applied sciences

MDPI

Article

An Algorithm for the Detection of Hidden Propaganda in Mixed-Code Text over the Internet †

Andrea Tundis [1,*], Gaurav Mukherjee [2] and Max Mühlhäuser [1]

[1] Department of Computer Science, Technische Universität Darmstadt, 64289 Darmstadt, Germany; max@tk.tu-darmstadt.de
[2] Brox IT-Solutions GmbH, 70771 Stuttgart, Germany; gmukherjee@brox.de
* Correspondence: tundis@tk.tu-darmstadt.de; Tel.: +49-6151-16-23205

† This paper is an extended version of the paper published in the 15th International Conference on Availability, Reliability and Security (ARES 2020), Virtual Event, Dublin, Ireland, 25–28 August 2020; Article 76, pp. 1–7.

Abstract: Internet-based communication systems have become an increasing tool for spreading misinformation and propaganda. Even though there exist mechanisms that are able to track unwarranted information and messages, users made up different ways to avoid their scrutiny and detection. An example is represented by the mixed-code language, that is text written in an unconventional form by combining different languages, symbols, scripts and shapes. It aims to make more difficult the detection of specific content, due to its custom and ever changing appearance, by using special characters to substitute for alphabet letters. Indeed, such substitute combinations of symbols, which tries to resemble the shape of the intended alphabet's letter, makes it still intuitively readable to humans, however nonsensical to machines. In this context, the paper explores the possibility of identifying propaganda in such mixed-code texts over the Internet, centred on a machine learning based approach. In particular, an algorithm in combination with a deep learning models for character identification is proposed in order to detect and analyse whether an element contains propaganda related content. The overall approach is presented, the results gathered from its experimentation are discussed and the achieved performances are compared with the related works.

Keywords: propaganda detection; mixed-code identification; text analysis; machine learning; internet-based crimes; cyber terrorist networks; cyber-criminality

Citation: Tundis, A.; Mukherjee, G.; Mühlhäuser, M. An Algorithm for the Detection of Hidden Propaganda in Mixed-Code Text over the Internet. *Appl. Sci.* **2021**, *11*, 2196. https://doi.org/10.3390/app11052196

Academic Editor: David Megías

Received: 29 December 2020
Accepted: 24 February 2021
Published: 3 March 2021

1. Introduction

Language is a medium of expression in human society. Communication is a form of social interaction that involves exchange or negotiation of information primarily through the use of the *language* [1]. As it is discussed in [2], languages are socio-cultural as well as individual products and, as such, they change over time, keeping up with the evolution of the social environment. IT has revolutionised the communication way [3] as evidenced by the amount of information generated and spread via social media, such as Facebook, Instagram, Twitter and YouTube. In particular, the growth of the Internet-based communication has become one of the basic necessities of the world. Thanks to such technology, many users attempt to use the Internet even for malicious purposes. The high number of users make it very easy for any kind of information to diffuse very quickly so as to influence the perception of the user itself [4]. This kind of information, mostly known as *viral messages*, is often not verified, from its authenticity point of view and it can result in outrage or hysteria. Furthermore, as discussed in [5], such methods can be used to shape public opinion, finances or even create panic in the society, that means it can be treated as an *infection of the mind*. Terrorist organisations, for example, use online networking to generate and promote propaganda based content to legitimise or instruct illegal activities to the public, as the huge amount of online registered users make the information spreading

faster and easier [6–8]. That is why criminals exploit the Internet as an active means to perpetrate some crimes [9–12].

Especially, online social media is skillfully utilised by terrorists as a tool of persuasion and behaviour change (by including political opinion, marketing and sales, humanitarian and community acts) in order to cause significant damage as well as social unrest in sensitive places by using fake news and propaganda, which might result in individuals joining terrorist networks [13–16].

