Document Processing: Methods for Semantic Text Similarity Analysis

Conference Paper · September 2020

DOI: 10.1109/INISTA49547.2020.9194665

3 authors, including:

Abdul Wahab Qurashi
University of Huddersfield
2 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Violeta Holmes
University of Huddersfield
86 PUBLICATIONS **1,086** CITATIONS
SEE PROFILE

Document Processing: Methods for Semantic Text Similarity Analysis

Abdul Wahab Qurashi, Violeta Holmes, and Anju P. Johnson
School of Computing and Engineering
University of Huddersfield, United Kingdom
{abdul.qurashi, v.holmes, a.johnson}@hud.ac.uk

***Abstract*—The document text similarity measurement and analysis is a growing application of Natural Language Processing. This paper presents the results of using different techniques for semantic text similarity measurements in documents used for safety-critical systems. The research objective of this work is to measure the degree of semantic equivalence of multi-word sentences for rules and procedures contained in the documents on railway safety. These documents, with unstructured data and different formats, need to be preprocessed and cleaned before the set of Natural Language Processing toolkits, and Jaccard and Cosine similarity metrics are applied. The results demonstrate that it is feasible to automate the process of identifying equivalent rules and procedures and measure similarity of disparate safety-critical documents using Natural language processing and similarity measurement techniques.**

***Index Terms*—Natural Language Processing, Semantic, Jaccard Similarity, Cosine Similarity**

I. Introduction

Measuring text similarity is an essential task in many Natural Language Processing (NLP) applications such as information retrieval, machine translation, and text summarization [1]. Data is generated from different sources and formats such as tweets, email, online reviews, surveys, and spreadsheets. This unstructured data is challenging to process and analyse. As the number and size of text documents are increasing exponentially, there is a growing need for new automated systems for document analysis.

The traditional approach is to represent words as one-hot vectors in which each word is represented by a vector with a unique one for each word in vector space. This technique is not suitable for an extensive dictionary with billions of words. Bag-of-Words (BOW) [2] is a technique used for fixed-length vector representation where features are extracted for modeling. One drawback of the method is that the order of words is lost during the approach leading to the same vector representation for different sentences with the same words [3]. Term Frequency (TF) or Inverse Term Frequency (IDF) are considered to be a useful search query methods as they find the number of occurrences of words in a document, but do not provide measures of any semantic or lexical similarities [4].

Distributed representation or word embedding applies a distributional hypothesis stating that words that appear in a similar syntactic context tend to have similar semantic meanings, i.e., similar words tend to appear in the same places [3]. In [5], authors introduced a method known as word2vec representation as a breakthrough in word embedding. The Skip-gram neural network model of 'Word2Vec' predicts the surrounding words in sentences without using hidden layer neurons. Here the Artificial Neural Network (ANN) is trained by feeding the word pairs extracted from documents, considering the window size as a critical parameter of the algorithm. Skip-gram neural network model consists of weights and biases that are updated with each iteration of input dataset and training of large word dataset would be a time-consuming task [6].

The parallelization in convolutional layers provides good applicability of Convolutional Neural Networks (CNN) for NLP [7]. However, in CNNs, as the information traverses through multiple layers, the output of each layer gets condensed, leading to a compression of sentence structure and loss of information [8].

The aim of this research is to evaluate suitable techniques to measure accurately text similarity between documents. These techniques will be used to assess the feasibility of automating the process of identifying and propagating the changes between two documents containing information on railway safety rules and procedures, the Enterprise Architecture model (EA) and Rail Safety and Standards Board (RSSB) Rule Book.

In this paper, two different text similarity metrics are evaluated: Jaccard and Cosine similarity metric [9]. Jaccard similarity measures the lexical similarity or character matching between texts. In Cosine similarity, the text is mapped into a vector space model, and their distance is measured using the 'Word2Vec' approach [10]. These two measures are used to study the feasibility of automating the process of identifying and propagating the changes between two documents related to railway safety rules and procedures–the Enterprise Architecture model (EA), and Rail Safety and Standards Board (RSSB) Rule Book.

