [5, 43] is a geometrical method that seeks to classify points into two different categories by finding the best separating hyperplane. As illustrated by Fig. 23 in the two-dimensional case (two attributes), the best separating hyperplane is the line that separates the two groups of points with the greatest possible margin on each side. Training the SVM, that is, finding the best hyperplane, comes down to optimizing the $\vec{w}$ and b parameters (see Fig. 23).

In test mode, a new observation is classified based on the side of the hyperplane on which it falls, which corresponds to the sign of the dot product of the observation with the vector orthogonal to the hyperplane ($\vec{w}$).

Because in practice, points may not all be separable (e.g., due to outliers), when searching for the best separating hyperplane, the SVM is allowed to misclassify certain points. The tolerance level is controlled by a parameter traditionally referred to as C in the literature. The smaller C, the more tolerant the model is towards misclassification.

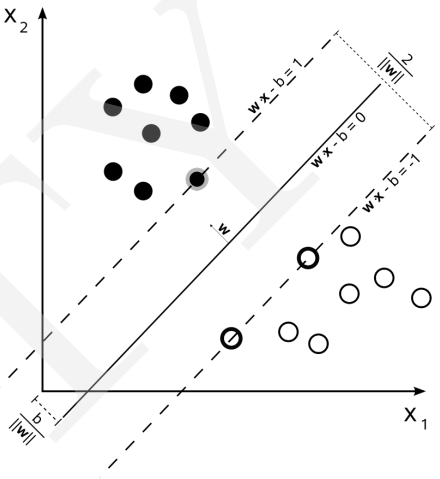



Figure 23: Linear SVM decision boundary in the two-dimensional case (two attributes).

C plays a crucial *regularization* role, i.e., it has a strong impact on the generalization ability of the SVM. Indeed, for large values of C (low misclassification tolerance), a smaller-margin hyperplane will be favored over a larger-margin hyperplane if the former classifies more points correctly, at the risk of overfitting the training data. On the other hand, small values of C will favor larger-margin separating hyperplanes, even if they misclassify more points. Such solutions tend to generalize better.

When the target variable features more than two categories, a *one-versus-rest* approach is used to redefine the problem as a set of binary classification tasks. More precisely, as many SVMs as there are categories are trained, and the goal of each SVM is to predict whether an observation belongs to its associated category or not.

**Time complexity**. Finding the support vectors scales quadratically with the number of training examples $n$. More precisely, it is $O(n^2)$ when the $C$ parameter is small and $O(n^3)$ when it gets large [6]. In practice, this is the main limitation of SVMs compared to RF or XGBoost. On some large datasets, using a SVM might just be intractable.

### 6.2.5 Attribute importance measures

**Random Forest**. The out-of-bag (OOB) observations (see subsection 6.2.2) can be used to compute a relative importance score for each predictor. For a given predictor and tree, the procedure consists in randomly permuting the values of the predictor in the set of observations that have not taken part in the training of the tree (i.e., the OOB sample), and comparing the prediction error of the

tree on the permuted OOB sample with the prediction error of the tree on the untouched OOB sample. This process is repeated for all the trees in the forest, and the predictor is given an importance score proportional to the overall increase in error that its permutation induced. The most important variables are the ones leading to the greatest loss in predictive accuracy when noised-up [9].

**XGBoost**. Like Random Forest, the Boosting algorithm allows the calculation of importance scores for the predictor variables. In our classification case, the procedure is as follows: for a given tree in the sequence, and for a given non-terminal node of this tree, the reduction in node purity weighted by the number of observations in the node is attributed to the predictor the split was made on. This process is repeated for every non-terminal node of the tree, and the variable importance scores are averaged over all trees.

**Linear SVM**. Recall from subsection 6.2.4 that the $\vec{w}$ vector, orthogonal to the best-separating hyperplane, is used to determine whether a given observation belongs to the class of interest or to any of the other classes (one-vs-rest approach). Since by definition, observations can only have non-negative coordinates in our attribute space (0 or 1), a given observation belongs to the class of interest if its dot product with $\vec{w}$ is positive.

Furthermore, the $\vec{w}$ vector contains the contribution of each attribute in making the classification decision. More precisely, the magnitudes of the coordinates of $\vec{w}$ indicate the strength of the contributions, while their signs indicate if the attributes attract or reject observations to/from the class of interest. Attributes for which the coefficients of $\vec{w}$ are large and positive (resp. negative) are strongly indicative of belonging (resp. not belonging) to the category of interest.

### 6.2.6 Model stacking

To see whether performance could be further improved, we experimented with *stacking*, a popular technique among machine learning practitioners. With stacking, the goal is to automatically learn how to combine the individual strengths of each model with a meta-model, in order to obtain a better classifier. We used simple logistic regression models[14]as our meta-models, with $C = 0.2$ for all outcomes (no tuning was performed). More precisely, a given logistic regression model was trained to predict the levels of a given outcome based on the sum of the probabilistic forecasts of XGB, RF, and SVM. We used the same train/test split as described in subsection 6.1.2.

By design, the `linearSVC` implementation of the linear SVM model only returns discrete predictions, that is, a single label corresponding to the most likely class, rather than a probability distribution over *all* classes. Therefore, we re-trained the best SVM models using the more general `SVC` implementation[15], specifying a linear Kernel.

## 6.3 Experimental setup

### 6.3.1 Hyperparameter optimization

All grid searches were performed on the validation set (see subsection 6.1.2). The final models, corresponding to the best parameter combinations, were then trained on the union of the training and validation sets and tested on the test set. The best parameter values for each model and each outcome are shown in Tables 6 to 8.

For Random Forest[16], we searched the number of trees (`ntree` parameter, from 100 to 1200 with steps of 100), the number of variables to try when making each split (`mtry`, from 5 to 45 with steps of 5), and the leaf size (`nodesize`, 1, 2, 5, and 10).

[14] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[15] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[16] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

For XGBoost[17], we searched the maximum depth of a tree in the sequence (`max_depth`, from 3 to 6 with steps of 1), the learning rate (`learning_rate`, 0.01, 0.05, and 0.1), the minimum leaf size (`min_child_weight`, 1,3, and 5), the percentage of training instances to be used in building each tree (`subsample`, 0.3, 0.5, 0.7, and 1) , and the percentage of predictors to be considered in making each split of a given tree (`colsample_bylevel`, 0.3, 0.5, 0.7, and 1). The number of trees in the sequence (`ntrees`) was automatically selected with an early stopping strategy. More precisely, the validation loss had to decrease at least once every 200 iterations to continue training. The loss was the multinomial or binary log loss, depending on the dataset. Note that the maximum allowed number of trees in any case was 2000.

Finally, for the SVM model, we optimized the C parameter (`C`, $10^x$ with $x$ taking 800 evenly spaced values in $[-7, 7]$).

### 6.3.2 Performance metrics

Due to the large class imbalance for all outcomes, measuring classification performance with accuracy was inadequate. Rather, we recorded a *confusion matrix* at the end of each batch on the validation set during the learning rate range tests and on the test set during training. The confusion matrix $C$ is a square matrix of dimension $K \times K$ where $K$ is the number of categories, and the $(i, j)^{th}$ element $C_{i,j}$ of $C$ indicates how many of the observations known to be in category $i$ were predicted to be in category $j$. From the confusion matrix, we computed precision, recall and F1-score for each class. Precision, respectively recall, for category $i$, was computed by dividing $C_{i,i}$ (the number of correct predictions for category $i$) by the sum over the $i^{th}$ column of $C$ (the number of predictions made for category $i$), respectively by the sum over the $i^{th}$ row of $C$ (the number of observations in category $i$).

$$\text{precision} = \frac{C_{i,i}}{\sum_{j=1}^{K} C_{j,i}} \qquad \text{recall} = \frac{C_{i,i}}{\sum_{j=1}^{K} C_{i,j}} \tag{2}$$

Finally, the F1-score was computed as the harmonic mean of precision and recall:

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

### 6.3.3 Configuration

Experiments were run in Python 3.6, using a 4-core 2.4GHz CPU and a high-end GPU. The machine had 16GB of RAM, and was running a 64-bit linux-based operating system.

Note that for the XGBoost grid searches, we used the GPU-accelerated implementation of the fast histogram algorithm (`gpu_hist`) as the tree method[18], which led to a signficant 8X speedup. However, we made sure to use the exact algorithm (`exact`) for training and testing the final models.

## 6.4 Results

The optimal parameter values are shown for each model and each outcome in what follows.

| | max_depth | learning_rate | min_child_weight | subsample | colsample_bylevel | ntrees |
|---|---|---|---|---|---|---|
| Classification | 5 | 0.10 | 5 | 0.3 | 1.0 | 31 |
| Part.Of.Body | 3 | 0.10 | 1 | 0.3 | 0.7 | 86 |
| Nature.Of.Injury | 4 | 0.10 | 1 | 0.3 | 0.5 | 59 |

Table 6: Best hyperparameter values for XGBoost.

[17] https://xgboost.readthedocs.io/en/latest/parameter.html
[18] https://xgboost.readthedocs.io/en/latest/gpu/

| | n_estimators | max_features | min_samples_leaf |
|---|---|---|---|
| Classification | 600 | 40 | 2 |
| Part.Of.Body | 300 | 30 | 1 |
| Nature.Of.Injury | 200 | 40 | 2 |

Table 7: Best hyperparameter values for Random Forest.

| | $x$ |
|---|---|
| Classification | 1.253 |
| Part.Of.Body | -3.548 |
| Nature.Of.Injury | -1.849 |

Table 8: Best hyperparameter values for linear SVM, where $C = 10^x$

Performance for each prediction task is shown in Tables 9 to 11. The three models, XGB, RF, and SVM are compared with a random classification baseline. Overall, all three models perform largely better than the random baseline, and XGB is always the best of the single models. Moreover, performance is quite high, meaning that by using attributes that are only observable before incident occurrence, it is possible to *predict well* three safety outcomes independently extracted by human annotators.

**Model stacking**. The XGB+RF+SVM ensemble outperforms the other models on some categories (but not overall) for the `Classification` and `Part Of Body` outcomes, and reaches best overall performance for the `Nature Of Injury` outcome.

| | | FRST-AD | LST-TM | RCRDBL-INJRY | RPRT-ONLY | RSTRCTD-TM | mean |
|---|---|---|---|---|---|---|---|
| XGB | prec | 33.05 | 14.29 | 27.91 | 39.02 | 26.32 | 28.12 |
| XGB | rec | 56.52 | 5.26 | 29.27 | 34.78 | 12.20 | 27.61 |
| XGB | F1 | 41.71 | 7.69 | 28.57 | 36.78 | **16.67** | **26.28** |
| RF | prec | 30.51 | 14.29 | 25.58 | 39.02 | 21.05 | 26.09 |
| RF | rec | 57.14 | 5.13 | 20.00 | 36.36 | 11.76 | 26.08 |
| RF | F1 | 39.78 | 7.55 | 22.45 | 37.65 | 15.09 | 24.50 |
| SVM | prec | 21.19 | 21.43 | 27.91 | 36.59 | 15.79 | 24.58 |
| SVM | rec | 51.02 | 4.23 | 36.36 | 41.67 | 6.52 | 27.96 |
| SVM | F1 | 29.94 | 7.06 | **31.58** | **38.96** | 9.23 | 23.35 |
| XGB+RF+SVM | prec | 48.31 | 14.29 | 25.58 | 31.71 | 10.53 | 26.08 |
| XGB+RF+SVM | rec | 54.29 | 9.09 | 23.40 | 36.11 | 8.00 | 26.18 |
| XGB+RF+SVM | F1 | **51.12** | **11.11** | 24.44 | 33.77 | 9.09 | 25.91 |
| random | prec | 19.87 | 18.43 | 20.51 | 20.73 | 19.95 | 19.90 |
| random | rec | 49.72 | 5.54 | 18.70 | 18.12 | 8.03 | 20.02 |
| random | F1 | 28.40 | 8.52 | 19.57 | 19.34 | 11.45 | 17.45 |

Table 9: Test set performance for `Classification`. Best F1 score per column in **bold**.

