applied sciences

MDPI

Article

Applying BERT for Early-Stage Recognition of Persistence in Chat-Based Social Engineering Attacks

Nikolaos Tsinganos, Panagiotis Fouliras and Ioannis Mavridis *

Department of Applied Informatics, University of Macedonia, 156 Egnatia Str., 54636 Thessaloniki, Greece
* Correspondence: mavridis@uom.edu.gr

Featured Application: Methods and techniques demonstrated in this work can be used to increase the effectiveness of chat-based social engineering attack detection systems.

Abstract: Chat-based social engineering (CSE) attacks are attracting increasing attention in the Small-Medium Enterprise (SME) environment, given the ease and potential impact of such an attack. During a CSE attack, malicious users will repeatedly use linguistic tricks to eventually deceive their victims. Thus, to protect SME users, it would be beneficial to have a cyber-defense mechanism able to detect persistent interlocutors who repeatedly bring up critical topics that could lead to sensitive data exposure. We build a natural language processing model, called CSE-PersistenceBERT, for paraphrase detection to recognize persistency as a social engineering attacker's behavior during a chat-based dialogue. The CSE-PersistenceBERT model consists of a pre-trained BERT model fine-tuned using our handcrafted CSE-Persistence corpus; a corpus appropriately annotated for the specific downstream task of paraphrase recognition. The model identifies the linguistic relationship between the sentences uttered during the dialogue and exposes the malicious intent of the attacker. The results are satisfactory and prove the efficiency of CSE-PersistenceBERT as a recognition mechanism of a social engineer's persistent behavior during a CSE attack.

Keywords: cybersecurity; sensitive data; chat-based social engineering attack; persistence; deep learning; natural language processing; transfer learning; BERT

check for updates

Citation: Tsinganos, N.; Fouliras, P.; Mavridis, I. Applying BERT for Early-Stage Recognition of Persistence in Chat-Based Social Engineering Attacks. *Appl. Sci.* **2022**, *12*, 12353. https://doi.org/10.3390/app122312353

Academic Editors: Konstantinos Rantos, Konstantinos Demertzis and George Drosatos

Received: 13 November 2022
Accepted: 29 November 2022
Published: 2 December 2022

1. Introduction

In a chat-based dialogue e.g., between an SME employee and a potential customer, the interlocutors exchange written sentences during their communication. The ability to identify one or more characteristics [1] that can discriminate a normal interlocutor from a malicious one can lead to sufficiently protecting the SME employee from a potential social engineering attack. To achieve her goal, a malicious interlocutor repeats her arguments many times in a conversation to convince and manipulate her partner. This was also confirmed, after processing our CSE corpus [2], where we observed that 83% of social engineering attackers tend to insist on their arguments to exfiltrate the targeted type of critical information. The persistence of an interlocutor to regurgitate the same topic that could lead to sensitive data exfiltration can be considered an enabler of a successful CSE attack. Thus, there is a need to develop a mechanism for recognizing when an interlocutor is continuously trying to lead the conversation to a specific and previously mentioned topic.

To cope with such a problem, we can adopt different approaches depending on the pros and cons of each one. Traditional machine learning models require substantial amounts of labeled data to be trained for a recognition task, but labeling data is a time-consuming and expensive activity. In pursuing persistence recognition, we choose to leverage the latest trends in deep learning, especially the Transformers [3] technology. Transformers use substantial amounts of unlabeled raw text for training and take advantage of the linguistic information contained to overcome the difficulty of creating a large amount of labeled data.

The Transformer models are very efficient in learning the context and relationships between sequential data, such as words in a sentence. Even if supervised learning using labeled data is an option, learning robust word vector representations in an unsupervised manner of learning can lead to a significant performance boost.

In this paper, we describe a hybrid approach for the paraphrase detection task, in the context of CSE attacks, based on a combination of unsupervised pre-training and supervised fine-tuning based on the Transformer-based BERT model [4]. This approach results in our proposed model, called CSE-PersistenceBERT, which learns a universal representation that transfers knowledge with only minimal adaptation to the paraphrase recognition task. Initially, the CSE-PersistenceBERT model has access to a large corpus of unlabeled text on which BERT has been pre-trained, and later it uses the CSE-Persistence corpus, which emerged from the CSE corpus of our previous work [2]. CSE-Persistence corpus generated after manually annotating the CSE corpus and utilizing the CSE ontology [2]. This corpus which is relevant to the downstream task was used to fine-tune CSE-PersistenceBERT. The complete training procedure was performed in two stages: firstly, the BERT model parameters were learned by training the model using unlabeled data on a language modeling objective, and secondly, a part of the pre-trained parameters was fine-tuned on the paraphrase recognition task using labeled data from the CSE-Persistence corpus. The evaluation results confirmed that CSE-PersistenceBERT can recognize the persistence of an interlocutor in an early stage of a chat-based conversation. This can result in an indicator of a potential social engineering attack, given that the topic of the conversation is an entity in our CSE ontology.

This work contributes to improving knowledge about utilizing existing state-of-the-art deep learning models as cyber defense mechanisms. Our study provides a better understanding of how we can tailor a BERT-based language model to expose the malicious intent of an interlocutor. Furthermore, detailed implementation details are given about fine-tuning such a model using an in-context and appropriately annotated corpus for the paraphrase recognition task.

The remainder of this paper is organized as follows: Section 2 provides an overview of the relevant background information. In Section 3, related work regarding the application of machine learning and deep learning approaches to the detection of social engineering attacks is presented. The proposed approach is presented in Section 4 and the implementation details are given in Section 5. The evaluation results are discussed in Section 6, respectively. In Section 7, we discuss our findings and the paper concludes with Section 8.

2. Background

2.1. Natural Language Processing

In Natural Language Processing (NLP) terms, persistence recognition can be cast as a Natural Language Understanding (NLU) task. Furthermore, there are a plethora of relative NLU tasks that can be utilized to describe persistence recognition, such as Semantic Textual Similarity, Natural Language Inference, and Paraphrase Recognition. All these NLU tasks are closely related, but they also differ in several aspects:

- Semantic Textual Similarity (STS) [5,6] is the semantic task of inferring the relation between different text data units. It is usually measured as a numerical score in the range of [0, 1] and quantifies the semantic similarity between different text data units. In earlier days, techniques such as Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) were used to represent text as vectors to aid the calculation of semantic similarity. These techniques were designed to identify if two text units contained the same words or a similar group of characters. However, sentences can have different meanings while containing the exact, same words or can contain different words while representing semantically similar concepts.
- Natural Language Inference, sometimes also called Textual Entailment [7–9], is the classification task of determining, given a text premise and a text hypothesis, whether the hypothesis is entailed by, contradictory to, or independent of the premise. The

most popular categories used for textual entailment relationships are entailment, contradiction, and neutral.

- Paraphrase Recognition [10] can be described as another text classification task that determines whether two text data units have a similar meaning (or not) in a specific context.

