applied sciences

MDPI

Article

Classification of Problem and Solution Strings in Scientific Texts: Evaluation of the Effectiveness of Machine Learning Classifiers and Deep Neural Networks

Rohit Bhuvaneshwar Mishra * and Hongbing Jiang

School of Management Engineering, Zhengzhou University, Zhengzhou 450001, China; jhbymx@foxmail.com
* Correspondence: rohit.bnmishra123@gmail.com

Abstract: One of the central aspects of science is systematic problem-solving. Therefore, problem and solution statements are an integral component of the scientific discourse. The scientific analysis would be more successful if the problem–solution claims in scientific texts were automatically classified. It would help in knowledge mining, idea generation, and information classification from scientific texts. It would also help to compare scientific papers and automatically generate review articles in a given field. However, computational research on problem–solution patterns has been scarce. The linguistic analysis, instructional-design research, theory, and empirical methods have not paid enough attention to the study of problem–solution patterns. This paper tries to solve this issue by applying the computational techniques of machine learning classifiers and neural networks to a set of features to intelligently classify a problem phrase from a non-problem phrase and a solution phrase from a non-solution phrase. Our analysis shows that deep learning networks outperform machine learning classifiers. Our best model was able to classify a problem phrase from a non-problem phrase with an accuracy of 90.0% and a solution phrase from a non-solution phrase with an accuracy of 86.0%.

Keywords: discourse analysis; problem–solution pattern; automatic classification; machine learning classifiers; deep neural networks

Citation: Mishra, R.B.; Jiang, H. Classification of Problem and Solution Strings in Scientific Texts: Evaluation of the Effectiveness of Machine Learning Classifiers and Deep Neural Networks. *Appl. Sci.* **2021**, *11*, 9997. https://doi.org/10.3390/app11219997

Academic Editor: Arturo Montejo-Ráez

Received: 20 September 2021
Accepted: 20 October 2021
Published: 26 October 2021

1. Introduction

Problem-solving is not a standardized exercise. Problems are different in domain, content, type, and linguistic properties [1–7]. In the most general sense, a problem is an unknown that arises from any situation where a person aims to satisfy a need or accomplish a goal. A problem comes into consideration when there is a "felt desire" to seek a solution to eliminate the problem or to find ways to solve the differences [8]. Problems traditionally have a problem area or domain, a problem category, a problem-solving approach, and a solution. The area or domain defines the problem constructs, laws, and fundamentals [9]. The problem category defines the type or nature of the problem [10,11]. The problem-solving strategy is then determined, and finally, we present the solution.

Mayer and Wittrock [10] categorized problem types as "poorly-defined", "well-defined", "routine," and "non-routine." Jonassen [11] classified well-structured problems from ill-structured problems by identifying individual variations in cognitive functioning. Smith [12] identified external variables from internal problem-solver characteristics, including domain and complexity. According to the researchers, there is increasing consensus that problems differ in content, structure, and method [13]. Problems also differ concerning their form, sophistication, and abstractness (domain specificity). While these three factors are similar, they are neither independent nor identical. Among these variables, there is enough independence to merit separate consideration.

Problem-solving is widely recognized as the most significant cognitive task [14–16]. However, the exploration of problem-solving techniques is severely limited in academic

papers. According to studies, to decipher an unknown phenomenon we apply problem-solving methods. Previous knowledge, imaginative guesswork, and logical inference are the tools to solve problems in day-to-day life [17–19]. However, systematic problem-solving can be understood only after formulating a problem that we want to solve. Therefore, discovering the problem to be solved is the first component in problem-solving practice [20,21]. After identifying the problem, a search for a suitable or ideal solution starts in the problem-solving process. In the final step, we apply algorithmic analysis.

In the past, various problem-solving techniques were developed. One of the most well-known problem-solving models, the IDEAL model [22], describes problem-solving as a "structured process of identifying potential problems, defining the problem, representing the problem, exploring potential solutions, implementing strategies to find the solution to the problem, and then reflecting on and evaluating the activities' outcomes." Although the IDEAL model recommends applying these approaches to various problems in different ways, there are no specific guidelines for how to do so. Another model by Gick [2], as shown in Figure 1, describes a problem-solving model that must involve the following three processes: creating a problem representation, looking for solutions, and implementing and tracking solutions. Similarly, Smith [12] tried to offer a uniform problem-solving theory, but it was not entirely successful.

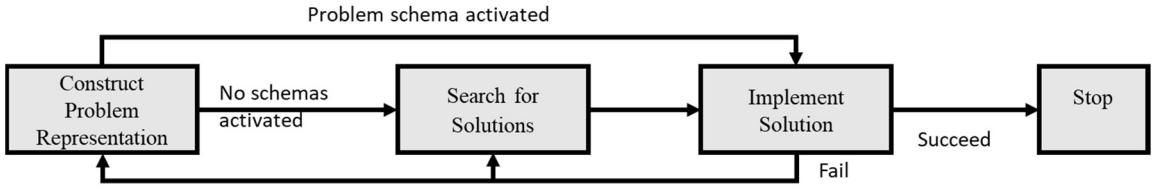

Figure 1. Problem-Solving Model (Gick, 1986).

Problem-solving methods develop knowledge. However, with the increase in content from journals, social media, business press releases, and scientific articles and discourse, knowledge generation will be highly challenging in the future [23–28]. Digitization initiatives in nearly all sectors would only increase the volume and variety of unstructured data. To apply outdated problem-solving models on this ever-increasing unstructured data is already out of scope. Hence, we need innovative and automatic techniques for the data and problems of the twenty-first century.

In the backdrop of all these developments, it is essential to look for automated techniques of problem–solution differentiation for information generation. With this paper, we aim to improve the linguistic and educational aspects of studying the problem–solution patterns. Furthermore, we seek to increase our understanding and associated assessments of problem–solution patterns. We look at real-world examples in published scientific literature to understand the complex problem–solution patterns. This analysis uses a collection of sentence features to apply various machine learning classifiers and deep learning models [29,30] in order to intelligently identify a string as a problem string or a solution string. We examine the parsed dependencies of our test word (problem, solution, or their synonyms) in the subject position in a sentence structure, then select its syntactic argument as a test phrase for our automatic classification. By extracting the problem and solution strings from scientific articles, our method can significantly improve knowledge mining. It will aid us in learning the gist of the papers and significantly improve information processing and visualization. Our methods can aid in the collection of scientific information and improve the efficiency of scientific searches. It will also aid in comparing related papers and, in the long run, lead to the automated production of field-specific review papers. In comparison with the previous studies, the work presented in this paper makes the following innovative and distinguishable key findings:

- Based on our review of the relevant literature, the proposed technique is the first to compare Machine Learning Classifiers and Deep Neural Networks for problem and solution string classification.
- Our approach is unique in applying both data iteration and cross-validation approach in assessing the effectiveness of Machine Learning Classifiers and Deep Neural Networks.
- Additionally, we perform parameter tuning to enhance the accuracy of our models.

2. Literature Review

In academic literature, the problem–solution pattern is pervasive [13,31,32]. Our writings, according to Jordan [33], represent our problem-solving, thought-action process. The research focused on linguistic and educational studies to develop a complete view of the complex problem–solution mechanism and to explain how we communicate these systems in the literature. The structure, domain specificity (abstractness), and complexity of the problems were defined by Jonassen [34]. He specified the continuum of outcomes for problem-solving learning. He also differentiated between well-structured problems and ill-structured problems in terms of the instructional design criteria.

