

SmarTxT: A Natural Language Processing Approach for Efficient Vehicle Defect Investigation

Transportation Research Record
1–14© National Academy of Sciences:
Transportation Research Board 2022
Article reuse guidelines:sagepub.com/journals-permissions

DOI: 10.1177/03611981221125744

journals.sagepub.com/home/trrJack Francis¹, Kim Cates¹ , and Gianluigi Caldiera² 

Abstract

The investigation of vehicle defects, which is generally led by the National Highway Traffic Safety Administration (NHTSA) in the U.S., is critical to the continued trust of the general public in the safety of vehicles. NHTSA routinely receives millions of reports of potential defects, complaints, recalls, and manufacturer communications, which may provide evidence of a new vehicle defect. However, the large quantity and text-based communication make efficiently identifying defect trends difficult for analysts. To accelerate the investigation of defect reports, we introduce a natural language processing (NLP) application that identifies key topics and similar defect reports to assist analysts and investigators. Further, our application is built to provide users with a web interface for interacting with the NLP models. The integration of NLP with current NHTSA datasets provides a method for quickly identifying defect trends in large text-based datasets. To demonstrate the effectiveness of our method, we apply our approach to two publicly available NHTSA datasets, namely the Technical Service Bulletins and Recalls Dataset.

Keywords

data and data science, machine learning (artificial intelligence), unsupervised learning, safety

Like in many other domains, the size and frequency of data has increased exponentially in recent years in transportation-related fields. This drastic increase in data availability has led to artificial intelligence and machine learning (AI/ML) techniques being frequently used for a wide range of transportation applications. This effect has led to the number of AI/ML papers to the TRB Annual Meeting to skyrocket from 5 in 2015 to 162 in 2020 (1). Most of the AI/ML work is concentrated on navigation applications, advanced driver assistance, and smart traffic systems (2–7). A common technology used for these applications, deep learning, is often used to detect defects in the manufacture of vehicles and vehicle components (7, 8). In these applications, researchers use convolutional neural networks, recurrent neural networks, and depth residual neural networks, to name a few, to extract features from images and map them to categories of defects. The investigation and resolution of defects in vehicles are critical to ensure public safety and confidence.

Two key challenges arise when an analyst, from the National Highway Traffic Safety Administration

(NHTSA), U.S., or a safety advocacy organization, like the Insurance Institute for Highway Safety (IIHS), U.S., wants to investigate and address vehicle defects. First, defect data is generally recorded in text with associated metadata, such as the make, model, and year of the vehicle. The text data makes performing AI/ML approaches difficult, because most classical machine learning algorithms do not work with text data directly (9). Because of this, an analyst must read each defect record line by line to understand the defect and identify patterns among defects. For example, analysts in NHTSA's Office of Defect Investigation (ODI) identify defect issues through manual evaluation of consumer complaints, equipment defects, and injury reports. The results of these manual preliminary investigations are

¹KBR, Lexington Park, MD²KBR, Fulton, MD

Corresponding Author:

Gianluigi Caldiera, gianluigi.caldiera@us.kbr.com

summarized and stored in the Artemis database (10, 11). Although historical investigations are stored in Artemis, there is currently no method for identifying the similarity of a current investigation to any past historical investigations. This is a key challenge for ODI analysts, who often examine defects that occur on components used in multiple makes and models. Second, the data size is exceedingly large. NHTSA publicly posts a variety of datasets relating to defects, including consumer complaints, recalls, and manufacturer communications. These datasets have millions of entries, which makes the analyst's job of manually reading defect reports line by line to identify patterns essentially infeasible. Natural language processing (NLP) techniques, that will be introduced in this paper, help the analyst in dealing with this problem. NLP is a subfield of machine learning that aims to train computers to understand spoken words and text like humans and is further described in the next section.

In this paper, we propose an application, namely SmarTxT, that not only efficiently processes the large text datasets of defects using NLP but also provides a user-friendly interface to assist analysts in the investigation and resolution of defects. To enhance the analyst's investigation, we implement two NLP approaches, namely topic modeling and document similarity. Topic modeling identifies latent topics for a given dataset, which provides an analyst with the major trends for a data selection. Document similarity provides analysts with similar defect records. To implement document similarity, we use Bidirectional Encoder Representations from Transformers (BERT) to embed defect records and then use cosine similarity to identify similarity among defect records. To demonstrate the effectiveness of our NLP approaches, we apply these methods to the NHTSA Technical Service Bulletins (TSB) dataset as a case study. These methods are unsupervised, so the model results are validated by a human expert. Finally, we built the application using Flask and Python to provide an accessible approach for analysts to interact with our modeling approaches.

Our proposed application built on NLP provides major benefits to analysts in defect investigations:

1. Our NLP approach provides a fast, accurate method for investigating millions of defect records. These approaches significantly reduce the number of hours analysts must spend on reading individual defect records. Further, the immediate aggregation of similar documents may lead to the identification of new patterns and trends that would have previously been missed.
2. SmarTxT provides an easy and accessible interface for analysts to understand the results from

our NLP models. SmarTxT is built so that NLP model understanding is not required for analysts, which greatly increases the number of analysts that can benefit from this tool. It is a tool designed for what Gartner Group has described as the "citizen data scientist."

The remainder of this paper is structured as follows. The next section provides a literature review. The section after that provides a description of the data and our preprocessing approach to convert the data into a format appropriate for NLP. The NLP approaches, namely topic modeling and document similarity, are described in the following section. The antepenultimate section provides an overview of the SmarTxT application. The penultimate section details our validation approach for assessing model quality in determining similar documents. The final section concludes our study with conclusions and future work.

Literature Review

NLP is a subfield of machine learning that aims to understand the contents of documents, where a document in the NLP setting is any collection of text, which could include words, sentences, paragraphs, and/or papers (12, 13). The key challenge in NLP is associating documents with a lower-level embedding that captures the semantic similarity and context between two documents. There have been multiple methods proposed for this problem, including Word2Vec, Doc2Vec, term frequency-inverse document frequency (TF-IDF), topic modeling, and, more recently, transformer models (13–17). In each of these methods, the objective is to take in a corpus of documents and assign a lower-dimension numerical embedding to capture the relationship between documents. Because documents are actually collections of documents, there are varying levels of abstraction that each of these models attempt to embed. For example, Word2Vec operates on individual words, Doc2Vec and TF-IDF operate on paragraphs, and transformers can be used for either use case depending on the model selected. Some of the widely used transformer models for NLP are BERT, ELMo, and GPT-3 (18–20). These transformer models are trained on large corpora and distributed with the learned weights from this training procedure. For example, BERT was trained using the BooksCorpus (800M words) and English Wikipedia (2,500M words) (18). A major benefit of these large pre-trained transformer models is the relative ease of applying transfer learning for new domains (21). By freezing all weights except the top layer and then retraining the model on domain-specific corpora the models can be quickly adapted for new

domains. Examples of domain-specific retraining include BioBERT and LegalBERT (22, 23).

