future internet

MDPI

Article

A Hybrid CNN-LSTM Model for SMS Spam Detection in Arabic and English Messages

Abdallah Ghourabi [1,2,*], Mahmood A. Mahmood [1,3] and Qusay M. Alzubi [1]

[1] Department of Computer Science, Jouf University, Tabarjal 74728, Saudi Arabia; mamahmood@ju.edu.sa (M.A.M.); qmalzubi@ju.edu.sa (Q.M.A.)
[2] Higher School of Sciences and Technology of Hammam Sousse, University of Sousse, Hammam Sousse 4011, Tunisia
[3] Department of Information and Technology Systems, Cairo University, Giza 12613, Egypt
* Correspondence: aghourabi@ju.edu.sa

Received: 24 August 2020; Accepted: 14 September 2020; Published: 18 September 2020

Abstract: Despite the rapid evolution of Internet protocol-based messaging services, SMS still remains an indisputable communication service in our lives until today. For example, several businesses consider that text messages are more effective than e-mails. This is because 82% of SMSs are read within 5 min., but consumers only open one in four e-mails they receive. The importance of SMS for mobile phone users has attracted the attention of spammers. In fact, the volume of SMS spam has increased considerably in recent years with the emergence of new security threats, such as SMiShing. In this paper, we propose a hybrid deep learning model for detecting SMS spam messages. This detection model is based on the combination of two deep learning methods CNN and LSTM. It is intended to deal with mixed text messages that are written in Arabic or English. For the comparative evaluation, we also tested other well-known machine learning algorithms. The experimental results that we present in this paper show that our CNN-LSTM model outperforms the other algorithms. It achieved a very good accuracy of 98.37%.

Keywords: SMS spam detection; deep learning; CNN; LSTM; SMS Classification

1. Introduction

Human uses short message service (SMS) in mobile as a way of communication or business. Recently SMS is the most used data service in the world. The world sent 8.3 trillion SMS messages in the year 2017, the number of SMS messages sent monthly is 690 billion, so SMS is important for business communications [1].

Recently, SMS spam target mobile phones. SMS spam refers to any illusion text message that is delivered using the mobile network. They are disturbing to users [2]. A survey exposes that 68% of mobile phone users are affected by SMS Spam [3]. In some cases, SMS spam contains malicious activities, such as smishing. Smishing is a cyber-security attack for mobile user aimed at deceiving the user via SMS spam messages that may include a link or malicious software or both. Smishing is combined of two words: SMS and Phishing [3].

Nowadays, attackers find the sms is a simple way to communicate with victims [4]. The most victims of smishig and phishing attacks are users that have a smartphone [5]. The attackers attempt to steal user's secret information, like credit card number, bank account details, etc., by sending a link or direct contact with victims by SMS texts [6].

In addition, the filtration of SMS spam in smartphones is still not very robust as compared to email spams that are supported by advanced methods of spam filtering [7]. Among the recent solutions that have proven to be effective in solving these kinds of problems is the use of deep neural

networks. Deep neural network-based architecture, such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM), have been used for many classification problems for images, videos and texts. Deep learning is used for automatic pattern recognition as well as traditional clustering machine learning techniques (unsupervised techniques). It has several layers of information processing stages [2]. The components of deep neural network are employed in all stages of deep learning classifiers to minimize the classification errors. However, deep neural networks are not yet well exploited in the classification of SMS messages.

In the context of Arabic NLP, Deep Learning Models have proven their capabilities in several areas, like handwritten Arabic text recognition [8–10], recognition of Arabic text overlaid in videos [11], Automatic Speech Recognition [12], Machine translation [13,14], Text Categorization [15], and Sentiment Analysis [16–18].

In this paper, we want to focus on the SMS messages that are delivered in the Arabic speaking countries. These messages are generally written in Arabic or English. Thus, the challenge that we face is: (i) how to collect a significant dataset supporting both Arabic and English language and allowing researchers to conduct studies on SMS spam; and, (ii) how to find a robust classification model to detect spam messages in this mixed environment. For the first problem, we collected a set of 2730 Arabic messages from several local smartphones in Saudi Arabia and tagged them as spam and not-spam. For the second problem, we chose to harness the strength of deep neural networks to identify SMS spam. The proposed model is based on a hybrid deep learning architecture combining the CNN and LSTM algorithms. CNN performs excellently in extracting n-gram features at different positions of a message through the use of convolutional operations [19] and, hence, can be useful to identify the most common words in spam messages. However, LSTM is able to handle word sequences of any length and capture long-term dependencies [19]. This can be useful for identifying dependencies between words in a spam message and, hence, LSTM can determine whether the message is spam or not from the initial words of the message. We designed a hybrid model combining these two algorithms in order to benefit from the advantages of both CNN and LSTM. In addition, to compare this model with other classification techniques, we tested several known machine learning algorithms. The results show that our CNN-LSTM model gives the best performance based on several evaluation measures.

The main contributions of this paper are as follows:

- Collection of an Arabic SMS dataset labeled as spam or not-spam, which can be helpful in performing studies on Arabic SMS spam.
- Proposal of a model for detecting SMS spam in a mixed mobile environment that supports Arabic and English, based on a hybrid deep learning architecture combining the CNN and LSTM algorithms.
- Comparison of the proposed model with other machine learning algorithms.
- The proposed model outperforms traditional machine learning algorithms by achieving a remarkable accuracy of 98.37%.

The remaining parts of the paper are organized, as follows: Section 2 reviews the related works. Section 3 describes the model design of our approach. Section 4 presents the experimental results. Finally, we conclude the paper in Section 5.

2. Related Work

In recent years, several researchers have proposed solutions to identify SMS spam. A significant number of these works are based on the use of machine learning and datamining techniques. Some researchers have proposed the use of naive bayes algorithm to classify text messages as spam or ham [7,20,21]. Other researchers have tried other classifiers, such as Random Forest, Decision Tree, Support Vector Machine, and AdaBoost [22,23], or even a rule-based classification [24]. Other researchers have been interested in standardizing and expanding the content of the messages to

improve the classifiers performance [25]. Additionally, recently, researchers began using deep learning techniques for this task [2].