The rest of the paper is organized as follows: Section II presents motivation for this work, followed by NLP frameworks and tools in Section III. Section IV shows an overview of different metrics for text similarity. Section V presents the details of datasets, data pre-processing along with document processing using lexical and semantic similarity analysis, and Section VI presents the experimental results. Finally, in Section VII, we conclude by providing future directions.

978-1-7281-6799-2/20/$31.00

II. Motivation

Safety in the railway industry is of paramount importance. There are several documents designed to capture different aspects of safety, such as rules, objects, and responsible persons. Often different versions of the documentation need to be consolidated and ensure that they are providing matching coherent advice and procedures based on the similar principles of operational safety. Today there is a range of manuals, operating procedures, and guidance containing essential safety-critical information regarding the running of the railway. Digital Railway UK [11], a significant transformation program utilizing a whole-system approach, is accelerating the digital modernisation of the railway to deliver sustainability and safety. It is using the Enterprise Architecture (EA) framework to represent whole-systems changes.

It is crucial to ensure that all requirements specified in other relevant documents are captured in EA. This could be achieved by comparing the EA document with the 'Operational Concept in RSSB document for the GB Mainline Railway' herein referred to as the Operation Concept Document (OCD).

It is possible to identify similarities between the documents for the selected text, using human expertise (through visual inspection), which is a laborious and time-consuming process. This motivates us to deploy suitable techniques to mimic the above human behavior. The aim of this research is to employ NLP techniques to automate the process of identifying similarities and differences in the safety-critical statements within these disparate documents.

III. Natural Language Processing Tools

NLP requires some essential tools for the analysis of big data and to process unstructured data. Various programming languages, software, tools, and libraries have been designed to enhance NLP's performance. Tensorflow and Python are the open-source programming languages that are used for big data analysis generated by computing devices. Python contains many libraries that help to perform different NLP tasks such as Natural Language Toolkit (NLTK), and Pandas support importing, pre-processing, and cleaning of unstructured data. Numpy library helps in scientific computing of large multi-dimensional array and matrices.In this work, we used below mentioned tools for measuring document semantic text similarity.

A. CUDA and CUDA Toolkit

The CUDA toolkit for high-performance parallel computation on the Graphics Processing Unit (GPU), built by NVIDIA, is used to perform standard computational primitives that are incredibly optimized for GPU- accelerated applications. The CUDA Toolkit consists of GPU accelerated libraries, a compiler, development tools, and the CUDA runtime environment. It is used in multiple domains such as deep learning, image processing, and graph analytics [12].

B. Anaconda Navigator

Anaconda consists of a GUI (Graphical User Interface) that supports the running of different applications such as JupyterLab, Jupyter NoteBook, Spyder, and VSCode. It manages a 'conda' (abbreviated form of Anaconda) environment and packages and provides tools that can collect data from different sources and in multiple formats. Its interactive interface makes it easy to use and supports the installation of different packages in a single environment without interfering with other environments.

C. Jupyter Notebook

The Jupyter Notebook is a web-based, open-source interactive development environment that enables the creation and sharing of documents that consist of code, equation, graphs visualizations, data modeling, and many more. It supports various languages and is used for Deep Learning (DL) and NLP tasksThe web browser enables running a live code and sharing it with others. Code can be written in small parts named cells, and each cell can be run separately and debugged. Along with that, there are tools to support a brief explanation and data visualization. Many options for text such as bold, colour and size are also available.

D. Python

It is a programming language that is fast and integrates systems efficiently. It is used for many application domains and is considered best for big data analysis [13]. Many Application Programming Interface (API), software, and libraries provide support for Python.

E. Gensim

Gensim stands for "generate similarity." It is a free Python library that supports topic modeling and semantic similarity between documents. It offers tools like Latent Dirichlet Allocation (LDA) and Latent Semantic Indexing (LSI), which are keys to building high-quality topic modeling. Gensim is considered as one of the best packages for text processing and working with models such as word2vec and FastText [13].