It is not trivial to monitor all ongoing activities even if such cyber communication takes place in the clear, without adopting any anonymous or encrypted communication techniques [17,18]. It is difficult to recognise it, especially if it is structured in such a fashion that avoids raising the necessary security flags while reaching the intended targets. The writing style, for example, is a strong way to spread propaganda while avoiding the authorities [19]. It can represent a big obstacle from a machine perspective, to automatically detect and correctly interpret a message written in clear as a plain text. There are a few ways to enable it, such as code-mixing and mixed-code text methods. *Code-mixing* is the term used when a person writes something using two different languages especially of two different scripts. For example, "Aap kaise ho?" is a text written for a Hindi person but using a Roman script. This is common parlance called a *Hinglish* text, and likewise: *Romanji* (Japanese in roman script), *Benglish* (Bengali in Roman script), *Roman Urdu* (Urdu in Roman script) and so on. Furthermore, when using code-mixing, the spelling of words tends to differ for different persons, while the variation of the phonetic usage of the Roman letters can result in varying interpretation adding to its complexity. The major challenge relies on properly monitoring these harmful messages circulating in the social network and removing or delegitimising them. This is a big concern in multilingual societies like in India where correctly identifying a potential threat is a huge hurdle. This creates a challenge for the machine to properly "understand" the semantics. In countries like India where multiple languages are used and social media is predominantly expressed in Roman script, a wide usage of mixed code to converse on a daily basis can be observed. However, this poses a problem to recognise the intended message in a mixed code environment. On the other hand, *mixed-code* text consists of using special characters to write words in such a way they would graphically resemble the intended words but from the machine point of view, it is considered a no-sense sequence of symbols. The question should be asked—What is the function of mixed-code text? The answer lies in the simplicity of the method. Mixed-code text only requires the user to use a bit of creativity and on the reader's part only the knowledge of the language as the reader in a flow would easily understand the intended text. However, a machine will take the text literally and would check for its content which in most cases would be dismissed as garbage. Consequently, there is a need for careful deliberation to re-evaluate text and to provide a method for improving its interpretation.

In this panorama, this article, which is an extended version of [20], focuses on the detection of hidden propaganda in mixed-code text, by proposing an algorithm for supporting the analysis of suspicious mixed-code text, based on the optical character recognition (OCR) analogy [21]. Specifically, a segregation approach is adopted to analyse the characters' combinations, which in turn are transformed into images. A character recognition model, based on machine learning techniques, is trained in order to recognise such modified letters of the alphabet. The process iterates through all possible combinations to find the intended word. A second model deals with the text classification, to determine whether a text contains hidden propaganda messages or not. Since such problem falls within the detection field, accuracy, precision, recall and f1-score have been employed as evaluation criteria for assessing the proposal's performances for the detection of hidden propaganda in mixed-code. From the conducted experiments, it results that each of those metrics reaches more than 91%, as a symptom of good performances, that are inline in comparison to the related works described in Section 2.

The rest of the paper is structured as follows: Section 2 provides an overview of the related works by distinguishing between mixed-code analysis and propaganda detection. An overview on optical character and image recognition, which represent the enabling technologies, have been introduced in Section 3. In Section 4, the most common identified mixed-code categories, the overall adopted analysis process along with the proposed algorithm are elaborated. Whereas its evaluation is presented and discussed in Section 5. Finally, the conclusion and future works are highlighted in Section 6.

2. Related Work

This section provides an overview on the related scientific contributions both in the field of (i) mixed-code analysis (see Table 1), which takes place when, in a conversation or communication, two distinct languages, syntax or written forms are employed [22]. It is generally used to overcome linguistic shortfalls, to help in understanding groups' dynamics and characteristics, as well as to grab various ethnic peculiarities which can define their group identities [23]; (ii) online propaganda detection, which is devoted to identify specific content that aims to incite the masses by influencing their mindset. An overview of the most popular related works is presented in Sections 2.1 and 2.2 respectively.

2.1. Mixed Code Analysis

A contribution centred on machine translation of the so called *Hinglish* text is proposed in [24], where Hinglish represents a communication way that uses a mix of Hindi and English terms within a conversation. The authors proposed a system centred on machine learning in order to translate Hinglish script. They discussed how the constraints of mixed code should be considered when defining a model in order to obtain an accurate interpretation of the mixed-code. They claimed that acceptable results were achieved in more than 90% of cases; however, any metrics and the quantitative results related to the evaluation approach have not been presented. Furthermore, the application context in which such experimentation was conducted has not been clearly specified.

In [25], the hate speech detection problem in code-mixed texts on social media has been faced, by exploiting a Hindi-English code-mixed dataset consisting of online posts on Twitter. Different types of textual-based features have been used to define a detection model: n-grams, word n-grams, punctuations, negation words and hate lexicon. The annotation consisted of two Factor-World Level and Hate/Normal speech, which were used to indicate which words of the text were English and which words were Hindi. N-gram was adopted to support the feature extraction (both at character and word level) in order to detect relevant words. Furthermore, punctuation and negative words were considered as a benchmark to determine potential hate speech or not. The experiment showed that the best result was achieved by combining all such features to train a SVM, thus achieving 71.7% accuracy.