Flowerdew [35] studied how particular keywords can be commonly utilized to discover specific aspects of the discourse structure. The keyword study was conducted on two corpus forms: the technical corpus and the student corpus. The phraseology of the keywords in both corpora was examined, and the examination revealed that the students' writing lacked various grammatical and lexical patterns used in expressing the problem–solution patterns and its elements. He discussed the pedagogical implications of these learning issues as well as the data-driven learning concepts.

In their analysis, the researchers [36] established adverbials that belong to the semantic category of "Result and Inference." Upton and Connor [37] used the corpus method to propose a text–linguistic approach that considers the unique characteristics of the genre-specific corpora. Charles [38] analyzed the problem–solution trend using discourse markers instead of a keyword-based approach. He analyzed adverbials such as "therefore, hence, and then" in two different corpora. He analyzed about 190,000 words from politics and about 300,000 words from materials science to check how they signal a pattern of problem-solving. According to the findings, combining corpus methods with discourse analysis would provide richer insights into academic discourse.

Technical texts have a four-part structure, according to Winter [39], which includes a situation, a problem, a solution, and an evaluation. This pattern is similar to Van Dijk's [40] pattern of "Introduction-Theory, Problem-Experiment-Comment, and Conclusion." SPRE, one of the most commonly used problem-solving patterns, is introduced by Hoey [41,42]. S stands for the situation; P for the query, purpose, problem, or the knowledge required; R for the reply, answer, response, the methods applied, and so on (depending on the case); and E for evaluation, in which a successful evaluation (the sequence comes to an end), or an unfavorable evaluation is given (the sequence is recycled).

In academic research texts, the assumption is that the problem identification or formulation comes before the solution [1,5,6,31,32]. In most scientific texts, the problem's condition is set at the start of the solution and remains unchanged. However, in some cases [43–46], the problem's initial specification is eventually reformulated or re-specified as the problem is solved.

The CARS ('Create a Research Space') model is one of the most well-known models of research article introductions [47]. The model's different movements cover similar ground to that of the SPRE model. The first move, 'Establishing a Territory', is similar to the Situation from SPRE; the second move, 'Establishing a Niche', is similar to the Problem in SPRE, and it identifies a knowledge void or an issue in the research field; and the third move, 'Occupying the Niche', enables the researcher to fill the void by announcing their findings, thus forming a Response step.

Based on the extensive literature review, in this work we address the challenge of defining patterns for problem-solving in the scientific text [48,49]. We chose the problem-solving model defined by Hoey [42] for our study. We aimed to classify strings of problems and solutions through various machine learning classifiers and deep learning networks [30,50,51]. We restricted ourselves to ML classifiers and deep neural networks, and we did not include non-standard techniques, such as morphological neural networks with dendritic processing and spiking neural networks for our study [52–57]. We were more concerned with evaluating sentence features in various syntactic variations [48,49] than trying all available classifiers. We theorized the multiple explications for our results and tried to ascertain the source of the failure/success of our models in increasing the accuracy.

3. Research Methodology

The methodology section consists of five sub-sections. In Section 3.1, we describe the SPRE problem–solution model with an example. In Section 3.2, we talk about the corpora and dataset preparation. Section 3.3 describes the wide range of syntactic variations, which we consider for the problem and solution strings. In Section 3.4, we discuss the training data preparation. Finally, in Section 3.5, we discuss our algorithm and model development.

3.1. Problem–Solution Pattern

A sentence's various features aid in the identification of problem–solution patterns. We use the SPRE model [42] for our study, which consists of Situation, Problem, Response, and Evaluation. We analyzed several sentences and found that the situation is an optional element in many sentences. As a result, we agreed to focus our model-building efforts on Problem, Response, and Evaluation. It is an excellent place to start describing the pattern with the problem because it is unusual for an author to present a problem and then leave the problem without any answer. There are exceptional cases, as when authors present a problem and then leave it for future research; however, in principle, the problem variable should have enough information for our model to work. The second knowledge parameter we are searching for is Response and Evaluation in the same sentence. We consider providing sufficient information to help our automatic classifier recognize the pattern externally for Response and Evaluation. Let us go through each part of the SPRE model with a real-life illustration. It will assist us in effectively understanding the functionality of the SPRE model.

Situation: Background information on situations; details about the persons, topic, case, or location involved in the debate. Example: John was doing experiments in the school laboratory.

Problem: An aspect of a situation that requires a solution; a need, a dilemma, a puzzle, or an obstacle that is being discussed; flaws in the current situation. Example: John discovered an error in the experiment code.

Response: Problem–solution(s); discussion of a way(s) to deal with or solve the problem. Example: John modified and executed the code.

Evaluation: A determination of the efficacy of the proposed solution(s); if several solutions are available, which one is the best. Example: The code ran successfully, and John got the desired output.

The flowchart explaining the flow of processes in the SPRE model is shown in Figure 2 below. As shown, the process ends if the evaluation is positive, or else the whole process is recycled again.

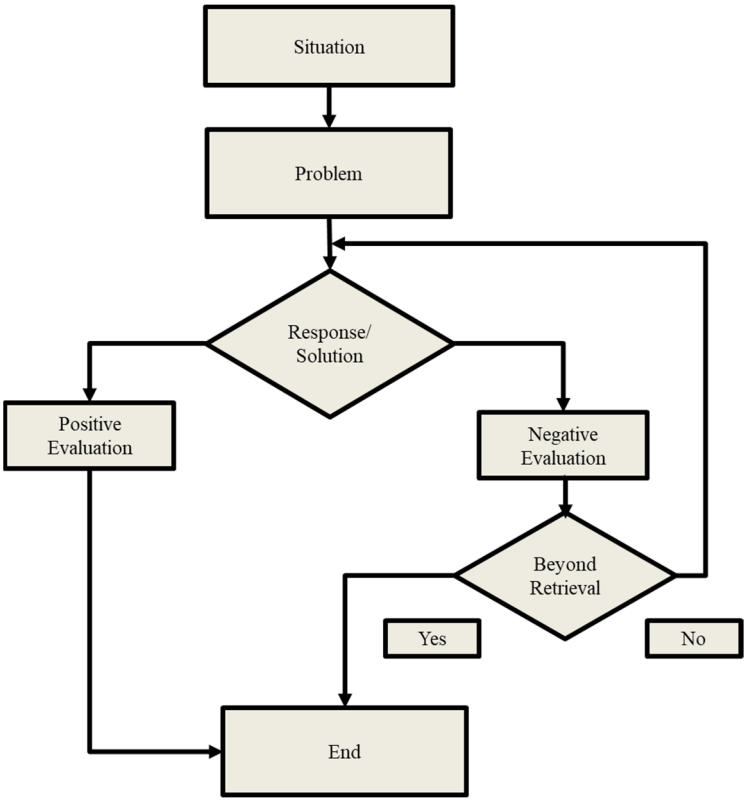

Figure 2. SPRE Model given by Hoey.

The word "problem" has various meanings. For example, the word "problem" can mean that something needs to be accomplished (a task), and another meaning is that something is troublesome, negative, and needs a solution. We are restricted to the use of the problem only for the second scenario. The use of the problem-defining task is beyond the scope of this research.

