

# A Machine Learning Guided Path for Optimal Literature Review

Denitsa Panova<sup>1</sup>

<sup>1</sup> *Shumen University "Bishop Konstantin of Preslav", Bulgaria*

**Abstract** – This paper introduces a novel machine learning framework to address the challenge of optimizing literature research by identifying the optimal path. To create dataset and ensure the versatility of the solution for different applications, we developed an online scraping tool designed to extract articles from ResearchGate based on a specific search query. The proposed machine learning model leverages contextual embeddings and graph theory, translating intricate scholarly work into informative steps for one to go wider rather than deeper in their research. By employing a Christofides approximation of the Traveling Salesman Problem algorithm, our model efficiently navigates through more than 1000 article embeddings. We prove that the resulting path not only accelerates the knowledge gaining process, but also evidently diversifies the findings. Moreover, we evaluated multiple PDF reader libraries to arrive at the most suitable one for the purpose. This adaptability allows the framework to be applied not only to scraped articles, but also to those stored as PDF files, giving an option for multiple data sources. In conclusion, this paper presents a transformative approach for literature research optimization, equipping researchers with a potent tool to efficiently explore articles.

**Keywords** – Travelling salesmen problem, graph theory, sentence transformers, web scraping, PDF libraries.

---

DOI: 10.18421/TEM131-64

<https://doi.org/10.18421/TEM131-64>

**Corresponding author:** Denitsa Panova,  
*Shumen University "Bishop Konstantin of Preslav", Bulgaria.*


**Email:** [denitsa.panova@gmail.com](mailto:denitsa.panova@gmail.com)

*Received: 21 September 2023.*

*Revised: 04 January 2024.*

*Accepted: 15 January 2024.*

*Published: 27 February 2024.*

 © 2024 Denitsa Panova; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License.

The article is published with Open Access at <https://www.temjournal.com/>

## 1. Introduction

In the realm of scientific research, the accumulation of knowledge is the cornerstone of advancement. Literature review sections within research papers serve as vital conduits for building upon existing knowledge and fostering innovation. However, with the exponential growth of published articles, the process of comprehensive review from an extensive body of literature in a fast manner has become a daunting challenge.

This paper introduces a novel machine learning framework designed to address this challenge by appointing the optimal path for literature research. In the context of this study, optimal path and extensive research refer to set of articles which provide different angles on the same generic topic, giving a wide rather than deep knowledge. Our proposed model leverages the power of natural language processing, in particular neural network embeddings and graph theory, to sift through vast volumes of scholarly work, distilling their core contributions into concise and informative map. By harnessing the potential of the framework, scientists can significantly enhance their ability to extract relevant information swiftly, identify key trends, and ultimately expedite the process of knowledge assimilation.

In the subsequent sections, we delve into the architecture, methodologies, and evaluation metrics of our proposed model. Through comprehensive experiments and comparisons with existing methods, we showcase the model's effectiveness in accurately providing guided path to inspect various domains. Our findings not only underscore the efficiency of our approach but also highlight its potential to revolutionize the way researchers navigate the ever-expanding landscape of scientific literature.

The rest of this paper is organized as follows: Section 2 provides a summary of related work and outlines the methodology and data used in the proposed framework. Section 3 presents the results of the experiments, followed by a comprehensive analysis of the findings.

Finally, Section 4 concludes the paper with a summary of contributions, implications, and potential

future directions in the domain of automating literature research.

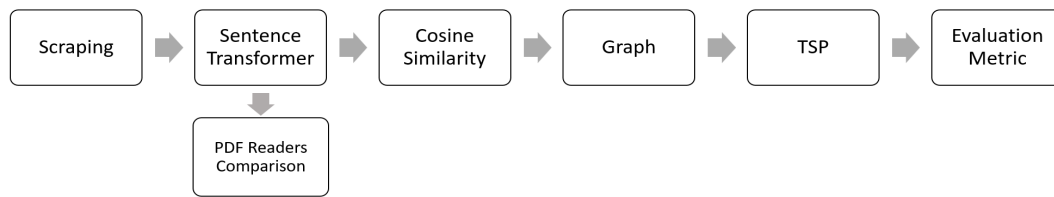


Figure 1. Framework overview

## 2. Methodology

To the knowledge of the authors of this research, there is no published machine learning approach to provide a guided path for literature research optimization. An overview of the proposed framework architecture is displayed in Figure 1. To initiate any research endeavour, the first step is to identify a reliable source. In our case, we have opted for ResearchGate as our primary location for articles. ResearchGate is a European commercial social networking platform designed for scientists, established in 2008 [1]. Since then, it has increased its popularity. In 2016 it was the website with highest number of active users from the academic world [9], as many scholarly profiles as Google Scholar [26]. The platform has been chosen for its widespread popularity, and we will not delve into an exhaustive discussion of the websites' pros and cons.