F. GoogleNews Dataset

It is a free, open-source, dataset provided by Google. It includes 3 million words and phrases that are trained on 100 billion words from Google News Dataset. In this research, Google's pre-trained model is used in to assess the similarity between two documents. Gensim library is used to import this model by passing the path of the model file [14].

G. NLTK

The Natural Language Toolkit (NLTK) is open-source and supports Python to work with textual data. It provides over 50 corpora, including different features for text processing such as tokenization, classification, stemming, lemmatization, parts of speech tagging, parsing, and semantic reasoning.

IV. Metrics for Text Similarity

Search engines use text similarity for ranking the search results to a query. Information retrieval with a vector model is an essential application for text similarity, where documents are ranked according to their relevance to an input query [15]. Text similarity is defined by two methods: lexical similarity and semantic similarity. Lexical similarity measures the surface closeness or sequence of strings that are similar to each other. Semantic similarity measures are based on the context of words [16]. In this research Jaccard and Cosine similarity metrics are used for text similarity measures.

A. Jaccard Similarity

Jaccard similarity is a count-based measuring technique defined as the intersection of two sets divided by the union of two sets. It uses a lexical method that provides similarity based on characters, words, strings, and statements matching. Jaccard similarity between two sentences can be represented as in Equation (1) [17].

$$J(S1, S2) = \frac{|S1 \cap S2|}{|S1 \cup S2|} \tag{1}$$

where $'|'$ represent mode, $'\cap'$ and $'\cup'$ denotes union and intersection respectively. An example of Jaccard similarity score calculation for the considered documents is illustrated below (using rules described in Table I):

- **Sentence 1:** 2.3.1.2 Obstruction of the line at a level crossing can occur when:
- **Sentence 2:** 2.3 Obstruction of the line: level crossings and infrastructure work

Both sentences can be represented in sets.

- **S1:** = 2.3.1.2, Obstruction, of, the, line, at, a, level, crossing, can, occur, when, :
- **S2:** = 2.3, Obstruction, of, the, line, :, level, crossings, and, infrastructure, work

The Jaccard similarity based on intersection of two sentences is represented using the Venn diagram shown in figure 1.

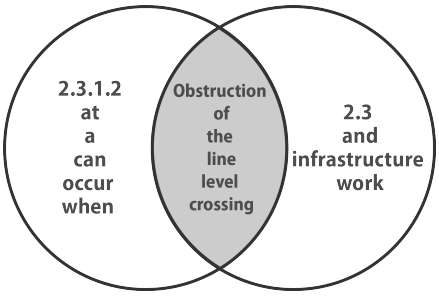

Fig. 1: Jaccard similarity between two sentences.

B. Cosine Similarity

Cosine similarity measures the similarity between documents. Sentences from documents are converted into vectors using the word2vec approach. The similarity is measured by calculating the cosine angle between these vectors [18]. A smaller angle between vectors indicates a higher similarity between the texts. An angle higher than 90 degrees means vectors are orthogonal to each other. This shows that the sentences are not related to each other, as seen in figure 2. The Cosine similarity between two sentences can be represented as in Equation (2)

$$\cos(\mathbf{S1}, \mathbf{S2}) = \frac{\mathbf{S1}.\mathbf{S2}}{\|\mathbf{S1}\|\|\mathbf{S2}\|} = \frac{\sum_{i=1}^{n} \mathbf{S1}_i \mathbf{S2}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{S1}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{S2}_i)^2}} \tag{2}$$

where $S1.S2$ is the dot product of two sentences, divided by the product magnitude of two sentences $||S1||, ||S2||$.