Because of the large success of NLP over recent years, these methods have been applied in a wide range of applications. Of particular interest for SmarTxT are applications that leverage topic modeling, document similarity, and transformer models. Topic modeling is an NLP technique for identifying latent topics from a set of documents and is described in further detail in the section dedicated to Topic Modeling. Topic modeling has been used in a wide variety of applications including engineering, bioinformatics, and automated vehicles (24–27). Document similarity is a natural extension of embeddings, where we identify the most-similar document to a selected reference document. A common use case for document similarity is to return contextually relevant information for search queries (28). Additional applications of document similarity include providing semantic search capability for aviation maintenance records and searching clinical notes in health centers (29, 30).

Topic modeling and document similarity can enhance the current data review and investigation of defects by NHTSA. First, the identification of latent topics in maintenance records provides an overview of trending defects. The identification of topics allows NHTSA ODI analysts to evaluate trending defects and then compare against previous defect investigations to identify long-term defect trends. Currently, this is not feasible because of the large amount of data and the dynamic standard operating procedures (SOPs) (11). Further, topic modeling allows for cross-make and cross-model analyses to be completed. By combining all documents from various vehicle makes and models, analysts can identify defect trends among parts shared between multiple automakers. Second, document similarity provides a semantic, contextual search for analysts to identify similar defects. Defect records are not written in a consistent format, which leads to further challenges in comparing cross-make and cross-model defect reports. With document similarity, NHTSA ODI analysts would have the ability to input full defect descriptions or key phrases to query semantically similar records within the Artemis database. Our solution, SmarTxT, proposes to leverage NLP techniques for vehicle defect investigation, which to the author's knowledge has not previously been completed.

Dataset Description

Vehicle Defects

To demonstrate the effectiveness of the proposed method, we use publicly available data from NHTSA, namely the TSB and Recalls Datasets, which are available at this web site: <https://www-odi.nhtsa.dot.gov/downloads/>. These datasets are representative of defect

datasets commonly used by NHTSA analysts and share the following characteristics:

- An identifier (Bulletin Number or Record ID): A code associated with the defect report, which is not necessarily unique. In some cases, technicians will use the same code for similar, but non-identical, defects. For SmarTxT, we use the Bulletin Number as the identifier for technical reports in the TSB Dataset. For the Recalls Dataset, the Record ID is used as the identifier.
- A set of metadata (NHTSA ID, Bulletin Date, Component Name, Make, Model, Year, Date Added): A set of values, generally categorical, that provide an outline of the problem that was identified. The NHTSA ID is an ID provided by NHTSA for each new technical report. The Bulletin Date denotes when the original bulletin was drafted. The Component Name provides the code and description of the failing component. The Make, Model, and Year are used to identify a specific car (e.g., Ford Focus 2012 or Mazda CX-5 2021). The Date Added is the date when the bulletin was added into the NHTSA database.
- A textual description (Technical Summary): A description of the problem or other information associated with the defect report (e.g., in some cases there is also a description of the proposed fix).

An example defect report from the TSB dataset is shown below in Figure 1.

As shown above, in Figure 1, the Technical Summary provides the textual description for the TSB dataset. The TSB dataset, downloaded directly from NHTSA, contains almost 3M defect reports at the time of writing, with new reports being added daily. However, we find many duplicated reports, namely, we find multiple reports where the BULNO (i.e., the identifier) and the Summary match exactly. We remove duplicated records but add a column to note the number of duplicates to provide analysts with an idea of the frequency of each defect. After removing duplicated records, the dataset is reduced to 160k defect reports. The Recalls Dataset contains 190k recall reports with no duplicates, so no reports are removed from this dataset. Once the duplicated records are removed, we then proceed to preprocess the text fields to be appropriate for applying NLP.

The technical defect reports that we analyze with NLP techniques are often written as a memo for the technicians rather than explicit documentation of the issue. This means that the language used is technical and the specific issue may not be clearly stated. Our selected class of neural networks, called transformers, can link words

Bulletin Number	NHTSA ID	Bulletin Date	Component Name	Make	Model	Year	Date Added	Technical Summary
SSM 46303	10095164	20170113	060000 ENGINE (PWS)	FORD	FOCUS	2014	20170315	THE 2012-2017 FOCUS WORKSHOP MANUAL PEROCEDURE FOR OIL PAN AND REAR MAIN OIL SEAL INSTALLATION HAS BEEN REVISED. THE SEALANT APPLICATION LOCATION HAS BEEN REVISED.
SSM 46312	10095165	20170118	110000 ELECTRICAL SYSTEM	FORD	ESCAPE	2017	20170215	SOME 2017 ESCAPE VEHICLES EQUIPPED WITH SYNC CONNECT AND ORDERED WITH A FORD ORIGINAL ACCESSORY (FOA) REMOTE STARTSYSTEM MAY EXHIBIT A CONCERT THAT THE ENGINE CANNOT BE REMOTE STARTED USING THE FORDPASS APPLICATION.
PI1276A	10095166	20140910	102000 POWER TRAIN: MANUAL TRANSMISSION	CHEVROLET	CORVETTE	2014	20170315	THIS PRELIMINARY BULLETIN PROVIDES A PROCEDURE TO ADJUST THE SHIFTER TO CORRECT THE CUSTOMERS CONCERN OF THE TRANSMISSION HARD TO SHIFT OR POPS OUT OF GEAR
...

Figure 1. Example Technical Service Bulletins (TSB) defect reports. We use the Bulletin Number as the record identifier and apply our Natural Language Processing (NLP) workflow to the Technical Summary.

with the context and link sentences with other sentences, which helps to address this kind of language problem.

Text Preprocessing for NLP

NLP is generally applicable for a wide range of text-based datasets; however, it is generally recommended to perform a set of preprocessing steps to improve the inference driven from NLP models (31). Our text preprocessing procedure is completely automated and follows the six steps outlined below:

1. Remove punctuation
2. Remove numbers
3. Tokenize the text
4. Remove stopwords
5. Lemmatize the text
6. Remove words less than three characters

For many NLP tasks, the punctuation and numbers contained in the text descriptions do not add meaningful information to understanding the text. This is the case for our study, where numbers are generally used for additional information and do not directly impact the defect report. One exception is the year range, which is often mentioned in the summary. However, the year is captured in the metadata of each defect report, so there is no loss of information. Similarly, punctuation, such as full-points, commas, and hyphens, do not provide meaningful textual information for our study. Next, we tokenize the text and remove any stopwords. Stopwords (e.g., a, an, the) are context-independent words in English that do not impact the context of the text. We use a stopwords

list provided by Natural Language Toolkit (NLTK), a commonly used package for NLP (32). Next, we lemmatize the text using the WordNet Lemmatizer from NLTK (32). The WordNet Lemmatizer replaces words with a base form of the word only if the base form is a valid dictionary word. For example, the words “removes” and “removed” would be lemmatized to “remove.” Similarly, the words “saving” and “saved” would be lemmatized to “save.” Lemmatization helps NLP models to learn the importance of individual words more quickly. Finally, we remove any words less than three characters, which are generally acronyms.