In [7], the authors proposed a model called "Smishing Detector" in order to detect smishing messages with a reduced false positive rate. The proposed model contains four modules. The goal of the first module is to analyze the content of text messages and identify the malicious contents by the use of Naive Bayes classification algorithm. The second one is used to inspect the URL contained in the messages. The third module is dedicated to analyze the source code of the website linked in the messages. The last module is an APK download detector, its role is to identify if a malicious file is downloaded when the URL is called. The experimental tests that were carried out by the authors on this model gave an accuracy of 96.29%.

In recent paper [2], Roy et al. proposed the use of deep learning to classify Spam and Not-Spam SMS messages. The objective of their approach is to employ two deep learning techniques together: CNN and LSTM. The idea is to classify text messages and identify those that are spam and those that are not spam. In order to evaluate the performance of the proposed approach, they compared it with other machine learning algorithms, like Naive Bayes, Random Forest, Gradient Boosting, Logistic Regression, and Stochastic Gradient Descent. The obtained results showed that the CNN and LSTM model is much better when compared to other machine learning models.

In another work that was designed to detect smishing messages, Joo et al. [20] proposed a model, called “S-Detector”. This model employs four modules: a component to monitor the SMS activities, an analyzer to analyze the content of the SMS, a determinant to classify and blocks Smishing text messages, and a database to store the SMS data. Naive Bayes is the classification algorithm used in this model.

In another paper [24], Jain and Gupta proposed a rule-based classification technique to detect phishing SMS. Their approach has identified nine rules that can identify and filter phishing SMS from legitimate ones. Experimental tests that were conducted by the authors yielded 99% of true negative rate and 92% of true positive rate.

In [22], Sonowal et al. proposed a model, called “SmiDCA”, for the detection of smishing messages based on machine learning algorithms. In the model, the authors chose to use correlation algorithms to extract the 39 most relevant features from smishing messages. Subsequently, they applied four machine learning classifiers to evaluate the performance of their model. The four classifiers were: Random Forest, Decision Tree, Support Vector Machine, and AdaBoost. The experimental evaluation of this model showed an accuracy of 96.4% with Random Forest Classifier.

In [23], the authors proposed a feature-based approach to detect smishing messages. This approach extracts ten features that the authors claim to be able to distinguish false messages from ham. Subsequently, the features were implemented on benchmarked dataset with the use of five classification algorithms to judge the performance of the proposed approach. The experimental evaluation showed that the model can detect smishing messages with 94.20% of true positive rate and 98.74% overall accuracy.

In [25], Almeida et al. proposed a processing method to normalize and expand text messages in order to improve the performance of classification algorithms when applied with these text messages. The proposed method is based on lexicography, semantic dictionaries and techniques of semantic analysis and disambiguation. The main idea was to standardize the words and create new attributes in order to expand the original text and reduce the factors that can degrade performance, such as redundancies and inconsistencies.

In [21], the authors proposed a method for filtering SMS spam based on two data mining techniques: FP-growth and Naive Bayes. FP-growth algorithm is used to extract frequent itemset in text messages and Naive Bayes Classifier is used to classify the messages and filter those that are spam. The experimental evaluation that was performed by the authors on this approach showed an average accuracy of 98.5%.

Concerning the detection of SMS spam in foreign languages, we only found a few works dealing with this problem. For example, in [26], the authors proposed a mobile-based system called "SMSAssassin" dedicated to filter SMS spam messages in India. This system is based on the application of Bayesian learning and Support Vector Machine techniques. For the experimental evaluation, the authors presented a new SMS Spam dataset collected from Indian users in the real-world. In another paper [27], the authors presented an analysis of different machine learning techniques for detecting SMS spam on a corpus of Indian messages. The authors used the UCI SMS spam dataset to which they added a set of Indian messages collected manually. For the experiments, they tested four simple algorithms: Multinomial Naive Bayes, Support Vector Machine, Random Forest, and Adaboost.

The contribution of our paper, as compared to existing work, is the proposal of an efficient system for detecting SMS spam that can deal with both English and Arabic messages. To the best of our knowledge, the model that we propose in this paper is the first approach that uses a combination of the deep learning algorithms in order to classify Arabic short messages.

In Table 1, we present a comparative summary on different works that are discussed in this section.

Table 1. Comparative Summary of related work.

Paper Reference	Approach Objective	Used Methods	Dataset Type
[2]	Classify SMS and identify spam messages	CNN and LSTM	• UCI machine learning repository: SMS spam collection dataset [28]
[7]	Identify spam messages and inspect included URL	Naive Bayes	• UCI machine learning repository: SMS spam collection dataset [28] • Pinterest Smishing message images
[20]	Classify SMS and identify spam messages	Naive Bayes	• private dataset
[24]	Classify SMS and identify spam messages	Rule-based classification	• UCI machine learning repository: SMS spam collection dataset [28]
[22]	Classify SMS and identify spam messages	Random Forest, Decision Tree, Support Vector Machine and AdaBoost.	• UCI machine learning repository: SMS spam collection dataset [28] • no-English data from Yadav et al. [26]
[23]	Identify spam messages and inspect included URL	Feature-based technique, Random Forest, Naïve Bayes, Support Vector Machine and Neural Network.	• UCI machine learning repository: SMS spam collection data set • NUS SMS corpus [29] • Pinterest Smishing message images
[25]	Normalize and expand text messages to improve the classification performance	Lexicography, semantic dictionaries and techniques of semantic analysis and disambiguation	• UCI machine learning repository: SMS spam collection dataset [28]
[21]	Classify SMS and identify spam messages	FP-growth and Naive Bayes	• UCI machine learning repository: SMS spam collection dataset [28]
[26]	Classify SMS and identify spam messages	Bayesian learning and Support Vector Machine	• Indian SMS dataset
[27]	Classify SMS and identify spam messages	Multinomial Naive Bayes, Support Vector Machine, Random Forest and Adaboost	• UCI machine learning repository: SMS spam collection dataset [28] • Indian SMS dataset

3. The Proposed Model

In this section, we describe, in detail, our proposed model. The main idea of this detection system is to process the collected SMSs and apply a machine learning method to classify them and identify those that are considered to be spam or phishing messages. In Figure 1, we present the architecture of the proposed model. In this model, we have chosen to apply two directives of machine learning

classification. The first is based on traditional machine learning algorithms, like Naive Bayes, SVM, Decision Tree, KNN, etc. The second directive is based on deep learning algorithms. The objective of this variety of algorithms is to try several classification methods in order to ultimately choose the one that gives the best results.