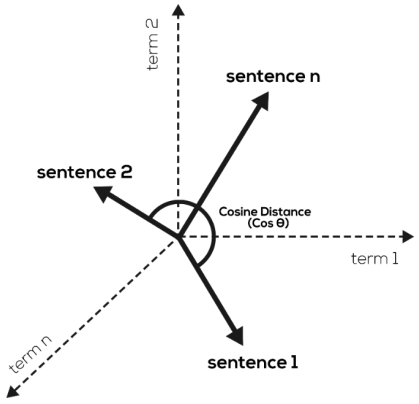

Fig. 2: Cosine similarity between sentences

V. Dataset Details and Processing

Relevant documents and datasets were provided to the Institute of Railway Research at University of Huddersfield as part of a collaborative project with the RSSB [19]. The dataset of interest for this research is derived from the documents containing a set of rules, procedures and responsible persons/actors for railway safety and security.

Initially, two documents were considered, OCDMapping2.CSV and RSSB Enterprise Architecture (EA-lite) in Unified Modelling Language (UML) format. The Enterprise Architecture (EA) file was viewed using software SPARX SYSTEMS. The OCDMapping2.CSV file included information organised in sections such as tranches, modules, principles, OCD lookup, OCD clauses, and responsibility indicators [20].

It was established that there are similarities between the RSSB EA and OCD files, especially concerning the numbering of clauses that were identical in EA and OCD documents. However, it was evident that the UML model did not include the complete text of the clauses from the OCD document, but an abbreviated version of the text. Hence, it is necessary to compare the documents and quantify the degree of similarity between them. In particular, it is required to ascertain the differences in clauses that were observed in the close examination of OCD and EA (UML).

A model was devised using NLP tools, ML techniques, and text similarity measures to compare the text for a selection of clauses from EA and OCD documents and prove the feasibility of comparing and measuring similarity automatically. The model consists of different stages of the EA and OCD documents processing and measuring document similarity as show in figure 3.

1) **Pre-process:** convert both documents to .CSV format and additionally export free text from EA (UML) file into EA.CSV.
2) **Import:** import documents to Jupyter NoteBook using Python language environment.
3) **Tokenize:** tokenize documents using the NLTK library.
4) **Clean data:** remove free text not starting with a sequence of digits, remove stop-words in clauses such as *of, a, an*; remove duplicate words in text on actors/responsible persons.
5) **Vectorize:** use Word2vec pre-trained model (on Google News Dataset) to transform both .CSV documents into a vectorized format.
6) **Document processing:** measure similarity of vectorized documents using different metrics; match text on actor/responsible persons.

A. Pre-Processing

Data pre-processing is one of the most crucial and demanding tasks. In order to perform similarity analysis of the rules and actors/responsible persons in the OCD and EA documents, both files need to have the same format. From the OCDMapping2.CSV file and its OCD lookup section, data was extracted and exported to OCDlookup.CSV file. This document includes rules and principles for command and safety on railway – OCD Rulebook. Data from EA UML file was exported into EA.CSV file format, containing information about actors, principles, and rules in the EA file. The EA.CSV and OCDlookup.CSV files were imported into the Python environment, for lexical and semantic similarity analysis.

B. Tokenization

Tokenization is a process of breaking the paragraphs or documents into sentences or words. It is one of the critical steps in text processing. Machine learning (ML) models take vectors as input. When working with text, it is essential to convert strings into numbers, or "vectorize" the text, before feeding it to the ML model. NLTK is a Python-based library used for various text processing tasks such as tokenization, stemming and lemmatization [21]. It was used to tokenize both EA and OCD.CSV files from large documents to sentences and words, before data cleaning.

C. Data Cleaning

Visual inspection of the documents confirmed that the corresponding rules in EA and OCDlookup documents have the same index numbers, which provides the basis for rule similarity analysis. Data cleaning was performed on both EA and OCD lookup.CSV documents to remove any sentences (free text) which are not starting with a sequence of numbers. The Python library "re"(Regular Expression), a module that provides regular matching expression or matching of strings, was used for the above process. 'Re' can be concatenated to form a new regular expression, which helps to identify a specific string.