| | | FT/ANKL | HD/NCK | HND/WRST | LWEXT | OTHER | TRK | UPEXT | mean |
|---|---|---|---|---|---|---|---|---|---|
| XGB | prec | 30.77 | 20.83 | 42.86 | 18.60 | 81.82 | 25.00 | 16.98 | 33.84 |
| XGB | rec | 21.05 | 45.45 | 36.92 | 32.00 | 26.47 | 18.75 | 26.47 | 29.59 |
| XGB | F1 | **25.00** | 28.57 | **39.67** | 23.53 | **40.00** | 21.43 | **20.69** | **28.41** |
| SVM | prec | 7.69 | 25.00 | 76.79 | 9.30 | 36.36 | 25.00 | 5.66 | 26.54 |
| SVM | rec | 12.50 | 46.15 | 26.38 | 36.36 | 44.44 | 42.86 | 27.27 | 33.71 |
| SVM | F1 | 9.52 | **32.43** | 39.27 | 14.81 | **40.00** | **31.58** | 9.37 | 25.28 |
| RF | prec | 23.08 | 20.83 | 28.57 | 32.56 | 36.36 | 25.00 | 11.32 | 25.39 |
| RF | rec | 10.71 | 17.24 | 35.56 | 29.79 | 19.05 | 25.00 | 20.00 | 22.48 |
| RF | F1 | 14.63 | 18.87 | 31.68 | 31.11 | 25.00 | 25.00 | 14.46 | 22.96 |
| XGB+RF+SVM | prec | 15.38 | 20.83 | 30.36 | 44.19 | 54.55 | 25.00 | 13.21 | 29.07 |
| XGB+RF+SVM | rec | 12.50 | 21.74 | 36.17 | 33.93 | 28.57 | 26.47 | 17.95 | 25.33 |
| XGB+RF+SVM | F1 | 13.79 | 21.28 | 33.01 | **38.38** | 37.50 | 25.71 | 15.22 | 26.41 |
| random | prec | 14.85 | 14.96 | 14.77 | 13.60 | 14.36 | 14.22 | 14.09 | 14.41 |
| random | rec | 5.70 | 10.49 | 25.48 | 17.29 | 4.68 | 15.45 | 21.53 | 14.37 |
| random | F1 | 8.24 | 12.33 | 18.70 | 15.23 | 7.06 | 14.81 | 17.04 | 13.34 |

Table 10: Test set performance for `Part Of body`. Best F1 score per column in **bold**.

| | | BT/STNG | CNT | FOB | FRACTR | LCRTN | OTHER | SCRTCH/ABR | SPR/STR/TR | mean |
|---|---|---|---|---|---|---|---|---|---|---|
| XGB | prec | 96.67 | 21.88 | 100.00 | 6.67 | 31.03 | 50.00 | 40.00 | 48.10 | 49.29 |
| XGB | rec | 93.55 | 25.00 | 50.00 | 8.33 | 30.00 | 16.67 | 20.00 | 65.52 | 38.63 |
| XGB | F1 | **95.08** | **23.33** | 66.67 | 7.41 | 30.51 | 25.00 | **26.67** | 55.47 | 41.27 |
| SVM | prec | 96.67 | 15.62 | 100.00 | 13.33 | 31.03 | 33.33 | 70.00 | 18.99 | 47.37 |
| SVM | rec | 93.55 | 29.41 | 66.67 | 11.76 | 30.00 | 18.18 | 9.86 | 68.18 | 40.95 |
| SVM | F1 | **95.08** | 20.41 | **80.00** | 12.50 | 30.51 | 23.53 | 17.28 | 29.70 | 38.63 |
| RF | prec | 96.67 | 12.50 | 100.00 | 6.67 | 34.48 | 50.00 | 30.00 | 27.85 | 44.77 |
| RF | rec | 93.55 | 14.81 | 44.44 | 7.14 | 28.57 | 17.65 | 8.57 | 59.46 | 34.27 |
| RF | F1 | **95.08** | 13.56 | 61.54 | 6.90 | 31.25 | 26.09 | 13.33 | 37.93 | 35.71 |
| XGB+RF+SVM | prec | 96.67 | 18.75 | 100.00 | 13.33 | 34.48 | 50.00 | 30.00 | 51.90 | 49.39 |
| XGB+RF+SVM | rec | 93.55 | 23.08 | 44.44 | 15.38 | 32.26 | 21.43 | 17.65 | 64.06 | 38.98 |
| XGB+RF+SVM | F1 | **95.08** | 20.69 | 61.54 | **14.29** | **33.33** | **30.00** | 22.22 | **57.34** | **41.81** |
| random | prec | 12.93 | 12.31 | 13.25 | 12.73 | 12.97 | 13.00 | 10.80 | 12.66 | 12.58 |
| random | rec | 15.10 | 15.03 | 2.03 | 7.77 | 15.00 | 2.99 | 4.27 | 38.45 | 12.58 |
| random | F1 | 13.93 | 13.53 | 3.53 | 9.65 | 13.91 | 4.86 | 6.12 | 19.05 | 10.57 |

Table 11: Test set performance for `Nature Of injury`. Best F1 score per column in **bold**.

## 6.5 Attribute importance

Since the one-vs-rest approach allows the linear SVM to offer attribute importance measures at the categorical level (unlike Random Forest and XGBoost), and since these measures are easily interpretable, as was explained in subsection 6.2.5, we selected the linear SVM as our method of choice for computing attribute importance scores. Results can be seen in Figures 24 to 26. Note that for the sake of completeness, attribute importance scores according to XGBoost and Random Forest are shown in Appendices A and B.

More precisely, for each category of each outcome, the barplots show the 6 attributes which are the most predictive of belonging to that category, and the 6 attributes which are the least predictive.

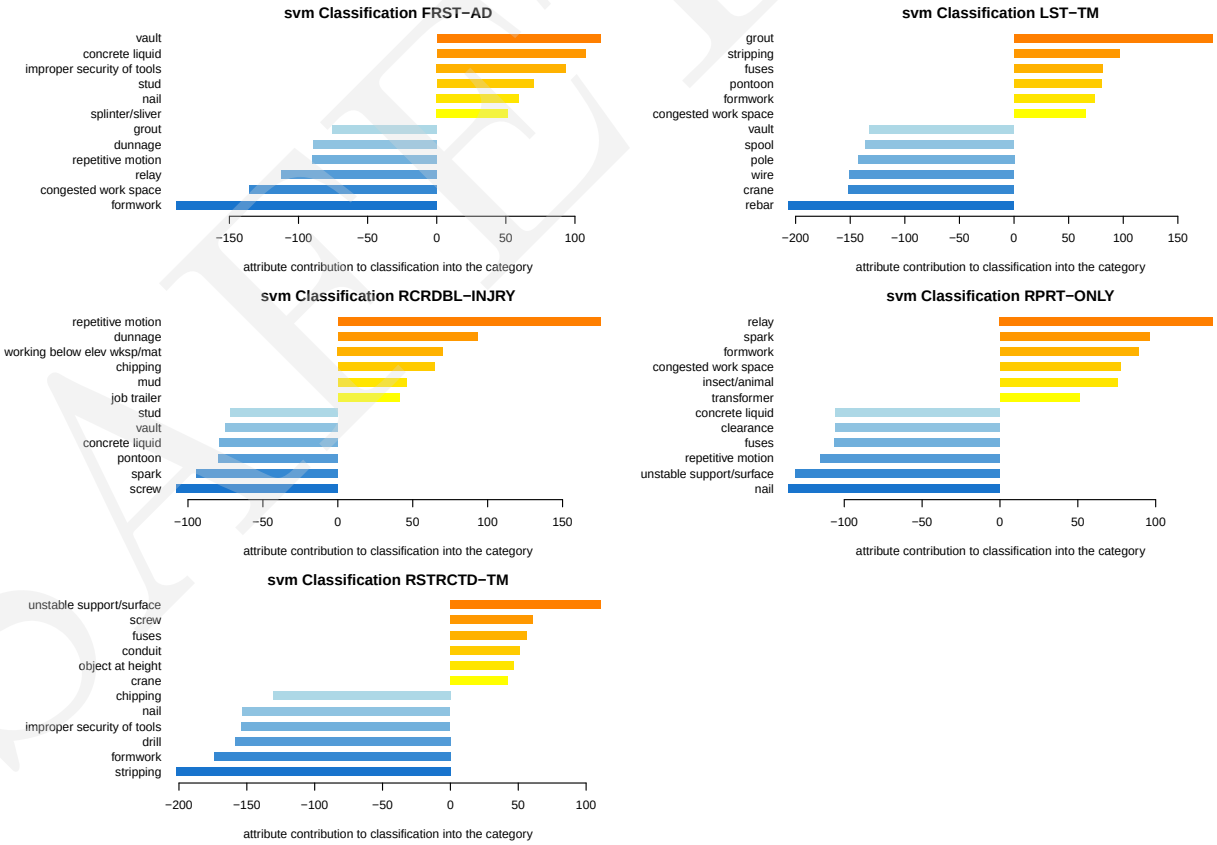



Figure 24: Per-category attribute contribution for `Classification`.

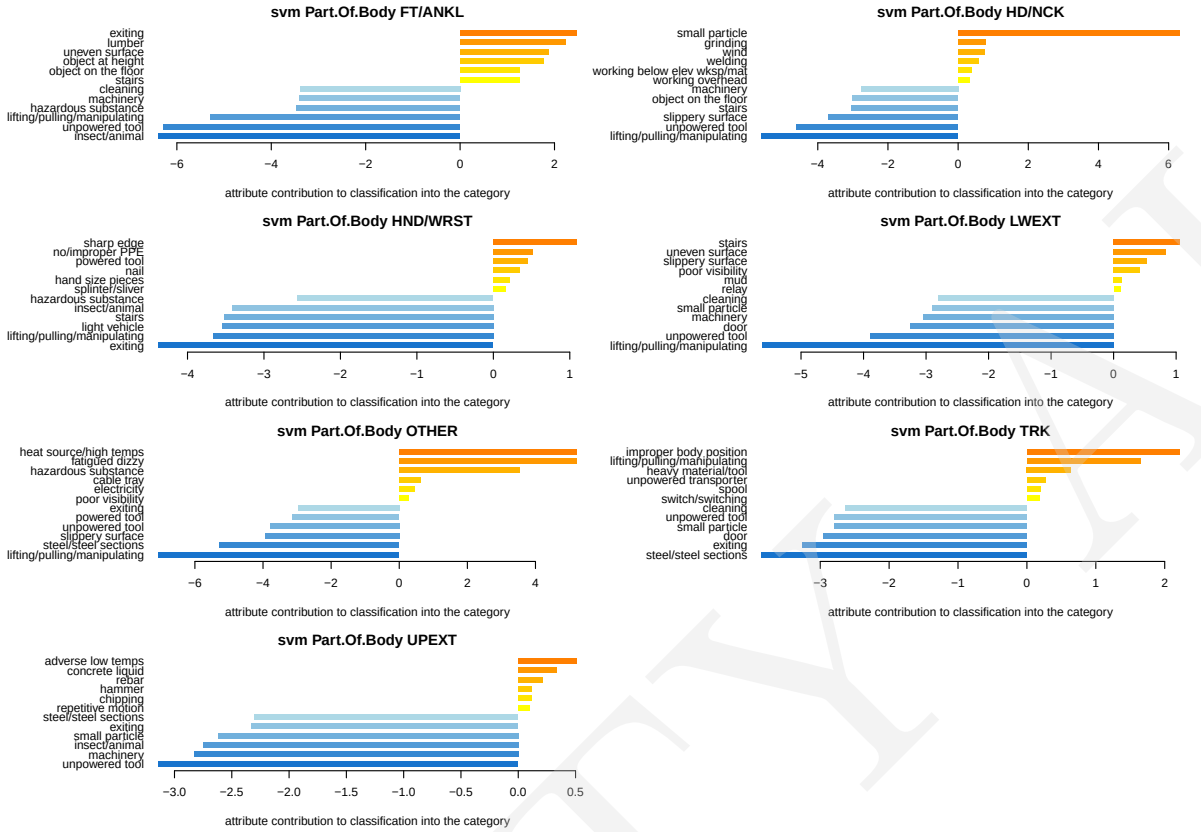



Figure 25: Per-category attribute contribution for `Part of body`.