### 2.1. Web Scraping

When scientists embark on their research journey, the initial and instinctive step involves entering a general query into the search bar. This query serves as the starting point for assessing the relevance of all articles by reviewing their abstracts. The proposed framework aims to replicate this human-like behaviour; therefore, a Python scraper is developed, using selenium [2]. The code has the following functionality:

1. Logs in to ResearchGate with credentials.
2. Searches a vague statement – ‘bee acoustic.’
3. Loops over the pages, downloads all the articles and their abstracts.
4. Saves the file names.

It is important to be noted that if anyone wants to recreate the procedure, it is required that some sleep time between the download periods is added to avoid being rate limited. We added a random number between 30 and 60 seconds. The same waiting time is advisable to be applied to the fourth step as well, since process time is required for the browser to fully download the PDF.

### 2.2. Sentence Transformer

To work programmatically with text, a mathematical representation is necessary. There are three primary types of embeddings – traditional (such as Term Frequency - Inverse Document Frequency), static (such as one-hot encoding), and contextualized (such as Bidirectional Encoder Representations from Transformers, BERT) [21]. Since the aim of this paper is to find an optimal path for reading articles based on their context, naturally we utilize the contextualized embeddings. We chose to represent the article abstracts using sentence transformers, state of the art Python implementation of Siamese BERT-Networks [13], [22]. The framework provides pretrained models for general use which resonates to the purposes of this research.

‘Huggingface’ is used as a library for pretrained models. In particular ‘all-mpnet-base-v2’ instance is chosen [3]. As per the benchmarks, it proves to perform with best quality – transforms the text to 768-dimensional vector, even though it is not the fastest option [10]. For the purposes of this research, we prefer quality over speed. Additionally, it is the model which considers the greatest number of words before truncating the text – 384. To validate how many of the abstracts fit that criterion, abstracts are split into words, using the ‘nltk’ Python package [14]. It is observed that 93% of the articles will not be truncated. The rest are considered as outliers. Figure 2 shows a histogram of the word count distribution over the dataset.

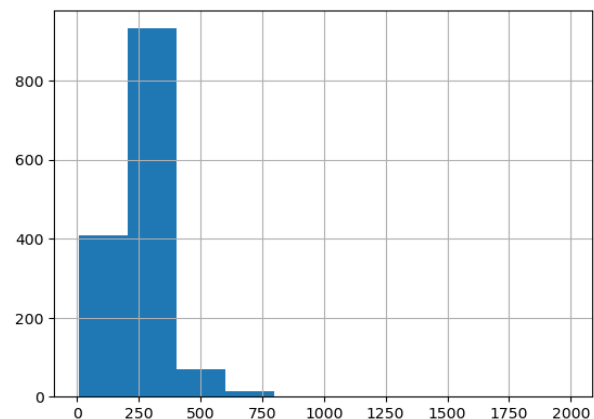


Figure 2. Word count distribution over the dataset

### 2.3. Similarity Metric

There are two metrics which are usually used for comparing contextual embeddings – cosine similarity and Pearson coefficients. Research has demonstrated the equivalence of both metrics when applied to such vectors [27]. Consequently, we opt to utilize cosine similarity as the preferred comparison metric.

### 2.4. Social Network Analysis (Graph Theory)

The goal of the social network analysis is to illustrate social interactions between different agents (known as nodes) and thus reflect reality in a mathematical manner. It is a standard practise to represent text embeddings in a network [15]. To construct a graph, one needs nodes (in this instance the abstracts) and edges between them, the calculated pairwise cosine similarity. By design, the resulting graph is undirected; fully connected (every two nodes have an edge); signed and weighted, since the edges have value between (-1;1); and homogeneous, since all nodes have the same function. As Python implementation, we will use the ‘networkx’ package [12].