NLP Approaches: Topic Modeling and Document Similarity

Topic Modeling

The objective of topic modeling is to identify latent semantic patterns in unstructured text that uncover previously unknown groupings in the text that might correspond to common vehicle issues. In topic modeling, documents of unstructured text are clustered semantically based on words that appear in documents. The key assumption is that documents containing similar words belong to the same topic and describe similar problems. These topics are not known ahead of time and thus require unsupervised, probabilistic approaches, such as Latent Dirichlet Allocation (LDA) or Probabilistic Latent Sentiment Analysis (PLSA), where documents are modeled as a random mixture of topics and topics are characterized as a distribution of words. In these models, a “bag-of-words” approach is used, where word

frequency is retained but the word order is lost. Because of this approach, word co-occurrence is strongly influential in determining the latent topics.

The approach followed in this paper uses the power of LDA to cluster similar defects (or, better, similar defect descriptions as found in the Technical Bulletins) by looking at term recurrence. The model will be described with some accuracy in the next section but the important point to make, when applying LDA to vehicle defects, is that many recurring terms are part of the technical lexicon, and their frequency does not indicate their importance. LDA focuses on the probability that certain terms recur together or close to each other simulating, in this way, the perception of someone who is reading the document.

LDA Model. To identify topics in the NHTSA datasets, we use LDA, a generative probabilistic model, which was first introduced by Blei et al., where they provide the following definitions (16):

- A word (w) is the basic unit of data, which is an item from the vocabulary.
- A document (d) is a sequence of N words denoted by $d = (w_1, w_2, \dots, w_n)$, where w_n is the n th word in the sequence. In our case, a document is a pre-processed summary of a technical bulletin or a recall description.
- A corpus (D) is a collection of M documents denoted by $D = (d_1, d_2, \dots, d_n)$, where d_n is the n th document in the corpus. In our case, a corpus is a collection of technical bulletins or recalls.

Documents are represented as a random mixture over latent topics (Z), where each topic (z_i) is characterized by a distribution over words. LDA assumes a generative process for creating documents as presented in Griffiths and Steyvers (33):

1. Choose $\theta_i \tilde{Dir}(\alpha)$
2. Choose $\phi_i \tilde{Dir}(\beta)$
3. Choose topic $z_i \tilde{Multinomial}(\theta_i)$
4. Choose word $w_i \tilde{Multinomial}(\phi_{z_i})$

Here, α and β are hyperparameters for the document-topic (θ) and topic-word (ϕ) Dirichlet distributions, respectively. While these hyperparameters are typically symmetric, they are not required to be. LDA, while unsupervised, can be optimized through hyperparameter tuning and model selection criteria like Akaike or Bayesian information criteria, by comparing the log-likelihood of various models. This allows for an informed selection of the number of topics, alpha, and beta.

Document Similarity

Introduction to Document Embedding and Cosine Similarity. Document similarity is an NLP method that quantifies pairwise semantic measures of all documents for a given text field. Semantically similar documents can be queried for a reference document ordering the documents in order of decreasing similarity score. Identifying semantically similar documents is particularly useful when locating other records related to a central problem summarized in a previous document (34). In the TSB dataset, document similarity is utilized to link similar detect defects in vehicles to one another.

Semantic similarity is quantified by applying cosine distance to embedded document vectors. Embedded vectors represent the optimized conditional probability of a given word occurring in relation to all other words in the corpus (13). Capturing the conditional probability of each word w given all context words c can be represented by the *softmax* function in Equation 1.

$$P(w|c) = \frac{\exp(u_w^T v_c)}{\sum_{w \in \text{Vocab}} \exp(u_w^T v_c)} \quad (1)$$

where

- the numerator captures the similarity between the input word u_w and the context word vector transposed from the weight matrix v_c , and
- the denominator provides a sum of all context words v_c multiplied by input word u_w to normalize similarity for a given context word in relation to all other words.

In place of a simple embedding, we use a document embedding model which adds a paragraph ID as another input vector. The product outputs from both the word embedding layer and the document embedding layer are concatenated together and averaged before applying the *softmax* activation function to produce the document embedded vector (14). After generating these document vectors, we apply a cosine similarity metric to each document vector allowing for each document's semantic value to be quantified. In the following example, we generate vectors from the italicized text and associated similarity score with *A.* being the reference document, both *B.* and *C.* having high similarity scores compared with *A.* and with *D.* having very little similarity:

- A. *Some vehicles may exhibit a knocking noise heard from the front of the vehicles with a slight floor vibration while traveling long distances during high ambient temperatures*
Similarity score = 1 (this means that the reference document is similar to itself)

- B. *Region email: The subject vehicles may have been filled with an improper concentration of coolant and water. Under high load driving conditions this could cause the high engine coolant temperature*
Similarity score = 0.85
- C. *Service bulletin: in high ambient temperatures, the steering column screeches or squeals when it moves in or out, potentially caused by premature wear of the steering column gearbox components*
Similarity score = 0.82
- D. *Subject regarding cellular telephone/CD changer/optional voice recognition system installation*
Similarity score = 0.20

Bidirectional Encoder Representation From Transformers (BERT) Document Embeddings. BERT, a pre-trained document embedding model developed by Google in 2018, was used to ensure validity when quantifying semantic values in technical summaries to capture document similarity (18). BERT performs superior to traditional embedding approaches because of BERT's ability to bidirectionally learn contextual relationships among phrases. This means that words that are positioned before or after a target word are considered for contextual relevance. This is unique compared with previous approaches, which only use one-directional embedding models to predict the next word in a sequence. Furthermore, LSTM models could only predict words immediately before and immediately after a given word (35). In contrast, BERT predicts the contexts of word phrases as they relate to each word for a given text string. It does this through a masked language modeling (MLM) approach that involves an encoder-decoder structure in which each encoder layer is fed into all decoder layers allowing for all input words to be processed simultaneously (18). As a result, different contexts of the same word are captured, which was not possible in prior embedding techniques. For instance, the word "gauge" could be distinguished between the noun form in "tire pressure gauge" and the verb form "gauge".

Even more so, BERT implements a transformer model architecture. Transformer models apply constant steps of "attention" mechanisms that enhance the inference among connections between all words in a sentence. These attention mechanisms work by capturing both the hierarchical structure of language (parsing) along with context (composition) (18, 36). The BERT model will quickly identify, or selectively attend to, words in a sentence that are salient to the context of that sentence rather than identifying the word's overall context. More specifically, the attention mechanisms allow for local attention for a given word as it focuses on words of relative importance (36). Figure 2 provides a visual representation of attention mechanisms comparing the words within sentence A from the previous example. The word

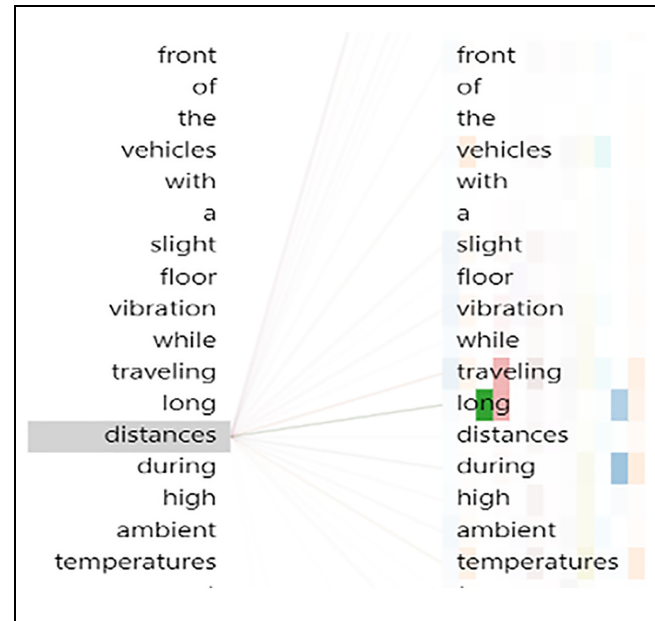


Figure 2. Visualizing the attention head mechanism for the word "distances" in an example technical summary.