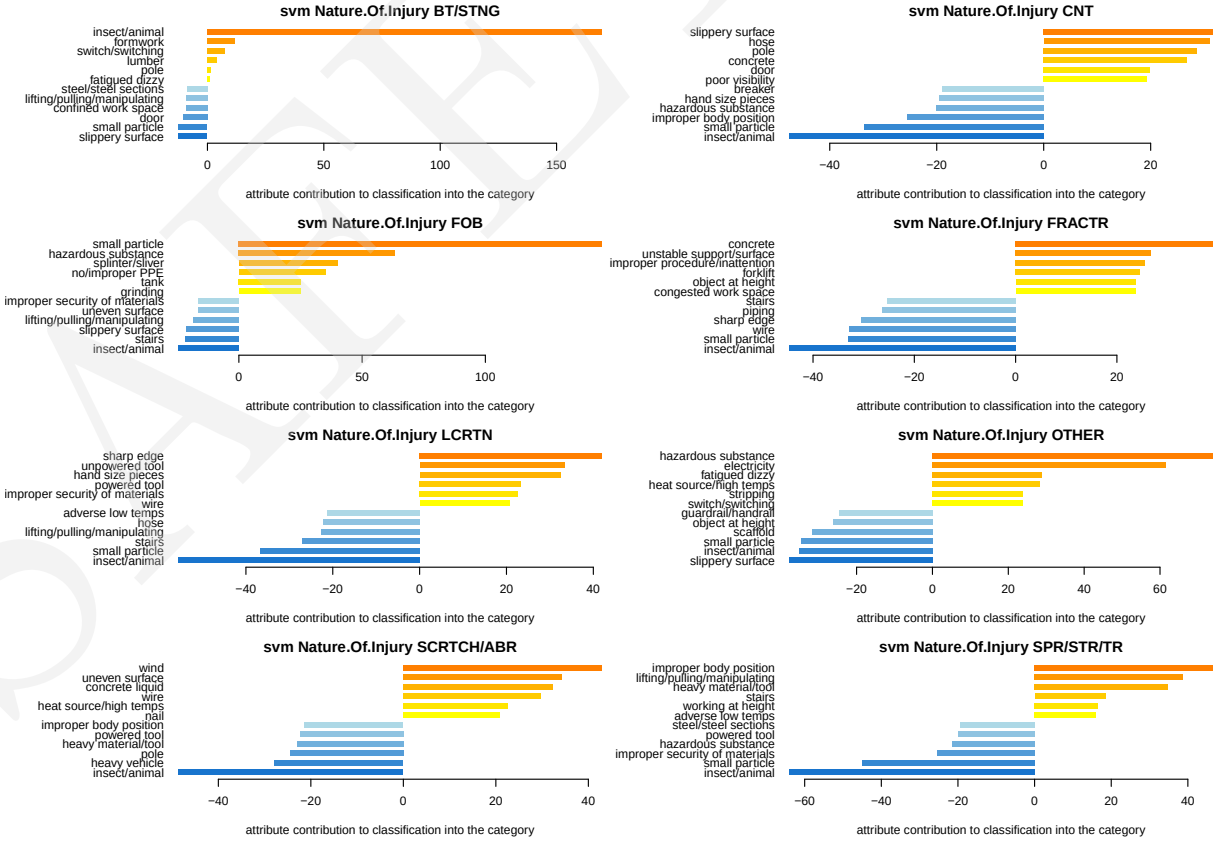



Figure 26: Per-category attribute contribution for `Nature of injury`.

Results make sense. For instance, the attributes most predictive of the `FT/ANKL` (foot/ankle) category of the `Part Of Body` outcome include *exiting*, *uneven surface*, *object on the floor*, and *stairs*. For the `HD/NCK` (head/neck) category of the same outcome (which includes eyes, nose and mouth), the most predictive attributes are *small particle*, *grinding*, *wind*, *working below elevated workspace/material* and *working overhead*. For the laceration (LCRTN) level of the `Nature Of Injury` outcome, the most predictive attributes include *sharp edge*, *powered tool*, and *wire*.

# 7 Risk analysis

**Live video demonstration**

https://vimeo.com/407641827

**Relevant journal paper**

Tixier, A. J. P., Hallowell, M. R., & Rajagopalan, B. (2017). Construction safety risk modeling and simulation. *Risk analysis*, 37(10), 1917-1935. **[link]**

## 7.1 Overview

This tab exposes the results of the risk analysis. Its left pane features two subtabs, namely Bubble chart and Detailed table, which provide two ways of making an attribute selection. Once the user has selected a combination of attributes describing a construction situation, a work package, etc., the situational-level risk associated with the selection is then assessed. The estimate is provided both in terms of where the risk stands in Company's distribution of situational-level risk (histogram), and what fraction of the maximum simulated risk does it represent (gauge).

The Detailed table subtab (see Fig. 27) lets the user select rows in an interactive variant of Table 12, while with the Bubble chart subtab (Fig. 28), attributes are displayed as circles in the *relative risk* vs. *global risk* space (both types of risk will be defined in the rest of this section). The diameters of the circles are proportional to the attribute counts in the injury part of Company's dataset. In the Bubble chart subtab, the user can thus select attributes in a visual way, simply by clicking on the attribute circles.

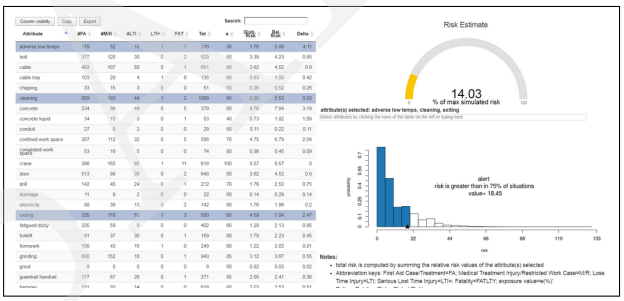



Figure 27: Detailed table subtab.

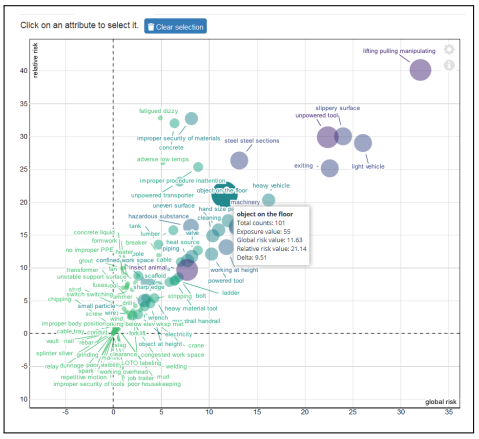



Figure 28: Bubble chart subtab.

**What questions can this part answer?**

- How risky is a particular work package?
- What is the distribution of risk in time and space on a particular site?
- On which aspects of a given task should the JHA/safety meeting/toolbox talk focus?

› The risk of a work package or of any construction situation is estimated by summing the relative risk values of the attributes identified as present. The risk estimate is provided in terms of where the risk stands in Company's distribution of situational-level risk (histogram), and what fraction of the maximum simulated risk does it represent (gauge).

› Assuming that the repartition of tasks is known in time and space, a distribution of risk along these dimensions can be obtained at scheduling time. Attribute clashes creating high risk situations can therefore be mitigated by separating the incompatible attributes in time and/or space.

› The third question can be answered by looking at the attributes that are involved in the task, and by focusing the discussion on the riskiest attributes (in terms of relative risk, global risk, or both). Moreover, the filtering interface of the Diagnostics tab can be used to check whether there is any history of previous injuries or near misses for the particular combination of attributes selected. If relevant reports are found, they can be read aloud to workers. Identifying what went wrong in the reports can then help recognizing the hazards associated with the task, and determining how to eliminate them from the work environment or at least how to reduce workers' exposure to them.

## 7.2 Attribute-level risk

From now until the end of the risk section, we summarize the definitions and methods presented in [38], and explain how they were applied to Company's dataset. For more theoretical justifications, technical details, and references, we invite the reader to refer to the original paper. In what follows, we use the terms *attribute* and *precursor* interchangeably.

### 7.2.1 Global risk

We define construction safety risk as the product of $frequency$ by $severity$, as shown in Eq. 4:

$$risk = frequency \times severity \tag{4}$$

The global safety risk $R_p$ accounted for by $precursor_p$ is computed as the product of the number $n_{p_s}$ of injuries attributed to $precursor_p$ for the severity level $s$ (given by Table 12) and the severity score $S_s$ of this severity level (given by Table 13), summed across all severity levels:

$$R_p = \sum_{s=1}^{5} \left( n_{p_s} \cdot S_s \right) \tag{5}$$

We considered the five following severity levels from the `Classification` column of Company's dataset: report only ($s = 1$), first aid ($s = 2$), recordable ($s = 3$), restricted time ($s = 4$), and lost time ($s = 5$). We abbreviated these levels RPRT-ONLY, FRST-AD, RCRDBL-INJRY, RSTRCTD-TM, and LST-TM. As can be seen in Table 13 and Fig. 29, the scores of the severity levels are exponentially increasing.

The accident cases that were not associated with any attribute, or for which the severity was unknown (OTHER), and of course the non-accident cases, were not used for computing risk values