Semantic textual similarity identifies the semantic equivalence between text data units as a continuous value, while textual entailment and paraphrase detection produce a categorical output. However, it is possible to quickly transfer, e.g., from the STS measure area to the Paraphrase Detection task, by introducing a paraphrase detection threshold. For example, one can set a certain value of the STS measurement above which two different text data units can be considered as related. The aforementioned NLU tasks can also broadly be classified based on the approach used to achieve the objective, as follows:

- knowledge-based, where ontologies, databases, or dictionaries are used (e.g., WordNet [11])
- corpus-based, where information retrieved from large corpora is used (e.g., word2vec [12])
- deep neural network-based, where recent developments of neural networks are utilized (e.g., CNNs [13], Transformers [3], etc.), and
- hybrid-based, where characteristics from all the above-mentioned methods are combined [14].

For each of these NLU tasks, there are several popular datasets used for the performance evaluation of the corresponding algorithms. These datasets are composed of sentences or pairs of sentences with associated classification labels.

2.2. Neural Networks

A neural network model that transforms one sequence of tokens into another performs a sequence-to-sequence (Seq2Seq) task, as it accepts a sequence of tokens as input and produces another sequence of tokens as output. Known applications of such tasks are machine translation [15], spell-checker [16], etc. This paradigm, until recently, was implemented by neural networks based on an encoder-decoder architecture such as recurrent neural networks (RNNs) [17]. The encoder e.g., a Long Short-Term Memory (LSTM) [18] neural network, takes a sequence of tokens and outputs a fixed-size vector representation of the input. Then, the decoder, which is also an RNN, uses this vector to produce the output sequence of tokens. The limitation of this workflow lies in the information compression that occurs during the transformation of the input sequence into a fixed-length vector. No matter how lengthy the input sequence, its representation will always have to fit into a fixed vector size, and consequently a significant part of the information is lost.

This bottleneck was overcome using a mechanism called *Attention* [19,20]. Attention can produce a summary for each input token, which is context-dependent. This summary is a list of vectors that act as a memory that the decoder can look up during output production. These vectors represent the hidden states of the encoder and, in NLP, we can think of them as key-value pairs representing input tokens. An attention function f is applied on every key producing a weight, and then the input values are weighted by the corresponding weight and summed up to create the summary vector. This weighted sum is then appended to the decoder's hidden states to produce the final output sequence.

A *Transformer* [19] is a relatively new type of encoder-decoder neural network architecture that utilizes a specific type of attention mechanism based on the concept of *self-attention*. Due to their outstanding performance over RNNs, Transformers are now the de facto standard architecture to use in related NLP tasks (machine translation, etc.). The self-attention mechanism produces, for each input token, a summary of the entire input, using as a context the specific token. Furthermore, a Transformer uses *multi-head self-attention* by using multiple sets of key-value pairs and queries per token, thus producing multiple sets of weights focused on different input sequence characteristics. By repeatedly applying several layers, the input sequence is transformed from a raw word embedding to a more abstract form representing the input's semantics. While a Transformer is a powerful model by itself, it also acts as the underlying architecture of well-known pre-trained models such as GPT-2 ([21], p. 2) and BERT [4].

Pre-trained models are used in *transfer learning* [22], which is a collection of techniques that improve the performance of a neural network model in a task using data and/or a model trained for a different task. Transfer learning consists of at least two steps [23];

- pre-training, where the model learns a general-purpose representation of inputs, and
- adaptation, where the input representation is transferred to a downstream task. The adaptation has two main paradigms:
  a. Feature extraction, where the model's weights remain unchanged and are used as features in a downstream task.
  b. Fine-tuning, where (some of) the model's weights are unfrozen and fine-tuned for the new downstream task.

In the first step, the model is trained for one task (pre-training), and in the second step, it is adjusted for another task (fine-tuning). Transfer learning uses word embeddings, which are learned vector representations. Thus, semantically similar words share similar representations. Although these word embeddings can be used for downstream NLP tasks, they are limited in the sense that they are trained per token and thus ignore context.

BERT is a transformer-based pre-trained language model that employs the transformer encoder part to transform the input into contextualized embeddings through a series of layers that gradually summarize the input sequence. Due to its transformer-based architecture, BERT can capture long-term dependencies between input tokens, considering the context of the token in both directions. BERT is trained using self-supervised learning, which means there is no need for human intervention, e.g., data annotation. BERT's training comes from the data itself, as the humungous datasets used for training contain deep linguistic knowledge such as collocation, syntactic, grammatical, and semantic information.

In the literature, the big pre-trained language models are sometimes referred to as base or foundation models [24], and their availability gave rise to the terms of fine-tuning, transfer learning, and classification heads. Fine-tuning implies updates to some of the model's layers, while classification heads treat the last layer of the model as input features: they take input X and predict the outcome Y performing classification if the labels are categorical or regression if the labels are continuous [25].

3. Related Work

To the best of our knowledge, our work is the first attempt to utilize deep learning techniques to identify an interlocutor's persistence in a CSE attack. Nevertheless, many works are related to the paraphrase recognition task and similar tasks of semantic textual similarity and textual entailment.

Gupta et al. [26] introduce an interesting approach to generating paraphrases from a sentence, using an LSTM and a Variational Autoencoder as generative components. They propose a simple, modular, deep neural network architecture for question paraphrase generation from a question and a sentence based on a novel combination of generative adversarial networks and sequence-to-sequence models. Their method automatically generates paraphrases of an input sentence in multiple languages. This work, although interesting, has no specific target, and it is difficult to be utilized in the cyber security domain.

Victor U. Thompson and Chris Bowerman [27] approach the problem of recognizing texts that look different but are similar in meaning by identifying several phenomena related to paraphrasing. These phenomena include word and phrase reordering, the substitution of lexical parts, the addition and deletion of words, etc. They argue that the most effective method is to combine the results of identifying the above-mentioned techniques that commonly appear in plagiarism. Their proposed model shows promising performance by combining the aforementioned techniques. However, the complexity of the syntactical and grammatical checks is a deterrent factor for adopting such an approach for a real-time cyber security defense mechanism.

Mohamed I. El Desouki et al. [14] used deep learning techniques related to semantic textual similarity to detect paraphrase phenomena. They propose a hybrid model, which they evaluate using the Microsoft Research Paraphrase Corpus. Their model presents an F1 performance of 83.5%. The hybrid model follows a three-stage workflow where initially, the sentence is converted to a semantic vector via a skip-through approach. Then, a plethora of different string-based, knowledge-based, and corpus-based algorithms are executed. Finally, classical machine learning algorithms related to semantic textual similarity are applied. The authors' proposed solution is composed of several layers of processing employing traditional machine learning algorithms, too. Updating such a complex recognizer would be difficult after a deployment in production.

Mahtab Ahmed [28] uses a Tree-LSTM, a variant of LSTM, to represent the sentence structure in a tree topology. His model performed better using attention than other Tree-LSTMs without the use of an attention framework. The two different techniques presented can be applied for both dependency and constituency tree structures and proved superior for semantic relatedness tasks, such as textual entailment and paraphrase detection. This approach is promising but dependency and constituency can be difficult to maintain in a lengthy dialogue.

El Mostafa Hambi et al. [29] perform a comparative study of the different methods and datasets used regarding semantic textual similarity and other natural language inference tasks. They conclude that most of the research attempts are around word granularity using the word2vec method for word vector representation. This may be a disadvantage in the pursuit of understanding the meaning of a sentence. As mentioned by the authors the meaning of a sentence is of major importance and it is context related.