### 2.5. Travelling Salesmen Problem (TSP)

The travelling salesmen problem is a well-known optimization problem. In its essence, it tries to answer the following question – “If one has a list of cities and the distance between each pair of cities, what is the shortest route to visit each city?” For the first time the problem has been mentioned in a handbook for travelling salesmen from 1832, from where the naming convention arises [23].

TSP is a NP-hard problem and does not have a polynomial-time algorithm to find optimal solution. There are different solutions for approximating the solution. This paper utilizes the Christofides algorithm, which in its essence uses the minimum spanning tree notion [5], [20]. The approximation ensures the solution will be a factor of 3/2 of the optimal path length. The method has been developed in 1976 and it is one of the base algorithms for solving the challenge [8]. We deliberately employed it to demonstrate that even with a standard algorithm, the framework would showcase its utility.

### 2.6. Evaluation Metric

The goal of the paper is to demonstrate employment of the proposed framework that will yield a more effective path—one that includes a greater diversity of articles, facilitating comprehensive literature research. Furthermore, the aim of the social network analysis is to simulate real-world conditions.

In typical research practices, scientists often cycle through a restricted number of result pages, let's assume 50 articles this case. The evaluation metric introduced in this paper is visually described in Figure 3. We will compare the average similarity of articles within the first 50 articles of results obtained originally from ResearchGate and the proposed TSP path.

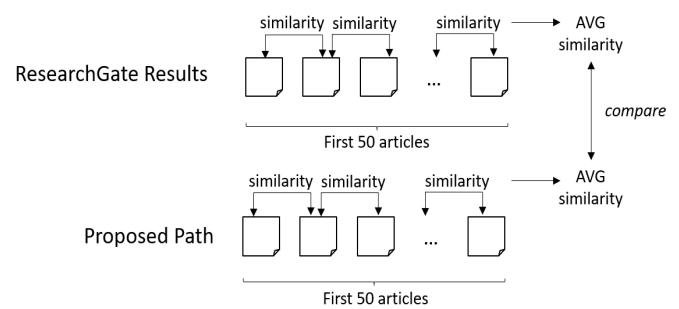


Figure 3. Evaluation metric design

Additionally, we introduce a loss metric. If researchers choose to follow the article order suggested by ResearchGate, it will require a certain amount of reading time to reach the articles recommended by TSP. The reading rate for studying is from 100 to 200 words per minute depending on the different sources [28], [17]. For the purposes of this research, we will assume 150 words per minute. In its essence, the “loss” metric is the number of extra articles a researcher should read if following the ResearchGate path, to cover the first 50 articles recommended by TSP. For instance, if the first article in the TSP sequence is the 100th in the original ResearchGate order, the reader would have to peruse 99 articles before reaching the suggested one. In our calculation, we will exclude articles that are already part of the TSP-recommended path. Building on the previous example, if article 50 is already in the suggested path, it will not be factored into the loss function because the reader would need to read it anyway. Once we have identified the “loss” articles, we will proceed to compute the word count and the time lost. The loss function will be applied to the initial 50 articles as well.

### 2.7. PDF Libraries

As a final step of the research, we evaluate various Python packages for processing PDF documents. The objective is to ensure that the framework remains applicable even when the source of information consists of saved PDF documents, which is a prevalent method for archiving articles. From our web scraping efforts, we have access to both the abstract and the stored PDF files. However, it is worth noting that converting PDF files to text is a non-trivial task.

There are numerous Python packages available for this purpose, and we will assess their effectiveness based on several metrics:

- 1) Keyword search misses - inability to find text between keywords “abstract” and “introduction.”
- 2) Time - Time required to process the PDF files.
- 3) Average similarity - average Levenshtein distance between the scraped and the PDF-extracted abstracts, using the ‘thefuzz’ Python package [24], [11].

In this study, we will assess the performance of five packages that were newly updated this year (‘pypdf’, ‘pypdfium2’, ‘borb’, ‘pdfnet’, and ‘tika’) alongside one package released in 2017, which is ‘fitz’ [6], [16], [19], [3], [4], [18], [25]. Furthermore, it is worth noting that one of these packages, ‘pdfnet’, is a paid solution.

It is essential to emphasize that our objective is to offer a comparison of the effectiveness of these packages rather than delving deeply into the reasons for their performance.