| | RPRT-ONLY | FRST-AD | RCRDBL | RSTRCTD | LST-TM | Total | $e$ (%) | Global Risk | Relative Risk | $\Delta$ |
|---|---|---|---|---|---|---|---|---|---|---|
| bolt | 4 | 34 | 29 | 5 | 6 | 78 | 80 | 8.99 | 11.24 | 2.25 |
| breaker | 1 | 13 | 7 | 2 | 4 | 27 | 40 | 4.81 | 12.03 | 7.22 |
| cable | 10 | 19 | 13 | 5 | 5 | 52 | 65 | 6.92 | 10.65 | 3.73 |
| cable tray | 2 | 8 | 4 | 0 | 0 | 14 | 30 | 0.36 | 1.19 | 0.84 |
| chipping | 0 | 1 | 3 | 0 | 0 | 4 | 25 | 0.19 | 0.76 | 0.57 |
| cleaning | 27 | 82 | 31 | 9 | 13 | 162 | 70 | 17.38 | 24.83 | 7.45 |
| clearance | 0 | 6 | 7 | 1 | 1 | 15 | 70 | 1.67 | 2.38 | 0.71 |
| concrete | 8 | 23 | 23 | 3 | 7 | 64 | 20 | 8.96 | 44.81 | 35.84 |
| concrete liquid | 1 | 5 | 0 | 0 | 1 | 7 | 15 | 1.01 | 6.75 | 5.74 |
| conduit | 6 | 10 | 6 | 4 | 0 | 26 | 40 | 1.45 | 3.64 | 2.18 |
| confined work space | 16 | 46 | 16 | 4 | 2 | 84 | 45 | 4.47 | 9.94 | 5.47 |
| congested work space | 6 | 2 | 3 | 1 | 2 | 14 | 60 | 2.33 | 3.89 | 1.55 |
| crane | 8 | 15 | 9 | 6 | 1 | 39 | 80 | 3.11 | 3.89 | 0.78 |
| door | 19 | 99 | 30 | 9 | 14 | 171 | 90 | 18.47 | 20.53 | 2.05 |
| drill | 7 | 9 | 11 | 1 | 1 | 29 | 60 | 1.97 | 3.28 | 1.31 |
| dunnage | 0 | 0 | 1 | 0 | 0 | 1 | 35 | 0.06 | 0.17 | 0.11 |
| electricity | 18 | 19 | 13 | 7 | 3 | 60 | 90 | 5.55 | 6.16 | 0.62 |
| exiting | 33 | 102 | 50 | 13 | 19 | 217 | 90 | 25.35 | 28.17 | 2.82 |
| fan | 6 | 14 | 6 | 3 | 2 | 31 | 30 | 3.15 | 10.5 | 7.35 |
| fatigued/dizzy | 8 | 12 | 13 | 1 | 5 | 39 | 15 | 5.88 | 39.18 | 33.3 |
| forklift | 4 | 9 | 11 | 2 | 3 | 29 | 80 | 4.06 | 5.08 | 1.02 |
| formwork | 3 | 0 | 2 | 0 | 1 | 6 | 15 | 1.06 | 7.09 | 6.03 |
| fuses | 0 | 2 | 2 | 1 | 1 | 6 | 20 | 1.32 | 6.58 | 5.26 |
| grinding | 3 | 10 | 5 | 1 | 0 | 19 | 55 | 0.68 | 1.24 | 0.56 |
| grout | 0 | 0 | 0 | 0 | 1 | 1 | 10 | 0.94 | 9.36 | 8.42 |
| guardrail/handrail | 8 | 21 | 11 | 4 | 3 | 47 | 80 | 4.72 | 5.9 | 1.18 |
| hammer | 5 | 8 | 11 | 1 | 2 | 27 | 70 | 2.88 | 4.12 | 1.24 |
| hand size pieces | 13 | 43 | 41 | 11 | 9 | 117 | 70 | 14.07 | 20.1 | 6.03 |
| hazardous substance | 28 | 69 | 39 | 7 | 5 | 148 | 50 | 9.71 | 19.41 | 9.71 |
| heat source/high temps | 10 | 47 | 20 | 3 | 4 | 84 | 70 | 6.34 | 9.05 | 2.72 |
| heater | 3 | 9 | 3 | 1 | 2 | 18 | 20 | 2.42 | 12.12 | 9.69 |
| heavy material/tool | 16 | 35 | 23 | 9 | 6 | 89 | 80 | 9.63 | 12.04 | 2.41 |
| heavy vehicle | 9 | 42 | 26 | 6 | 13 | 96 | 80 | 15.73 | 19.67 | 3.93 |
| hose | 12 | 35 | 19 | 8 | 3 | 77 | 75 | 6.34 | 8.46 | 2.11 |
| improper body position | 7 | 17 | 6 | 4 | 5 | 39 | 50 | 6.24 | 12.48 | 6.24 |
| improper procedure/inattention | 10 | 15 | 26 | 4 | 8 | 63 | 35 | 10.2 | 29.13 | 18.94 |
| improper security of materials | 10 | 33 | 18 | 12 | 8 | 81 | 25 | 11.86 | 47.45 | 35.59 |
| improper security of tools | 0 | 5 | 1 | 0 | 0 | 6 | 20 | 0.13 | 0.66 | 0.53 |
| insect/animal | 123 | 106 | 31 | 1 | 5 | 266 | 80 | 8.72 | 10.9 | 2.18 |
| job trailer | 1 | 5 | 2 | 0 | 1 | 9 | 65 | 1.13 | 1.74 | 0.61 |
| ladder | 16 | 36 | 19 | 5 | 4 | 80 | 80 | 6.61 | 8.26 | 1.65 |
| lifting/pulling/manipulating | 39 | 143 | 62 | 28 | 28 | 300 | 80 | 38.6 | 48.25 | 9.65 |
| light vehicle | 36 | 83 | 55 | 11 | 21 | 206 | 90 | 26.78 | 29.76 | 2.98 |
| LOTO/labeling | 4 | 9 | 9 | 2 | 1 | 25 | 80 | 2.08 | 2.59 | 0.52 |
| lumber | 11 | 31 | 17 | 2 | 6 | 67 | 40 | 7.57 | 18.92 | 11.35 |
| machinery | 30 | 98 | 54 | 19 | 11 | 212 | 80 | 19.44 | 24.29 | 4.86 |
| manlift | 1 | 3 | 3 | 0 | 0 | 7 | 80 | 0.22 | 0.28 | 0.06 |
| mud | 4 | 4 | 9 | 1 | 1 | 19 | 65 | 1.77 | 2.72 | 0.95 |
| nail | 0 | 8 | 2 | 0 | 0 | 10 | 50 | 0.23 | 0.47 | 0.23 |
| no/improper PPE | 3 | 9 | 4 | 1 | 2 | 19 | 20 | 2.48 | 12.41 | 9.93 |
| object at height | 7 | 21 | 10 | 6 | 1 | 45 | 85 | 3.26 | 3.83 | 0.57 |
| object on the floor | 23 | 65 | 23 | 9 | 11 | 131 | 55 | 14.78 | 26.86 | 12.09 |
| piping | 21 | 74 | 27 | 7 | 6 | 135 | 70 | 9.99 | 14.27 | 4.28 |
| pole | 9 | 16 | 13 | 4 | 1 | 43 | 30 | 2.9 | 9.66 | 6.76 |
| pontoon | 0 | 0 | 0 | 0 | 1 | 1 | 16.67 | 0.94 | 5.61 | 4.68 |
| poor housekeeping | 1 | 4 | 2 | 0 | 0 | 7 | 30 | 0.18 | 0.6 | 0.42 |
| poor visibility | 5 | 6 | 3 | 0 | 3 | 17 | 30 | 3.09 | 10.29 | 7.21 |
| powered tool | 7 | 51 | 36 | 9 | 9 | 112 | 85 | 13.4 | 15.77 | 2.36 |
| rebar | 2 | 9 | 3 | 1 | 0 | 15 | 10 | 0.55 | 5.48 | 4.93 |
| relay | 2 | 0 | 0 | 0 | 0 | 2 | 30 | 0.01 | 0.02 | 0.02 |
| repetitive motion | 0 | 1 | 1 | 0 | 0 | 2 | 70 | 0.07 | 0.1 | 0.03 |
| scaffold | 25 | 25 | 27 | 2 | 1 | 80 | 65 | 3.44 | 5.29 | 1.85 |
| screw | 1 | 4 | 1 | 1 | 1 | 8 | 60 | 1.29 | 2.15 | 0.86 |
| sharp edge | 7 | 36 | 10 | 5 | 4 | 62 | 65 | 6.05 | 9.31 | 3.26 |
| slag | 1 | 9 | 5 | 1 | 0 | 16 | 35 | 0.66 | 1.89 | 1.23 |
| slippery surface | 20 | 82 | 37 | 13 | 19 | 171 | 80 | 24.25 | 30.32 | 6.06 |
| small particle | 12 | 64 | 22 | 3 | 1 | 102 | 65 | 3.9 | 6 | 2.1 |
| spark | 2 | 1 | 0 | 0 | 0 | 3 | 50 | 0.02 | 0.04 | 0.02 |
| splinter/sliver | 1 | 8 | 2 | 0 | 0 | 11 | 50 | 0.24 | 0.48 | 0.24 |
| spool | 4 | 14 | 5 | 0 | 1 | 24 | 20 | 1.45 | 7.24 | 5.79 |
| stairs | 28 | 75 | 24 | 6 | 12 | 145 | 80 | 15.23 | 19.04 | 3.81 |
| steel/steel sections | 34 | 90 | 63 | 17 | 9 | 213 | 50 | 17.52 | 35.04 | 17.52 |
| stripping | 3 | 9 | 9 | 1 | 6 | 28 | 75 | 6.52 | 8.69 | 2.17 |
| stud | 0 | 2 | 2 | 0 | 1 | 5 | 30 | 1.08 | 3.61 | 2.52 |
| switch/switching | 6 | 13 | 6 | 2 | 1 | 28 | 30 | 1.97 | 6.55 | 4.59 |
| tank | 12 | 47 | 22 | 5 | 3 | 89 | 35 | 5.99 | 17.12 | 11.13 |
| transformer | 9 | 8 | 13 | 2 | 1 | 33 | 55 | 2.31 | 4.21 | 1.89 |
| uneven surface | 12 | 49 | 27 | 11 | 10 | 109 | 70 | 14.27 | 20.38 | 6.11 |
| unpowered tool | 40 | 123 | 87 | 25 | 20 | 295 | 75 | 31.59 | 42.12 | 10.53 |
| unpowered transporter | 5 | 25 | 7 | 5 | 6 | 48 | 30 | 7.58 | 25.25 | 17.68 |
| unstable support/surface | 1 | 6 | 2 | 3 | 0 | 12 | 30 | 0.91 | 3.03 | 2.12 |
| valve | 13 | 56 | 22 | 11 | 4 | 106 | 70 | 8.47 | 12.1 | 3.63 |
| vault | 0 | 4 | 0 | 0 | 0 | 4 | 20 | 0.06 | 0.29 | 0.23 |
| welding | 7 | 15 | 5 | 1 | 1 | 29 | 65 | 1.71 | 2.63 | 0.92 |
| wind | 2 | 13 | 4 | 1 | 0 | 20 | 50 | 0.67 | 1.33 | 0.67 |
| wire | 12 | 35 | 18 | 5 | 1 | 71 | 75 | 3.71 | 4.95 | 1.24 |
| working at height | 12 | 29 | 18 | 6 | 5 | 70 | 90 | 7.6 | 8.45 | 0.84 |
| working below elev wksp/mat | 5 | 6 | 14 | 0 | 1 | 26 | 70 | 1.86 | 2.66 | 0.8 |
| working overhead | 2 | 6 | 3 | 1 | 0 | 12 | 80 | 0.5 | 0.63 | 0.13 |
| wrench | 6 | 26 | 14 | 1 | 2 | 49 | 80 | 3.33 | 4.16 | 0.83 |

Table 12: Attribute counts at each severity level, median exposure values, and risk values, on the injury portion of Company's dataset (2,189 reports). This table can interactively be explored on the Risk tab of the application.

at the attribute level[19]. This left us with $n = 2189$ cases.

| Severity level | RPRT-ONLY | FRST-AD | RCRDBL-INJRY | RSTRCTD-TM | LST-TM |
|---|---|---|---|---|---|
| Scores | 8 | 32 | 128 | 512 | 2048 |

Table 13: Severity level impact ratings, adapted from [21].

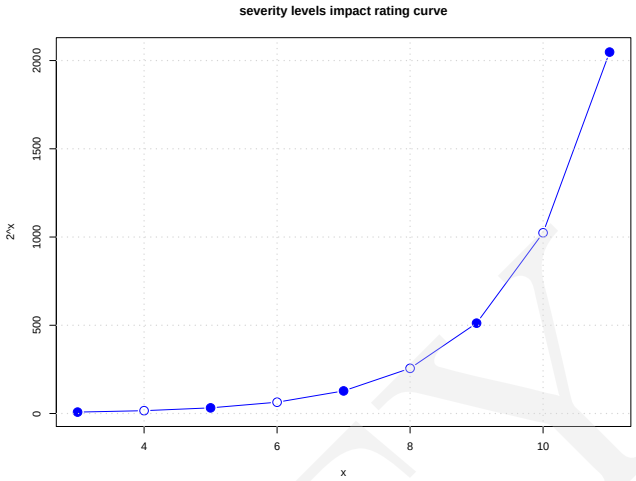



Figure 29: Severity level impact ratings follow an exponential function $f(x) = 2^x$

### 7.2.2 Relative risk

The global risk $R_p$ of $attribute_p$ computed in Eq. 5 was weighted by its exposure value $e_p$ to obtain the relative risk $RR_p$ of $precursor_p$:

$$RR_p = \frac{1}{e_p} \cdot R_p = \frac{1}{e_p} \cdot \sum_{s=1}^{5} \left( n_{p_s} \cdot S_s \right) \tag{6}$$

**Relative risk interpretation**

If two attributes have the same global risk value, the attribute having the *lowest* exposure value will be associated with the *greatest* relative risk. The assumption is that if a rare attribute causes as much damage as a more common one, the rare attribute should be considered riskier by proportion.

Note that risk values allow comparison but do not have an absolute physical meaning. What matters, rather than the risk value itself, is the range in which it falls in comparison to the other values in the data set. E.g., is it lower or greater than the average risk value?

**Exposure values**. Exposure values are probabilities of occurrence of the attributes on any random Company site. They can be seen as *exterior* knowledge, that is combined with the *local* statistics computed from the dataset. Exposure values were obtained through two web-based surveys[20] that were sent out to Company from August to September 2019. 18 and 17 complete responses were obtained[21]. For all attributes but *pontoon*, we computed the median of the responses for each

[19] However, as will be explained in subsection 7.3, *all* 10,250 cases associated with at least one attribute were used to compute Company's empirical risk distribution at the situational level.

[20] https://forms.gle/JNeHzjyfFWxe2fwV9 - https://forms.gle/HQHssPgMRtNfGXwE9

[21] The responses are part of the files delivered, see Section 3.

attribute to get their final exposure values (reported in Table 12). For *pontoon*, the median exposure value was zero, so we took the mean instead.

### 7.2.3 High-delta attributes

It is interesting to inspect the attributes having the greatest relative/global risk difference. The top ten attributes for this quantity are shown in Table 14. Of these top delta attributes, three are related to human factors: *fatigued/dizzy*, *improper security of materials*, and *improper procedure/inattention*, while one is related to environmental conditions: *adverse low temperatures*.

| Attribute | Global risk | $e$ (%) | Relative risk | $\Delta$ |
|---|---|---|---|---|
| adverse low temps | 9.50 | 20.00 | 47.50 | 38.00 |
| concrete | 9.00 | 20.00 | 44.80 | 35.80 |
| improper security of materials | 11.90 | 25.00 | 47.50 | 35.60 |
| fatigued/dizzy | 5.90 | 15.00 | 39.20 | 33.30 |
| improper procedure/inattention | 10.20 | 35.00 | 29.10 | 18.90 |
| unpowered transporter | 7.60 | 30.00 | 25.30 | 17.70 |
| steel/steel sections | 17.50 | 50.00 | 35.00 | 17.50 |
| object on the floor | 14.80 | 55.00 | 26.90 | 12.10 |
| lumber | 7.60 | 40.00 | 18.90 | 11.40 |
| tank | 6.00 | 35.00 | 17.10 | 11.10 |
| mean | 6.36 | 54.36 | 11.64 | 5.28 |
| min | 0.01 | 10.00 | 0.02 | 0.02 |
| max | 38.60 | 90.00 | 48.25 | 38.04 |

Table 14: Top delta attributes. Mean, min and max are over all 91 attributes (one attribute, *soffit*, was removed because it did not appear in any report).

Some of the top delta attributes (e.g., *fatigued/dizzy*, *lumber*) have global risk values close to or below average, but their exposure values are very low. This means that these attributes were estimated to be rare by Company's survey respondents, but that in Company's dataset, they contribute normally to injuries. As a result, the relative risk values of these attributes are much higher than their global ones (since $RR_p = \frac{1}{e_p} \cdot R_p$).

Another class of attributes among the top delta ones is that of the attributes with large global risk values and relatively modest exposure values, such as *steel/steel sections*, and *object on the floor*. These attributes are associated with many injuries at all severity levels (see Table 12), i.e., they are top injury contributors. Nevertheless, they were not estimated as being found everywhere, every time, by Company's survey respondents: their exposure values are only slightly greater than average (in $[50, 55]$), and far below the maximum $(90)$. As a result, the relative risk values of these attributes are almost equal to twice their global risk values.