Kai Shuanga [30] focuses on word representation, presenting a method called Convolution-Deconvolution Word Embedding that embeds context-specific information and task-specific information. The results are presented by applying his model and method to two NLP tasks such as text classification and machine translation. Kai Shuanga's end-to-end embedding method is the first that extends deconvolution to word vector embedding generation. Although an interesting approach, there are no results for tasks such as paraphrase detection.

In another survey regarding the natural language inference tasks Divesh R. Kubal et al. [31] present several traditional and more modern methods, such as statistical methods, machine learning, and deep learning methods. A well-established comparative analysis is conducted for the word vector representation techniques. The overall survey is a thorough journey from statistical tools to machine learning, and deep learning tools and techniques. This work confirms the importance of a robust word vector representation and it is aligned with our approach to fine-tuning using a task-focused training dataset.

Tedo Vrbanec [10] examines corpus-based models for various NLP tasks such as paraphrase detection, textual entailment, etc. Using combinations of eight different algorithms and three different datasets, he presents his findings regarding model hyper-parameters, model selection techniques, similarity measures, and semantic textual similarity thresholds. The author after conducting several experiments confirms our approach for a simple model design.

Hien T. Nguyen [32] presents a novel approach, where each sentence is modeled using two vectors. In the first vector, the sentence is represented using pre-trained word vectors, while in the second vector, the sentence is represented on the basis of other sources of knowledge. He evaluated his model on different datasets, such as Microsoft Research Paraphrase Corpus, STS2015 ([33], p. 2015), and P4PIN [34]. The complexity of the aforementioned approach makes it difficult to scale in targeted tasks.

Although several different approaches exist, as presented above, they all use complex model architectures which have an impact on the computational resources and the training data required. Most authors aim to develop generic solutions with no focus on specific tasks. The lack of a focused training dataset decreases the efficiency of the models on a specific task something that we overcome by utilizing an appropriately tailored dataset such as the CSE-Persistence corpus.

4. The Proposed Approach

4.1. The CSE-Persistence Corpus

The Stanford Natural Language Inference (SNLI) corpus [35] is a well-known extensive collection of English-written sentence pairs manually annotated with entailment, contradiction, and neutral labels. It is usually used to determine the semantic relationship between two different text data units, a *premise*, and a *hypothesis.* Following the SNLI paradigm, we produced the CSE-Persistence corpus by modifying and annotating our CSE corpus [2] to be suitable for the task of paraphrase recognition. For this purpose, we also utilized the CSE ontology [2], which is asset-oriented and connects social engineering concepts with cybersecurity ones. The CSE ontology focuses on sensitive data that could leak from an SME employee during a chat-based conversation. It groups similar in-context concepts to facilitate the hierarchical categorization of an SME's assets and does not exceed three levels of depth to be efficient for text classification algorithms. The CSE ontology was created using a custom information extraction system and several text documents as input (corporate IT Policies, IT professionals' CVs, ICT Manuals, and others). An excerpt of the CSE ontology is depicted in Figure 1 where the '+' symbol means that the entity can be expanded.

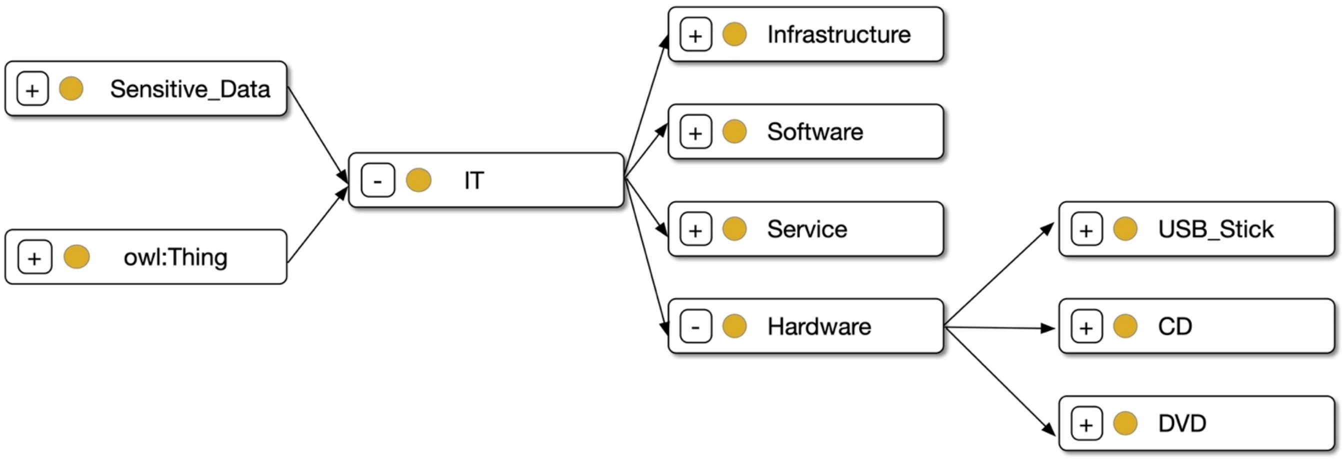

Figure 1. Excerpt of the CSE ontology.

Each instance of the CSE-Persistence corpus is composed of two sentences and is manually labeled as being a member of one of the following three categories:

- *Identical (I)*: The two sentences are semantically close *and* share a common term targeting the same leaf entity in the CSE Ontology (e.g., USB_Stick in Figure 1).
- *Similar (S)*: The two sentences are semantically related *and* share a common intent, which translates into a higher-level entity in the CSE ontology, targeting a different leaf entity (e.g., hardware as the higher-level entity and CD as the leaf entity in Figure 1).
- *Different (D)*: The two sentences are not semantically related, *and* they do not share a common higher-level or leaf entity.

We propose likening each CSE-Persistence corpus instance to a tiny drama play composed of only two sentences. Thus, by borrowing some drama structure terms, we named the two sentences as follows:

- The first sentence is referred to as the Prologue in the sense that the social engineering attacker uses it to introduce her intention to the play.
- The second sentence is referred to as the *Epilogue* in the sense that it concludes the play and informs us how the story ends.

Thus, the CSE-Persistence corpus contains instances of training examples where each instance is composed of a string for the Prologue, a string for the Epilogue, and a Paraphrase Recognition label (Identical, Similar, Different).

4.2. The Training Process

During the training process, three columns were considered from the CSE-Persistence corpus: ‘Prologue’, ‘Epilogue’, and ‘Paraphrase Recognition label’. The Paraphrase Recognition label is the ‘gold_label’ given to the instance after the manual annotation of the dataset. An excerpt of our CSE-Persistence corpus that is used for persistence recognition follows in Table 1:

Table 1. Excerpt from the CSE-Persistence corpus.