### 3. Results

The scraper code was executed twice, first in May and then in August 2023. ResearchGate's website typically provides a maximum of 1000 results for each query. After these two scraping sessions, we accumulated a total of 1472 articles. Subsequently, a manual inspection of the downloaded files revealed that only three had not been downloaded correctly, and we intervened manually to rectify this issue. This translates to an impressive success rate of approximately 99.8%. Additional 45 results do not have saved file since it is not available on the website, which we exclude from the results.

The figure shows a screenshot of the ResearchGate search interface. The search query is 'bee acoustic'. The results are sorted by relevance. The first result is 'Bee detection in bee hives using selective features from acoustic data' by Furqan Rustam, Muhammad Zahid Sharif, and Wajdi Aljedaani, published in August 2023. Below the screenshot is a table summarizing the data extracted from the search results.

	Title	URL	Authors	Date	Left Category	Abstract	Footer	File
0	Landscape conser...	https://www.rese...	Rafaela Mendes A...	Oct 2022	Publisher previe...	Landscape struct...	3 Recommendation...	C:\Users\dpanova...
1	Impact of the Pl...	https://www.rese...	R. Shumkova;Rali...	Sep 2022	Full-text availa...	Winter is the se...	1 Recommendation...	C:\Users\dpanova...
2	A contemporary s...	https://www.rese...	Kaleigh Fisher;K...	Mar 2022	Full-text availa...	Bumble bees (gen...	1 Recommendation...	C:\Users\dpanova...
3	Canopy sampling ...	https://www.rese...	Guthrie Allen;Ri...	Oct 2022	Full-text availa...	1. Woodlands can...	1 Recommendation...	C:\Users\dpanova...
4	Enhanced bubble-...	https://www.rese...	Jianfeng Wang;Yu...	Aug 2023	Full-text availa...	The role of the ...	2 Recommendation...	C:\Users\dpanova...
...	...	...	...	...	...	...	...	...
1421	Dialect classifi...	https://www.rese...	Mohammad Ali Hum...	Jun 2023	Full-text availa...	span lang="EN-US...	:18 Reads	C:\Users\dpanova...

Figure 4. Data in ResearchGate and Python

It is crucial to highlight why we store the file names. There are no standardized naming conventions for these files, resulting in situations where the title can be quite lengthy, like "Seasonal Dynamics of the Honey Bee Gut Microbiota in Colonies Under Subtropical Climate: Seasonal Dynamics of Honey Bee Gut Microbiota," while the file name may be something like 'Castelli2021\_Article\_SeasonalDynamicsOfTheHoneyBeeG.pdf.' This meticulous step is vital for our research, ensuring that we can confidently match each file with its corresponding abstract and accurately compare the PDF readers.

As part of this process, we also make cosmetic adjustments, extracting the file type (e.g., 'Article', 'Proceeding' etc.). Additionally, we convert the date column into a datetime dimension and break down the year. Furthermore, we create tag columns based on the footer information to discern which results contain recommendations, reads, and citations. Given that the dataset includes information from two separate scraping sessions. There is a need to address the presence of duplicate entries for the result positions. In cases where duplications occur, we prioritize the most recently published article, considering it as the primary entry. It is worth noting that most of the scraped results pertain to articles published after 2022. Figure 4 shows how the data looks on the website and, in the 'pandas' data frame used for visualization.

We compute the pairwise similarity between the 1425 abstracts. Figure 5 shows its distribution. As expected, they are similar to some extent, proven by the lack of extreme negative values in the cosine similarity.

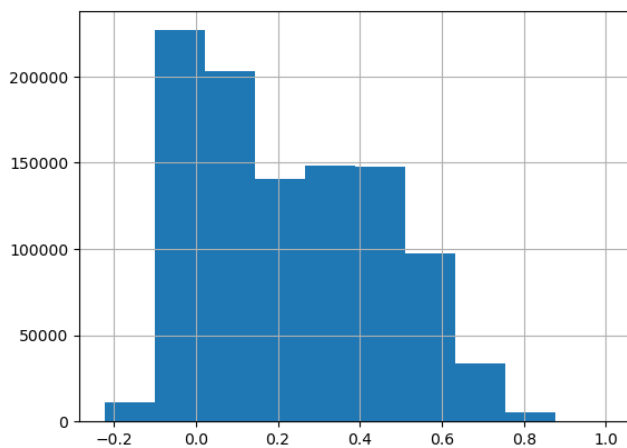


Figure 5. Pairwise Similarity

A graph is constructed, and the Traveling Salesman Problem (TSP) algorithm is applied to it, resulting in an execution time of 1,506 seconds, executed on a computer, equipped with a 11th Gen Intel(R) Core(TM) i7-11850H running at 2.50GHz and 32.0GB of system memory.