3.2. Corpora

There are numerous public datasets available for citation and text analysis. The datasets with citation data are used for cluster analysis with network and edge information, analyzing influence in the citation network, recognizing the most impactful papers, and for topic modelling analysis [58–60]. Text analysis datasets on the other hand are used to implement different techniques for practical problems in artificial intelligence, computational science, scientific data analysis, and other fields [61–64].

We use the dataset provided by [48]. This dataset is prepared from the March 2016 ACL anthology corpus [65]. The data provided by [48] checks the parsed dependencies and the searches for "problem/solution" or their synonyms in the subject position. The data include synonyms of problem and solution words in order to maximize the search. The dataset manually selects 28 synonyms for the "problem" word and 19 synonyms for the "solution" word. The synonyms are manually selected from the semantically closest words, trained using Word2Vec on PubMed articles [66,67]. The words chosen for the problem candidate phrase extraction, as from [48], are "*Bottleneck, Caveat, Challenge, Complication, Conundrum, Difficulty, Dilemma, Disadvantage, Drawback, Fault, Flaw, Impediment, Issue, Limitation, Mistake, Obstacle, Pitfall, Problem, Quandary, Riddle, Shortcoming, Struggle, Subproblem, Threat, Tragedy, Trouble, Uncertainty*, and *Weakness*." Similarly, the words chosen for the solution candidate extraction are "*Alternative, Answer, Answers, Approach, Approaches, Idea, Method, Methodology, Proposal, Remedy, Solution, Suggestion, Scheme, Step, Strategy, Technique, Task, Way*, and *Workaround*."

We chose exact and straightforward strings in suggesting a positive or negative condition for the problem/solution argument. A problem string denotes an unexplained event, a research query, or an item that has failed to meet its specified requirements. A solution argument, on the other hand, is a successful answer to the evaluation. However, we chose the phrase so that its status as a problem or solution phrase was not revealed lexically. This additional check was used to reject inputs to the classifier that were overly evident. For example, we rejected the sentence if it contained the words "troublesome" or "shortcomings" because it would be far too easy for the classifier to pick up on such signals. For both the problems and the solutions, there are corresponding negative examples (non-problem and non-solution). The negative strings were chosen to imitate as closely as possible the apparent characteristics of the positive examples, hence offering no extra information for differentiation on the surface. Negative examples were selected from a collection of sentences which had a similar length and a similar POS (part of speech) to that of the test problem and the test solution strings. Negative examples were carefully sampled to match the positive examples' POS pattern and sentence length. Sentences that lacked words for problems/solutions were flagged, and one syntactic subtree within them was chosen at random to create negative samples. Finally, 500 problem strings, 500 non-problem strings, 500 solution strings, and 500 non-solution strings made up the data sample. These 2000 strings were used to train and test our machine learning and deep learning models.

3.3. Selection of Problem and Solution Strings

Our model aimed to identify strings of problems and solutions that appear in a wide range of syntactic variations. Furthermore, we wanted to test the problem and solution scenarios even when the problem or solution status was not explicitly specified. Moreover, in certain instances, two or more sentences are used to explain the problem or the solution. These kinds of sentences are beyond the research scope of the present study. Here, we just looked at one-sentence-long problem and solution phrases.

3.4. Creating the Training Sample

The dataset was limited, so we applied two methods for preparing the training and testing data. In the first method, we randomly divided the test and train data and performed various iterations. In each iteration, the train dataset and the test dataset were different. However, in each iteration, the percentage of train data was 67%, and the test data was 33% [68–70]. This method achieved a better estimate of accuracy (average accuracy), even on this limited dataset. In the second method, we applied multifold cross-validation to train and test the data [71,72]. We applied 5-fold and 10-fold validation, and we realized that the accuracy changed. We have mentioned the accuracies obtained from the best scenarios in this paper.

3.5. Method and Model Development

The dataset [48] listed a collection of features without considering the context of the phrase. We used traditional machine learning classifiers, and deep learning networks to test features, such as BOW (bag of words), transitivity, modality, polarity, syntax, doc2vec, word2vec, and word2vec smoothed, as discussed by Heffernan and Teufel [48]. We improved the findings with a comparison with those from [48]. We theorized different explanations for our findings and attempted to determine the cause of failure/success in terms of the classification-accuracy improvement.

The semantic disambiguation ability of the problem/solution definition was tested using traditional machine learning classifiers and deep learning networks. The problem/solution keywords enabled template-based search. The syntactic complement of the keyword in the subject position was used. To ensure that there were no "give-away" phrases (phrases that provide additional information) inside the phrases, we modeled only the keyword from the sentence and excluded the rest of the sentence. This was conducted

to aid in the generalization of our models for real-world problem/solution differentiation tasks. The following is the computational algorithm:

1. Obtain the features from the dataset. Start with baseline feature (bag of words).
2. Method 1: Divide the data into 67% training and 33% testing. Obtain the accuracy for differentiating the problem strings from the non-problem strings. In the next iteration, different train and test data with the same proportion are selected, and the classification is performed again. The absolute accuracy is the average of the accuracies in all the iterations. We apply the same steps for differentiating the solution strings from the non-solution strings.
3. Method 2: Perform multifold cross-validation of data. Obtain the accuracy for differentiating the problem strings from the non-problem strings. We apply the same steps for differentiating the solution strings from the non-solution strings.
4. Add one feature extra on top of the BOW. Apply method 1/method 2. Repeat.
5. Compare the results after applying the machine learning classifier/deep learning model on all the available features.
6. Improve the best model by hyperparameter tuning.

4. Evaluation

We evaluated the machine learning classifiers and the deep neural network models separately for methods one and two. The results varied for the different methods. For example, the accuracy values were high for both the machine learning classifiers and the deep neural networks when we used the data iteration method, while accuracy decreased when we used the multifold validation.

4.1. Evaluation of Machine Learning Classifiers

Machine learning is all about discovering patterns and applying those to new datasets. To evaluate an algorithm, we can divide the dataset into two parts: train and test. In the first method, we divided the data into 67% training and 33% testing. We trained the data in 5–10 iterations and applied ML classifiers, such as logistic regression (LR), multilayer perceptron (MLP), support vector classifier (SVC), random forest (RF), Naive Bayes classifier (NV), AdaBoost (AB), and gradient boosting (GB). We used the scikit-learn machine learning library for our experiments [73]. It is a free, open-source, and trendy machine learning package for Python [74].

We used a bag of words (BOW) for the first feature, which is also our baseline feature. We applied the ML classifiers and checked the accuracy. In the next step, we added the transitivity feature on top of the baseline feature. Similarly, we kept adding extra features in the given order: bag of words (BOW), transitivity, modality, polarity, syntax, doc2vec, word2vec, and word2vec smoothed [48]. We calculated the disambiguation capacity of the problem/solution through these ML classifiers. The accuracy of each model for both scenarios is shown in Tables 1 and 2 below.

Table 1. Classification of problems from non-problems using ML classifiers by training data iteration.

Feature Set	LR	MLP	SVC	RF	NB	AB	GB
Baseline	0.69	0.70	0.66	0.68	0.66	0.68	0.67
+transitivity	0.69	0.69	0.64	0.68	0.65	0.68	0.66
+modality	0.69	0.68	0.66	0.68	0.65	0.67	0.67
+polarity	0.70	0.70	0.65	0.69	0.65	0.67	0.67
+syntax	0.76	0.75	0.75	0.77	0.65	0.73	0.73
+doc2vec	0.77	0.77	0.76	0.76	0.66	0.75	0.68
+word2vec	0.73	0.69	0.61	0.70	0.66	0.72	0.72
+word2vecSmoothed	0.76	0.74	0.75	0.77	0.65	0.73	0.72

Table 2. Classification of solutions from non-solutions using ML classifiers by training data iteration.