#	Prologue	Epilogue	Paraphrase Recognition Label
1	I have my resume on this USB key	Are you using a USB extension cord	identical
2	See, without your password, nobody can access your mail	As smart as they are, they didn’t the password	identical
3	When did you last change your password	I can cut some corners and save some time but I’ll need your username and password	identical
4	My network connection just went down like you said	I am just working on an audit	different
5	Hello how may I help you	Is there anything else I can help you with today	similar
6	It’s called Doctors Database and I believe that they are located in Denver Colorado	Hello John. This is Bill Jenkins from Doctor’s Database in Denver	identical
7	We’re trying to troubleshoot a computer networking problem	In the back of the computer can you recognize the network cable	identical
8	I am sorry for interrupting you, but I am experiencing a problem with my Charge 2	When I tried to turn on the Charge 2 I saw that the battery was leaking	similar
9	I need that info to report back to my boss	My boss will probably fire me if I don’t have it for the morning	similar
10	The chief executive character is in a meeting with important clients and would like the password reset as his current email account	But we need the details or we can’t give you any information	different

The CSE-Persistence corpus was divided into train, validation, and test set, where all the source data were elicited from successful or unsuccessful social engineering attacks gathered from websites, books, and logs as described in our previous work [2].

4.3. The CSE-PersistenceBERT Model

Our approach incorporates the technology of Transformers, self-supervised learning, pre-trained language models, and transfer learning to boost the performance of the paraphrase recognition task in the context of a CSE attack. For this purpose, we developed CSE-PersitenceBERT, a fine-tuned version of the BERT model. CSE-PersistenceBERT defines a sentence vector-based model that performs text classification on pairs of fixed-size sentence representations that are computed independently of one another. CSE-PersitenceBERT takes as input an instance composed of two sentences and outputs a classification result for them.

BERT comes in two flavors: BERT-base and BERT-large; we chose to work with BERT-base, a model that consists of twelve layers of transformer encoder blocks and 110 million parameters. The transfer learning technique is used to improve the performance of the downstream NLP task, which in this study is the paraphrase detection task, using a model that is already trained (pre-trained) on a different task (e.g., language modeling). Transfer learning, as adopted in our work, is depicted in Figure 2.

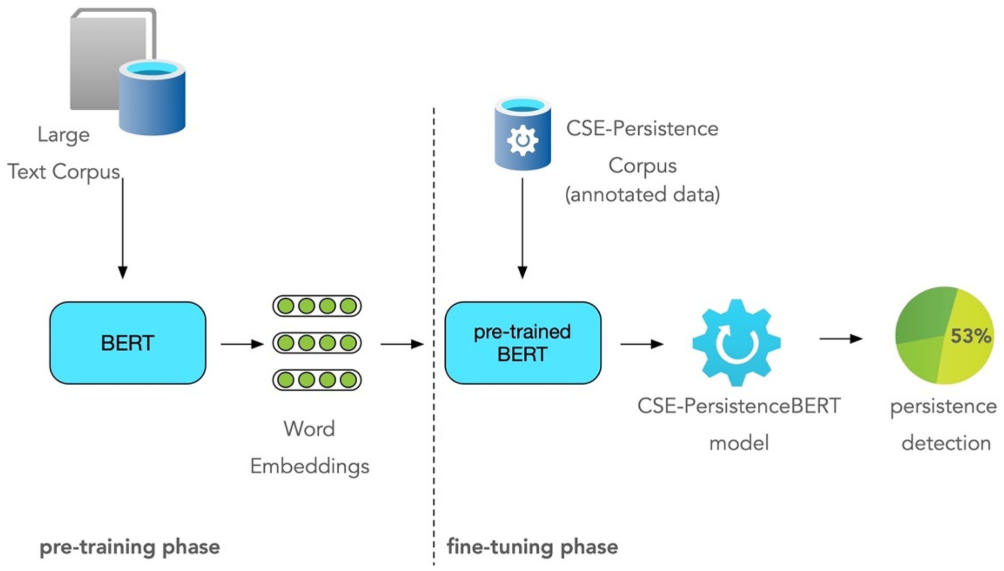

Figure 2. The proposed Transfer Learning approach.

The end-to-end learning process of the complete neural network is divided into a pre-training phase where the BERT-base model is trained by masking word tokens in each sentence of a large text corpus and a fine-tuning phase where the pre-trained BERT model is learning the persistence recognition labels, once again, from the CSE-Persistence corpus. This last training is performed only on the last BERT layer to extract features that will allow the model to use the representations of the pre-trained model.

BERT uses the [CLS] token that is placed at the beginning of a token indicating the start of the input sequence. In addition, the [SEP] token separates the two sentences, and the [PAD] token is used for padding. Thus, an example instance would be the following:

[CLS] <sentence 1> [SEP] <sentence 2> [SEP] [PAD]

The [CLS] vector acts as an input to the dense layer, and the similarity is calculated using a cosine similarity measure. The cosine similarity of two vectors *a* and *b* is computed according to Formula (1):

$$cosine\_similarity(a,b) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}, \tag{1}$$

where a, and b are vectors of dimension n.

During the fine-tuning phase, we do not modify the neural network architecture. More specifically, the classification pipeline (see Figure 3) is as follows: The pre-trained BERT tokenizer sends to the transformer encoder a sequence of tokens, which is composed of the two input sentences concatenated and separated by a special [SEP] token. A [CLS] token is prepended to the sequence denoting the start of the sequence. This token is used to extract the final embedding of the input instance. [PAD] token is a way to keep the input size constant. The loss function is minimized by continuously training the entire neural network on the CSE-Persistence corpus data. The loss function used is cross-entropy, which is computed according to Formula (2):

$$l_{cross-entropy}(\hat{y}, y) = -\sum_{i=1}^{C} t_i \log(\hat{y}_i), \tag{2}$$

where C is the number of different classes in the data, $\hat{y}$ is the predicted probabilities vector over the classes for the instance, y is the correct label for the instance, t_i indicates binary if i is the correct label for the specific instance, and $\hat{y}_i$ is the predicted probability that the instance belongs to class i.