When examining the first 50 results of the generated path, the average similarity score is found to be 0.15, while for the ResearchGate articles, it is 0.49. This strongly suggests that the proposed algorithm has successfully diversified the results, making the suggested path more optimal for conducting comprehensive literature research. Recognizing that the average metric can be deceptive, we turn our attention to the distribution of results displayed in the histogram. It becomes evident from the histogram (Figure 6) that a significant majority of the results lack similarity.

When quantifying the "loss" articles, they constitute 672 results. This translates to 178,515 words, or a time investment of approximately 1190 minutes, which is equivalent to about 20 hours of consecutive reading required to attain a comparable range of knowledge diversity.

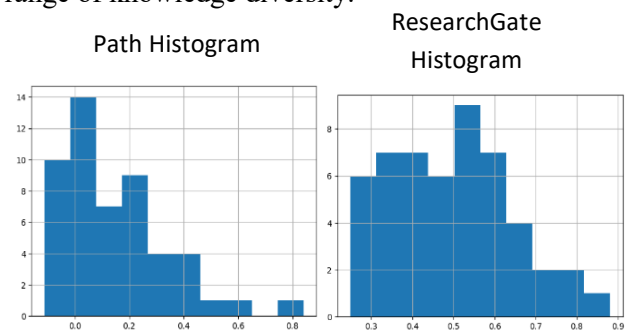


Figure 6. Pairwise similarity for comparison

The final phase of our research involves identifying the most suitable PDF reader. Table 1 presents a summary of the evaluation criteria results. We would recommend 'pypdfium2' as the optimal choice. This recommendation is based on several factors: it demonstrates one of the highest accuracy levels, boasts a relatively low execution time, records the lowest number of missing keyword texts, and has received the most recent updates.

Interestingly, 'pdfnet' does exhibit a notable quantity of missing data, considering it is paid tool, but an examination of the similarity distribution reveals that the majority of abstracts have a similarity score of 80% or higher. Despite being considered a state-of-the-art PDF reader, 'tika's performance is heavily influenced by PDF formatting and does not align well with the keyword search evaluation method.

Another state-of-the-art package, 'borb', while capable, suffers from slow execution times. It is worth noting that the reported results are based on a sample of only 20 articles, and the execution time is not practical for handling a large volume of files.

Table 1. Package comparison

Package	Updated date	Time (sec.)	Key-word Misses	AVG similarity
Pypdf	Aug 2023	1387	640	84.24
Fitz	Feb 2017	80	659	84.47
Pypdfium2	Jul 2023	91	606	85.96
Pdfnet	May 2023	219	1066	80.97
Tika	Jan 2023	263	1425	0
Borb*	Jul 2023	448	5	77.26

\*Calculated only for 20 files

#### 4. Discussion

In this rapidly evolving era of scientific research, where the volume of published articles continues to grow exponentially, the task of efficiently navigating the vast sea of knowledge has become a formidable challenge. As we introduced in the preceding sections, literature review plays a pivotal role in advancing research, acting as a bridge between existing knowledge and novel discoveries. However, the sheer magnitude of available literature can be overwhelming, often hindering the pace of innovation.

Our paper has presented a novel machine learning framework that seeks to address this challenge by providing researchers with an optimized path for literature exploration. In the context of this study, an "optimal path" refers to a curated set of articles that offer diverse perspectives on a given topic, facilitating a broad understanding rather than a deep dive into a specific area. This framework harnesses the power of natural language processing, specifically transformer models, and incorporates principles from graph theory to distill the essence of extensive scholarly work into a concise and informative map. By leveraging this framework, scientists can significantly expedite their ability to access pertinent information, identify emerging trends, and streamline the knowledge assimilation process.