The process of the proposed system begins by cleaning up unnecessary information found in the text messages. Subsequently, a pre-processing task will be applied to these messages to represent the textual data in a compatible form, which will be given as input to the machine learning methods. After completing the preparation of the data, the classification algorithms will be applied to this data in order to distinguish spam messages and those that are not spam.

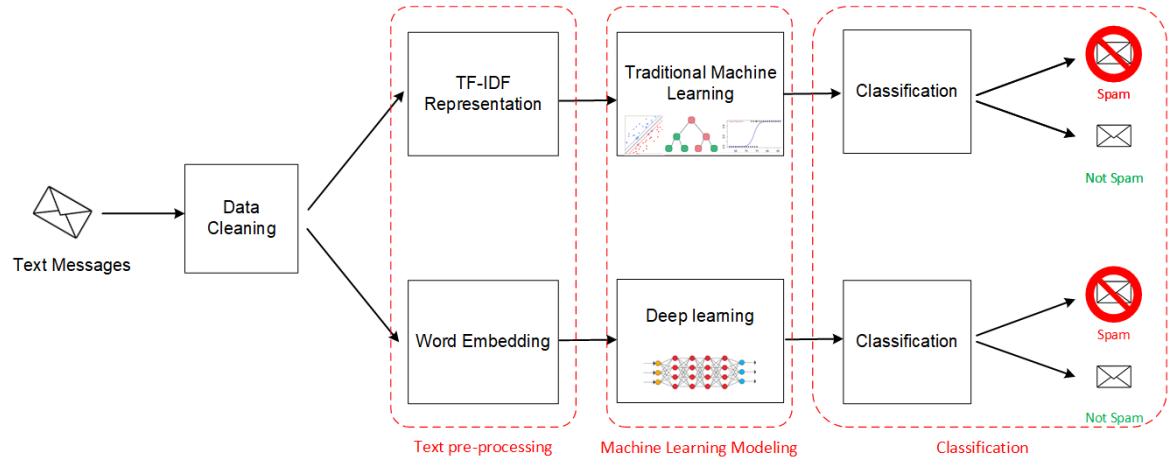

Figure 1. Architecture of the spam detection model.

3.1. Data Cleaning

Data cleaning is the first step in the proposed model. The purpose of this task is to remove unnecessary words and symbols from the text messages in order to improve the performance of the machine learning model. The elements to be removed at this step are:

- Punctuation: remove unnecessary punctuation marks and symbols from the text.
- Capitalization: remove the capital-letters by transforming all words to lower-case.
- Arabic and English stop-words: Stop-words refer to the most common words in a language that are not important for understanding the text. These include words such as "the", "is", "a", "which", and "on" for English language and إلى ، من ، في for Arabic language. In NLP tasks, it is often useful to remove these types of words before training the models in order to reduce the amount of 'noise'.

3.2. Text Pre-Processing

Text pre-processing is the second step in the model. The problem that we are trying to solve in this step is the fact that machine learning models only accept numerical data and cannot deal with textual data. Therefore, the objective is to transform the textual data collected from short messages into an understandable format which can be interpreted by machine learning algorithms. For this task we propose two separate methods due to the different nature of the machine learning algorithms that were used in this approach.

3.2.1. TF-IDF Representation

TF-IDF (term frequency–inverse document frequency) is a numerical statistic model used in several fields related to NLP, like information retrieval and text mining. It aims to convert the text documents into vector models based on a weight measure which is used to evaluate how important a

word is to a document in a collection or corpus. The TF-IDF measure increases proportionally to the number of times a word appears in the document and it is offset by the number of documents in the corpus that contain the word. Equation (1) is the formula of TF-IDF used for word weighting [30].

$$W_{i,j} = tf_{ij} * log(\frac{N}{df_i}) \quad (1)$$

where $W_{i,j}$ is the weight for word i in document j, N is the number of documents in the collection, tf_{ij} is the term frequency of term i in document j and df_i is the document frequency of term i in the collection. The purpose of this step is to use the TF-IDF method to convert the text data into numerical data. The result of this transformation is a matrix of TF-IDF features, which will then be used as input to the traditional machine learning algorithms.

3.2.2. Word Embedding

Word embedding is a technique that aims to represent words or sentences in a text by real numbers vectors, described in a vector model (or Vector Space Model). Semantically similar words are mapped close to each other in the vector space. Word embedding is very useful for representing words, to be given as input to a Deep learning model. In recent years, it is considered among the best representations of words in NLP. There are several pre-trained word embedding models that are ready to use, such as Word2Vec [31] and GloVe [32]. However, in this paper, we chose to train our own word embedding model, because our text messages contain mixed words from the Arabic and English language.

In the proposed approach, the purpose of word embedding is to prepare the textual data for the deep learning model. Figure 2 describes the steps performed to accomplish the word embedding process. When considering that we have an input data containing a collection of N text messages, the first step in the process is to vectorize the text corpus, by transforming each text into a sequence of integers (each integer being the index of a token in a dictionary). The result is a list of sequences with variable sizes (since the text messages do not have the same length). Subsequently, the "Pad sequences" task makes the sentences in uniform length, by padding sentences smaller than "MAX_SEQ_LENGTH" with empty values and truncating the sentences longer than "MAX_SEQ_LENGTH". At the end of this process, the text data are ready to train with the word embedding layer and give them as input to the deep learning model.