Feature Set	LR	MLP	SVC	RF	NB	AB	GB
Baseline	0.71	0.70	0.65	0.65	0.67	0.67	0.66
+transitivity	0.70	0.70	0.68	0.69	0.68	0.66	0.68
+modality	0.70	0.70	0.68	0.67	0.67	0.66	0.68
+polarity	0.71	0.70	0.68	0.69	0.68	0.67	0.69
+syntax	0.74	0.73	0.74	0.76	0.67	0.71	0.74
+doc2vec	0.76	0.75	0.75	0.75	0.68	0.76	0.64
+word2vec	0.76	0.73	0.71	0.79	0.68	0.80	0.73
+word2vecSmoothed	0.77	0.72	0.70	0.78	0.66	0.80	0.71

For method one, the three most effective ML classifiers for the classification of the problems from the non-problems by training data iteration were **logistic regression (LR), multilayer perceptron (MLP),** and **random forest (RF),** giving an accuracy of **77%**.

For the classification of the solutions from the non-solutions, the most effective ML model in method one was **AdaBoost**, giving an accuracy of **80%**.

In the second method, we applied a 5–10-fold cross-validation for the same ML models. In a similar way to method 1, we started with the BOW, added the other features in the same order as before, and calculated the accuracy. We used the same scikit-learn machine learning library for method two as well. The results show that for most cases, the accuracy in method 2 decreases when compared to that of method 1. Moreover, the accuracy of the models increases from 5-fold validation to 10-fold validation. Hence, we present the results from the 10-fold validation. The accuracy of each model for method two is shown in Tables 3 and 4 below.

Table 3. Classification of problems from non-problems using ML classifiers by 10-fold cross validation.

Feature Set	LR	MLP	SVC	RF	NB	AB	GB
Baseline	0.71	0.71	0.68	0.71	0.66	0.70	0.71
+transitivity	0.70	0.71	0.67	0.70	0.66	0.70	0.70
+modality	0.70	0.71	0.67	0.69	0.66	0.69	0.67
+polarity	0.71	0.71	0.67	0.70	0.66	0.71	0.66
+syntax	0.73	0.72	0.69	0.71	0.66	0.70	0.68
+doc2vec	0.74	0.73	0.70	0.76	0.66	0.73	0.73
+word2vec	0.74	0.67	0.61	0.67	0.66	0.70	0.67
+word2vecSmoothed	0.73	0.74	0.69	0.71	0.66	0.70	0.67

Table 4. Classification of solutions from non-solutions using ML classifiers by 10-fold cross validation.

Feature Set	LR	MLP	SVC	RF	NB	AB	GB
Baseline	0.72	0.71	0.68	0.69	0.69	0.68	0.70
+transitivity	0.72	0.71	0.68	0.70	0.69	0.68	0.70
+modality	0.72	0.69	0.69	0.70	0.69	0.69	0.70
+polarity	0.72	0.70	0.69	0.72	0.69	0.69	0.70
+syntax	0.71	0.71	0.68	0.70	0.69	0.69	0.68
+doc2vec	0.73	0.72	0.69	0.75	0.69	0.75	0.67
+word2vec	0.75	0.71	0.72	0.79	0.69	0.79	0.75
+word2vecSmoothed	0.75	0.70	0.72	0.79	0.69	0.81	0.75

For method two, the most effective ML model for classifying problems from non-problems was **random forest (RF)**, giving an accuracy of **76%**.

For the classification of solutions from non-solutions, the most effective ML model for method two was **AdaBoost**, giving an accuracy of **81%**.

4.2. Evaluation of Deep Learning Models

Although the simple machine classifiers performed well as per our experimental settings, they still can be improved. If an AI algorithm returned an incorrect prediction

for a few scenarios, we had to interfere. A deep learning model can efficiently improve our results because a deep learning model may use neural networks to figure out on its own whether a prediction is correct or not [75,76]. Hence, we compared the classifications between the ML classifiers and the deep learning models. A deep learning model's reasoning structure is close to how a person can conclude. This is achieved using a layered system of algorithms called an artificial neural network. An artificial neural network's architecture was inspired by the human brain's neural network [77,78], contributing to a learning mechanism that is far more capable than the machine learning classifiers discussed earlier.

For the first method, we applied three deep learning models to train the dataset. We divided the data into 67% training and 33% testing. We trained the data in 5–10 iterations and applied the deep learning models: Long short-term memory (LSTM), neural network (NN), and convolutional neural network (CNN) [79,80]. As with the ML classifiers, we started with BOW, added the other features in the same order as before, and calculated the accuracy. We used Keras, an open-source library, for the application of the deep learning models [74,81]. It offers a python interface for artificial neural networks and also serves as a frontend for TensorFlow. The accuracy of each model for method 1 (data iteration) is shown in Tables 5 and 6 below.

Table 5. Classification of problems from non-problems using deep learning networks by training data iteration.

Feature Set	LSTM	NN	CNN
Baseline	0.49	0.70	0.82
+transitivity	0.49	0.69	0.82
+modality	0.50	0.71	0.83
+polarity	0.50	0.69	0.83
+syntax	0.50	0.75	0.86
+doc2vec	0.47	0.76	0.84
+word2vec	0.50	0.71	0.82
+word2vecSmoothed	0.50	0.75	0.82

Table 6. Classification of solutions from non-solutions using deep learning networks by training data iteration.

Feature Set	LSTM	NN	CNN
Baseline	0.50	0.72	0.77
+transitivity	0.48	0.71	0.85
+modality	0.49	0.70	0.82
+polarity	0.48	0.71	0.78
+syntax	0.52	0.75	0.81
+doc2vec	0.50	0.77	0.83
+word2vec	0.54	0.71	0.83
+word2vecSmoothed	0.57	0.76	0.83

For method one, the most effective deep learning model for the classification of the problems from the non-problems by training data iteration is **CNN**, giving an accuracy of **86%**.

For the classification of solutions from non-solutions, the most effective deep learning model in method one is again **CNN**, giving an accuracy of **85%**.

For the second method, we applied 5–10-fold validation cross-validation for the same deep learning models. The results show that for most cases, the accuracy in method 2 decreases when compared to method 1. Moreover, the accuracy of the models decreases from 5-fold validation to 10-fold validation. Hence, we present the results from the 5-fold validation. In a similar way to method 1, we started with BOW, added the other features in the same order as before, and calculated the accuracy. For method two, we also used the Keras library for the application of the artificial neural networks.

The accuracy of each model for method two is shown in Tables 7 and 8 below.

Table 7. Classification of problems from non-problems using deep learning networks by 5-fold cross validation.

Feature Set	LSTM	NN	CNN
Baseline	0.53	0.63	0.61
+transitivity	0.53	0.65	0.61
+modality	0.47	0.65	0.65
+polarity	0.55	0.66	0.57
+syntax	0.50	0.67	0.57
+doc2vec	0.50	0.70	0.55
+word2vec	0.48	0.61	0.59
+word2vecSmoothed	0.52	0.67	0.59

Table 8. Classification of solutions from non-solutions using deep learning networks by 5-fold cross-validation.