D. Data Transformation using Word Embedding

Word2Vec is one of the ML-based algorithms that generate word embedding - an efficient, dense representation in which similar words have a similar encoding. It creates vectors that are distributed numerical representations of word features such as the context of individual words (self-supervised learning). The input to the algorithm is a large corpus of words, and the output produces a vector space, with each unique word in the corpus being assigned a corresponding vector in the space. The Word2vec neural network output is a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net or queried to detect relationships between words. Word2vec data transformation was performed before the application of different text similarity measures in our research.

E. Document Processing using lexical and semantic similarity analysis

To conduct document similarity analysis, initially, only a section of the text was extracted for a clause "Separation of Trains" from the EA and OCD. The result of a comparison using the 'difflib' library in Python generated an HTML file which indicated the defence using a colour legend. This was an excellent indication to illustrate a proper mapping of a portion of text from OCD to the EA system. A similar process was conducted for two more cases: "Movement authority for a train" and "Obstruction of the line: level crossings and infrastructure work." The results of initial comparisons demonstrated that an entire document could be parsed to compare the clauses contained in these documents.

Although the visualisation of document similarity in the HTML file was useful it is necessary to compare the documents and quantify the similarity. In particular, it is required to ascertain the differences in clauses that were observed in the close examination of the documents. Hence, a model shown in figure 3 was used to apply Jaccard and Cosine similarity measures to vectorized EA and OCD documents, and calculate similarity scores between each clause of EA and OCD. The threshold of similarity is set to $> 80\%$ to identify the text with high similarity between EA and OCD documents. A **zero** score shows no similarity and **one** represents that the sentences/clauses are identical.

In addition to rules and procedures, important information for safety is to identify responsible persons in OCD document, represented as actors in the UML version of the EA document. Hence, it was required to analyse and compare the EA and OCD documents for consistency. The goal was to identify the text for responsible persons and actors in the documents and compare them. A counter function is used that counts the word

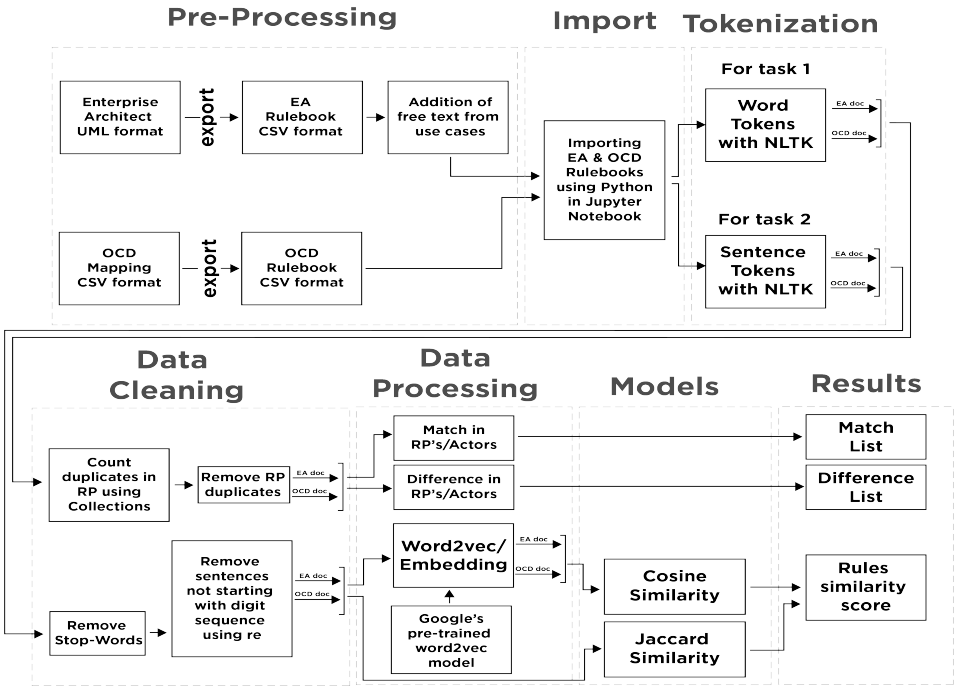

Fig. 3: Model for text similarity using different algorithms

frequency for the **responsible persons** in OCD and **actors** in the EA document. The function result shows match/difference of occurrence of actors/responsible person in both documents. A number of experiments were conducted, using Jaccard, and Cosine algorithms to perform lexical and semantic similarity analysis of the documents, which will be presented in the following section.