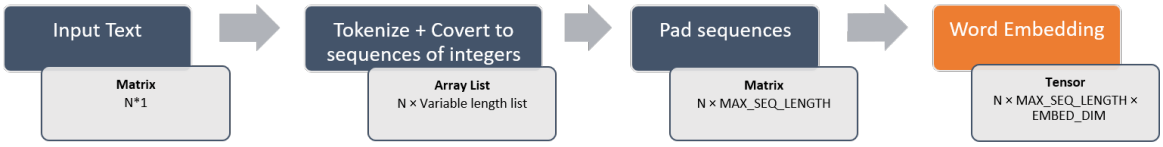

Figure 2. Word embedding process.

3.3. Machine Learning Modeling

Machine learning modeling is the core of our classification approach. In this step, we have chosen to try several machine learning algorithms to classify text messages as spam or not-spam. We then prepared two categories of algorithms: traditional machine learning algorithms and deep learning algorithms. The idea is to test these different algorithms and choose the one that gives the best performance. Based on the experimental results that are detailed in the next section, we concluded that a hybrid deep learning model based on the combination of two methods (CNN and LSTM) gave the best results as compared to the others.

3.3.1. Traditional Machine Learning

Even though the main idea of our approach was to implement a model based on deep learning, in this paper we have chosen to compare several algorithms that belong to the family of traditional machine learning. The algorithms tested in this category are:

- Support Vector Machine
- K-Nearest Neighbors
- Multinomial Naive Bayes
- Decision Tree
- Logistic Regression
- Random Forest
- AdaBoost
- Bagging classifier
- Extra Trees

3.3.2. Deep Learning

In this paragraph, we describe the deep learning-based model for detecting spams from a collection of Arabic and English text messages. To implement this model, we tried three different deep learning architectures. Firstly, we created an architecture using the CNN (Convolutional Neural Network) method. Secondly, we tried a model based on LSTM (Long Short-Term Memory). Finally, we implemented a hybrid model combining the two previous methods CNN and LSTM. Based on the experimental results that we will explain in the next section, the hybrid model gave us the best results in terms of performance. For this reason, here we will limit ourselves to the description of the CNN-LSTM hybrid model.

In Figure 3 and Algorithm 1, we describe the architecture and operating principle of the CNN-LSTM model. The output of the Embedding layer, as described above, is connected with a CNN layer with a type "Convolution 1D". Subsequently, a MaxPooling layer is applied to reduce the dimensionality of the CNN output. Next, we connect an LSTM layer. Finally, we apply a Dense output layer with a single neuron and a sigmoid activation function to make decision for the two classes spam and not-spam. In the following, we describe, in detail, the role and configuration of each layer.

Algorithm 1: Convolutional Neural Network (CNN)-Long Short-Term Memory (LSTM) model

Input: M (A set of n text messages, $M = M_1, M_2, ..., M_n$)
Output: Y (Message label : 0 or 1)
foreach *message* M_i *in* M **do**
  V_i = Word_Embedding(M_i)
end
foreach V_i **do**
  C_i = CNN(V_i)
end
foreach C_i **do**
  O_i = LSTM(C_i)
end
foreach O_i **do**
  $Y_i = sigmoid(O_i)$ `// Dense layer with sigmoid function`
end

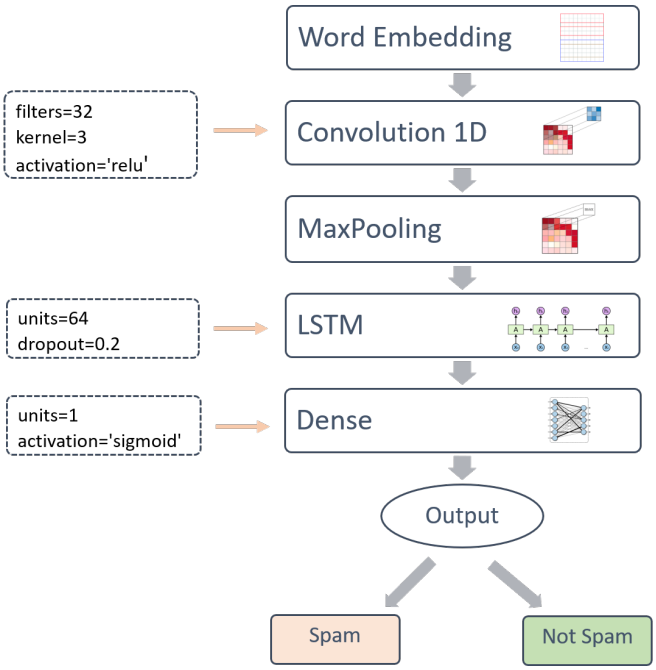

Figure 3. Architecture of the CNN-LSTM model.

- **Convolution 1D**

The main interest of the Convolution layer is to extract relevant features from the text data. This is done by applying the convolution operation on the word vectors generated by the Word Embedding layer. The convolution task that we present in this paragraph is based on the work that was presented in [19]. Let $x_i \in \mathbb{R}^d$ be the d-dimensional word vector corresponding to the i-th word in the message. Let $x \in \mathbb{R}^{L \times d}$ denote the input message, where L is the length of the message. For each position j in the message, consider a window vector w_j with k consecutive word vectors, represented as:

$$w_j = [x_j, x_{j+1}, \ldots, x_{j+k-1}] \tag{2}$$

A convolution operation involves a filter $p \in \mathbb{R}^{k \times d}$, which is applied to the window w to produce a new feature map $c \in \mathbb{R}^{L-k+1}$. Each feature element c_j for window vector w_j is calculated, as follows:

$$c_j = f(w_j \odot p + b) \tag{3}$$

where $\odot$ is element-wise product, $b \in \mathbb{R}$ is a bias term, and f is a nonlinear function. In our case, we used Rectified linear unit (ReLU) as nonlinear function. It is defined as:

$$f(x) = max(0, x) \tag{4}$$

ReLU activation function returns x if the value is positive, elsewhere returns 0. For the configuration of the convolution layer, we used a one-dimensional convolution associated with a filter window $k = 3$. Algorithm 2 describes the detailed working of the CNN algorithm.