We have proven that using the proposed methodology will significantly increase the diversification of the set of articles compared to the original order provided by the search engine. The average cosine similarity drops from 0.49 to 0.15. Additionally, we have shown the amount of time a researcher will save to arrive at the same comprehensive articles – 20 hours reading time.

The innovation highlighted in this paper lies in the framework itself and its practical application within the realm of optimizing literature reviews. It represents a comprehensive solution that researchers can directly utilize, whether by sourcing their information from ResearchGate or a collection of PDF articles.

Our comprehensive experiments and comparisons with existing methods have demonstrated the effectiveness of our model in providing accurate guidance for exploring various domains of scientific literature. These findings underscore not only the efficiency of our approach but also its potential to revolutionize the way researchers navigate the ever-expanding landscape of scholarly publications. As the research advances, one avenue for enhancing the model could involve exploring various TSP algorithms to optimize and expedite the resolution time. Additionally, an improvement could entail incorporating cluster creation based on title embeddings, the number of reads, recommendations, and citations. These resultant clusters could then be leveraged to apply TSP between them, complementing the existing solution. This approach would not only offer an optimal path but also enable the suggestion of similar articles if a particular article is deemed "relevant."

#### 5. Conclusion

The framework introduced in the paper is designed to address the challenge of finding the optimal path for literature research. The focus is on breadth, rather than depth, of knowledge which allows researchers to gain a comprehensive understanding of their chosen topics swiftly.

The proposed solution harnesses the power of contextual embeddings and graph theory, enabling us to translate complex scholarly work into informative maps. With an impressive success rate of approximately 99.8% in data acquisition (mapping scraped articles to files), we ensured the quality and reliability of our test dataset.

Through a Christofides approximation of the Traveling Salesman Problem (TSP) algorithm, our framework efficiently navigates through the 1425 abstracts in reasonable time. The resulting path not only enhances the speed of knowledge assimilation (20 hours of continuous reading time saved) but also diversifies results, as reflected in the average cosine similarity scores and the histogram distribution.

Furthermore, our evaluation of PDF readers has identified 'pypdfium2' as the optimal choice, based on factors such as accuracy, execution time, and the finding text based on key words. While other packages like 'pdfnet', 'tika', and 'borb' offer unique features, they come with certain limitations that may affect their suitability for comprehensive literature research using the proposed framework. In this way the solution can be applied not only to articles which are scraped but also stored as pdf files with a high degree of accuracy.



In conclusion, this paper presents a transformative approach to literature research optimization, offering researchers a powerful tool to navigate the ever-expanding scientific literature efficiently. With a strong foundation in data acquisition, preprocessing, and analysis, our framework holds the promise of revolutionizing the way researchers explore and synthesize knowledge in the digital age.