Feature Set	LSTM	NN	CNN
Baseline	0.50	0.59	0.64
+transitivity	0.51	0.56	0.63
+modality	0.38	0.59	0.64
+polarity	0.51	0.58	0.65
+syntax	0.37	0.57	0.61
+doc2vec	0.50	0.62	0.56
+word2vec	0.63	0.64	0.63
+word2vecSmoothed	0.58	0.64	0.68

For method two, the most effective deep learning model for the classification of the problems from the non-problems is **NN**, with an accuracy of **70%**.

For the classification of the solutions from the non-solutions, the most effective model in method two is **CNN**, with an accuracy of **68%**.

4.3. Hyperparameter Tuning

To improve the accuracy further, we performed hyperparameter tuning for the best neural network model. Hyperparameter tuning methods include scanning through the space of possible hyperparameter values to find possible model architecture candidates [82,83]. Finding the optimal values is also referred to as "searching" the hyperparameter space. If the learning rate is too low for a model, the model will miss important data patterns. If the model is too heavy, it can have collisions leading to decreased accuracy. Hence, choosing suitable hyperparameters is key to the success of our neural network architecture. Even for small models, the number of hyperparameters may be significant. It is quite an intricate task to tune the hyperparameters but doing so improves the efficiency of our model significantly. Here, in Table 9, we can see that our model improves on performing the hyperparameter tuning. We applied hyperparameter tuning to CNN networks in method one as it was the best performing model and method.

Table 9. Hyperparameter tuning the CNN model for the data iteration method.

Feature Set	Problem Phrase	Solution Phrase
Baseline	0.85	0.84
+transitivity	0.90	0.85
+modality	0.87	0.81
+polarity	0.86	0.85
+syntax	0.88	0.86
+doc2vec	0.85	0.84
+word2vec	0.85	0.85
+word2vecSmoothed	0.83	0.84

As shown in the above results, hyperparameter tuning impacts the classification results immensely. The accuracies improved for most of the feature list when we tuned our models. For our analysis, the accuracy improved to **90%** for classifying the problems from the non-problems and **86%** for classifying the solutions from the non-solutions.

To get an idea of the impact of hyperparameter tuning, we show the results for the top classifier (after hyperparameter tuning) in the figures below. First, we present the best CNN model results for the classification of the problem from the non-problem string, as shown below in Figure 3. The top 10 values are in Table 10, and the graph is in Figure 3 to help us understand the effect of tuning on a model.

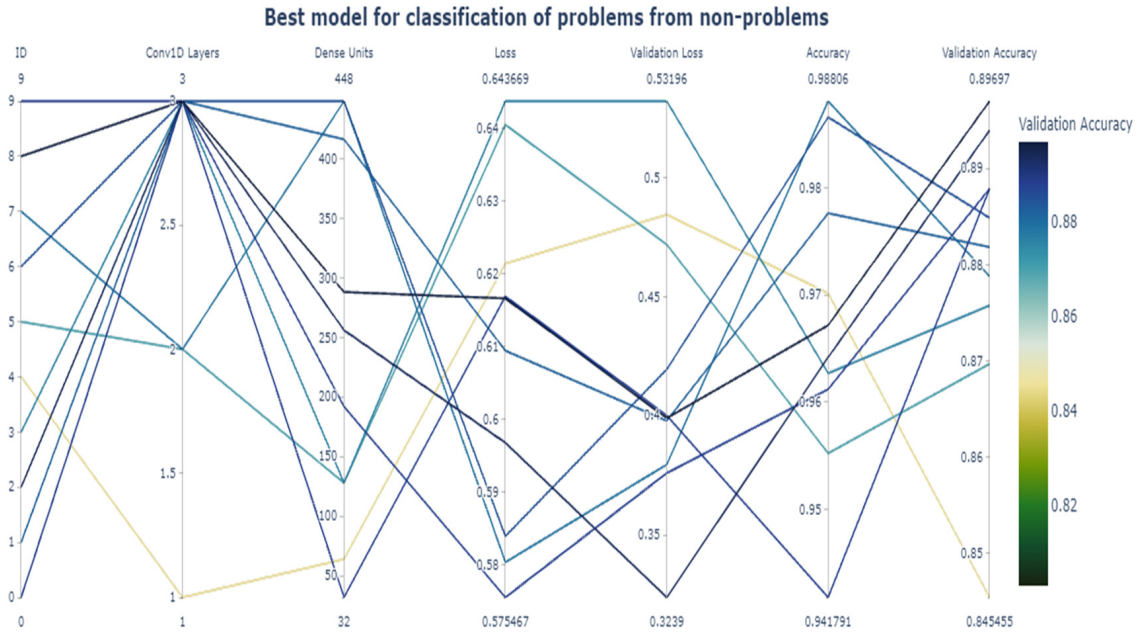

Figure 3. Hyperparameter tuning results for the classification of problems from non-problems.

Table 10. Hyperparameter tuning results for the classification of problems from non-problems.

Uid	Num_Conv1D	Units	Accuracy
0	3	192	0.89
1	3	416	0.88
2	3	256	0.89
3	3	128	0.88
4	1	64	0.85
5	2	128	0.87
6	3	448	0.88
7	2	448	0.88
8	3	288	0.90
9	3	32	0.89

The best accuracy of **90%** occurs for **three one-dimensional convolutional networks** and **288 units**. The number of units is an essential hyperparameter. The neural networks are universal function approximators, and they need enough 'power' to learn for the prediction task. The number of units is the critical indicator of the learning ability of the model. The simple role can require fewer units. The greater the number of units, the more complex the role of the parameter tuning. The number of units used for the best model indicates that our model is highly complex. Hence, it achieves good accuracy.

In a similar way to the tuning of the problem phrase classifier, we present the hyperparameter tuning results for the top 10 values of the best CNN model for the classification of the solutions from the non-solutions in Table 11 and Figure 4.

Table 11. Hyperparameter tuning results for the classification of solutions from non-solutions.

Uid	Num_Conv1D	Units	Accuracy
0	2	192	0.86
1	1	512	0.83
2	2	64	0.82
3	1	480	0.81
4	2	416	0.81
5	1	256	0.77
6	2	320	0.82
7	1	192	0.78
8	4	288	0.82
9	3	480	0.82

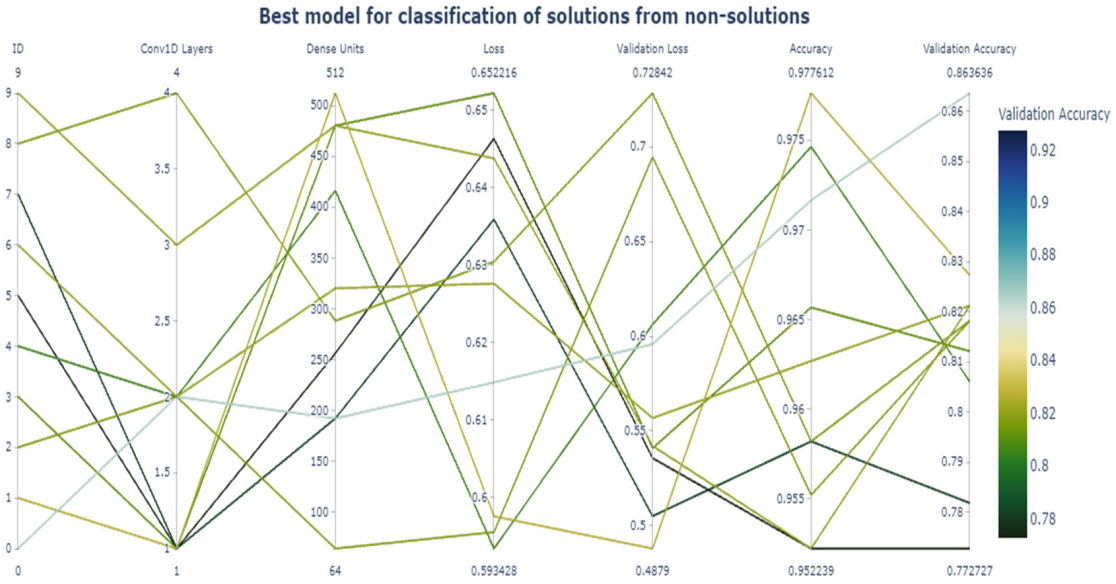

Figure 4. Hyperparameter tuning results for classification of solutions from non-solutions.