Note: For each of the 12 attention heads (the 12 colored boxes on the right), the relationship between the word "distances" and other words in the sentence is denoted by the intensity of the color.

on the left-hand side, "distances," is semantically linked to the words on the right-hand side. The highlighted words "vehicles," "traveling," "long," "during," and "temperatures" are emphasized because of their contextual relevance to the current sentence. The intensity of the color represents the strength of the link for the given attention head.

The outcome of the attention compatibility measures reflects aspects of syntax, or function to parse language. Concatenation of attention mechanisms into a multi-head within each layer captures context dictated by the individual word's local attention within each sentence. Coreference links, or expressions that are semantically equivalent to phrases in other documents independent of syntax structure, are identified through the stacking of multiple attention layers in the BERT architecture (37). The pre-trained BERT base model was used for developing the initial SmarTxT prototype. The base model consists of 12 layers (transformer blocks), 12 attention heads, and 110M parameters. The outcome of the iterative encoding-decoding attention mechanisms of the BERT model generates a dense vector representation of documents in which coreference links are quantified. BERT's sophisticated embedding technique was used to generate document embeddings for the technical summaries to calculate similarity.

Attention mechanisms play an important role in the detection of defects because, by modeling reader expectations when reviewing a text, they provide a context to

the recurrence of certain words, groups of words, or sentences. Obviously, this context is strongly dependent on the domain in which the model is applied. In our case, this model is vehicle defect detection, therefore the model should be trained not only to understand the English language but also the English language as it is used in technical bulletins. This goal is achieved by fine-tuning the model over the Technical Bulletins dataset.

Fine-Tuning BERT. We fine-tuned the BERT base model with the TSB dataset to establish more data-specific document embeddings for capturing document similarity. Fine-tuning, also called transfer learning, is a common procedure in deep learning for large neural networks (38–40). The motivating idea is to take a well-trained model in a similar domain and then retrain only the top layer(s) for a new domain. The model retains the general understanding of the domain, while also improving for a new domain through fine-tuning. The fine-tuning of document embeddings requires a supervised model architecture for the learned embeddings to be optimized. Therefore, pairwise documents are fed into the pre-trained BERT model with associated similarity scores for each pair. Reimers and Gurevych demonstrate a fine-tuned sentence embedding approach using predefined similarity labels for each document pair (41). In this use case, similarity scores were captured by calculating cosine similarity for each document pair after embeddings were generated by the Doc2Vec model. Following this approach, a dictionary consisting of each document pair and associated similarity score was the input into the BERT model. The fine-tuning occurs through feeding each tokenized sentence into the BERT network followed by a mean-pooling layer. A pooling layer is a method for down-sampling and is the default for Siamese BERT-Networks, like the ones provided by the Python package sentence-transformers (41). Siamese BERT-Networks are one type of fine-tuning scheme for BERT and the one we use. The averaged vector (i.e., the output of the mean-pooling layer) represents the pre-trained BERT vectors tuned on incoming sentence pairs and their respective similarity scores. Cosine similarity is calculated for the averaged vectors then compared against input similarity scores. Model weights and document embeddings are updated based on loss between two similarity scores. Tuning the pre-training model on the supervised output has been determined to make moderate changes to the attention mechanisms in the network. More specifically, fine-tuning BERT has shown to be most effective by enhancing domain-specific (in our case, vehicle defects) representations of sentences. In contrast, the pre-trained model is well adapted to identify global syntactic patterns found in everyday language.

SmarTxT Application

General Approach

The intended user of our application is a vehicle defect investigator at NHTSA or a similar role in another organization. A key challenge for current NHTSA vehicle defect investigators is the inability to access sensor data from vehicles directly. Instead, to perform defect investigations they must rely on technical bulletins and identify patterns and trends in the text data. Our tool, SmarTxT, provides NHTSA vehicle defect investigators the ability to identify key topics in the technical summaries and identify similarities among defects to expedite the in-depth vehicle investigations.

Our application provides five web pages to assist the user in investigating defects in the NHTSA data. A summary of each step is given below:

1. **Login:** A user logs in to the application and the application assigns the correct privileges (e.g., datasets that are visible to the specific user).
2. **Overview of the Dataset:** The user selects a dataset to analyze from the currently available datasets in the tool. A user can view the selected dataset contents and retrieve summary statistics.
3. **Train Models Page:** The user filters the dataset by make, model, and year (if needed) and trains the topic model on the filtered dataset or chooses to use the pre-trained NLP models for topic modeling and document similarity. The user can also see summary statistics for the filtered dataset.
4. **Topic Modeling:** The user views the results of topic modeling, where each topic, along with the reports that best fit in each topic, are displayed, as shown in Figure 3, *a* and *b*.
5. **Document Similarity:** The user provides the identifier of a specific document (the Bulletin ID in the case of Technical Bulletins) and the tool returns a list of documents (Figure 4) that have similar content, ranked in order of similarity.

SmarTxT Implementation

Our application was built using Python, Flask, and Bootstrap. Flask was selected as the web framework of choice because of its ease of use with Python and customization features. The web pages were developed using Bootstrap to match current industry standards and allow for the re-use of code throughout various parts of the application. All other code, including the NLP models, was written in Python. All timing data was calculated on a Dell Precision 5540 with an i7-9850H CPU and 32 GB

(a)	Topic	Keywords	View Similar Documents
1		provided data summary future stock newsletter overview solutions pending campaigns	Topic 1
2		oil fuel model engine leak pump coolant valve new models	Topic 2
3		engine vehicles exhibit model equipped start condition year idle cold	Topic 3
4		vehicles campaign service certain warranty dealer parts vehicle recall program	Topic 4
5		bulletin service information technical provides procedure instructions update repair	Topic 5
6		control module diagnostic dtc code trouble software lamp engine indicator	Topic 6
...