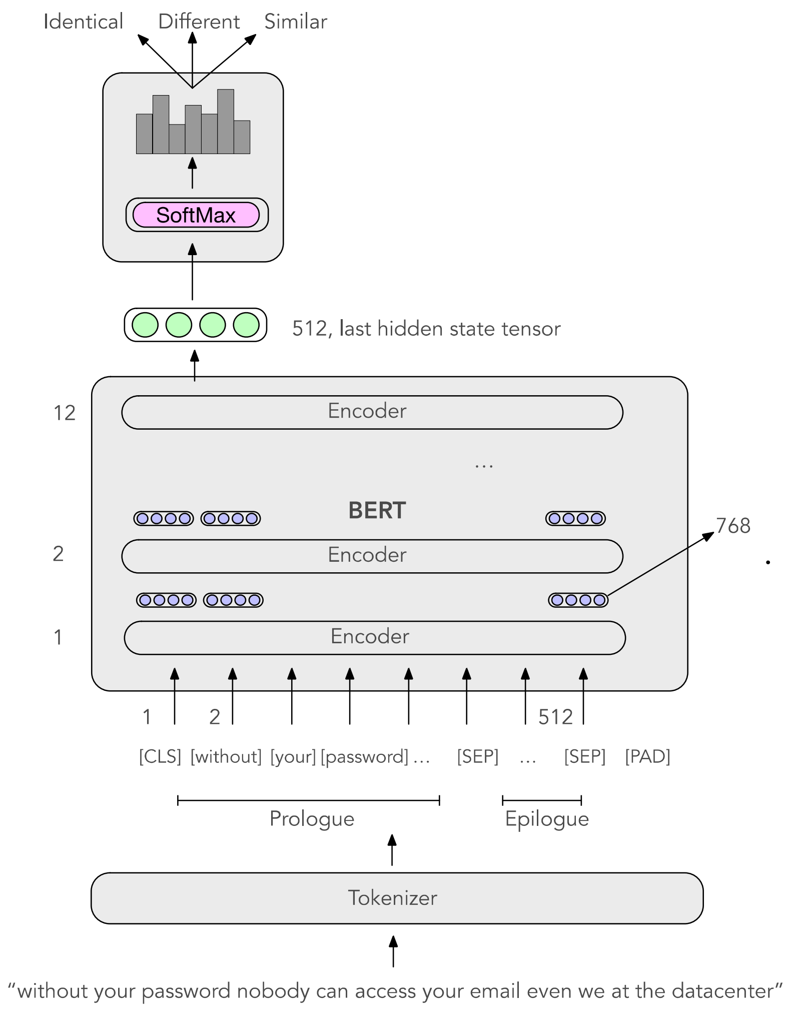
Identical
Different
Similar
SoftMax
512, last hidden state tensor
12
Encoder
...
BERT
768
2
Encoder
1
Encoder
1
2
512
[CLS] [without] [your] [password] ... [SEP] ... [SEP] [PAD]
Prologue
Epilogue
Tokenizer
"without your password nobody can access your email even we at the datacenter"

Figure 3. The proposed classification pipeline.

The training objective is to find the parameters that minimize the function, as mentioned above, which equally translates to:

$$arg_{\theta}min \frac{1}{|D|} \sum_{(x,y)\in D} l(\hat{y}, y), \tag{3}$$

where D is the dataset consisting of n data points (x_i, y_i) usually referred to as input vectors, x_i is the vector inputted into the neural network, and y_i is the accompanying label.

Instead of being initialized randomly, the CSE-PersistenceBERT model weights are inherited by the pre-trained BERT model, and the neural network is trained from scratch. The final sentence representations, which are a set of values called *logits*, are then passed to the *SoftMax* [36] function to derive a probability distribution regarding the sentences' paraphrase recognition task. Some of the BERT's weights, which were initialized with the pre-trained values, are also fine-tuned through backpropagation. The combination of the linear layer with SoftMax is called a *head*. Therefore, it is said that we are attaching a classification head to BERT to solve the prediction task. The SoftMax function that outputs the probabilities of the instance belonging to each class is the following:

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}, \tag{4}$$

where z_i is each element in the last layer of the neural network.

5. Implementation

5.1. CSE-Persistence Corpus

Table 2 presents the identity of the CSE-Persistence corpus, providing several key statistics:

Table 2. Key statistics of the CSE-Persistence corpus.

Data set size	16,900 sentences
Type of text units	Pairs of sentences
Source of judgment	Three judges
Training pairs	13,520
Development pairs	1690
Test pairs	1690
Identical sentences	6484
Similar sentences	5023
Different sentences	5393

Several text pre-processing steps occurred on the CSE-Persistence corpus, such as identification and removal of one-word sentences, ensuring appropriate text file encoding, and noise removal, e.g., stopwords, emoji, etc. to prepare it as input to the CSE-PersistenceBERT model.

In a separate validation phase, apart from our own annotation, we collected two more judgments for each label of the 16,900 examples, where a 96% annotator consensus emerged. In Table 1, three random sentence pairs with the selected gold label (Identical, Similar, Different) in bold, and each annotator's complete set of labels are depicted. E.g., the first sentence pair was classified as Identical (I) by us and the two annotators. Thus, the three labels are III, and the final Paraphrase Recognition Label (gold label) is Identical.

The underlined word is the entity found in the CSE Ontology. The first pair of sentences in Table 3 uses the leaf-entity "*password*" of the CSE ontology. The second pair of sentences does not share a leaf entity but a higher-level entity "*IT*" (Figure 1). The higher-level entity "IT" contains the leaf entities "icon" and "file". Finally, the third pair of sentences has neither a leaf entity nor a higher-level entity in common.

Table 3. Examples of sentence pairs from the annotated CSE-Persistence corpus.

Prologue	Epilogue	Annotators Labels	Paraphrase Recognition Label
Without your password, nobody can access your mail, even we at the data center	I can cut some corners to save some time, but I'll need your username and password [1]	III	**Identical** [2]
Just double-click on the icon when it downloads	You must open the file when it's done	SSS	**Similar**
Are you using a USB extension cord	Thanks for the quick replies	DDD	**Different**

[1] Underlined words represent CSE ontology entities. [2] Bold words represent gold labels.

To produce a well-balanced dataset, each *Prologue* appears five times in the CSE-Persistence corpus with a different *Epilogue* and the corresponding label. An example of a corpus instance in JSON format is presented in Figure 4:

```
{
  "prologue": "without your password, nobody can access your mail, even we at the data center",
  "epilogue": "I can cut some corners to save some time, but I 'll need your username and password",
  "label": "Identical"
}
```

Figure 4. Example of a CSE-Persistence instance in JSON format.

5.2. CSE-PersistenceBERT

For the implementation, the Hugging Face [37] library of transformers was used along with AllenNLP [38] and Pytorch [39]. The Hugging Face library has become the standard library for NLP researchers, and several state-of-the-art model implementations exist, such as GPT-2, BERT, RoBERTa, etc., with or without pre-trained model parameters. Using AllenNLP and BERT-as-service [40], we implemented a text classification model that embeds the input sequence, encodes it with a seq2vec ([41], p. 2) encoder, and finally classifies it with the help of a SoftMax layer coupled with a classification head. The embedding is done by BERT, BertPooler [38] did the encoding, and Adam [38] was employed as the optimizer.

For the fine-tuning of the CSE-PersistenceBERT model, we set the hyperparameter values as recommended in [4]: a maximum length of the input sentence to the model of 128 (max_length = 128), a batch size of 32 (batch_size = 32), a learning rate of 3e5, four training epochs (epochs = 4), and a dropout probability of 0.1.

During the persistence recognition task, we also keep a top-five ranking of the most similar sentences. E.g., for the sentence "The chief executive character wants to change her password", the top-5 ranking returns the following (Table 4):

Table 4. Top 5 of similar sentences.

Index	Sentence	Similarity
1	The chief executive character is in a meeting with important clients and would like the password reset as his current email account password no longer works	0.7804
2	And listen we just installed an update that allows people to change their passwords	0.6937
3	Now go ahead and type your password but don't tell me what it is	0.5255
4	You should never tell anybody your password not even tech support	0.3490
5	In this case, would you like to reset your password	0.3488

6. Evaluation

We evaluated the CSE-PersistenceBERT model using the following standard measures that are used to assess the performance of classification tasks:

- True Positives (TP): Sentence pairs where the true tag is positive and whose category is correctly predicted to be positive.
- False Positives (FP): Sentence pairs where the true tag is negative and whose category is incorrectly predicted to be positive.
- True Negatives (TN): Sentence pairs where the true tag is negative and whose category is correctly predicted to be negative.
- False Negatives (FN): Sentence pairs where the true tag is positive and whose class is incorrectly predicted to be negative.