The best accuracy of **86%** is achieved for **two one-dimensional convolutional networks** and **192 units**.

4.4. Final Results

We report the results for the best model, method, and hyperparameter tuning parameters in Table 12 below.

Table 12. Best performing CNN model.

Strings	Best Accuracy	Model	Method	Hyperparameters
Problem Strings	90.0%	CNN	Method 1	Number of Conv1D: 3 Units: 288
Solution Strings	86.0%	CNN	Method 1	Number of Conv1D: 2 Units: 192

5. Discussion

The research discussed in this paper focuses on comparing machine learning classifiers with deep learning models that consist of an empirical assessment. In our paper, the theory, arguments, and solution methods proposed were empirically tested. Even though the dataset size was small, the high accuracy of the models indicates that our models were highly efficient. We addressed our views on how to improve the accuracy of classification using different methods. We performed hyperparameter tuning to improve the accuracy for the used models. We provided an overview of some of the potential ways to increase accuracy. We did our best to minimize the challenges to the problems of validity. We extracted various characteristics from the sentence, but we did not conduct any relation or meta-data analysis that could be regarded as external variables.

We used a dataset that is available in the public domain. The dataset was annotated and checked by more than one person [48], ensuring that the annotation of the dataset was of good quality and that no errors in the annotation and calculation were present. Moreover, we carried out the experiments more than once to ensure that there were no mistakes when performing the experiments and that our findings were replicable.

We would also like to highlight the challenges we faced while performing the classifications. The first hurdle was locating the correct dataset for our study. Then, it was challenging to apply various ML and neural networks to the feature list present in the dataset. The next issue we faced was identifying various permutations and combinations of the models and the data training approach discussed in the paper. To ensure that our algorithms were robust, we tried various approaches and models, but mentioned only the significant ones in this paper. This was the most challenging part of our analysis.

Next, we would talk about the future scope of our experiments. In future research, we would like to have a few valuable additions. Firstly, our corpus consisted entirely of scientific articles from computer linguistic research, constituting a particular subset of textual information. We want to test our model on scientific articles from different countries and domains of science. We used an existing dataset; hence, we could not change the existing parameters, such as the feature set, the synonyms of problem words, the synonyms of solution words and the number of sentences in the corpus. In the future, we would like to prepare our corpus. It would offer us more flexibility in the way we approach our research question. We would like to make a corpus that contains articles from different domains. It would help us to identify different characteristics of problem–solution patterns specific to a research domain. We would like to apply the model to a corpus containing research articles from multiple research domains and to try to find a generalized model for problem–solution classification. Once we achieve a generalized model for problem–solution classification, the next stage of our research would be linking the problem string with its corresponding solution string. This would help in knowledge mining from research articles which would eventually lead to idea discovery from the scientific texts. This would improve the existing mechanisms of text summarization and text abstraction. It would also help review articles and help in the classification of research domains based on the texts from the research papers.

6. Conclusions

We presented a technique focused on machine learning classifiers, such as logistic regression (LR), multilayer perceptron (MLP), support vector classifier (SVC), random forest (RF), Naive Bayes classifier (NV), AdaBoost (AB), gradient boosting (GB), and deep

learning neural networks, such as long short-term memory (LSTM), neural network (NN), and convolutional neural network (CNN). We constructed various classification models in which we used sentence features to distinguish the problem from the non-problem strings and the solution strings from the non-solution strings. For the distinction of the solution from the non-solution, our best model was able to achieve an 86% accuracy. Likewise, an accuracy of 90% was obtained by the best model for differentiating the problems from the non-problems. In both cases, CNN worked the best, and we performed hyper-parameter tuning to achieve the best CNN outcome.

Author Contributions: Conceptualization, R.B.M. and H.J.; formal analysis and writing—review and editing, R.B.M.; data curation and writing—original draft preparation, R.B.M.; supervision, H.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, Project no. 71801195.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the three anonymous reviewers for providing their constructive comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Albay, E.M. Analyzing the Effects of the Problem Solving Approach to the Performance and Attitude of First Year University Students. *Soc. Sci. Humanit. Open* **2019**, *1*, 100006. [CrossRef]
2. Gick, M.L. Problem-Solving Strategies. *Educ. Psychol.* **1986**, *21*, 99–120. [CrossRef]
3. Hembree, R. Experiments and Relational Studies in Problem Solving: A Meta-Analysis. *J. Res. Math. Educ.* **1992**, *23*, 242–273. [CrossRef]
4. Hidayati, N.; Permana, D. Assessment of Problem Solving Abilities and Student Learning Activities Based on Learning Tools: The Basis of Problem Based Learning Development. *Int. J. Sci. Technol. Res.* **2019**, *8*, 453–456.
5. Molnár, G.; Csapó, B. The Efficacy and Development of Students' Problem-Solving Strategies during Compulsory Schooling: Logfile Analyses. *Front. Psychol.* **2018**, *9*, 302. [CrossRef] [PubMed]
6. Priemer, B.; Eilerts, K.; Filler, A.; Pinkwart, N.; Rösken-Winter, B.; Tiemann, R.; Belzen, A.U.Z. A Framework to Foster Problem-Solving in STEM and Computing Education. *Res. Sci. Technol. Educ.* **2020**, *38*, 105–130. [CrossRef]
7. Rausch, A.; Schley, T.; Warwas, J. Problem Solving in Everyday Office Work—A Diary Study on Differences between Experts and Novices. *Int. J. Lifelong Educ.* **2015**, *34*, 448–467. [CrossRef]
8. Sinnott, J.D. *Everyday Problem Solving: Theory and Applications*; Praeger: New York, NY, USA, 1989; ISBN 978-0-275-92691-5.
9. Kim, D.-K.; El Khawand, C. An Approach to Precisely Specifying the Problem Domain of Design Patterns. *J. Vis. Lang. Comput.* **2007**, *18*, 560–591. [CrossRef]
10. Mayer, R.E.; Wittrock, M.C. Problem-solving transfer. In *Handbook of Educational Psychology*; Prentice Hall International: London, UK, 1996; pp. 47–62, ISBN 978-0-02-897089-9.
11. Jonassen, D.H. Instructional Design Models for Well-Structured and III-Structured Problem-Solving Learning Outcomes. *ETRD* **1997**, *45*, 65–94. [CrossRef]
12. Smith, M.U. *Toward a Unified Theory of Problem Solving: Views from the Content Domains*; Erlbaum: Hillsdale, MI, USA, 1991; ISBN 0-8058-0510-9.
13. Hoey, M. Problem-Solution Patterns. *Encycl. Lang. Linguist.* **2006**, *1*, 112–115. [CrossRef]
14. Delahunty, T.; Seery, N.; Lynch, R. Exploring Problem Conceptualization and Performance in STEM Problem Solving Contexts. *Instr. Sci.* **2020**, *48*, 395–425. [CrossRef]
15. Greiff, S.; Holt, D.; Funke, J. Perspectives on Problem Solving in Cognitive Research and Educational Assessment: Analytical, Interactive, and Collaborative Problem Solving. *J. Probl. Solving* **2013**, *5*, 71–91. [CrossRef]
16. Huitt, W.G. Problem Solving and Decision Making: Consideration of Individual Differences Using the Myers-Briggs Type Indicator. *J. Psychol. Type* **1992**, *24*, 33–44.
17. Bronkhorst, H.; Roorda, G.; Suhre, C.; Goedhart, M. Logical Reasoning in Formal and Everyday Reasoning Tasks. *Int. J. Sci. Math. Educ.* **2020**, *18*, 1673–1694. [CrossRef]
18. Galotti, K.M. Approaches to Studying Formal and Everyday Reasoning. *Psychol. Bull.* **1989**, *105*, 331–351. [CrossRef]
19. Hintikka, J. Is Logic the Key to All Good Reasoning? *Argumentation* **2001**, *15*, 35–57. [CrossRef]