#### References:

- [1]. ResearchGate (n.d.). *ResearchGate search*. ResearchGate. Retrieved from: <https://www.researchgate.net/> [accessed: 29 August 2023].
- [2]. The Python Package Index (n.d.). *Selenium 4.16.0*. PyPI. Retrieved from: <https://pypi.org/project/selenium/> [accessed: 30 August 2023].
- [3]. Huggingface (n.d.). *all-mpnet-base-v2*. Huggingface. Retrieved from: <https://huggingface.co/sentence-transformers/all-mpnet-base-v2> [accessed: 30 August 2023].
- [4]. Jorisschellekens. (n.d.). *GitHub - jorisschellekens/borb*. Git Hub. Retrieved from: <https://github.com/jorisschellekens/borb> [accessed: 02 September 2023].
- [5]. Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. *Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group*.
- [6]. PyMuPdf. (n.d.). *Module Fitz*. PyMuPdf Retrieved from: <https://pymupdf.readthedocs.io/en/latest/module.html> [accessed: 03 September 2023].
- [7]. *GetRepo*. (n.d.). Retrieved from: <https://github.com/dpanova/A-Machine-Learning-Guided-Path-for-Optimal-Lit-Review> [accessed: 02 September 2023].
- [8]. Karlin, A. R., Klein, N., & Gharan, S. O. (2021). A (slightly) improved approximation algorithm for metric TSP. *the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. New York.
- [9]. Matthews, D. (n.d.). *Do academic social networks share academics' interests?* Times Higher Education. Retrieved from: <https://www.timeshighereducation.com/features/do-academic-social-networks-share-academics-interests> [accessed: 03 September 2023].
- [10]. Sbert. (n.d.). *Model Overview*. Sbert. Retrieved from: [https://www.sbert.net/docs/pretrained\\_models.html](https://www.sbert.net/docs/pretrained_models.html) [accessed: 04 September 2023].
- [11]. Navarro, G. (2001). A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, 33(1). Doi:10.1145/375360.375365
- [12]. *Networkx*. (n.d.). NetworkX documentation. Networkx. Retrieved from: <https://networkx.org/> [accessed: 05 September 2023].
- [13]. Nils Reimers, I. G. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *EMNLP*. Doi: 10.48550/arXiv.1908.10084
- [14]. *NLTK*. (n.d.). Natural Language Toolkit. NLTK. Retrieved from: <https://www.nltk.org/> [accessed: 06 September 2023].
- [15]. Paul Compagnon, K. O. (2017). *Graph Embeddings for Social Network Analysis: State of the Art*. Institut National Des Sciences Appliquees Lyon.
- [16]. pdfnet. (n.d.). *Custom Python Wrapper PDF Library for Windows | Apyrse SDK*. docs.apyrse Retrieved from: <https://docs.apyrse.com/documentation/python/get-started/python3/windows/> [accessed: 08 September 2023].
- [17]. Primativo, S., Spinelli, D., Zoccolotti, P., De Luca, M., & Martelli, M. (2016). Perceptual and cognitive factors imposing "speed limits" on reading rate: a study with the rapid serial visual presentation. *PLoS one*, 11(4), e0153786. Doi:10.1371/journal.pone.0153786
- [18]. PyPDF. (n.d.). *Welcome to PyPDF2 &mdash; PyPDF2 documentation*. PyPDF. Retrieved from: <https://pypdf2.readthedocs.io/en/3.0.0/> [accessed: 10 September 2023].
- [19]. Pypdfium2. (n.d.). *pypdfium2 &mdash; pypdfium2 documentation*. Pypdfium2. Retrieved from: <https://pypdfium2.readthedocs.io/en/stable/> [accessed: 10 September 2023].
- [20]. van Bevern, R., & Slugina, V. A. (2020). A historical note on the 3/2-approximation algorithm for the metric traveling salesman problem. *Historia Mathematica*, 53, 118-127.
- [21]. Selva Birunda, S., & Kanniga Devi, R. (2021). A review on word embedding techniques for text classification. *Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020*, 267-281.
- [22]. SentenceTransformers Documentation. (n.d.). *SentenceTransformers Documentation &mdash; Sentence-Transformers documentation*. Sbert.net. Retrieved from: <https://www.sbert.net/> [accessed: 11 September 2023].
- [23]. Voigt B. F. (1832) *The traveling salesman as he should be and what he has to do in order to receive orders and to be certain of successful success in his business*. Jenaische Allgemeine Literaturzeitung. Retrieved from: [https://zs.thulb.uni-jena.de/receive/jportal\\_jparticle\\_00248075](https://zs.thulb.uni-jena.de/receive/jportal_jparticle_00248075) [accessed: 13 September 2023].

- [24]. Seatgeek. (n.d.). *GitHub - seatgeek/thefuzz: Fuzzy String Matching in Python*. Github. Retrieved from: <https://github.com/seatgeek/thefuzz> [accessed: 14 September 2023].
- [25]. Chrismattmann. (n.d.). *GitHub - chrismattmann/tika-python: Tika-Python is a Python binding to the Apache Tika™ REST services allowing Tika to be called natively in the Python community*. Github. Retrieved from: <https://github.com/chrismattmann/tika-python> [accessed: 14 September 2023].
- [26]. Utrecht, U. (2016). *Innovations in Scholarly Communication*. Web.archive. Retrieved from: <https://web.archive.org/web/20161209150914/https://101innovations.wordpress.com/> [accessed: 16 September 2023].
- [27]. Zhelezniak, V., Savkov, A., Shen, A., & Hammerla, N. Y. (2019). Correlation coefficients and semantic textual similarity. *arXiv preprint arXiv:1905.07790*.
- [28]. Curcic, D. (n.d.). *Reading Speed Statistics & #8211*. WordsRated. Retrieved from: <https://wordsrated.com/reading-speed-statistics/#:~:text=The%20average%20oral%20reading%20speed,learning%20is%20100%2D200%20wpm.> [accessed: 19 September 2023].