(b)	Bulletin Number	Make	Model	Year	Data Added	Technical Summary	Topic Likelihood
	14911	FCA	ALL MODELS	YEAR	20171013	HEADLAMP ASSY-TRT R/L. YOU HAVE BEEN IDENTIFIED AS EITHER HAVING STOCK OR HAVE STOCK IN TRANSIT TO YOUR DEALERSHIP. RETURN PARTS 68223467AB & 68214567AB	0.880607
	TT-ALI	SUBARU	SUBARU	9999	20180611	TECH TIPS ARTICLE LOCATOR INDEX	0.849986
...

Figure 3. (a) SmarTxT topic modeling results page (clicking on “Topic 1” generates Figure 3b) and (b) SmarTxT topic modeling results deep dive page.

Note: To ensure the highest quality representation of the table for the paper, we have removed the NHTSA ID, Bulletin Date, and Component Name columns from the table that normally appear when viewing this table in SmarTxT.

Reference Document

	Bulletin Number	NHTSA ID	Bulletin Date	Component Name	Make	Model	Year	Date Added	Technical Summary	Best_Topic
143039	SSM 48551	10171954	20200214	110000 ELECTRICAL SYSTEM	FORD	TRANSIT CONNECT	2020	20200303	SSM 48551 - 2020 Various Vehicles Equipped With SYNC 3 Various SYNC Performance Related Concerns Some 2020 Ford and Lincoln vehicles equipped with SYNC 3 may exhibit various performance related concerns such as: intermittent frozen or blank	6

Similar Documents

	Bulletin Number	NHTSA ID	Bulletin Date	Component Name	Make	Model	Year	Date Added	Technical Summary	Best_Topic
148227	SSM 48833	10176119	20200522	110000 ELECTRICAL SYSTEM	FORD	ESCAPE	2020	20200604	Some 2020 Ford and Lincoln vehicles may exhibit a symptom of no ability to schedule a remote start event using the FordPass or LincolnWay app. If only the scheduling portion of the remote start system is not functioning, please ensure that	6
31877	44872	10057837	20150109	113000 ELECTRICAL SYSTEM:STARTER ASSEMBLY	FORD	MUSTANG	2015	20150701	FORD: VARIOUS 2015 FORD MUSTANG VEHICLES EQUIPPED WITH THE REMOTE START FEATURE MAY EXPERIENCE DTC P025A STORED IN PCM MEMORY WITH CRANK NO-START OR STALL PROBLEMS. *TA	5
143256	SSM 48558	10172019	20200218	110000 ELECTRICAL SYSTEM	FORD	F-250 SD	2020	20200303	SSM 48558 - 2017-2020 Various Vehicles Equipped With Telematics Control Unit (TCU) - Various FordPass/LincolnWay Mobile App Concerns Some 2017-2020 Ford and Lincoln vehicles equipped with a TCU may exhibit inoperative remote features via F	6
122013	SSM 47444	10144930	20180802	110000 ELECTRICAL SYSTEM	FORD	EXPLORER	2016	20180924	Some 2016-2018 Ford and Lincoln vehicles equipped with SYNC 3 not at the latest software level may exhibit Various Performance Related concerns such as 6 Display Operation, Navigation, Voice Recognition, Phone Pairing, AppleLink, Travel Link, S	6
109733	SSM 46860	10125078	20171006	353500 EQUIPMENT:ELECTRICAL:NAVIGATIONAL SYSTEM(GLOBAL POSITIONING SYSTEM)	FORD	ESCAPE	2016	20171214	Some 2016-2018 Ford or Lincoln vehicles equipped with SYNC 3 may exhibit a blank screen or other various SYNC symptoms only when the vehicle is in transport mode. This is due to the power saving features enabled while in transport mode and	6
109734	SSM 46861	10125079	20171006	353000 EQUIPMENT:ELECTRICAL	FORD	ESCAPE	2016	20171214	Some 2016-2017 Ford and Lincoln vehicles with SYNC 3 may intermittently display an overhead image of a vehicle in the center of the SYNC touchscreen with no obstacle detection grids or warning chimes. SYNC may appear unresponsive when the o	7

Figure 4. SmarTxT document similarity tool output.

of RAM. Key aspects of the implementation are provided below in further detail:

- **Multiple dataset compatibility:** To increase the effectiveness of the tool, we built the functionality to analyze distinct datasets. To achieve this,

Python code was written functionally, such that specific dataset parameters were passed throughout the app instead of hard-coded. For each dataset, a JSON file is created with the appropriate metadata, which is then processed once a user selects a dataset to analyze.

- Topic Modeling:** Topic modeling (i.e., LDA) was implemented using scikit-learn in Python to provide results to users in real-time (42). The user selects the number of topics, but all other hyperparameters (i.e., alpha, beta, and the learning decay) are set to the default values provided by scikit-learn. From the topic modeling implementation in scikit-learn, we provide the user information on the topics, as well as the reports that best fit into each topic, to identify key defect reports more quickly in the dataset. For small datasets, topic modeling is computationally fast (i.e., less than 10 s). Thus, when users select any filters (i.e., make, model, year) in SmarTxT, a topic model is trained at runtime based on the filters selected. However, with the large size of the NHTSA datasets (e.g., 180k + records) topic modeling was too slow to apply at runtime, when analyzing the entire dataset. To avoid the user experiencing this slowdown, we implemented an offline pre-training procedure where the topic model for the entire dataset is pre-computed. For this model, we perform RandomSearchCV in scikit-learn to identify the optimal topic model based on the log-likelihood. We test the following hyperparameters and ranges: number of topics (3, 10), alpha (0, 1), beta (0, 0.5), and learning decay (0.5, 0.99). Our procedure generates 90 candidate topic models and returns the best model based on log-likelihood. Then the topic model is loaded into SmarTxT if a user does not filter the dataset. This option is important for analysts who want to investigate defects that may affect multiple makes or models.
- Document Similarity:** Document Similarity was implemented using the Sentence Transformers package in Python, which provides sentence-level embeddings using BERT (41). A user will provide a reference document and all similar documents are returned to the user. We only use the text field (e.g., technical summary for TSB) to identify similar documents. As mentioned previously, cosine similarity was the metric of choice for identifying similarity between documents. We return any document that has a cosine similarity score greater than a threshold value. To identify the similarity threshold to use, we tested multiple possible thresholds (e.g., 0.7, 0.8, and 0.9) for each dataset. Because of the lack of a validation set, we are unable to optimize the similarity threshold. Instead, we select the similarity threshold based on the average number of similar documents returned on average for each dataset. For the TSB dataset, the selected similarity threshold was 0.8, while the selected similarity threshold for the

Recalls Dataset was 0.9. The difference in similarity thresholds can be explained by the lack of exact duplicates in the Recalls Dataset. Instead, there are many similar, but non-identical, recalls. This is dissimilar to the TSB dataset, where many documents are exact duplicates, which we remove in our pre-processing procedure. Similar to topic modeling, performing document similarity at runtime is too slow for a user because of the large amount of computation required to calculate the BERT embeddings and perform the similarity calculation. To avoid the user experiencing this slowdown, we perform a global similarity calculation offline and save the result into a lookup table, which is described further in the section on improving user experience.