20. Christ, T.J.; Christ, T.J. *Best Practices in Problem Analysis*; National Association of School Psychologists: Bethesda, MD, USA, 2008.
21. Narula, S.C. Systematic Ways to Identify Research Problems in Statistics. *Int. Stat. Rev. Rev. Int. De Stat.* **1974**, *42*, 205–209. [CrossRef]
22. Bransford, J.; Stein, B.S. *The Ideal Problem Solver. A Guide for Improving Thinking, Learning, and Creativity*; Series of books in psychology; W. H. Freeman and Company: New York, NY, USA, 1984; ISBN 978-0-7167-1669-3.
23. Farrington, T.; Alizadeh, A. On the Impact of Digitalization on R&D: R&D Practitioners Reflect on the Range and Type of Digitalization's Likely Effects on R&D Management. *Res. Technol. Manag.* **2017**, *60*, 24–30. [CrossRef]
24. Hausberg, J.P.; Liere-Netheler, K.; Packmohr, S.; Pakura, S.; Vogelsang, K. Research Streams on Digital Transformation from a Holistic Business Perspective: A Systematic Literature Review and Citation Network Analysis. *J. Bus. Econ.* **2019**, *89*, 931–963. [CrossRef]
25. Nadkarni, S.; Prügl, R. Digital Transformation: A Review, Synthesis and Opportunities for Future Research. *Manag Rev. Q* **2021**, *71*, 233–341. [CrossRef]
26. Nelson, G.; Ellis, S. The History and Impact of Digitization and Digital Data Mobilization on Biodiversity Research. *Philos. Trans. R. Soc. B Biol. Sci.* **2019**, *374*, 20170391. [CrossRef]
27. Reis, J.; Amorim, M.; Melão, N.; Cohen, Y.; Rodrigues, M. Digitalization: A Literature Review and Research Agenda. In *Proceedings on 25th International Joint Conference on Industrial Engineering and Operations Management—IJCIEOM*; Anisic, Z., Lalic, B., Gracanin, D., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 443–456.
28. Reis, J.; Amorim, M.; Melão, N.; Matos, P. Digital Transformation: A Literature Review and Guidelines for Future Research. In *Trends and Advances in Information Systems and Technologies*; Rocha, Á., Adeli, H., Reis, L.P., Costanzo, S., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 411–421.
29. Bui, D.T.; Tsangaratos, P.; Nguyen, V.-T.; Liem, N.V.; Trinh, P.T. Comparing the Prediction Performance of a Deep Learning Neural Network Model with Conventional Machine Learning Models in Landslide Susceptibility Assessment. *CATENA* **2020**, *188*, 104426. [CrossRef]
30. Nanehkaran, Y.A.; Zhang, D.; Salimi, S.; Chen, J.; Tian, Y.; Al-Nabhan, N. Analysis and Comparison of Machine Learning Classifiers and Deep Neural Networks Techniques for Recognition of Farsi Handwritten Digits. *J. Supercomput.* **2021**, *77*, 3193–3222. [CrossRef]
31. Khaw, L.L. Problem-Solution Patterns in the Introductions of Chemical Engineering Research Articles: Pedagogical Insights. In Proceedings of the 2020 IEEE Global Engineering Education Conference (EDUCON), Porto, Portugal, 27–30 April 2020; pp. 78–84.
32. Khaw, L.L.; Tan, W.W. Creating Contexts in Engineering Research Writing Using a Problem-Solution-Based Writing Model: Experience of Ph.D. Students. *IEEE Trans. Prof. Commun.* **2020**, *63*, 155–171. [CrossRef]
33. Jordan, M.P. Short Texts to Explain Problem–Solution Structures–and Vice Versa. *Instr. Sci.* **1980**, *9*, 221–252. [CrossRef]
34. Jonassen, D.H. Toward a Design Theory of Problem Solving. *ETRD* **2000**, *48*, 63–85. [CrossRef]
35. Flowerdew, L. *Corpus-Based Analyses of the Problem–Solution Pattern*; John Benjamins Publishing Company: Amsterdam, The Netherlands, 2008; ISBN 978-90-272-2303-6.
36. Biber, P.D.; Finegan, E.; Johansson, S.; Conrad, D.S.; Leech, G. *Longman Grammar Spoken & Written English Cased*; Longman: Harlow, UK, 1999; ISBN 978-0-582-23725-4.
37. Upton, T.; Connor, U. Using Computerized Corpus Analysis to Investigate the Textlinguistic Discourse Moves of a Genre. *Engl. Specif. Purp.* **2001**, *20*, 313–329. [CrossRef]
38. Charles, M. Adverbials of Result: Phraseology and Functions in the Problem–Solution Pattern. *J. Engl. Acad. Purp.* **2011**, *10*, 47–60. [CrossRef]
39. Winter, E.O. A Clause-Relational Approach to English Texts: A Study of Some Predictive Lexical Items in Written Discourse. *Instr. Sci.* **1977**, *6*, 1–92. [CrossRef]
40. van Dijk, T.A. *Text and Context: Explorations in the Semantics and Pragmatics of Discourse*; Longman: London, UK, 1977; ISBN 978-0-582-55085-8.
41. Hoey, M. On the Surface of Discourse. *Language* **1983**, *61*, 734–735. [CrossRef]
42. Hoey, M. *Textual Interaction: An. Introduction to Written Discourse Analysis*, 1st ed.; Routledge: London, UK; New York, NY, USA, 2000; ISBN 978-0-415-23169-5.
43. Kurup, U.; Bignoli, P.G.; Scally, J.R.; Cassimatis, N.L. An Architectural Framework for Complex Cognition. *Cogn. Syst. Res.* **2011**, *12*, 281–292. [CrossRef]
44. Schön, D.A. *The Reflective Practitioner: How Professionals Think in Action*; Routledge: London, UK, 1992; ISBN 978-1-85742-319-8.
45. Smith, R.P.; Eppinger, S.D. Identifying Controlling Features of Engineering Design Iteration. *Manag. Sci.* **1997**, *43*, 276–293. [CrossRef]
46. Thomke, S.; Fujimoto, T. The Effect of "Front-Loading" Problem-Solving on Product Development Performance. *J. Prod. Innov. Manag.* **2000**, *17*, 128–142. [CrossRef]
47. Swales, J. *Genre Analysis: English in Academic and Research Settings*, 1st ed.; Cambridge University Press: Cambridge, UK, 2014; ISBN 978-0-521-33813-4.
48. Heffernan, K.; Teufel, S. Identifying Problems and Solutions in Scientific Text. *Scientometrics* **2018**, *116*, 1367–1382. [CrossRef]
49. Heffernan, K.; Teufel, S. *Identifying Problem Statements in Scientific Text*; University of Potsdam: Potsdam, Germany, 2016. [CrossRef]