Using the above measures, we calculate *Accuracy*, defined as the number of sentence pairs correctly identified as either true positive or truly negative out of the total number of entities

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{5}$$

For our baseline model, we used word2vec [12] to produce word vector representations in $\mathbb{R}^{300}$. Then, the sentence vector embeddings were generated by averaging the vector embeddings of all tokens in the sentence. Furthermore, we employ BERT-base without fine-tuning as a comparison model to support our proposed model's superiority.

During CSE-PersistenceBERT's fine-tuning, the only required architecture changes that are appropriate for the paraphrase recognition task concern the extra fully-connected layers. During the supervised learning of the downstream task, the parameters of these extra layers were learned from scratch, while some of the parameters of the pre-trained BERT model were fine-tuned. After fine-tuning, we compared CSE-PersitenceBERT to the baseline model and achieved the accuracy values presented in Table 5, which are also aligned with the work in [42].

Table 5. Model benchmarks on the CSE-Persistence corpus.

Model	Training Accuracy (%)	Training Loss	Validation Accuracy (%)	Validation Loss
Baseline	82.57	0.36	73.83	0.39
BERT-base	84.01	0.24	76.79	0.37
CSE-PersistenceBERT	84.96	0.21	78.03	0.36

Figure 5a, depicts the accuracy of the training and validation data sets of CSE-PersistenceBERT (indicated as CSE), and Figure 5b depicts the loss of training and validation datasets.

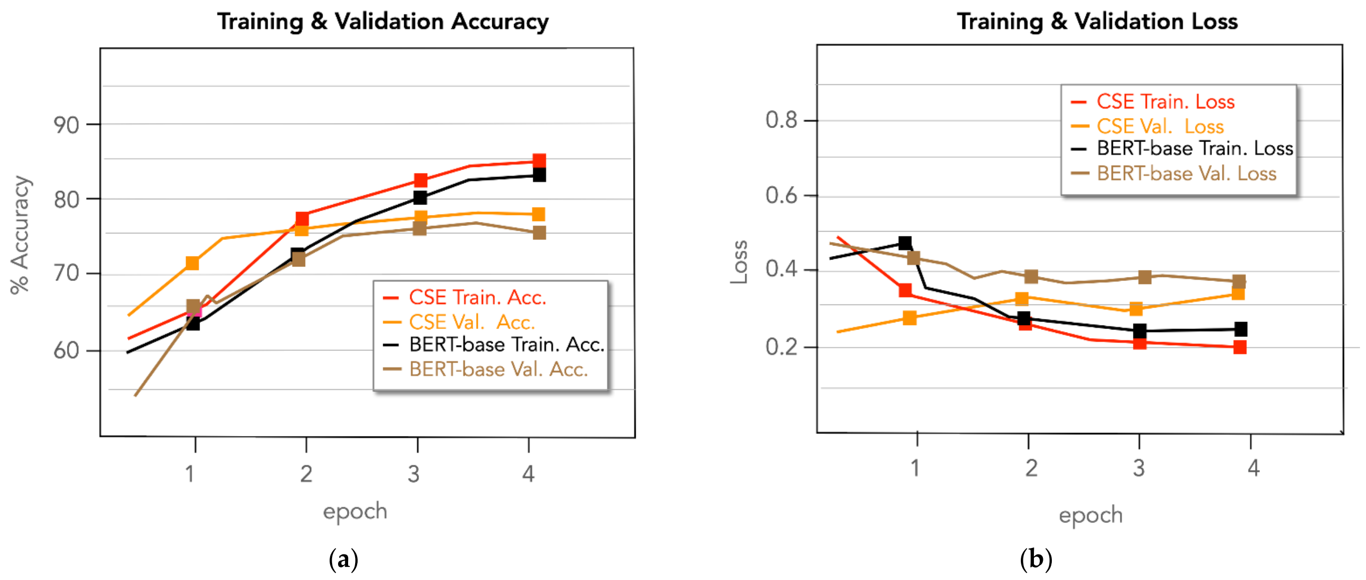

Figure 5. (**a**) Training and Validation Accuracy (**b**) Training and Validation Loss.

A comparison of the accuracy of the baseline model vs. CSE-PersistenceBERT is depicted in Figure 6 as the validation set of the CSE-Persistence corpus varied from 0% to 100%. In every percentage of data used, the CSE-PersistenceBERT model outperforms the baseline model. One key observation of the experimental results was that the difference between CSE-PersistenceBERT and baseline model's accuracy was bigger when the corpus was smaller (between 20% and 40%). This shows that even a complex BERT-based model can be efficient with a smaller corpus.

Furthermore, we investigated the impact of the number of layers transferred during fine-tuning from the unsupervised pre-training to the persistence recognition task. Figure 7, depicts the performance of the CSE-PersistenceBERT model as a function of the transferred layers. As an observation, we confirmed that each layer of the pre-trained BERT model contains valuable information for the persistence recognition task. Thus, CSE-PersistenceBERT improved its performance in the persistence recognition task by transferring its "knowledge".

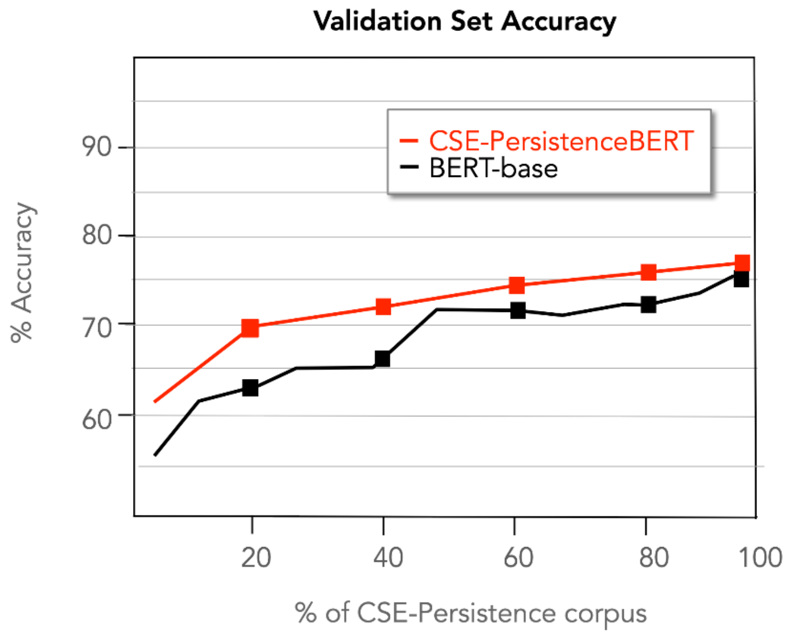

Figure 6. Comparison of accuracy on the validation set as a function of corpus percentage.