In [26], an approach for language identification on social media has been elaborated. A dataset with Facebook posts and comments that contains code mixing between Bengali, English and Hindi has been built. The authors considered a text as mixed-code not only when the script was different but also either when the words of different languages were used randomly or sentences switch languages within a single text or paragraph. Different analysis techniques have been experimented, some based on a unsupervised dictionary approach and others on supervised word-level classification. However, no clear evaluation has been provided. Instead, the work proposed in [27] focused on the classification of offensive tweets written in Hinglish language. A Twitter dataset containing tweets in Hindi-English code switched language has been collected. The dataset has been organised into three main classes: nonoffensive, abusive and hate-speech. Convolutional Neural Networks (CNNs) combined with transfer learning have been used in the classification process by reaching 83.9% accuracy.

In [28], the identification of hate speech from code-mixed text on social media has been faced, by using a Hindi-English mixed code corpus as the initial baseline. A Long Short

Term Memory (LSTM) technique, based on subword level, has been mainly considered. The idea behind the subword level selection is to identify the root words and to match their variations. This is because in a mixed code setting the variation may differ for each writer. The hierarchical LSTM model as well as Random Forest (RF) and Support Vector Machine (SVM) have been experimented by using subwords phonetic. As Hindi is a phonetically accurate for writing, a phonetic input of the text has been also provided. The words were divided into consonant-vowel sequences and these sequences form the base of their attention model. This enabled the model to determine both the words that make up the vocabulary and to match the phonetic to such words as well. The results showed that the hierarchical LSTM reached the highest Recall and F1-score, whereas the SVM reached the highest accuracy, equal to 70.7%.

In [29], different analysis techniques for supporting sentiment analysis of textual information, after having normalised the influence of multiple languages in mixed-code script, have been experimented. In particular, several machine learning techniques, which are able to characterise the text by determining the polarity of a text have been used. The mixed code text used to train the model was initially identified to mark the words as Hindi or English. Such model used an Artificial Neural Network (ANN) to identify the affinity of the text and to mark it with values from "very positive" to "very negative" (−5 to +5) so as to provide a sentimental score. The results showed circa 80% accuracy. However, regarding the meaning of polarity, it is not clearly specified whether positivity means happiness or joy and negativity anger or sadness.

Another research contribution related to code-mixed text centred on sentiment analysis is described in [30]. It is based on the combination of two models: N-gram probabilistic model, which was called Multinomial Naive Bayes (MNB), and LSTM model centred on Tri-gram characters. In particular, the LSTM model has been used to cache deep sequential patterns contained in the text, whereas the low-level word combination of keywords has been faced with the MNB model, in order to compensate grammatical inconsistencies. The overall achieved accuracy was equal to 70.8%. Whereas in [31], the authors have extended the NLP techniques to detect humour in a Hindi-English mixed code environment. N-gram, Bag-of-Words (BoW), Common words and hash-tags have been used and experimented on a manually annotated dataset. SVM, Random Forest and Naïve Bayes have been experimented with a highest accuracy of 69.3%. In [32], instead, another NLP technique to normalise words in a code-mixed environment—as the spelling tends to vary for different persons—has been proposed. A Skip-gram model, which clusters the words of variation to deliver the base word as output in the normalised text, has been used. The results have shown an accuracy level of 71% and a F1-score equals to 66%.

Table 1. Mixed code related works.

Ref	Objective	Approach and Contribution	Evaluation	Dataset
[24]	Mixed-code translation	Unspecified	Accuracy: 90%	Custom
[25]	Dataset Hate speech in Mixed code	Textual-based features, SVM	Accuracy: 71.7%	Custom
[26]	Language detection	Data collection	Unspecified	Custom
[27]	Tweets classification	Data collection, CNN	Accuracy: 83.9%	Custom
[28]	Hate Speech identification	SVM, LSTM, RF	Accuracy: 70.7%	Custom
[29]	Sentiment analysis	ANN	Accuracy: 80%	Custom
[30]	Sentiment analysis	N-Gram, MNB, LSTM	Accuracy: 70.8%	Custom
[31]	Sentiment analysis	N-Gram, BoW, SVM, RF,NB	Accuracy: 69.3%	Custom
[32]	Text normalisation	Skip-gram model	Accuracy: 71% F1-Score: 66%	Custom

A clear aspect that emerged from the above identified related works is that such previous research studies have focused on mixed-code analysis by mainly dealing with the language translation. Such research direction differs from the purpose of this current work, as it focuses on the detection of hidden propaganda in mixed code as *leet text* [33], that is a particular form of expression, which is primarily used on the Internet. The distinguishing characteristic of such textual expression form lies in the combination of characters and symbols of the computer keyboard in order to graphically represent a text. This writing style makes the *leet text* a powerful way of communicating and spreading propaganda, as it allows for easy bypass of its automatic identification.