50. Haq, A.U.; Li, J.; Memon, M.; Khan, J.; Din, S.U.; AHAD, I.; Sun, R.; Lai, Z. Comparative Analysis of the Classification Performance of Machine Learning Classifiers and Deep Neural Network Classifier for Prediction of Parkinson Disease. In Proceedings of the 2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 14–16 December 2018.
51. Apruzzese, G.; Colajanni, M.; Ferretti, L.; Guido, A.; Marchetti, M. On the Effectiveness of Machine and Deep Learning for Cyber Security. In Proceedings of the 2018 10th International Conference on Cyber Conflict (CyCon), Tallinn, Estonia, 29 May–1 June 2018. [CrossRef]
52. Franchi, G.; Fehri, A.; Yao, A. Deep Morphological Networks. *Pattern Recogn.* **2020**, *102*, 107246. [CrossRef]
53. Zamora, E.; Sossa, H. Dendrite Morphological Neurons Trained by Stochastic Gradient Descent. *Neurocomputing* **2016**, *260*, 420–431.
54. Arce, F.; Zamora, E.; Azuela, J.H.S.; Barrón, R. Differential Evolution Training Algorithm for Dendrite Morphological Neural Networks. *Appl. Soft Comput.* **2018**, *68*, 303–313. [CrossRef]
55. Sossa, H.; Guevara, E. Efficient Training for Dendrite Morphological Neural Networks. *Neurocomputing* **2014**, *131*, 132–142. [CrossRef]
56. Sussner, P.; Campiotti, I. Extreme Learning Machine for a New Hybrid Morphological/Linear Perceptron. *Neural Netw.* **2020**, *123*, 288–298. [CrossRef] [PubMed]
57. Jenkinson, G.; Khezeli, K.; Oliver, G.R.; Kalantari, J.; Klee, E.W. Universally Rank Consistent Ordinal Regression in Neural Networks. *arXiv* **2021**, arXiv:2110.07470.
58. Peroni, S.; Shotton, D. OpenCitations, an Infrastructure Organization for Open Scholarship. *Quant. Sci. Stud.* **2020**, *1*, 428–444. [CrossRef]
59. Sinha, A.; Shen, Z.; Song, Y.; Ma, H.; Eide, D.; Hsu, B.-J. (Paul); Wang, K. An Overview of Microsoft Academic Service (MAS) and Applications. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 243–246.
60. Tang, J.; Zhang, J.; Yao, L.; Li, J.; Zhang, L.; Su, Z. ArnetMiner: Extraction and Mining of Academic Social Networks. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; Association for Computing Machinery: New York, NY, USA, 2008; pp. 990–998.
61. NCBI Resource Coordinators Database Resources of the National Center for Biotechnology Information. *Nucleic Acids Res.* **2018**, *46*, D8–D13. [CrossRef]
62. Europe PMC. Over 15,300 Full Text COVID-19 Now Available in Europe PMC. Available online: http://blog.europepmc.org/2021/02/full-text-covid19-preprints.html (accessed on 15 October 2021).
63. Lo, K.; Wang, L.L.; Neumann, M.; Kinney, R.; Weld, D. S2ORC: The Semantic Scholar Open Research Corpus. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 4969–4983.
64. Lu Wang, L.; Lo, K.; Chandrasekhar, Y.; Reas, R.; Yang, J.; Eide, D.; Funk, K.; Kinney, R.; Liu, Z.; Merrill, W.; et al. CORD-19: The Covid-19 Open Research Dataset. *arXiv* **2020**, arXiv:2004.10706.
65. ACL Anthology. Available online: https://www.aclweb.org/anthology/ (accessed on 12 May 2021).
66. McKeown, K.; Daume, H.; Chaturvedi, S.; Paparrizos, J.; Thadani, K.; Barrio, P.; Biran, O.; Bothe, S.; Collins, M.; Fleischmann, K.R.; et al. Predicting the Impact of Scientific Concepts Using Full-Text Features. *J. Assoc. Inf. Sci. Technol.* **2016**, *67*, 2684–2696. [CrossRef]
67. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
68. Breiman, L.; Spector, P. Submodel Selection and Evaluation in Regression—The X-Random Case. *Int. Stat. Rev.* **1991**, *60*, 291–319. [CrossRef]
69. Stone, M. Cross-Validatory Choice and Assessment of Statistical Predictions. *J. R. Stat. Society. Ser. B* **1974**, *36*, 111–147. [CrossRef]
70. Xu, Y.; Goodacre, R. On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. *J. Anal. Test.* **2018**, *2*, 249–262. [CrossRef]
71. Jung, Y. Multiple Predicting K -Fold Cross-Validation for Model Selection. *J. Nonparametric Stat.* **2017**, *30*, 1–19. [CrossRef]
72. Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (II), Montreal, QC, Canada, 20–25 August 1995; Volume 2, pp. 1137–1143. [CrossRef]
73. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
74. Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Available online: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ (accessed on 13 May 2021).
75. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Netw.* **2014**, *61*, 85–117. [CrossRef] [PubMed]
76. Sutskever, I.; Vinyals, O.; Le, Q. Sequence to Sequence Learning with Neural Networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; MIT Press: Cambridge, MA, USA, 2014; p. 10.
77. Hassabis, D.; Kumaran, D.; Summerfield, C.; Botvinick, M. Neuroscience-Inspired Artificial Intelligence. *Neuron* **2017**, *95*, 245–258. [CrossRef] [PubMed]

78. Nwadiugwu, M.C. Neural Networks, Artificial Intelligence and the Computational Brain. *arXiv* **2020**, arXiv:2101.08635.
79. Núñez, J.C.; Cabido, R.; Pantrigo, J.J.; Montemayor, A.S.; Vélez, J.F. Convolutional Neural Networks and Long Short-Term Memory for Skeleton-Based Human Activity and Hand Gesture Recognition. *Pattern Recognit.* **2018**, *76*, 80–94. [CrossRef]
80. Vivekanandan, K.; Praveena, N. Hybrid Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM) Based Deep Learning Model for Detecting Shilling Attack in the Social-Aware Network. *J. Ambient Intell. Hum. Comput.* **2021**, *12*, 1197–1210. [CrossRef]
81. Gulli, A.; Pal, S. *Deep Learning with Keras*; Packt Publishing: Birmingham, UK, 2017; ISBN 978-1-78712-842-2.
82. Weerts, H.J.P.; Mueller, A.C.; Vanschoren, J. Importance of Tuning Hyperparameters of Machine Learning Algorithms. *arXiv* **2020**, arXiv:2007.07588.
83. Yu, T.; Zhu, H. Hyper-Parameter Optimization: A Review of Algorithms and Applications. *arXiv* **2020**, arXiv:2003.05689.