## 7.3 Report-level risk

As shown in Eq. 7, we define safety risk at the *situational level* as the sum of the risk values of all the attributes that were identified as present in the corresponding injury report:

$$R_{report_r} = \sum_{p=1}^{P} \left( RR_p \cdot \delta_{rp} \right) \tag{7}$$

Where $RR_p$ is the relative risk of $precursor_p$ as described in Eq. 6, $\delta_{rp} = 1$ if $precursor_p$ is present in $report_r$, and $\delta_{rp} = 0$ otherwise. While only the 2,189 reports that were associated with a known severity and with at least one attribute were used to compute the attribute-level risk values, all 10,250 cases associated with at least one attribute were used to compute Company's empirical distribution of risk at the situational-level. Indeed, a near miss, for example, is a legitimate snapshot of the work

environment that is associated with some amount of risk and has the potential to create an accident. Such observations are valuable and should be taken into account.

In what follows, Company's empirical risk values at the situational level are referred to as the *original* or *historical* sample, and we use the variable $X$ to denote them. Their histogram is shown in Fig. 30.

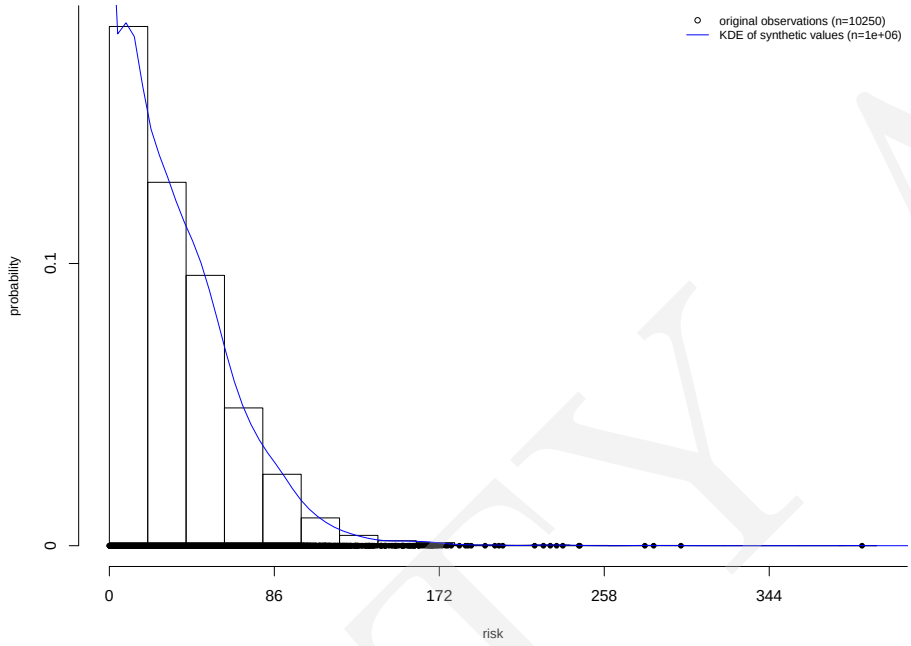



Figure 30: Histogram of original risk values with Kernel Density Estimate (KDE) of the simulated risk values.

**Situational risk distribution interpretation**

We observe that Company's risk distribution is **heavy-tailed**, and follows a **lognormal behavior**. That is, the bulk of the reports are associated with low to medium risk, while a smaller number of reports are associated with high to extreme risk (the tail of the distribution). Similar distributions have been observed in other construction datasets before [38].

## 7.4 Safety risk generator

### 7.4.1 Motivation

Since $X$ is a historical sample of finite, relatively small size ($n = 10,250$), risk estimates based on $X$ are biased. This is especially true for the high and extreme values (tail of the distribution), for which observations are sparser. For instance, the highest risk value observed in the original sample is surely not the highest value that can ever be reached. Had we recorded data over a longer period of time, we would likely have observed more extreme values. To address this issue, we generated synthetic risk values that are faithful to the historical observations, in a way that will be explained next. Populating the distribution with new observations enables more accurate risk estimation.

### 7.4.2 Quantile

The 100-quantile $Q$ at $p$ of a continuous random variable $X$ is defined as the threshold value $x$ under which $X$ would fall $p$ % of the time:

$$Q(p) = x \mid P[X \leq x] = p \tag{8}$$

**Quantile interpretation**

In finance and insurance, the quantile is known as the *value-at-risk*. For instance, the 99.95% value-at-risk $Q(99.95)$ at 10 days represents the amount of money that the loss can only exceed with 0.5% probability in the next 10 days. In other words, the corresponding fund reserve would cover 199 losses over 200 $(199/200 = 0.995)$.
The quantile is also associated with the notion of *return period* $T$ in hydroclimatology. For example, the magnitude of the 100-year flood $(T = 100)$ corresponds to the streamflow value that is only exceeded on average by 1% of the floods, assuming one flood per year. This value is given by $Q(1 - 1/T) = Q(0.99)$.

› In Company's dataset, $Q(0.99)$ corresponds to the risk value that is observed in one situation over a hundred. The median value, given by $Q(0.5)$, corresponds to the risk observed in one situation over two.

### 7.4.3 Methodology

Our simulation technique is based on the *smoothed bootstrap with variance correction* [35]. It can generate values outside of the limited historical range such as new extremes, while not reproducing spurious features of the original data like noise. The algorithm can be broken down into the following steps:

**Algorithm 2** Safety risk generator

**Input:** original situational risk values stored in $X$ of size $R$, desired number of simulated values $R_{sim}$
**Output:** $R_{sim}$ synthetic situational risk values stored in $X_{sim}$

1: **for** $j \in [1, ..., R_{sim}]$ **do**
2: **choose** $i$ uniformly with replacement from $[1, ..., R]$
3: **sample** $\epsilon_X$ from $\mathcal{N}(0, h_X^2)$
4: **do** $X_c = \overline{X} + (X_i - \overline{X} + \epsilon_X)/\sqrt{1 + h_X^2/{\sigma_X}^2}$
5: **if** $X_c \geq 0$ **then**
6: **record** $X_{sim}[j] \leftarrow X_c$
7: **else**
8: **continue**
9: **end if**
10: **end for**

Where $\mathcal{N}(0, h_X^2)$ is the standard normal distribution with variance $h_X^2$, and $\overline{X}$ and ${\sigma_X}^2$ are the mean and variance of the original risk values. In lines 5-8, we simply discard the synthetic value if it is negative to be consistent with the definition of risk. $h_X$ is computed using a widespread heuristic called Silverman's rule-of-thumb:

$$h_X = \frac{0.9 \min\left(\hat{\sigma}_X, \frac{Q_3 - Q_1}{1.34}\right)}{R^{1/5}} \tag{9}$$

Where $Q_1$ and $Q_3$ denote the first and third quartiles of $X$, respectively.

### 7.4.4 Results

Fig. 30 shows the histogram of Company's original sample and the Kernel Density Estimate (KDE) of $R_{sim} = 10^6$ simulated values. It can be clearly seen that the synthetic values are faithful to the original ones since the KDE of the simulated values fits the original observations very well. Also, while honoring the historical data, the smoothed bootstrap generated values outside the original range, as

desired. Indeed, the maximum risk value in the original sample was 396.4, while the maximum of the simulated values is 402.2. Table 15 compares the quantiles of the original observations to that of the simulated ones.

| risk observed in one situation over: | 2 | 5 | 10 | 100 | 500 | 1K | 10K | 50K | 500K |
|---|---|---|---|---|---|---|---|---|---|
| original observations | 29.50 | 57.90 | 75.80 | 129.30 | 175.90 | 217.60 | 297.60 | 373.00 | 390.40 |
| simulated observations | 29.70 | 57.90 | 76.00 | 129.80 | 175.80 | 214.70 | 379.20 | 393.60 | 396.00 |

Table 15: Quantile estimates based on original and simulated values.

**Quantile estimates interpretation**

The quantile estimates of Table 15 are roughly equivalent before reaching the tails. This makes sense, because the bulk of the original observations were in the low-to-medium range, enabling quite accurate estimation for this range in the first place. The smoothed bootstrap populated the tail with new observations, yielding a slightly higher, more accurate estimate of the extreme quantiles.

Based on the $10^6$ simulated values which represent decades of new (pseudo) observations, we propose the risk ranges shown in Table 16. These ranges are more robust and accurate than the ones that would have been estimated from Company's 10,250 original observations only.

| interpretation | quantile range | risk range |
|---|---|---|
| routine | 0-0.25 | 0-13 |
| caution | 0.25-0.5 | 13-30 |
| high caution | 0.5-0.75 | 30-52 |
| alert | 0.75-0.99 | 52-130 |
| high alert | 0.99-1 | $\geq$ 130 |

Table 16: Proposed risk ranges.

**Key points**

- The global risk computed from *local* occurrence statistics in the dataset is combined with the exposure values (*exterior* knowledge) to obtain the relative risk of a given attribute.
- The situational-level risk is the sum of the relative risk values of the attributes present.
- Company's situational risk follows a heavy-tailed lognormal-like distribution, which is consistent with previous research findings [38].
- Generating synthetic situational risk values allows for more accurate risk estimation.

## 8 Text-based search engines

**Live video demonstration**

https://vimeo.com/409516633

This tab exposes a text-based search engine that allows the user to perform two types of search over the full Company's dataset: exact search (binary) and semantic search (synonymy-aware).

With the binary search, only the reports containing the exact same keywords/keyphrases as that entered by the user are retrieved. One exception: when stemming is used, the query *engineer* will return reports containing *engineer*, but also *engineers*, *engineered*, *engineering*, *engine*, etc.

Also, when a query contains more than one item (word or phrase), the user can specify whether the results should contain *one of*, *all*, or *all* the items of the query *in order*.

With the semantic search, however, synonymy is captured, so the reports containing words or phrases similar to the query are retrieved, even if they do not contain any of the exact query words/phrases.

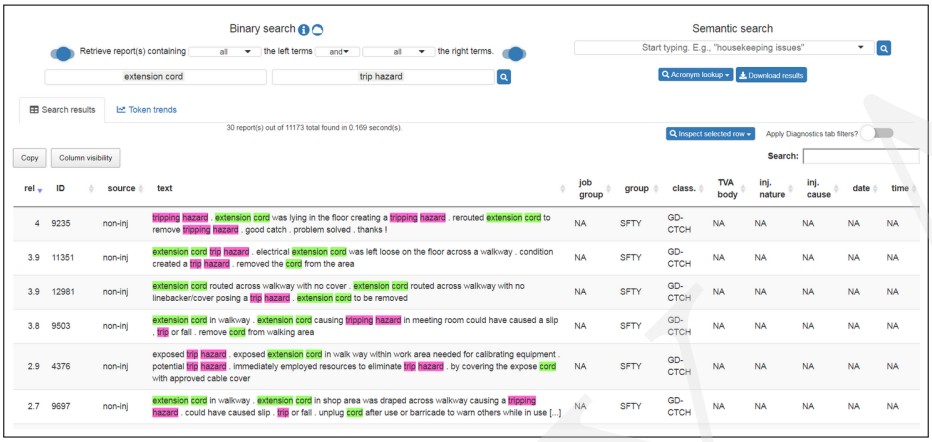



Figure 31: Text search tab.

**What questions can this part answer?**

- Which reports contain the phrases `extension cord`, and `trip hazard`?
- Which reports contain phrases equivalent to `extension cord` or `rolled ankle`?

› The binary search interface can be used to answer this question. 30 reports match the query.

› Here, it is the semantic search interface that should be used. Results for *extension cord* include *extension cord*, but also synonyms like *power cord*, *light cord*, and *electrical cord*. Likewise, results for the query `rolled ankle` include reports featuring the phrases `rolled foot`, `twisting their knee`, or `twisting his ankle`. The binary search engine would not have been able to retrieve such cases, despite the fact that they are relevant, because it does not capture synonymy and higher level semantic similarity. The semantic search interface is thus very useful when the user wants to retrieve the reports in the collection that are associated with a topic, meaning, or concept, rather than a precise combination of exact words and/or phrases.

**Key points**

- Use the **binary search engine** to retrieve reports containing the exact same words as the query (or the same root, when stemming is used).
- Use the **semantic search engine** to retrieve reports featuring words or phrases that have the same meaning as the query.

In what follows, we provide details about the implementation of the binary and semantic search capabilities.

## 8.1 Binary search: inverted index

The inverted index is the fundamental data structure in *information retrieval*. It is the basis of all traditional search engines. The inverted index is constructed offline from the entire collection, which

can take a lot of time (and quite some disk space), but it allows to respond to a variety of queries, including phrase queries, in real-time. It is a dictionary which contains, for each term in the vocabulary, the term's *positional list* in each document of the collection.

For instance, below is an example of the value of the inverted index for the key `googl` (which is the stem corresponding to e.g., `google` or `googles`):

```
{9799: 15,
 10405: 25,
 16838: [5, 25],
 22880: 26,
 26525: 14,
 27687: 0,
...
 202793: 23,
 203277: 18,
 219905: [10, 59],
 224259: [81, 203, 443],
 230996: 349}
```

This tells us, e.g., that `googl` appears in document `9799` at position 15, and in document 16838 at positions 5 and 25. Recording not only the IDs of the documents in which a given term appears but also the positions of the term in the document allows to answer *phrase* queries.
For instance, if `safety` and `googles` appear in the same document, and if their positions in that document are consecutive (e.g., 16 and 17), we can infer that the document contains the phrase `safety googles`.