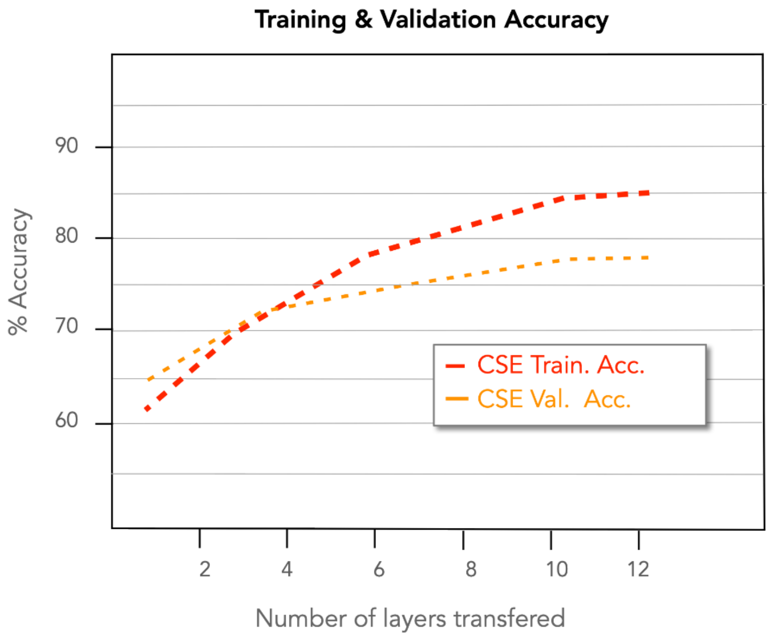

Figure 7. Performance improvement during transfer-learning.

7. Discussion

The high accuracy of the CSE-PersistenceBERT model indicates that it is capable of being used as an additional module in our proposed chat-based social engineering attack recognition system [1]. CSE-PersistenceBERT recognizes the persistent behavior of a malicious user by measuring the semantic similarity of the sentences uttered. Therefore, if an interlocutor tries to extract critical information by trying different approaches, her intention will be revealed by the model. Such a component is a useful add-on for a holistic system that combines several different models to recognize individual enablers of successful CSE attacks, e.g., personality characteristics, persuasion, and deception attempts, dialogue acts, etc.

Our research concludes that pre-trained models can be used for cyber-security-oriented tasks such as the recognition of chat-based social engineering attacks. The benefits are considerable:

- Less development and training time compared to an RNN model approach that is trained from scratch. The model weights are pre-trained, encoding valuable information trained on extensive corpora.
- Less training data as we only need to fine-tune the pre-trained model for a specific downstream task.
- Increased accuracy on the downstream task after fine-tuning for a few epochs (e.g., 2–4).

Initially, we encountered many errors of overestimating semantic similarity, and after further research, we concluded that punctuation played a crucial role in many missed cases. Additionally, the lexical overlap was another reason that led the model to mistakenly classify pairs of sentences as similar when they were not. The reasons were almost analogous for the cases where the CSE-Persistence model underpredicted similarity. Thus, the similarity was difficult to detect when there was a significant lack of lexical overlap or if completely different punctuation was used.

We experimented by using Euclidean distance instead of cosine similarity, and we observed slightly worse performance metrics. This can be explained because Euclidean distance is highly effective at clustering tasks, but a smaller distance is measured if the two different vectors have no common attribute values.

We explored word embeddings dimensionality reduction with Principal Components Analysis (PCA) and found a slight improvement in accuracy as in [43]. This approach may be helpful in the case of a smaller dataset; however, research is ongoing regarding making pre-trained models better few-shot learners [44,45] focusing on natural language inference tasks such as text entailment [46]. Few-shot learners, along with prompt engineering and GPT-3 [47], are the next step in our journey.

8. Conclusions

In this paper, CSE-PersistenceBERT was introduced, a pre-trained BERT-based model that was fine-tuned using the custom CSE-Persistence corpus to recognize persistent behavior as an enabler of successful chat-based social engineering attacks. Through pre-training on a corpus with long sections of contiguous in-context sentences, CSE-PersistenceBERT acquired the necessary language knowledge related to CSE attacks. This way the model acquires the capability to process long-range linguistic and semantic dependencies which are then successfully transferred to solve NLU tasks such as persistence recognition. The presented work confirms that the generic word vector representations produced by the pre-trained models can be employed in a range of natural language processing tasks related to the cyber security domain. Moreover, our fine-tuning using the appropriately annotated CSE-Persistence corpus generated in-context representations suitable for the paraphrase recognition task during a chat-based social engineering attack. The approach taken achieved satisfactory performance results while keeping the same model size. In the future, we plan to investigate the effects of data augmentation to improve the model's performance.

CSE-PersistenceBERT is intended to be part of a holistic system that aims to recognize CSE attacks in real time. The simple model design was one of our primary design principles in order to keep computational resources low. We are planning to integrate this model into an ensemble model which combines the output of several individual CSE attack recognizers on the basis of different characteristics e.g., personality characteristics, deception, and persuasion attempts detection, dialogue-act recognition, etc. The final prediction of the ensemble model will be performed using a heuristic, e.g., weighted majority vote.

Author Contributions: Conceptualization, N.T., P.F. and I.M.; methodology, N.T. and I.M.; software, N.T.; evaluation, N.T. and P.F.; writing—original draft preparation, N.T. and P.F.; writing—review and editing, N.T., P.F. and I.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Abbreviations

Abbreviation	Meaning
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag-of-Words
CD	Compact Disk
CLS	Classification
CNN	Convolutional Neural Network
CSE	Chat-based Social Engineering
DVD	Digital Video Disk
GPT	Generative Pre-trained Transformer
HD	Hard Disk
ICT	Information and Communication Technology
IT	Information Technology
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
NLU	Natural Language Understanding
P4PIN	Paraphrase for Plagiarism Include Non-Plagiarism
PAD	Padding
PCA	Principal Components Analysis
RNN	Recurrent Neural Networks
SEP	Separator
SEQ2SEQ	Sequence to sequence
SME	Small-Medium Enterprise
SNLI	Stanford Natural Language Inference
STS	Semantic Textual Similarity
TF-IDF	Term Frequency-Inverse Document Frequency
USB	Universal Serial Bus
WORD2VEC	Word to vector

References

1. Tsinganos, N.; Sakellariou, G.; Fouliras, P.; Mavridis, I. Towards an Automated Recognition System for Chat-based Social Engineering Attacks in Enterprise Environments. In Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, 27–30 August 2018. [CrossRef]
2. Tsinganos, N.; Mavridis, I. Building and Evaluating an Annotated Corpus for Automated Recognition of Chat-Based Social Engineering Attacks. *Appl. Sci.* **2021**, *11*, 10871. [CrossRef]
3. Lin, T.; Wang, Y.; Liu, X.; Qiu, X. A survey of transformers. *AI Open* **2022**, *3*, 111–132. [CrossRef]
4. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; Volume 1 (Long and Short Papers), pp. 4171–4186. [CrossRef]
5. Chandrasekaran, D.; Mago, V. Evolution of Semantic Similarity—A Survey. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–37. [CrossRef]
6. Agirre, E.; Diab, M.; Cer, D.; Gonzalez-Agirre, A. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In Proceedings of the First Joint Conference on Lexical and Computational Semantics—Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, Montreal, QC, Canada, 7–8 June 2012; pp. 385–393.