Algorithm 2: CNN

Input: $x \in \mathbb{R}^{L \times d}$ (the input message where L is the length of the message and d is the word vector dimension)
Output: $\hat{c}$
// $p \in \mathbb{R}^{k \times d}$ is the filter applied to each window vector w in the message, $b \in \mathbb{R}$ is a bias term
foreach *position j in the message* **do**
  $w_j = [x_j, x_{j+1}, \ldots, x_{j+k-1}]$
  $c_j = ReLU(w_j \odot p + b)$
end
$c = [c_1, c_2, ..., c_{L-k+1}]$
$\hat{c} = \max(c)$ // MaxPooling operation

- **MaxPooling**

The feature maps that are generated by the Convolution operation are characterized by a high-level vector representation. To reduce this representation, we added a MaxPooling layer after the Convolution layer to help select only important information by removing weak activation information. This is useful to avoid overfitting due to noisy text.

- **LSTM**

CNN is very useful in extracting relevant features from the text data. However, it is unable to correlate the current information with the past information. This can be done with another deep learning method, which is "LSTM".

LSTM (Long short-term memory) is a kind of RNN architecture that is capable of learning long-term dependencies. The architecture of LSTM contains a range of repeated units for each time step. Each unit, at a time step t, is composed of a cell c_t (the memory part of LSTM) and three gates to regulate the flow of information inside the LSTM unit: an input gate i_t, an output gate o_t and a forget gate f_t. These gates collectively decide how to update the current memory cell c_t and the current hidden state h_t. The transition functions betwen the LSTM units are defined, as follows [19]:

$$\begin{aligned} i_t &= \sigma(W_i.[h_{t-1} + b_i]) \\ f_t &= \sigma(W_f.[h_{t-1}, x_t] + b_f) \\ q_t &= \tanh(W_q.[h_{t-1}, x_t] + b_q) \\ o_t &= \sigma(W_o.[h_{t-1}, x_t] + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot q_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{5}$$

Here, x_t is the input vector of the LSTM unit, σ is the sigmoid function, tanh denotes the hyperbolic tangent function, the operator $\odot$ denotes the element-wise product, and W, and b are the weight matrices and bias vector parameters that need to be learned during training. In the architecture of our model, we used a single LSTM layer placed directly after the MaxPooling. This layer contains 64 LSTM units using a Dropout equal to 0.2 as a regularization parameter to prevent the model from overfitting. Algorithm 3 describes the detailed working of the LSTM algorithm.

Algorithm 3: LSTM

Input: x
Output: h
foreach *time step* t **do**
$i_t = \sigma(W_i.[h_{t-1} + b_i])$
$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$
$q_t = \tanh(W_q.[h_{t-1}, x_t] + b_q)$
$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$
$c_t = f_t \odot c_{t-1} + i_t \odot q_t$
$h_t = o_t \odot \tanh(c_t)$
end

- **Dense**

Dense is the last layer of our model. It is also called the fully connected layer, and it is used to classify text messages according to the output of the LSTM layer. Since our classification model is binary, we used a Dense layer with a single neuron and a sigmoid activation function to give predictions of 0 or 1 for the two classes (Not-spam and spam). The sigmoid function is a logistic function that returns a value between 0 and 1, as defined by the formula in Equation (6):

$$f(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

3.4. Classification

Classification is the final step in our model. Regardless of the used machine learning method, the goal here is to classify the text messages given as input to this model into two classes: spam and not-spam. The spam class refers to any undesirable message containing temptations of phishing or theft of information. The not-spam class designates normal messages that represent no danger for the users.

4. Experimental Evaluation

In this section, we present the results that were obtained from the experimental tests that we conducted on the model presented in this paper. We also present a comparison between the different machine learning techniques tried in our approach. The implementation of this model was performed in Python 3.7 with the help of the TensorFlow environment and the Keras 2.0 API.

4.1. Dataset Description

For the evaluation of our approach, we used two types of dataset: (i) the SMS Spam dataset from the UCI Repository [25] and (ii) a set of Arabic messages collected from local smartphones. The SMS Spam dataset is a public set of SMSs labeled messages that have been collected for SMS spam research. It contains 5574 English messages that were labeled according being legitimate (ham) or spam. For the second dataset, it contains a set of 2730 Arabic messages that were collected from several local smartphones in Saudi Arabia and which are labeled as spam and not-spam. For example, Figure 4 contains an Arabic spam message attempting to steal the confidential information of the user's bank card. In Table 2, we show some statistics regarding the dataset used.

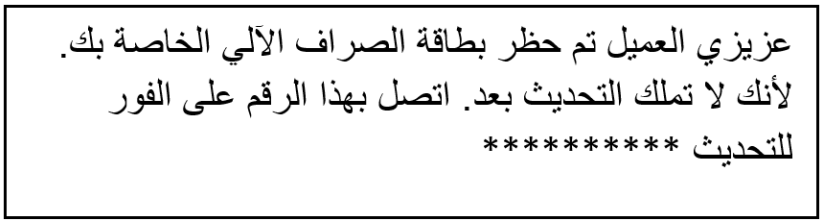

Figure 4. An example of an Arabic spam message.

Table 2. Statistics of the dataset.

	Number of the Messages	Training Size	Testing Size
Spam	785		
Not-Spam	7519	80%	20%
Total	8304		

4.2. Evaluation Measures

To evaluate the performance of the proposed model, we used the standard metrics for classification tasks, such as Accuracy, Precision, Recall, F1-Score, Confusion Matrix, ROC curve, and AUC.