SmarTxT is deployed in the AWS cloud using CloudFront, EC2, and Lambda. All cloud operations complete in less than 15s, which is the timeout applied for our Lambda functions. We use CloudFront to deploy the application with code stored in S3 and EC2. Specifically, we use an AWS Linux t2.large EC2 instance. Routes within the app call a Lambda function that invokes the function on EC2 and then returns the appropriate data to the webpage.

Figure 5 shows a screenshot of the SmarTxT tool presenting the output of a document similarity execution.

Custom Implementation to Improve User Experience

A common tactic for deploying machine learning models in production is to pre-train models offline and then make predictions at runtime. This improves the responsiveness and performance of the application, without sacrificing model accuracy.

Document similarity requires computing the embedding of each Technical Summary and then computing the cosine similarity between every combination of Technical Summaries. This process is quite slow, so we perform these steps offline. In general, to compute a similarity matrix for a vector v with n entries would create a matrix A , where A is nxn . For the TSB dataset, we have 157k entries, which would lead to a matrix requiring well over 100 GB to store in memory. Instead, we compute the similarity matrix in batches, where batch size $b < n$. This leads to a smaller matrix B , where B is bxn . We select b such that B will fit in the available RAM. Once B is constructed, each row i denotes the document in the i th index of the TSB dataset batch ($i \in \{1, 2, \dots, b\}$) and each column j denotes the document in the j th index of the TSB dataset ($j \in \{1, 2, \dots, n\}$). $B_{i,j}$ captures the similarity between document i and document j . For each document i , we first remove any entries with a similarity score

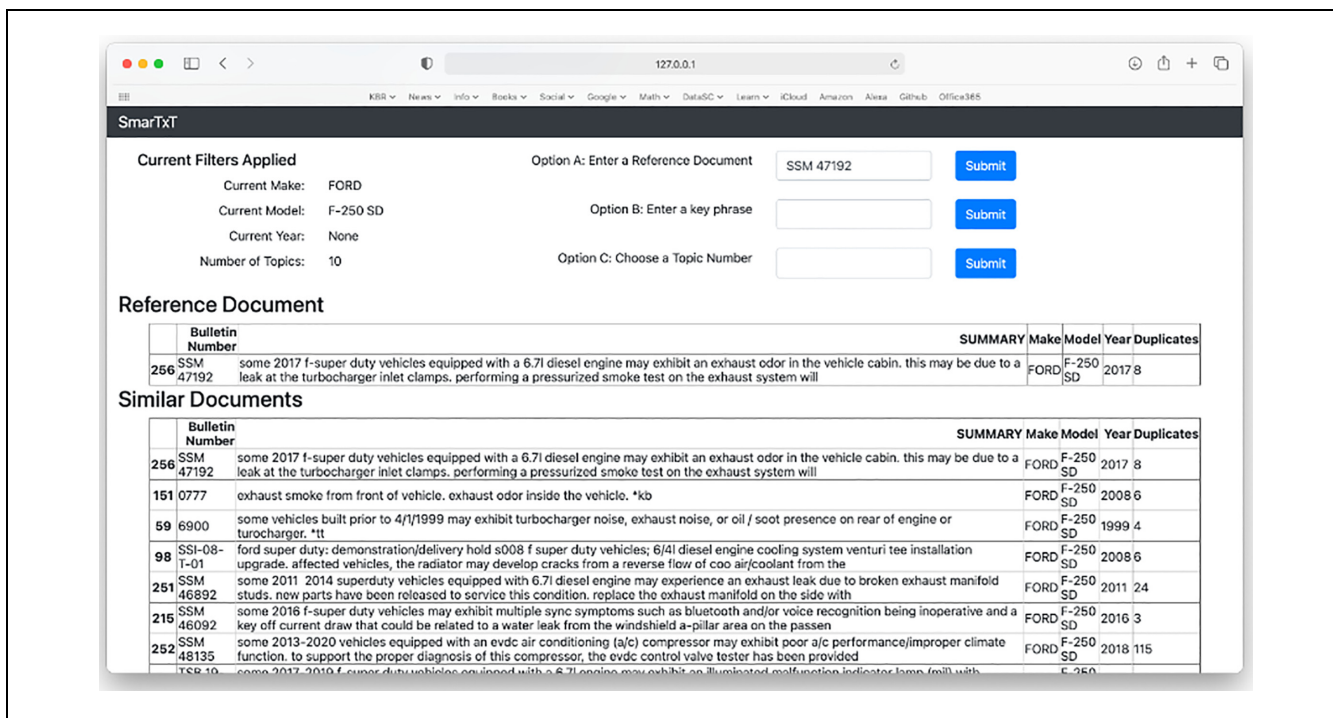


Figure 5. Screenshot of the SmarTtX tool.

less than the threshold. If more than 100 documents have a similarity score above the threshold, we keep the 100 most-similar documents. The 100-document limit is selected as a cutoff to balance, first, the size of the lookup table being generated and, second, to avoid overwhelming NHTSA analysts with too much information. As the tool progresses, we plan to work with NHTSA to identify the optimal number of similar documents to return for each query. This approach provides analysts with a selection of most-similar documents, without leading to an overwhelming amount of data. After applying this procedure for each document and each batch, the indices j of the most-similar documents are saved to a lookup table. When a user enters a reference document, SmarTtX queries the lookup table to provide the user with the most-similar documents. We find that it takes 64.2 min on average to generate the lookup table for the TSB data and 68.76 min to generate the lookup table for the Recalls Dataset. The discrepancy in time can be attributed to the additional records in the Recalls Dataset (roughly 30k more). It is important to note that, if a user applies any of the optional filters (make, model, year) before the document similarity page, these filters are applied after querying the lookup table. Thus, SmarTtX will always display the most-similar documents, according to BERT, for any query, regardless of the filtering options selected by the user. The query is nearly instantaneous and provides the user with a significantly faster and more usable experience. This implementation

provides the user a faster, more responsive experience with our application, which in turn could lead to faster adoption because of ease of use.

Result Validation

As previously mentioned, document similarity is an unsupervised task, which means that we do not have a set of labels to validate the results we have obtained. Thus, we use a human expert to validate our results. While this limits the amount of validation we can perform, because of the time required to manually review each defect report, it provides a repeatable benchmarking approach that accounts for the subjectivity of the outcome. We have randomly selected 10 documents and, for each of these documents, we have identified the top 10 similar documents using our approach. Then we have submitted the 100 resulting documents to a domain expert and asked the expert to review the results and give a subjective rating of their similarity to the reference document. The instructions were as follows:

1. Read the reference document.
2. Read each document deemed similar by our approach and for each document:
 - Rate it 0 if, in the expert opinion, the document does not describe a problem that is similar to the one described in the reference document.

Table 1. Validation Results for 10 Randomly Selected Documents

Reference document	Not similar	Similar
TSB 18-2352	2	8
99411	3	7
12925	1	9
SSM 46528	1	9
SSM 49061	3	7
04L23	2	8
996105	2	8
976011	1	9
TSB 20-2100	0	10
15-0109	2	8
Sum >>>>>>		83

- Rate it 1 if, in the expert opinion, the document describes a problem that is similar to the one described in the reference document.