**Preprocessing**. Before indexing the collection, some text cleaning is necessary. Whether stopwords should be removed is a tricky issue[22], because some queries might contain stopwords (e.g., `not safe`), or might even be composed of stopwords only (e.g., `do not`). To allow for more expressive search, we decided to keep stopwords in the index. The only tokens we removed were single-character punctuation marks.

We also used *stemming*[23] (this is why the previous example is for *googl* and not *google*). Stemming is another important issue in building an inverted index. It involves reducing a word to its root form (also known as its stem). Stemming allows to retrieve more relevant documents for a given query. For instance, for the query `scaffold piece`, the user might be interested in retrieving the reports containing `scaffold pieces` and `scaffolding piece(s)`.

However, stemming can also somewhat reduce the relevance of the results. For instance, since `engine`, `engines`, etc. are both mapped to the same stem as `engineer` (that stem being `engin`), a query like `engine room` will return documents containing `engineer room`, which might be irrelevant to the user. This is why the user has the ability to enable or disable stemming very easily, by clicking a button.

## 8.2 doc2vec

`doc2vec`, also known as *Paragraph Vector* [15, 24], is a simple but clever modification of `word2vec` that can learn *document embeddings* in addition to, or in lieu of, word embeddings. More precisely, each document in the collection is represented as a special word (*paragraph token*), and is as such associated with a vector, initialized at random.
$(context, target)$ pairs are then sampled from each document, and the same network as in `word2vec` is asked to perform one of two possible prediction tasks. In the first task, the goal is to predict the $target$ word given the $context$ words and the paragraph token. This configuration is known as the *distributed memory* variant of `doc2vec`.

[22] https://www.elastic.co/guide/en/elasticsearch/guide/current/pros-cons-stopwords.html
[23] https://www.elastic.co/guide/en/elasticsearch/guide/current/stemming.html

In the second task, the goal is to predict the *context* words from the paragraph token. In other words, using only the vector of the document, the network is asked to discriminate well between words sampled from this document and negative examples sampled at random from other documents in the collection. This second configuration is referred to as the *distributed bag-of-words* one. It is conceptually equivalent to the skip-gram variant of `word2vec`. We used this architecture in our experiments, because it was giving better results. One of the possible explanations is that this variant has less parameters. Indeed, unlike the distributed memory architecture, it does not train word embeddings in addition to document embeddings. Therefore, it requires smaller amounts of data to perform well.

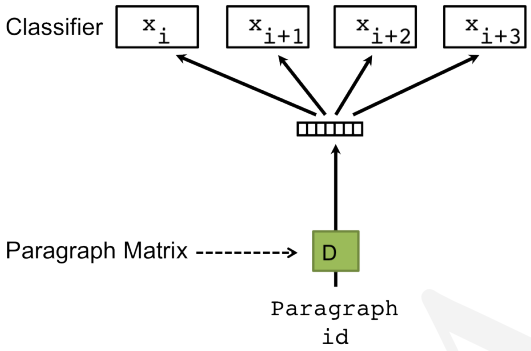



Figure 32: Distributed bag-of-words architecture of `doc2vec`.

After training, it is possible to use the `doc2vec` model in *inference mode* to obtain a vector for a new document, such as a user query. The cosine similarities between the query vector and the vectors of all documents in the collection can then be computed. Finally, results can be ranked by decreasing order of similarity and displayed to the user. This is the approach we used to develop the *semantic search* functionality of the app's Text search tab.

Parameter-wise, we trained document vectors of dimension 24, for 20 epochs, with gensim. All other parameters were left to their default values. At inference time, we use gensim's `infer_vector` method[24] with `steps` = 200 and `alpha` = 0.00025 to stabilize the stochastic nature of the process[25] and make sure results are consistent enough when entering the same query several times in a row.

## 9 Word embeddings

The results of this section are available in the Word Embeddings tab of the application. Among other things, the user can interactively visualize a 2D map of the word embedding space. By clicking on a point, or typing a word, the user can also inspect the nearest neighbors of that word.

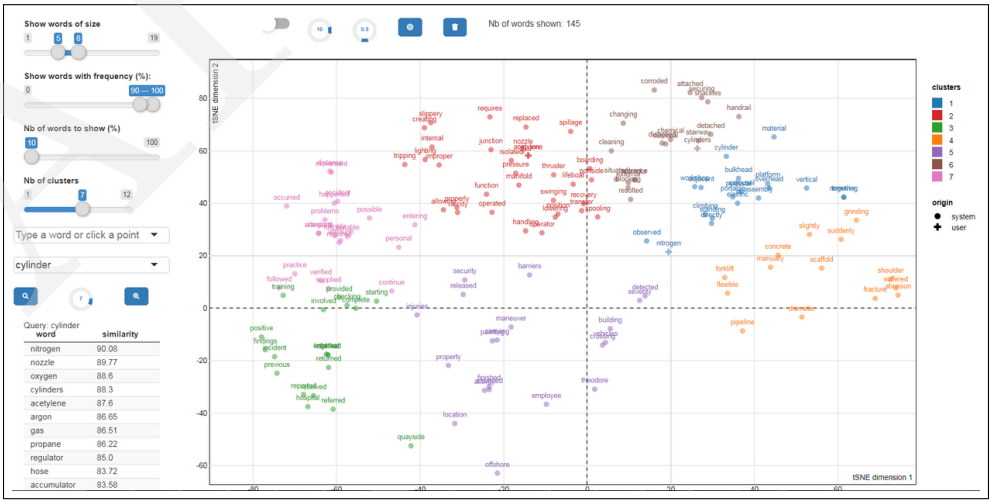



Figure 33: Word Embeddings tab.

[24] https://radimrehurek.com/gensim/models/doc2vec.html#gensim.models.doc2vec.Doc2Vec.infer_vector

[25] https://github.com/RaRe-Technologies/gensim/issues/447

**Relevant journal paper**

Tixier, A. J. P., Vazirgiannis, M., & Hallowell, M. R. (2016). Word embeddings for the construction domain. arXiv preprint arXiv:1610.09333. https://arxiv.org/pdf/1610.09333.pdf

**What questions can this part answer?**

- Which words frequently appear close to each other in Company's incident reports?
- What words are the most similar to a given word?

› The first question can be answered by inspecting the interactive 2D map of word embeddings. The words that appear close to each other in the map are either (1) words that frequently co-occur in the corpus or (2) words that never appear close to each other but that are frequently found in similar contexts (i.e., surrounded by the same words) in the corpus.
› To answer the second question, the user can simply click on a point in the map, or enter a query in the search bar on the left. The 10 most similar words, in terms of cosine similarity between vectors, will be displayed in the map and in a table below the search bar.

In what follows, we provide an overview of what are word embeddings and how they are learned. A relevant reference for the construction domain is [42].

## 9.1 Limitations of the vector space model

Traditionally, text has been represented with the vector space model [32]. Within this framework, each unique term (i.e., unigram, bigram, etc., up to a certain order) in the collection of documents is considered an independent dimension of the space, and is encoded as a so-called one-hot vector. Documents are then represented as bags of words with the TF-IDF weighting scheme [32].

This approach is limited because it considers terms as independent units. Therefore, semantic (meaning) and syntactic (grammar) dependence between individual terms are completely overlooked. For instance, the word *hammer* may be represented as the vector $[0, 0, 1, 0, ..., 0, 0, 0, 0]$ and the word *tool* as the vector $[0, 0, 0, 0, ..., 0, 0, 1, 0]$. It follows that $\overrightarrow{hammer} \cdot \overrightarrow{tool} = 0$. Put differently, according to the vector space model, *hammer* and *tool* are orthogonal i.e., they are as dissimilar as *hammer* and *truck*.

## 9.2 Distributed word representations

The aforementioned limitation has motivated the use of *distributed representations of words*, also known as *word embeddings* or *word vectors* [2]. As shown in Fig. 34, word embeddings map each word in the vocabulary to a real-valued vector in a dense *continuous* space of dimension $m \ll |V|$. The $m$ features represent concepts shared by all words. Typically, while $|V|$ lies in the $[10^5, 10^6]$ interval, $m$ takes values within $[100, 500]$.

Recently, [26] showed that high quality word embeddings could be obtained in a very fast way as a side ef-

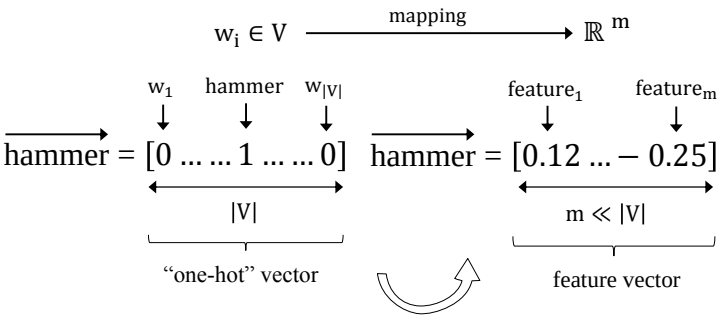



Figure 34: Traditional 'vector space' model (left) versus word embeddings (right). Where $V$ is the vocabulary of size $|V|$ (comprising words $w_1$ to $w_{|V|}$), and $m$ is the dimension of the embedding space.

fect of feeding very large amounts of text to a shallow neural network. This method was named `word2vec`.

Moreover, the embeddings obtained after training `word2vec` on large corpora were shown to encode very interesting syntactic and semantic regularities as simple vector operations [27]. For instance, linear translations were shown to capture many concepts like *pluralization*, *gender*, *country-capital*, or *genius-field*. Classical examples include $\overrightarrow{cats} - \overrightarrow{cat} = \overrightarrow{mice} - \overrightarrow{mouse}$, $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} = \overrightarrow{queen}$, $\overrightarrow{france} - \overrightarrow{paris} = \overrightarrow{italy} - \overrightarrow{rome}$, and $\overrightarrow{einstein} - \overrightarrow{scientist} + \overrightarrow{painter} = \overrightarrow{picasso}$.

### 9.3 word2vec

`word2vec` is based on the *Distributional Hypothesis* [22], which can roughly be summarized by the well-know quote by John Firth, "we shall know a word by the company it keeps", and is simply illustrated by Fig. 35.

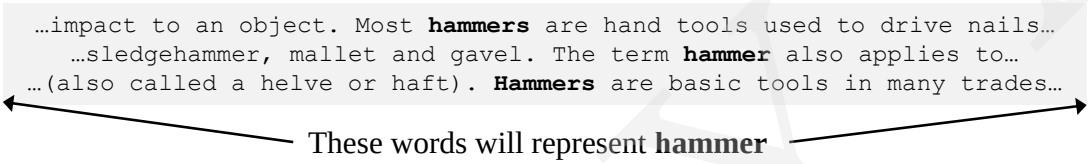



Figure 35: Intuition for the Distributional Hypothesis.

More precisely, `word2vec` first builds $(context, target)$ pairs by parsing the input text from start to finish, where $target$ is a given word and its $context$ of size $2n$ is made of the $n$ preceding and $n$ following words. Reasonable values of $n$ are around 5, which captures much richer word-word relationship information than 2-gram or 3-gram models. `word2vec` then iteratively passes the $(context, target)$ pairs to a shallow neural network, with the task of predicting either the $target$ word given its $context$ (CBOW architecture) or the $context$ of a given $target$ word (skip-gram architecture). The latter variant is shown in Fig. 36.

The network is qualified as being *shallow* as opposed to *deep* as it does not have any hidden layer, that is, any layer with a non-linear activation function. It is only made of an input layer, a projection layer, and an output layer, with two matrices of parameters: input → projection and projection → output. The only operations that this network performs are linear ones.
Formally, the objective of the skip-gram model is to minimize:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{i=t-n \\ i\neq t}}^{t+n} -\log P(w_i|w_t) \qquad (10)$$

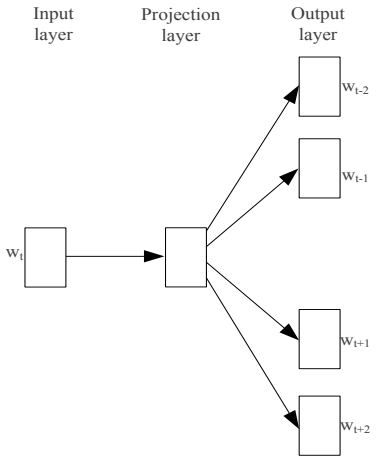



Figure 36: Skip-gram architecture with a context of size 4 (2 preceding words, 2 following words). The task is to predict the context words given the current word $w_t$.

Where $2n$ is the context size, and the training corpus is a sequence of words $w_1, w_2, ..., w_T$. Note that to obtain good results we must have $T \gg |V|$.