- The confusion matrix is a table which indicates the following measures:
  - True Positives (TP): the cases when the actual class of the message was 1 (Spam) and the predicted is also 1 (Spam)
  - True Negatives (TN): the cases when the actual class of the message was 0 (Not-Spam) and the predicted is also 0 (Not-Spam)
  - False Positives (FP): the cases when the actual class of the message was 0 (Not-Spam), but the predicted is 1 (Spam).
  - False Negatives (FN): the cases when the actual class of the message was 1 (Spam) but the predicted is 0 (Not-Spam).
- Accuracy: is the number of messages that were correctly predicted divided by the total number of predicted messages.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{7}$$

- Precision: is the proportion of positive predictions (Spam) that are truly positives.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

- **Recall**: is the proportion of actual Positives that are correctly classified.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

- F1-Score: is the harmonic mean of precision and recall.

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{10}$$

- Receiver Operating Characteristic (ROC) is the plot of the true positive rate (TPR) [Equation (11)] against the false positive rate (FPR) [Equation (12)] at various threshold settings.

$$TPR = \frac{TP}{TP + FN} \tag{11}$$

$$FPR = \frac{FP}{FP + TN} \tag{12}$$

- AUC (Area under the ROC Curve) is an aggregate measure of performance across all possible classification thresholds.

4.3. Parameters of the Deep Learning Layers

In the deep learning part of our model, we tested three architectures: CNN, LSTM, and a combination of CNN and LSTM together. In Figures 5–7, we present using the Keras API the parameters of these three architectures.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 150, 64)           384000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 150, 32)           6176
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 75, 32)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2400)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2401
=================================================================
Total params: 392,577
Trainable params: 392,577
Non-trainable params: 0
_________________________________________________________________
```

Figure 5. Parameters of CNN architecture.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 150, 64)           384000
_________________________________________________________________
lstm_1 (LSTM)                (None, 64)                33024
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 417,089
Trainable params: 417,089
Non-trainable params: 0
_________________________________________________________________
```