7. Manning, C.D. Local Textual Inference: It's Hard to Circumscribe, But You Know It When You See It—And NLP Needs It. 2006. Available online: http://nlp.stanford.edu/~{}manning/papers/LocalTextualInference.pdf (accessed on 18 April 2022).
8. Marelli, M.; Bentivogli, L.; Baroni, M.; Bernardi, R.; Menini, S.; Zamparelli, R. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), Dublin, Ireland, 23–24 August 2014; pp. 1–8. [CrossRef]
9. Dagan, I.; Glickman, O.; Magnini, B. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*; Quiñonero-Candela, J., Dagan, I., Magnini, B., d'Alché-Buc, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 3944, pp. 177–190. [CrossRef]
10. Vrbanec, T.; Meštrović, A. Corpus-Based Paraphrase Detection Experiments and Review. *Information* **2020**, *11*, 241. [CrossRef]
11. WordNet | A Lexical Database for English. Available online: https://wordnet.princeton.edu/ (accessed on 14 October 2021).
12. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781. Available online: http://arxiv.org/abs/1301.3781 (accessed on 3 March 2019).
13. Kim, Y. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 26–28 October 2014; pp. 1746–1751. [CrossRef]
14. Mohamed, I.; Gomaa, W.; Abdalhakim, H. A Hybrid Model for Paraphrase Detection Combines pros of Text Similarity with Deep Learning. *Int. J. Comput. Appl.* **2019**, *178*, 18–23. [CrossRef]
15. McCann, B.; Bradbury, J.; Xiong, C.; Socher, R. Learned in translation: Contextualized word vectors. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 4–9 December 2017; pp. 6297–6308.
16. Singh, S.; Singh, S. Systematic review of spell-checkers for highly inflectional languages. *Artif. Intell. Rev.* **2020**, *53*, 4051–4092. [CrossRef]
17. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv* **2015**, arXiv:1506.00019.
18. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef]
19. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 4–9 December 2017; pp. 6000–6010.
20. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
21. Budzianowski, P.; Vulić, I. Hello, It's GPT-2—How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems. In Proceedings of the 3rd Workshop on Neural Generation and Translation, Hong Kong, China, 4 November 2019; pp. 15–22. [CrossRef]
22. Ruder, S. Neural Transfer Learning for Natural Language Processing. Ph.D. Thesis, NUI Galway, Galway, Ireland, 2019. Available online: https://aran.library.nuigalway.ie/handle/10379/15463 (accessed on 23 November 2022).
23. Peters, M.E.; Ruder, S.; Smith, N.A. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. In Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), Florence, Italy, 2 August 2019; pp. 7–14. [CrossRef]
24. Wiggins, W.F.; Tejani, A.S. On the Opportunities and Risks of Foundation Models for Natural Language Processing in Radiology. *Radiol. Artif. Intell.* **2022**, *4*, e220119. [CrossRef]
25. Church, K.W.; Chen, Z.; Ma, Y. Emerging trends: A gentle introduction to fine-tuning. *Nat. Lang. Eng.* **2021**, *27*, 763–778. [CrossRef]
26. Gupta, A.; Agarwal, A.; Singh, P.; Rai, P. A deep generative framework for paraphrase generation. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 5149–5156.
27. Thompson, V. Methods for Detecting Paraphrase Plagiarism. *arXiv* **2017**, arXiv:1712.10309. Available online: http://arxiv.org/abs/1712.10309 (accessed on 27 April 2022).
28. Ahmed, M.; Samee, M.R.; Mercer, R.E. Improving Tree-LSTM with Tree Attention. In Proceedings of the 2019 IEEE 13th International Conference on Semantic Computing (ICSC), Newport Beach, CA, USA, 30 January–1 February 2019; pp. 247–254. [CrossRef]
29. Benabbou, F.; El Mostafa, H. A System for Ideas Plagiarism Detection: State of art and proposed approach. *Inf. Fusion* **2020**, *9*. [CrossRef]
30. Shuang, K.; Zhang, Z.; Loo, J.; Su, S. Convolution-deconvolution word embedding: An end-to-end multi-prototype fusion embedding method for natural language processing. *Inf. Fusion* **2020**, *53*, 112–122. [CrossRef]
31. Kubal, D.R.; Nimkar, A.V. A survey on word embedding techniques and semantic similarity for paraphrase identification. *Int. J. Comput. Syst. Eng.* **2019**, *5*, 36–52. [CrossRef]
32. Nguyen, H.T.; Duong, P.H.; Cambria, E. Learning short-text semantic similarity with word embeddings and external knowledge sources. *Knowl.-Based Syst.* **2019**, *182*, 104842. [CrossRef]

33. Agirre, E.; Banea, C.; Cardie, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Guo, W.; Lopez-Gazpio, I.; Maritxalar, M.; Mihalcea, R.; et al. SemEval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, CO, USA, 4–5 June 2015; pp. 252–263. [CrossRef]
34. Sánchez-Vega, J.F. Identificación de Plagio Parafraseado Incorporando Estructura, Sentido y Estilo de los Textos. Ph.D. Thesis, Instituto Nacional de Astrofísica, Optica y Electrónica, San Andrés Cholula, Mexico, 2016.
35. Bowman, S.R.; Angeli, G.; Potts, C.; Manning, C.D. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 632–642. [CrossRef]
36. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
37. Transformers. Available online: https://huggingface.co/docs/transformers/index (accessed on 8 April 2022).
38. Gardner, M.; Grus, J.; Neumann, M.; Tafjord, O.; Dasigi, P.; Liu, N.; Peters, M.; Schmitz, M.; Zettlemoyer, L. AllenNLP: A deep semantic natural language processing platform. *arXiv* **2018**, arXiv:1803.07640.
39. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An imperative style, high-performance deep learning library. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 8–14 December 2019; Curran Associates Inc.: Vancouver, BC, Canada, 2019; pp. 8026–8037.
40. Raval, S. Bert-as-Service. 2022. Available online: https://github.com/llSourcell/bert-as-service (accessed on 13 July 2022).
41. Kim, H.J.; Hong, S.E.; Cha, K.J. seq2vec: Analyzing sequential data using multi-rank embedding vectors. *Electron. Commer. Res. Appl.* **2020**, *43*, 101003. [CrossRef]
42. Phang, J.; Févry, T.; Bowman, S.R. Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks. *arXiv* **2019**, arXiv:1811.01088.
43. Huang, L.; Dou, Z.; Hu, Y.; Huang, R. Textual Analysis for Online Reviews: A Polymerization Topic Sentiment Model. *IEEE Access* **2019**, *7*, 91940–91945. [CrossRef]
44. Gao, T.; Fisch, A.; Chen, D. Making Pre-trained Language Models Better Few-shot Learners. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Online, 1–6 August 2021; Long Papers. Volume 1, pp. 3816–3830. [CrossRef]
45. Hu, Y.; Ding, J.; Dou, Z.; Chang, H. Short-Text Classification Detector: A Bert-Based Mental Approach. *Comput. Intell. Neurosci.* **2022**, *2022*, 8660828. [CrossRef] [PubMed]
46. Wang, S.; Fang, H.; Khabsa, M.; Mao, H.; Ma, H. Entailment as Few-Shot Learner. *arXiv* **2021**, arXiv:2104.14690.
47. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165.