At each iteration, the error gets backpropagated via stochastic gradient descent and accordingly updates the weights of the projection-output and input-projection matrices. Those weight matrices

precisely contain the desired word vectors. They are initialized randomly (i.e., words are initially dispersed at random within the space), and then, as training takes place, words get closer/more distant to/from each other as a reaction to the error, in an attraction-repulsion spring-like fashion [31]. The movements are coarse at first and then more and more subtle as the network gets better and the error diminishes. Upon several epochs of training with a decreasing learning rate, the vectors are ready. While both the input and the output parameter matrices can theoretically be used or combined, the input matrix of dimension $|V| \times m$ is generally used as the word vectors [27], and we followed this convention.

To sum up, the word embeddings are obtained as a side-effect of training `word2vec` on an artificial prediction task. That is, `word2vec` is never actually used for the task it was trained to perform. What we are interested in are its parameters.

**☑ Key points**

- Word vectors are obtained as a *side effect* of training `word2vec`.
- Thanks to its simplicity, `word2vec` is extremely fast to train.
- Since the process is stochastic, running `word2vec` twice on the same corpus will not return the same word vectors; however, word-word similarities and regularities are invariant.

`word2vec` was released as an open source project[26] by Google in 2013 and has since then been ported to other languages. In this project, we used the popular Python implementation `gensim` [30]. More precisely, we trained a vector of dimension 24 for each token that occurred at least 5 times in the collection of reports, and run the skip-gram variant of `word2vec` for 5 epochs, leaving all other parameters to their `gensim`'s default values. We used relatively small word vectors ($d = 24$) due to the small size of the corpus. Larger dimensionalities are justified when there are much more data to feed to the model, e.g., $d = 300$ for an entire Wikipedia dump.

We gathered all the text available for each report[27] in order to train the embeddings. Within the resulting embedding space, words sharing similar meanings occupy neighboring positions, as shown in Fig. 37 and in Table 17. While both the input and the output weight matrices can theoretically be used or combined, the input matrix of dimension $|V| \times m$ is generally used as the final word embeddings [27].

| worker | | relay | | sprained | | cord | |
|---|---|---|---|---|---|---|---|
| employee | 92.76 | circuit | 93.62 | sprain | 94.75 | extension | 93.07 |
| individual | 90.54 | input | 90.55 | straining | 93.9 | table | 88.03 |
| heo | 80.52 | relays | 90.44 | hip | 93.42 | cords | 85.14 |
| contractor | 80.51 | voltage | 90.3 | bruised | 93.28 | unplugged | 84.41 |
| laborer | 80.5 | carrier | 90.1 | dislocated | 93.11 | power | 83.94 |
| he | 80.35 | voltages | 88.63 | bruise | 92.93 | plugged | 83.19 |

Table 17: Top 6 words most similar to some query words, in terms of cosine similarity (in %) between word vectors.

[26] https://code.google.com/archive/p/word2vec/

[27] Like for the NLP tool (see section 4), text was extracted from the `Short Description`, `What Happened`, `Why Occur`, `Immediate Actions`, `Comments`, `Activity`, `Cause`, `Description`, and `Object` columns.

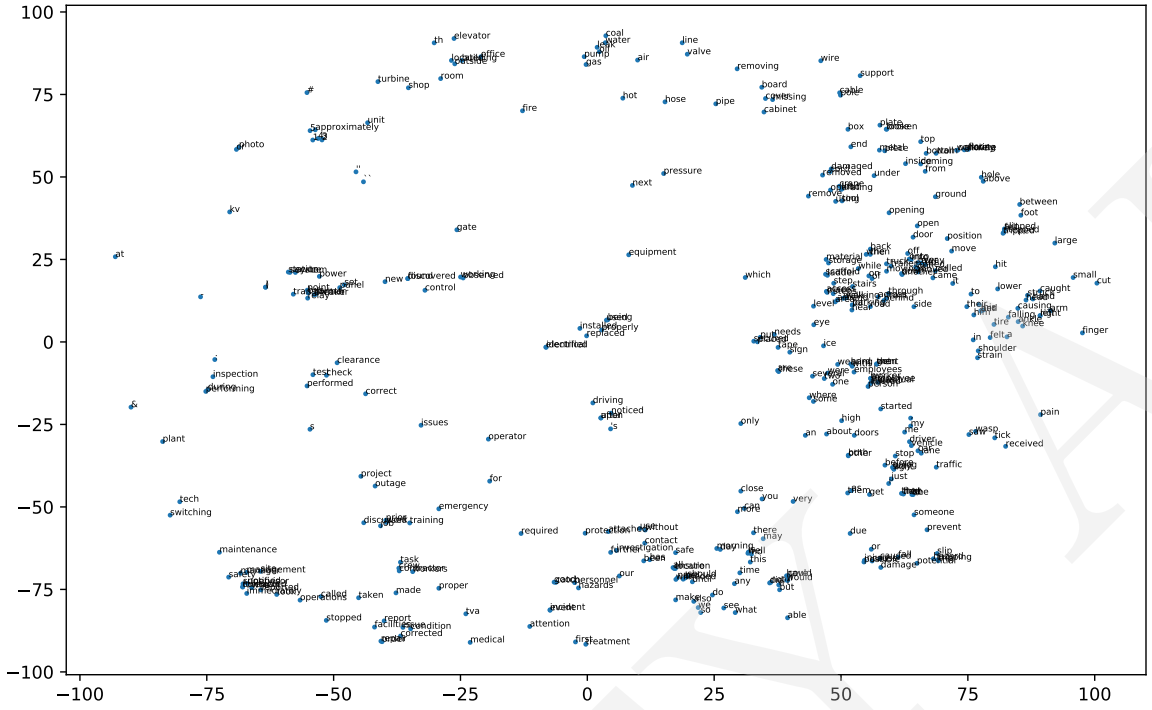


Figure 37: t-SNE visualization of a subset of the word embeddings. Note that this plot can interactively be explored in the web app.

## 10 Topic modeling

Here, we capitalized on the popular `LDAvis` R package [34] to build our visualization tool.

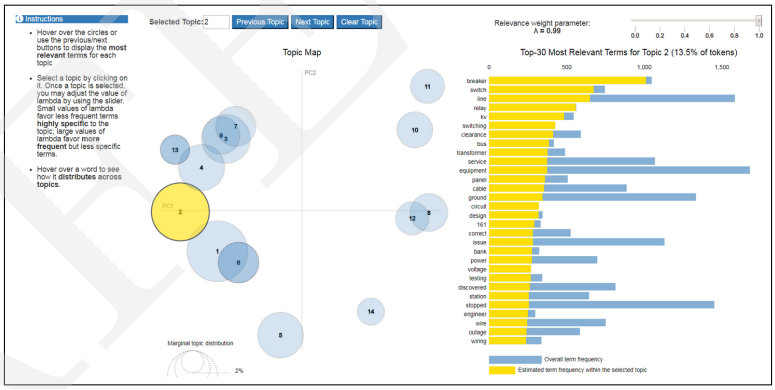


Figure 38: Topic modeling tab.

**What questions can this part answer?**

- What does each topic in Company's collection represent?
- How does a single word distribute over topics?
- What is the importance of each topic?
- Which topics are similar/dissimilar?

All of the questions above can be answered from interacting with the visualization tool in the Topic modeling tab of the app. We explain how in what follows.

➤ The answer to the first question is provided in subsection 10.2 below.

› To inspect the probability distribution of a word over topics, hover over a word in the barplot on the right. The greater the circle of a given topic, the likelier is the word to be found in that topic.
› The importance of each topic, in terms of the proportion of the collection in which this topic appears, is given by the size of the circles.
› The placement of the topics on the 2D map indicates topic similarity, in the sense that nearby topics are more similar than distant topics. For instance, topic 8 and topic 12 in the web app, which are very close to each other, are related to overexertion, sprains and strains. See the example provided in subsection 10.2 below for more details.

### 10.1 Topic modeling generalities

Topic models uncover shared *latent* topics in a collection of textual documents. The number $K$ of topics is prescribed by the user. Each topic is defined as a probability distribution over the terms in the vocabulary, and each document is defined as a probability distribution over the $K$ topics. With Latent Dirichlet Allocation (LDA) [3], both distributions are Dirichlet distributions[28].

The central LDA parameter is $K$, the number of topics. If this number is too large (i.e., more than an dozen), a large proportion of topics will overlap. This might not be problematic if LDA is used as a way to extract features to be passed to another machine learning algorithm downstream. However, if the goal is data exploration by humans, this is likely to make topic interpretation more difficult.

Two other important parameters are `alpha` and `eta`, the *concentration* parameters of the distributions of topics over documents and terms over topics (respectively), which control the shape of these distributions. More precisely, large values of `alpha` and `eta` (much greater than 1) lead to spread-out probability distributions: documents are assumed to feature many topics, and topics to be described by many words. Conversely, when `alpha` and `eta` are much less than 1, the distributions are sparse, and the probability mass is highly concentrated over a few components. That is, topics are described by a few words, and documents load on a couple of topics only. Usually, the `alpha` and `eta` parameters are set to equal heuristic values, such as for instance $1/K$. [16].

### 10.2 Term relevance

Once a topic model has been trained, the topics have to be interpreted, which is not an easy task. To this purpose, the most relevant terms for each topic are usually inspected. In this project, we used the definition of relevance proposed by [34], which is both simple and flexible. More precisely, according to this definition, the relevance of term $w$ to topic $k$ is defined as:

$$r(w, k|\lambda) = \lambda \log(\phi_{kw}) + (1 - \lambda) \log\left(\frac{\phi_{kw}}{p_w}\right) \tag{11}$$

where $\phi_{kw}$ is the probability that $w$ belongs to topic $k$, and $p_w$ is the marginal probability of $w$, that is, the normalized frequency of term $w$ in the collection. Finally, $\lambda \in [0, 1]$ is a tuning parameter that interpolates between two types of relevance: when $\lambda = 1$, the relevance of a term is solely based on how likely this term is to be observed in topic $k$; on the other hand, when $\lambda = 0$, the relevance of a term is based on the ratio of its topic-specific probability to its probability of occurrence in the collection. Therefore, using low values of $\lambda$ penalizes the very frequent terms.

Let's consider an example. For topic 12 in the application, the top 10 most relevant terms for $\lambda = 1$ are *felt*, *back*, *pain*, *shoulder*, *knee*, *left*, *strain*, *lower*, *medical*, and *pop*, which are quite generic. However, setting $\lambda = 0.1$, the top 10 most relevant terms become *pop*, *pain*, *felt*, *strain*, ***twinge***, ***ibuprofen***, ***kneeling***, ***groin***, *popped*, and ***bicep***. The more specific terms (in bold) help us towards interpreting topic 12 as related to overexertion, sprain and strain injuries.

[28] https://en.wikipedia.org/wiki/Dirichlet_distribution

**Relevance parameter interpretation**

- The relevance of a term to a given topic depends on the $\lambda$ parameter in $[0, 1]$.
- Small values of $\lambda$ favor terms that are **not necessarily very frequent** at the collection level, but that are **highly specific** to the topic.
- Large values of $\lambda$ favor **very frequent** terms that are **less specific** to the topic.
- The optimal value of $\lambda$ depends on the corpus and the user. $0.6$ is a good default [34].
- In the Topic modeling tab of the app, the user can interactively tune $\lambda$.

Note that when selecting a topic, setting $\lambda$ to a large value and hovering over terms, any topic can be associated with a large circle, not only the selected topic. This makes sense, since for large values of $\lambda$, very frequent terms are not penalized, and thus the most relevant terms are quite general and not very specific to the current topic. Conversely, when $\lambda$ is small, hovering over any term will always return a very large circle for the selected topic and much smaller circles for all other topics, because with this configuration, frequent terms are penalized, which favors terms that are highly specific to the current topic.

### 10.3 Experimental setup

Like for the NLP tool, word2vec and doc2vec, all the text fields associated with each report were retained, so as to get as much text as possible. To remove noise, only the top 6000 most frequent words that were at least 3 characters in length were retained. The probabilities $\phi_{kw}$ and $p_w$ were obtained via Collapsed Gibbs Sampling[29] (a Markov chain Monte Carlo algorithm) as implemented in the `lda` R package[30], with parameters $K = 14$, `G` $= 1600$ iterations, `alpha` $=$ `eta` $= 0.02$, and `compute.log.likelihood` $=$ `TRUE`. All other parameters were left to their default values.

## A Appendix: attribute importance for XGBoost

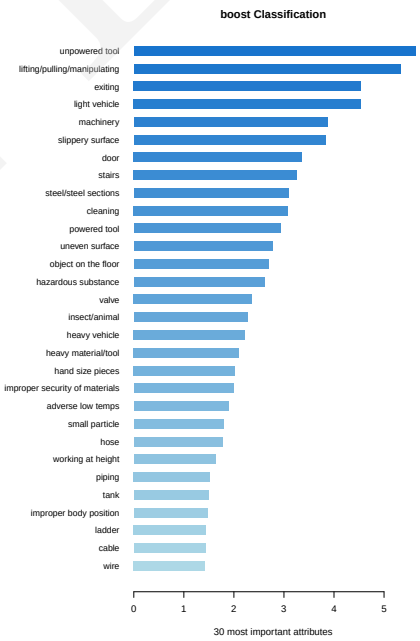



Figure 39: Most important attributes for `Classification` according to the XGBoost model.