Figure 6. Parameters of LSTM architecture.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 150, 64)           384000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 150, 32)           6176
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 75, 32)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 64)                24832
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 415,073
Trainable params: 415,073
Non-trainable params: 0
_________________________________________________________________
```

Figure 7. Parameters of CNN-LSTM architecture.

4.4. Experimental Results

To classify the text messages into Spam and Not-Spam, we tried in this paper several machine learning algorithms: (i) Support Vector Machine, (ii) K-Nearest Neighbors, (iii) Multinomial Naive Bayes, (iv) Decision Tree, (v) Logistic Regression, (vi) Random Forest, (vii) AdaBoost, (viii) Bagging classifier, (ix) Extra Trees, (x) CNN, (xi) LSTM, and (xii) a hybrid CNN-LSTM algorithm. To train these classifiers, we used the same distribution of data for all algorithms: 80% for training and 20% for the test. We calculated five measures to compare the performance of the classifiers: Accuracy, Precision, Recall, F1-Score, and ROC_AUC.

In Table 3 and Figure 8, we show the results that were found for all of the previously mentioned algorithms. Starting by the accuracy, the hybrid model CNN-LSTM gave the best score with a value equal to 0.983745, followed by the CNN algorithm with an accuracy of 0.981939. For the precision, Random Forest was the best with a value equal to 1, CNN-LSTM is in fourth position with a precision of 0.953947. Concerning the Recall, CNN-LSTM and Multinomial Naive Bayes share the best score with a value equal to 0.878788. Subsequently, for the F1-Score, CNN-LSTM is once again the best with a score of 0.914826 against 0.905063 for the CNN algorithm. Finally, for the AUC measurement, our CNN-LSTM model gave the best score with a value of 0.937054, Multinomial Naive Bayes is the second with a score of 0.934046. In Figures 9 and 10, we present the confusion matrix and the ROC curve of the CNN-LSTM model.

In conclusion, the hybrid model that is based on CNN and LSTM that we have proposed in this paper, has shown its effectiveness by giving the best result in four among five evaluation measures. The obtained results prove that a good combination of deep learning algorithms, as we have done in this paper, represents a very promising solution to create efficient systems for text messages classification and spam detection.

Table 3. Results of the classification algorithms.

	Accuracy	Precision	Recall	F1-Score	ROC AUC
Support Vector Machine	0.978326	0.977778	0.800000	0.880000	0.898997
K-Nearest Neighbors	0.900662	0.000000	0.000000	0.000000	0.500000
Multinomial Naive Bayes	0.978326	0.900621	**0.878788**	0.889571	0.934046
Decision Tree	0.965081	0.854305	0.781818	0.816456	0.883556
Logistic Regression	0.965081	0.942149	0.690909	0.797203	0.843115
Random Forest	0.978326	**1.000000**	0.781818	0.877551	0.890909
AdaBoost	0.972306	0.934307	0.775758	0.847682	0.884871
Bagging classifier	0.972306	0.883871	0.830303	0.856250	0.909135
Extra Trees	0.978928	0.977941	0.806061	0.883721	0.902028
CNN	0.981939	0.947020	0.866667	0.905063	0.930660
LSTM	0.981337	0.946667	0.860606	0.901587	0.927629
CNN-LSTM	**0.983745**	0.953947	**0.878788**	**0.914826**	**0.937054**

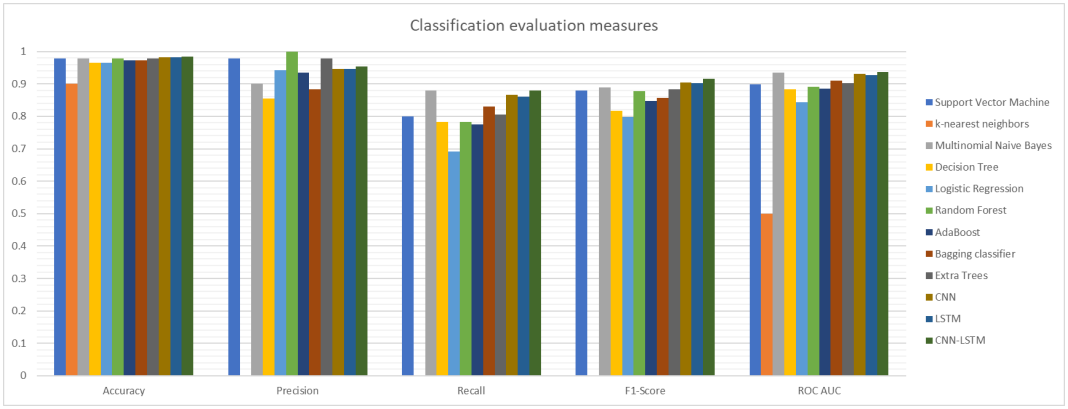

Figure 8. Comparison of the proposed CNN-LSTM model with other classifiers.

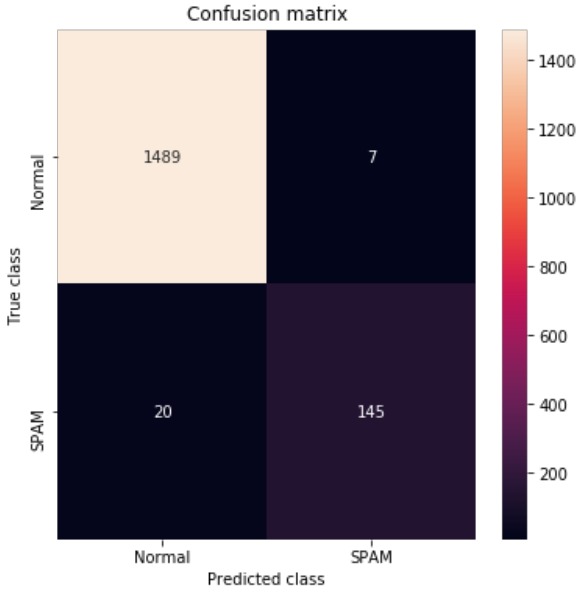

Figure 9. Confusion Matrix of CNN-LSTM model.

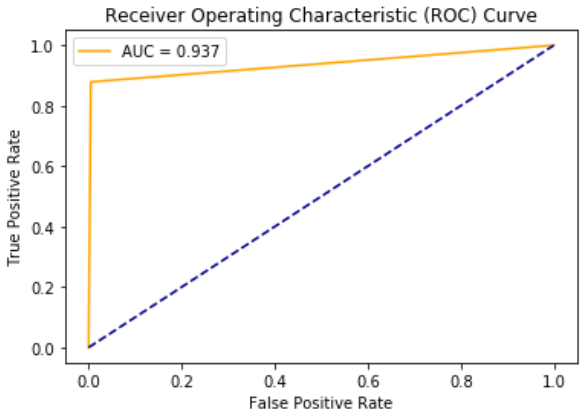

Figure 10. ROC curve of CNN-LSTM model.

5. Conclusions

In this paper, a hybrid model that is based on CNN and LSTM is presented for classifying SMS spam, which models SMS contexts (such as mobile network messages, Facebook messenger messages, WhatsApp messages). For the evaluation dataset, a set of messages in Arabic and English is collected in order to obtain a real dataset. Support vector machine (SVM), K-Nearest Neighbors (KNN), Multinomial Naive Bayes (NB), Decision Tree (DT), Logistic Regression (LR), Random Forest (RF), AdaBoost (AB), Bagging classifier, and Extra Trees are also utilized in order to classify SMS spam based on collected data. The experimental evaluation of the proposed approach has shown that the CNN-LSTM model performs better than other techniques for classifying SMS spam. The experimental results showed that our CNN-LSTM model achieved an accuracy of 98.37%, a precision of 95.39%, a recall of 87.87%, an F1-Score of 91.48%, and an AUC of 93.7%. This solution can significantly improve the security of smartphones by filtering spam messages and minimizing the risks that are related to smishing attacks in mobile environments.

As future work, we plan to create a rich framework capable of filtering spam messages in smartphones with better precision. The objective is to add more functionalities, such as the analysis of URLs or files attached to messages and the inspection of telephone numbers included in messages.

Author Contributions: Conceptualization, A.G. and M.A.M.; Methodology, A.G.; Software, A.G. and Q.M.A.; Validation, A.G., M.A.M. and Q.M.A.; Resources, M.A.M.; Data curation, A.G.; Formal analysis, A.G.; Investigation, A.G.; Project administration, A.G.; Supervision, A.G.; Visualization, A.G.; Writing—original draft, A.G. and M.A.M.; Writing—review & editing, A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Morreale, M. Daily SMS Mobile Usage Statistics. 2017. Available online: https://www.smseagle.eu/2017/03/06/daily-sms-mobile-statistics/ (accessed on 15 June 2020).
2. Roy, P.K.; Singh, J.P.; Banerjee, S. Deep learning to filter SMS Spam. *Future Gener. Comput. Syst.* **2020**, *102*, 524–533. [CrossRef]
3. Tatango. Text Message Spam Infographic. 2011. Available online: https://www.tatango.com/blog/text-message-spam-infographic/ (accessed on 15 June 2020).