At the end of this process, the human expert had reviewed 100 documents and obtained the ratings shown in Table 1. From these observations, we see that our approach to document similarity on this specific dataset gives us 83% accuracy. The observations we made are available for review on request. In general, we can say that our document similarity method routinely provided relevant documents that would be beneficial for NHTSA or other analysts.

This approach to the validation of the results relies on human expert examination. This human expert participation is, on one hand, unavoidable, because the approach emulates the way a human expert would review the information and find similarities among Technical Bulletins. On the other hand, it limits the number of bulletins that can be reviewed and compared.

Another approach to results validation that does not use the contribution of a human expert is based on the comparison of topic extraction with similarity calculation. If two documents are judged similar by the tool, they probably belong to the same topic. Since topic extraction uses a probability-based approach and document similarity uses a neural network approach, we are using two different approaches. It is therefore natural to ask ourselves how consistent the results are that we obtain using the two methods. Simplifying the reasoning, we ask ourselves: what is the average similarity of the documents in one topic? We restrict ourselves to one make, let us say Tesla, and we calculate average similarity and standard deviation of the documents for each topic.

Table 2. Similarity of Documents Belonging to the Same Topic (Make: Tesla)

Topic	Number of documents in the topic	Average similarity of documents in the topic	Standard deviation
1	37	0.60	0.08
2	24	0.70	0.04
3	5	0.71	0.07
4	31	0.70	0.07
5	24	0.72	0.05
6	9	0.74	0.04
7	111	0.65	0.06
8	3	0.80	0.05
9	7	0.71	0.02
10	21	0.75	0.04

Table 3. Similarity of Documents Belonging to the Same Topic (All Makes)

Total topics	Number of documents	Range of average similarity of documents in the same topic	Standard deviation range
10	3,192,528	0.60–0.95	0.01–0.12

The results for Tesla are shown in Table 2:

As we can see, the results of this analysis are quite promising. Repeating it for different vehicle makes we get average similarities shown in Table 3.

Conclusions

In conclusion, we have shown that NLP can be used effectively to extract patterns from a dataset documenting vehicle defects. SmarTxT possesses multiple advantages for NHTSA analysts and consumers of the NHTSA datasets:

1. SmarTxT provides an intuitive application, which greatly improves user experience in applying NLP to defect reports. This will allow analysts to quickly investigate new defects and identify trends much more quickly. Further, SmarTxT is housed in AWS. Thus, deploying SmarTxT to a new user is relatively straightforward and can be completely self-managed.
2. SmarTxT has built-in NLP models to assist analysts in identifying trends. First, topic modeling

allows analysts to identify the core issues among all defect reports almost instantly. This is a significant improvement over the current approach, where analysts must read reports one by one to manually identify patterns. Second, document similarity allows analysts to easily identify related defects for a current investigation. Again, this improves over the current method where analysts would have to manually search for similar documents by keyword or record IDs, which do not capture the context of the text.

3. SmarTxT provides practitioners with multiple datasets to analyze and provides a straightforward approach for adding new data to the tool. Users can select a dataset from the home page and immediately apply topic modeling and document similarity for an investigation.

Currently, we are experimenting with more sophisticated approaches to elicit topics by extracting features. In recent years, there has been growing interest in topological data analysis as a tool for extracting features from a textual dataset (43). This is a new development because, normally, topological techniques are not used for NLP. We have experimented with the Mapper algorithm in a different dataset from the one used in this paper and have obtained satisfactory results. Additionally, we are developing automated administrative functionality to allow users to add datasets to the tool directly. A user would provide a dataset, which would then be processed and pre-trained with topic modeling and document similarity. Further research will be needed to automatically identify the optimal similarity threshold for a new dataset. However, we envision this will require a validation set, which we did not have in this study. Automating these steps would allow users to upload any dataset to be used in SmarTxT. Additionally, an interesting research avenue is to determine if adding metadata would improve the quality of the returned similar documents. For example, using the make, model, and year columns along with the text could improve the quality. The key challenge in this approach is identifying the correct weighting scheme for each field and handling highly correlated fields (such as the make and model which are often discussed in the text). Further research is needed to determine the relationship between the structure of the models used and the data analyzed. For example, we used Siamese-BERT for fine-tuning, while other options for finetuning exist in the literature (41). We plan to explore the effect of model structure, additional hyperparameter tuning, and training BERT from scratch using vehicle defect data to develop a high-quality model for NHTSA and other organizations to use in the investigation of vehicle defects.

Further, these research efforts will provide a better understanding of how to extend the use of a similar application to other fields of study, such as LEGAL-BERT and BioBERT that have been tuned for the legal and medical industries, respectively (22, 23).

Acknowledgments

This work would not have been possible without the support of KBR and support from Jim King, Kelly Kim, Rohit Mital, Steve Scott, and Todd May.

Author Contributions

The authors confirm contribution to the paper as follows: study conception and design: G. Caldiera; data collection: J. Francis, K. Cates, G. Caldiera; analysis and interpretation of results: J. Francis, K. Cates, G. Caldiera; draft manuscript preparation: J. Francis, K. Cates, G. Caldiera. All authors reviewed the results and approved the final version of the manuscript.


Declaration of Conflicting Interests


The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was funded by KBR Science and Space.

ORCID iDs

Kim Cates  <https://orcid.org/0000-0002-3096-0200>

Gianluigi Caldiera  <https://orcid.org/0000-0002-4659-3978>

References

1. Transportation Research Board Artificial Intelligence Committee. A Primer on Machine Learning for Transportation. Presented at Annual Meeting of the Transportation Research Board, Washington, D.C., 2020.
2. Wu, C., A. Kreidieh, K. Parvate, E. Vinitzky, and A. M. Bayen. Flow: Architecture and Benchmarking for Reinforcement Learning in Traffic Control. *arXiv Preprint arXiv:1710.05465*, 2017.
3. Booth, J., B. Di Eugenio, I. F. Cruz, and O. Wolfson. Robust Natural Language Processing for Urban Trip Planning. *Applied Artificial Intelligence*, Vol. 29, No. 9, 2015, pp. 859–903.
4. Barua, L., B. Zou, and Y. Zhou. Machine Learning for International Freight Transportation Management: A Comprehensive Review. *Research in Transportation Business & Management*, Vol. 34, 2020, pp. 1004–1053.
5. Sligar, A. P. Machine Learning-Based Radar Perception for Autonomous Vehicles Using Full Physics Simulation. *IEEE Access*, Vol. 8, 2020, pp. 51470–51476.