[29] https://en.wikipedia.org/wiki/Gibbs_sampling
[30] https://cran.r-project.org/web/packages/lda/index.html

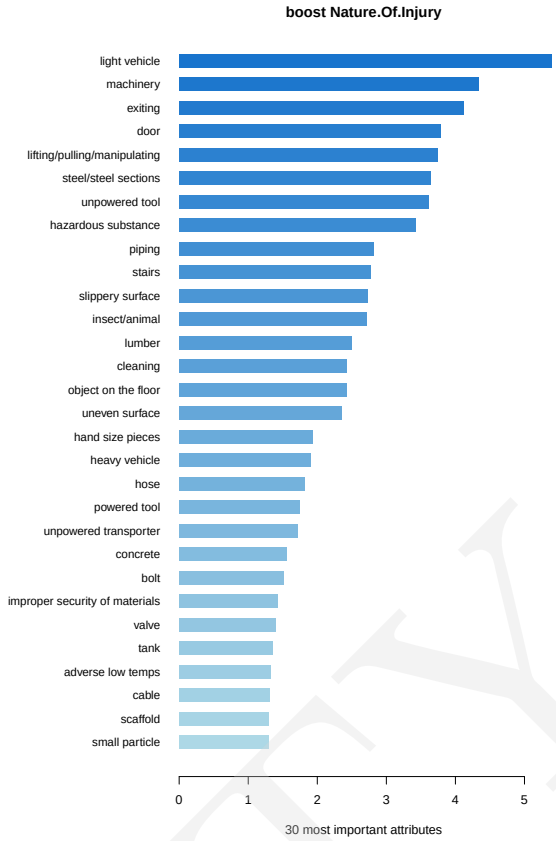



Figure 40: Most important attributes for `Nature Of Injury` according to the XGBoost model.

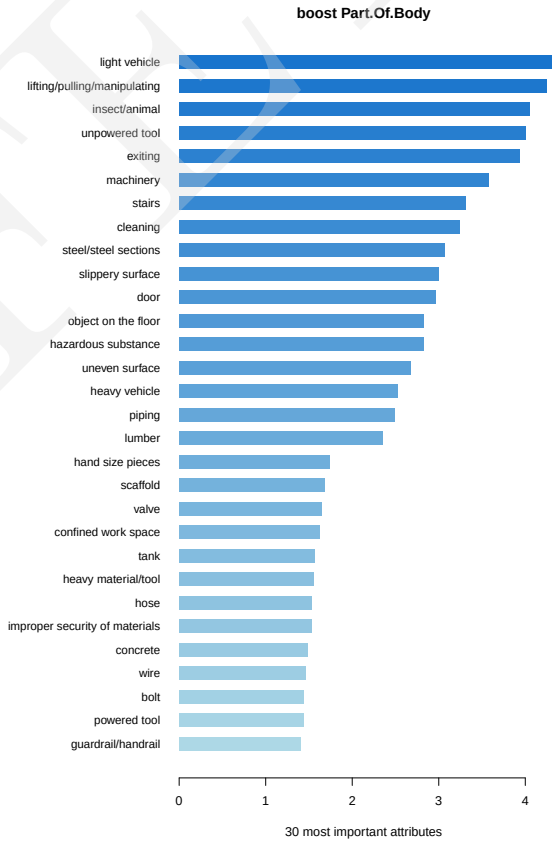



Figure 41: Most important attributes for `Part Of Body` according to the XGBoost model.

# B Appendix: attribute importance for Random Forest

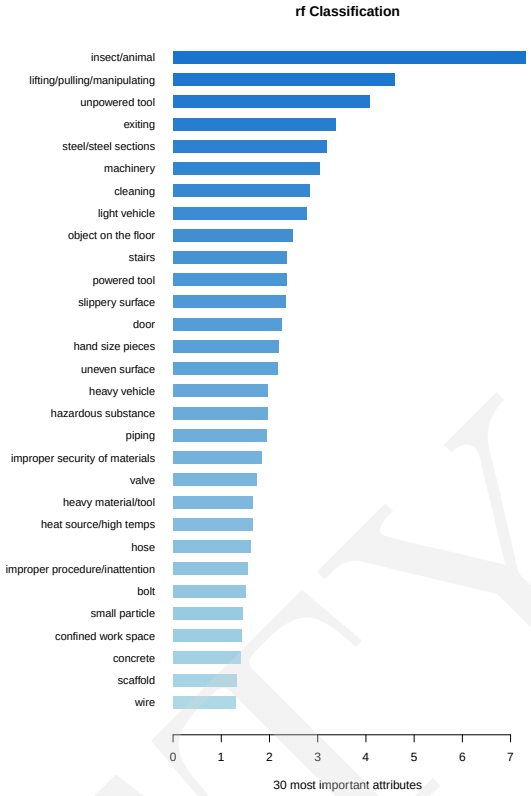



Figure 42: Most important attributes for `Classification` according to the Random Forest model.

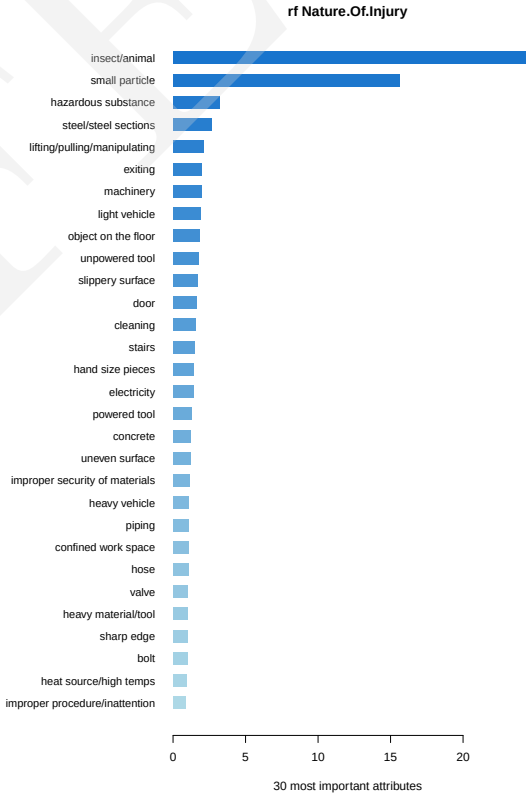



Figure 43: Most important attributes for `Nature Of Injury` according to the Random Forest model.

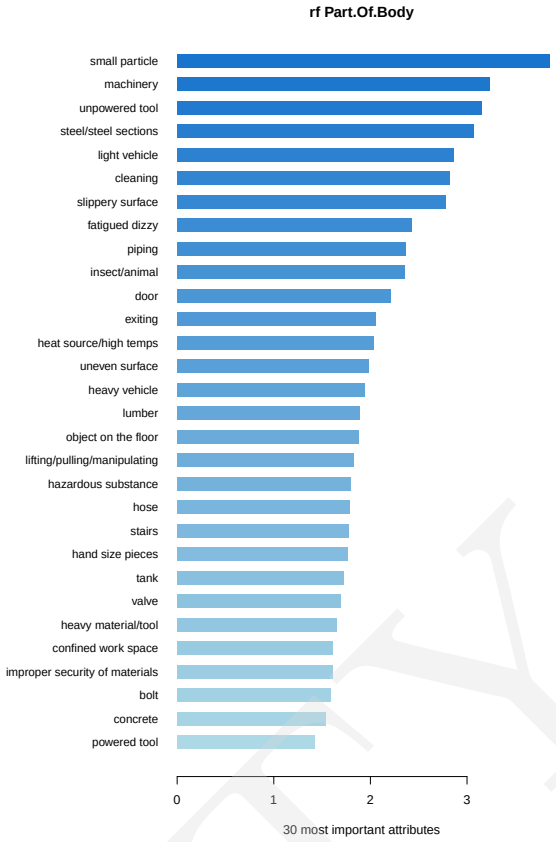



Figure 44: Most important attributes for `Part Of Body` according to the Random Forest model.

## References

[1] Alsamadani, R., Hallowell, M., and Javernick-Will, A. N. (2013). Measuring and modelling safety communication in small work crews in the us using social network analysis. *Construction Management and Economics*, 31(6):568–579. 17

[2] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155. 45

[3] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022. 49

[4] Borgwardt, K. M., Kriegel, H.-P., Vishwanathan, S., and Schraudolph, N. N. (2007). Graph kernels for disease outcome prediction from protein-protein interaction networks. In *Pacific symposium on biocomputing*, volume 12, pages 4–15. 17

[5] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM. 25, 28

[6] Bottou, L. and Lin, C.-J. (2007). Support vector machine solvers. *Large scale kernel machines*, 3(1):301–320. 28

[7] Breiman, L. (1996a). Bagging predictors. *Machine learning*, 24(2):123–140. 26

[8] Breiman, L. (1996b). Out-of-bag estimation. *Technical report*. 26

[9] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32. 25, 27, 29

[10] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. CRC Press. 25

[11] Bullmore, E. and Sporns, O. (2009). Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198. 17

[12] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM. 25, 28

[13] Chinowsky, P. S., Diekmann, J., and O'Brien, J. (2009). Project organizations as social networks. *Journal of Construction Engineering and Management*, 136(4):452–458. 17

[14] Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6):066111. 21

[15] Dai, A. M., Olah, C., and Le, Q. V. (2015). Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*. 43

[16] Debortoli, S., Müller, O., Junglas, I. A., and vom Brocke, J. (2016). Text mining for information systems researchers: an annotated topic modeling tutorial. *CAIS*, 39:7. 49

[17] Desvignes, M. (2014). *Requisite empirical risk data for integration of safety with advanced technologies and intelligent systems*. PhD thesis, University of Colorado at Boulder. 11

[18] Fortunato, S. (2010). Community detection in graphs. *Physics reports*, 486(3):75–174. 21

[19] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139. 27

[20] Girvan, M. and Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826. 20

[21] Hallowell, M. R. and Gambatese, J. A. (2009). Activity-based safety risk quantification for concrete formwork construction. *Journal of Construction Engineering and Management*, 135(10):990–998. 37

[22] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162. 46

[23] Kitsak, M., Gallos, L. K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H. E., and Makse, H. A. (2010). Identification of influential spreaders in complex networks. *Nature Physics*, 6(11):888–893. 17, 19

[24] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. 43

[25] Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into texts. Association for Computational Linguistics. 15, 17

[26] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. 45

[27] Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751. 46, 47

[28] Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113. 21

[29] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. 15, 17, 19

[30] Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en. 47

[31] Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*. 47

[32] Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523. 45

[33] Seidman, S. B. (1983). Network structure and minimum degree. *Social networks*, 5(3):269–287. 19

[34] Sievert, C. and Shirley, K. (2014). Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70. 48, 49, 50

[35] Silverman, B. W. (1986). *Density estimation for statistics and data analysis*, volume 26. CRC press. 40

[36] Tixier, A., Malliaros, F., and Vazirgiannis, M. (2016a). A graph degeneracy-based approach to keyword extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1860–1870. 15

[37] Tixier, A., Skianis, K., and Vazirgiannis, M. (2016b). Gowvis: a web application for graph-of-words-based text visualization and summarization. In *Proceedings of ACL-2016 System Demonstrations*, pages 151–156. 15

[38] Tixier, A. J.-P., Hallowell, M. R., and Rajagopalan, B. (2017a). Construction safety risk modeling and simulation. *Risk analysis*, 37(10):1917–1935. 35, 39, 41

[39] Tixier, A. J.-P., Hallowell, M. R., Rajagopalan, B., and Bowman, D. (2016c). Application of machine learning to construction injury prediction. *Automation in construction*, 69:102–114. 22

[40] Tixier, A. J.-P., Hallowell, M. R., Rajagopalan, B., and Bowman, D. (2016d). Automated content analysis for construction safety: A natural language processing system to extract precursors and outcomes from unstructured injury reports. *Automation in Construction*, 62:45–56. 11

[41] Tixier, A. J.-P., Hallowell, M. R., Rajagopalan, B., and Bowman, D. (2017b). Construction safety clash detection: identifying safety incompatibilities among fundamental attributes using data mining. *Automation in Construction*, 74:39–54. 17

[42] Tixier, A. J.-P., Vazirgiannis, M., and Hallowell, M. R. (2016e). Word embeddings for the construction domain. *arXiv preprint arXiv:1610.09333*. 45

[43] Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999. 28

[44] Villanova, M. P. (2014). *Attribute-based risk model for assessing risk to industrial construction tasks*. PhD thesis, University of Colorado at Boulder. 11