4. Goel, D.; Jain, A. Smishing-Classifier: A Novel Framework for Detection of Smishing Attack in Mobile Environment. In Proceedings of the Smart and Innovative Trends in Next Generation Computing Technologies (NGCT 2017), Dehradun, India, 30–31 October 2017; pp. 502–512.
5. Goel, D.; Jain, A.K. Mobile phishing attacks and defence mechanisms: State of art and open research challenges. *Comput. Secur.* **2018**, *73*, 519–544. [CrossRef]
6. Jain, A.K.; Yadav, S.K.; Choudhary, N. A Novel Approach to Detect Spam and Smishing SMS using Machine Learning Techniques. *IJESMA* **2020**, *12*, 21–38. [CrossRef]
7. Mishra, S.; Soni, D. Smishing Detector: A security model to detect smishing through SMS content analysis and URL behavior analysis. *Future Gener. Comput. Syst.* **2020**, *108*, 803–815. [CrossRef]
8. Graves, A. Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks. In *Guide to OCR for Arabic Scripts*; Springer: London, UK, 2012, pp. 297–313.
9. Chherawala, Y.; Roy, P.P.; Cheriet, M. Feature Set Evaluation for Offline Handwriting Recognition Systems: Application to the Recurrent Neural Network Model. *IEEE Trans. Cybern.* **2016**, *46*, 2825–2836. [CrossRef] [PubMed]
10. Elleuch, M.; Kherallah, M. An Improved Arabic Handwritten Recognition System Using Deep Support Vector Machines. *Int. J. Multimed. Data Eng. Manag.* **2016**, *7*, 1–20. [CrossRef]
11. Yousfi, S.; Berrani, S.A.; Garcia, C. Contribution of recurrent connectionist language models in improving LSTM-based Arabic text recognition in videos. *Pattern Recognit.* **2017**, *64*, 245–254. [CrossRef]
12. El-Desoky Mousa, A.; Kuo, H.J.; Mangu, L.; Soltau, H. Morpheme-based feature-rich language models using Deep Neural Networks for LVCSR of Egyptian Arabic. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 8435–8439.
13. Deselaers, T.; Hasan, S.; Bender, O.; Ney, H. A Deep Learning Approach to Machine Transliteration. In Proceedings of the Fourth Workshop on Statistical Machine Translation, Athens, Greece, 30–31 March 2009; StatMT '09; Association for Computational Linguistics: Stroudsburg, PA, USA, 2009; pp. 233–241.
14. Guzmán, F.; Bouamor, H.; Baly, R.; Habash, N. Machine Translation Evaluation for Arabic using Morphologically-enriched Embeddings. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016; The COLING 2016 Organizing Committee: Osaka, Japan, 2016; pp. 1398–1408.
15. Jindal, V. *A Personalized Markov Clustering and Deep Learning Approach for Arabic Text Categorization*; In Proceedings of the ACL 2016 Student Research Workshop, Berlin, Germany, 7–12 August 2016.
16. Dahou, A.; Xiong, S.; Zhou, J.; Haddoud, M.H.; Duan, P. Word Embeddings and Convolutional Neural Network for Arabic Sentiment Classification. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016; The COLING 2016 Organizing Committee: Osaka, Japan, 2016; pp. 2418–2427.

17. Al-Sallab, A.; Baly, R.; Hajj, H.; Shaban, K.B.; El-Hajj, W.; Badaro, G. AROMA: A Recursive Deep Learning Model for Opinion Mining in Arabic as a Low Resource Language. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2017**, *16*, 1–20. [CrossRef]
18. Al-Smadi, M.; Qawasmeh, O.; Al-Ayyoub, M.; Jararweh, Y.; Gupta, B. Deep Recurrent neural network vs. support vector machine for aspect-based sentiment analysis of Arabic hotels' reviews. *J. Comput. Sci.* **2018**, *27*, 386–393. [CrossRef]
19. Zhou, C.; Sun, C.; Liu, Z.; Lau, F.C.M. A C-LSTM Neural Network for Text Classification. *arXiv* **2015**, arXiv:cs.CL/1511.08630.
20. Joo, J.W.; Moon, S.Y.; Singh, S.; Park, J.H. S-Detector: an enhanced security model for detecting Smishing attack for mobile computing. *Telecommun. Syst.* **2017**, *66*, 29–38. [CrossRef]
21. Delvia Arifin, D.; Shaufiah.; Bijaksana, M.A. Enhancing spam detection on mobile phone Short Message Service (SMS) performance using FP-growth and Naive Bayes Classifier. In Proceedings of the 2016 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob), Bandung, Indonesia, 13–15 September 2016; pp. 80–84.
22. Sonowal, G.; Kuppusamy, K.S. SmiDCA: An Anti-Smishing Model with Machine Learning Approach. *Comput. J.* **2018**, *61*, 1143–1157. [CrossRef]
23. Jain, A.K.; Gupta, B.B. Feature Based Approach for Detection of Smishing Messages in the Mobile Environment. *J. Inf. Technol. Res.* **2019**, *12*, 17–35. [CrossRef]
24. Jain, A.K.; Gupta, B. Rule-Based Framework for Detection of Smishing Messages in Mobile Environment. *Procedia Comput. Sci.* **2018**, *125*, 617–623. [CrossRef]
25. Almeida, T.A.; Silva, T.P.; Santos, I.; Hidalgo, J.M.G. Text normalization and semantic indexing to enhance Instant Messaging and SMS spam filtering. *Knowl. Based Syst.* **2016**, *108*, 25–32. [CrossRef]
26. Yadav, K.; Kumaraguru, P.; Goyal, A.; Gupta, A.; Naik, V. SMSAssassin: Crowdsourcing Driven Mobile-Based System for SMS Spam Filtering. In Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, Phoenix, AZ, USA, 1–2 March 2011; HotMobile '11; Association for Computing Machinery: New York, NY, USA, 2011; pp. 1–6. [CrossRef]
27. Agarwal, S.; Kaur, S.; Garhwal, S. SMS spam detection for Indian messages. In Proceedings of the 2015 1st International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 4–5 September 2015; pp. 634–638.
28. Almeida, T.A.; Hidalgo, J.M.G.; Yamakami, A. Contributions to the Study of SMS Spam Filtering: New Collection and Results. In Proceedings of the 11th ACM Symposium on Document Engineering, Mountain View, CA, USA, 19–22 September 2011; DocEng '11; Association for Computing Machinery: New York, NY, USA, 2011; pp. 259–262. [CrossRef]
29. Chen, T.; Kan, M.Y. Creating a live, public short message service corpus: The NUS SMS corpus. *Lang. Resour. Eval.* **2013**, *47*, 299–355. [CrossRef]
30. Zhang, W.; Yoshida, T.; Tang, X. A comparative study of TF*IDF, LSI and multi-words for text classification *Expert Syst. Appl.* **2011**, *38*, 2758–2765. [CrossRef]
31. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of the 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, AZ, USA, 2–4 May 2013.
32. Pennington, J.; Socher, R.; Manning, C. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1532–1543. [CrossRef]