6. Ata, A., M. A. Khan, S. Abbas, M. S. Khan, and G. Ahmad. Adaptive IoT Empowered Smart Road Traffic Congestion Control System Using Supervised Machine Learning Algorithm. *The Computer Journal*, Vol. 64, No. 11, 2021, pp. 1672–1679.
7. Yang, J., S. Li, Z. Wang, H. Dong, J. Wang, and S. Tang. Using Deep Learning to Detect Defects in Manufacturing: A Comprehensive Survey and Current Challenges. *Materials*, Vol. 13, 2020, p. 5755.
8. Liqun, W., W. Jiansheng, and W. Dingjin. Research on Vehicle Parts Defect Detection Based on Deep Learning. *Journal of Physics: Conference Series*, Vol. 1437, No. 1, 2020, p. 012004.
9. Nadkarni, P. M., L. Ohno-Machado, and W. W. Chapman. Natural Language Processing: An Introduction. *Journal of the American Medical Informatics Association*, Vol. 18, No. 5, 2011, pp. 544–551.
10. NHTSA. *Risk-Based Processes for Safety Defect Analysis and Management of Recalls*. NHTSA, Washington, D.C., 2020.
11. NHTSA. *General Standard Operating Procedures (SOP)*. NHTSA, Washington, D.C., 2017.
12. Russell, S., P. Norvig. (eds.). Chapter 23: Natural Language Processing. In *Artificial Intelligence: A Modern Approach*, 4th ed, Pearson Education, London, UK, 2020, pp. 823–855.
13. Mikolov, T., K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv Preprint arXiv:1301.3781*, 2013.
14. Le, Q., and T. Mikolov. Distributed Representations of Sentences and Documents. *Proc., International Conference on Machine Learning*, Beijing, China, 2014.
15. Ramos, J. Using TF-IDF to Determine Word Relevance in Document Queries. *Proc., 1st Instructional Conference on Machine Learning*, Vol. 242, 2003, pp. 133–142.
16. Blei, D. M., A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, Vol. 3, 2003, p. 993–1022.
17. Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, et al. Huggingface's Transformers: State-of-the-Art Natural Language Processing. *arXiv Preprint arXiv:1910.03771*, 2019.
18. Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv Preprint arXiv:1810.04805*, 2018.
19. Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep Contextualized Word Representations. *arXiv Preprint arXiv:1802.05365*, 2018.
20. Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, et al. Language Models are Few-Shot Learners. *arXiv Preprint arXiv:2005.14165*, 2020.
21. Houlsby, N., A. Giurugu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-Efficient Transfer Learning for NLP. *Proc., International Conference on Machine Learning*, Long Beach, CA, 2019.
22. Lee, J., W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. BioBERT: A Pre-Trained Biomedical Language Representation Model for Biomedical Text Mining. *Bioinformatics*, Vol. 36, No. 4, 2020, pp. 1234–1240.
23. Chalkidis, I., M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos. LEGAL-BERT: The Muppets Straight out of Law School. *arXiv Preprint arXiv:2010.02559*, 2020.
24. Gurcan, F., and N. E. Cagiltay. Big Data Software Engineering: Analysis of Knowledge Domains and Skill Sets Using LDA-Based Topic Modeling. *IEEE Access*, Vol. 7, 2019, pp. 82541–82552.
25. Jelodar, H., Y. Wang, C. Yuan, X. Feng, X. Jiang, Y. Li, and L. Zhao. Latent Dirichlet Allocation (LDA) and Topic Modeling: Models, Applications, a Survey. *Multimedia Tools and Applications*, Vol. 78, No. 11, 2019, pp. 15169–15211.
26. Liu, L., L. Tang, W. Dong, S. Yao, and W. Zhou. An Overview of Topic Modeling and its Current Applications in Bioinformatics. *SpringerPlus*, Vol. 5, No. 1, 2016, pp. 1–22.
27. Alambeigi, H., A. D. McDonald, and S. R. Tankasala. Crash Themes in Automated Vehicles: A Topic Modeling Analysis of the California Department of Motor Vehicles Automated Vehicle Crash Database. *arXiv Preprint arXiv:2001.11087*, 2020.
28. Haveliwala, T. H., A. Gionis, D. Klein, and P. Indyk. Evaluating Strategies for Similarity Search on the Web. *Proc., 11th International Conference on World Wide Web*, Honolulu, HI, 2002.
29. Akhbardeh, F., T. Desell, and M. Zampieri. NLP Tools for Predictive Maintenance Records in MaintNet. *Proc., 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, Suzhou, China, 2020.
30. McEwan, R., G. B. Melton, B. C. Knoll, Y. Wang, G. Hultman, J. L. Dale, T. Meyer, and S. V. Pakhomov. NLP-PIER: A Scalable Natural Language Processing, Indexing, and Searching Architecture for Clinical Notes. *AMIA Summits on Translational Science Proceedings*, Vol. 2016, 2016, p. 150.
31. Kannan, S., V. Gurusamy, S. Vijayarani, J. Ilamathi, M. Nithya, S. Kannan, and V. Gurusamy. Preprocessing Techniques for Text Mining. *International Journal of Computer Science & Communication Networks*, Vol. 5, No. 1, 2014, pp. 7–16.
32. Bird, S., E. Loper, and K. Ewan. *Natural Language Processing With Python*. O'Reilly Media Inc., Sebastopol, CA, 2009.
33. Griffiths, T. L., and M. Steyvers. Finding Scientific Topics. *Proceedings of the National Academy of Sciences*, Vol. 101, 2004, pp. 5228–5235.
34. Oghbaie, M., and M. M. Zanjireh. Pairwise Document Similarity Measure Based on Present Term Set. *Journal of Big Data*, Vol. 5, 2018, pp. 1–23.
35. Maslej-Krešňáková, V., M. Sarnovská, P. Butka, and K. Machová. Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification. *Applied Sciences*, Vol. 10, 2020, p. 8631.

36. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you Need. In *Advances in Neural Information Processing Systems* (I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), 2017.
37. Clark, K., U. Khandelwal, O. Levy, and C. D. Manning. What Does BERT Look At? An Analysis of BERT's Attention. *arXiv Preprint arXiv:1906.04341*, 2019.
38. Peng, Y., S. Yan, and Z. Lu. Transfer Learning in Biomedical Natural Language Processing: An Evaluation of BERT and ELMo on Ten Benchmarking Datasets. *arXiv Preprint arXiv:1906.05474*, 2019.
39. Mozafari, M., R. Farahbakhsh, and N. Crespi. A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. In *Proc., Complex Networks 2019: International Conference on Complex Networks and Their Applications* (H. Cherifi, S. Gaito, J. Mendes, E. Moro, and L. Rocha, eds.), Lisbon, Portugal, December 10–12, 2019, Springer, Cham, pp. 928–940.
40. Sun, C., Q. Xipeng, X. Yige, and H. Xuanjing. How to Fine-Tune BERT for Text Classification? In *Proc., Chinese Computational Linguistics: China National Conference on Chinese Computational Linguistics* (M. Sun, X. Huang, H. Ji, Z. Liu, and Y. Liu, eds.), Kunming, China, October 18–20, 2019, Springer, Cham, pp. 194–206.
41. Reimers, N., and I. Gurevych. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. *Proc., 2019 Conference on Empirical Methods in Natural Language Processing*, Hong Kong, China, 2019.
42. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.
43. Gholizadeh, S., K. Savle, A. Seyeditabari, and W. Zadrozny. Topological Data Analysis in Text Classification: Extracting Features With Additive Information. *arXiv Preprint arXiv:2003.13138*, 2020.