VI. Results and Analysis

The focus of this research, as part of the railway safety and security project, was to perform semantic similarity analysis of two principle documents EA and OCD files provided by RSSB. A stand-alone machine with a specification of core i7, NVIDIA GPU Geforce 980M, 12GB RAM was used. The analysis was conducted using natural language processing techniques and tools, using the Jaccard and Cosine similarity algorithms. The scores with positive (+ive)/negative (-ive) note indicate accurate/inaccurate mapping between two documents sentences. As seen in table I the results from Jaccard similarity were not satisfactory as it uses a lexical method that measures the similarity based on character matching. When comparing EA rule 2.5.2.2 with rules in OCD Jaccard similarity score gives high similarity for rule 2.2.2.5 which is inaccurate result. Cosine Similarity provides more accurate results as it calculates the cosine of an angle between embedded sentences from two documents. Table I illustrates that the rule 2.1.1.4 from EA is matched with an equivalent rule in OCD rulebook with a positive similarity score of 1.0. These results provide a quantitative measure of similarity between the rules/clauses ranging from 0 to 1. The results show that there is a substantial match between some of the rules in two documents $> 80\%$ when matching threshold limit of 0.8 value was applied. Different threshold limits were tested during this research, and a decrease in threshold value resulted in detection of irrelevant rules since each rule from EA is compared with whole of OCD document. Also, there are some rules from EA that do not have a comparative rule in OCD architecture and are returning an empty list as a result with Cosine similarity and shows negative similarity score of 0.89 as seen in table I. The rules similarity in EA and OCD documents were quantified using different similarity metrics, with outputs compiled into CSV files. Based on the results, it was found that the Cosine similarity algorithm is giving more accurate results than the Jaccard similarity approach. However, each method was useful in rulebook document analysis since they are using both lexical and semantic text analysis methods. From the results of document analysis regarding the responsible persons/actors in EA and OCD files, respectively, it is evident that there are no significant similarities in the documents supplied for this project. However, further investigation is required to arrive at an agreeable remark.

VII. Conclusion

In this paper, we evaluated the performance of different similarity metrics for document analysis. We compared the approach on the dataset from the documents on railway safety and security provided by RSSB. The experimental results demonstrated that Cosine outperforms Jaccard metrics. Cosine measures the angle between vectorize sentences and gives higher accuracy in performing similarity analyses of two documents. The work points that measure of text similarity for rules and procedures, and identification of actors/responsible

Source rules or EA rulebook	Target Rules or OCD rulebook	Jaccard Similarity Score	Cosine Similarity Score
2.1 Separation of Trains	2.1 Separation of trains	0.93 (+ive)	1.0 (+ive)
2.1.1.4 - Written operating procedures or spoken instructions when the signalling system cannot be used to preserve the effectiveness of the space interval	2.1.1.4 Operating procedures using written or spoken instructions are applied when the signalling system cannot be used to preserve the effectiveness of the space interval:	0.87 (+ive)	1.0 (+ive)
2.2 Movement authority for a train	2.2 Movement authority for a train	1.0 (+ive)	0.99 (+ive)
2.2.1.1 Safety is compromised if a train proceeds without a movement autority	2.2.1.1 The safety benefits of a system for maintaining space intervals between trains (see section 2.1 of this operational concept document) are compromised if a train proceeds without an authority for its movement.	0.86 (-ive)	0.81 (+ive)
2.5.2.2 Trains without automatic brakes or through pipes cannot move on the mainline network without additional safeguards applied	2.2.2.5 Operating rules must prescribe the actions to be taken by signallers to mitigate the risks of collision or derailment in the event of a train proceeding without movement authority.	0.89 (-ive)	-
2.3 Obstruction of the line: level crossings and infrastructure work	2.3.1.2 Level crossing obstruction	0.88 (-ive)	0.81 (+ive)

TABLE I: Comparison of Jaccard and Cosine similarity score for two documents

persons in the RSSB documents helps maintain the consistency of safety instructions in the documents for safety-critical systems. Whenever a new rule/procedure or a new responsible person/actor is added, the method enables to perform a consistency check in both documents. Semantic text similarity analysis methodology, with the application of NLP techniques and tools and semantic text measures, was devised in this research and used for railway safety and security document analysis. This study has made achievements in this respect; it could be followed and implemented in document analysis in other domains and applications. Additionally, in the future, different hardware platforms such as CPU, GPU, and Field Programmable Gate Arrays (FPGA) will be considered to evaluate the performance of this model for document similarity analysis.

REFERENCES

[1] P. Chen, F. Wu, T. Wang, and W. Ding, "A semantic qa-based approach for text summarization evaluation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[2] J. Brownlee, *Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems*. Machine Learning Mastery, 2017.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[4] J. L. Neto, A. D. Santos, C. A. Kaestner, N. Alexandre, D. Santos *et al.*, "Document clustering and text summarization," 2000.

[5] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[6] C. McCormick, "Word2vec tutorial part 2 - negative sampling chris mccormick," 2020. [Online]. Available: http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/

[7] S. Chollampatt and H. T. Ng, "A multilayer convolutional encoder-decoder neural network for grammatical error correction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[8] H. Yao, H. Liu, and P. Zhang, "A novel sentence similarity model with word embedding based on convolutional neural network," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 23, p. e4415, 2018.

[9] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press, 2008.

[10] F. Gao, S. Wang, X. Li, H. Cao, and X. Cai, "Research on text mining of railway safety supervisors performance based on bilstm and crf," in *Journal of Physics: Conference Series*, vol. 1213, no. 5. IOP Publishing, 2019, p. 052016.

[11] "Supporting sustainable growth," Aug 2019. [Online]. Available: https://digitalrailway.co.uk/

[12] C. Nvidia, "Toolkit documentation," *NVIDIA CUDA getting started guide for linux*, 2014.

[13] V. Fedak, "5 heroic tools for natural language processing," 2020. [Online]. Available: https://towardsdatascience.com/5-heroic-tools-for-natural-language-processing-7f3c1f8fc9f0

[14] 2020. [Online]. Available: https://github.com/mmihaltz/word2vec-GoogleNews-vectors

[15] R. Mihalcea, C. Corley, C. Strapparava *et al.*, "Corpus-based and knowledge-based measures of text semantic similarity," in *Aaai*, vol. 6, no. 2006, 2006, pp. 775–780.

[16] N. Pradhan, M. Gyanchandani, and R. Wadhvani, "A review on text similarity technique used in ir and its application," *International Journal of Computer Applications*, vol. 120, no. 9, 2015.

[17] L. Zahrotun, "Comparison jaccard similarity, cosine similarity and combined both of the data clustering with shared nearest neighbor method," *Computer Engineering and Applications Journal*, vol. 5, no. 1, pp. 11–18, 2016.

[18] F. Rahutomo, T. Kitasuka, and M. Aritsugi, "Semantic cosine similarity," in *The 7th International Student Conference on Advanced Science and Technology ICAST*, vol. 4, no. 1, 2012.

[19] Y. Zhao, 2020. [Online]. Available: https://spark-uat.rssb.co.uk/Lists/Records/DispForm.aspx?ID=24168

[20] 2020. [Online]. Available: https://sparxsystems.com/

[21] N. Hardeniya, J. Perkins, D. Chopra, N. Joshi, and I. Mathur, *Natural Language Processing: Python and NLTK*. Packt Publishing Ltd, 2016.