2.2. Online Propaganda Detection

Online propaganda is a modern way of conducting campaign centred on IT-based tool and in particular online social networking. Twitter represents one of the most used medium for propaganda and it is heavily utilised by extremist and terrorist groups to reach the mass. The recent interest in detecting online propaganda is documented by the different research efforts conducted in this field (see Table 2).

In [34], for example, the authors focused on the detection of extremist ISIS groups, by proposing a method to actively identify tweets containing propaganda related content. The method aimed to recognise not only patterns containing specific Hashtags but also user accounts by profiling them so as to enable their potential predictions. The method was centred on the Term Frequency (TF) of the suspected words in order to derive relevant information, whereas a Regression-based Neural Network (RBNN) has been used as classification algorithm, without quantifying the achievements.

Similarly, the detection of radical content was faced in [35], through the definition of a model. It was used to analyse the text produced by the users from the psychological perspective by considering specific behavioural patterns. In particular, the authors utilised TF-IDF to identify suspicious terms related to radical behaviour, which were used to train a Random Forest (RF), a Support Vector Machine (SVM) and a K-Nearest Neighbor (KNN) classifier by obtaining 94% accuracy, as the highest result.

A deep learning model for supporting the identification of Sunni extremist propaganda via text analysis is presented in [36]. Here, word2vec and doc2vec techniques have been adopted to detect specific relationships among extremist-related terms which in turn were used to identify and classify new text as propaganda or not. Among the conducted tests, Artificial Neural Networks (ANNs) showed the best classification results by providing accuracy and precision of about 90%.

In [37], the authors focused on the creation of a free available dataset as a benchmark to support the propaganda detection research activities, that has been used in different research competitions related to propaganda detection. It contains sentences that are annotated from experts as propaganda and not. In addition, the paper presents preliminary classification results by achieving about 63% of accuracy by using the Neural Network classification. Using the same database, in [38] the authors exploited the BERT classification to identify single nonpropaganda rows by employing ELMo, BERT and RoBERTa approach, by achieving 79% recall, 66% precision and 60% f1-score.

Whereas in [39], the authors focused on the assessment of radical and extremist online propaganda, by proposing a pyramidal conceptual model that enables to distinguish propaganda related content at different level of radicalisation. The model is centred on three sociological human aspects which characterise traits of radicals and terrorists. A preliminary experimentation was carried out in order to illustrate how to allocate propaganda items to the pyramid model.

Table 2. Mixed code related works.

Ref	Technique	Contribution	Evaluation	Dataset
[34]	RBNN, TF	Hashtag-centred detection of ISIS groups	Unspecified	EMNLP19
[35]	TF-IDF, SVM, KNN	Text-based detection of Radical users	Accuracy: 94%	Acquired
[36]	Word2vec, doc2vec, ANN	Text analysis of Sunni extremist content	Accuracy: 90%	Acquired
[37]	Neural Network	Data collection and text analysis	Accuracy: 63%	Custom
[38]	ELMo, BERT, and RoBERTa	Text analysis propaganda detection	Precision: 66%	EMNLP19
[39]	K-Means	Classification of Pro-ISIS users	Unspecified	Acquired

From the above reviewed papers, it resulted that online propaganda detection is an active research field towards analysis of extremist related content. However, it emerged that all previous research effort are limited on the analysis of natural language of a plain-text. Our scope is instead to deal with a form of expression which is not directly attributable to plain-text but which is made up to "hide" the real message, by composing different symbols to graphically resemble the standard alphabet.

3. Background

For the sake of completeness and to make the paper's content self-contained, an overview on basic concepts regarding character recognition and writing ways is given in Sections 3.1 and 3.2 respectively.

3.1. Optical Character Recognition

The objective of character recognition is to enable a machine to recognise characters by using optical devices, such as a camera. The machine should be able to capture the image of the character and associate it with its corresponding symbolic identity [40]. Optical Character Recognition (OCR) technique can be found as early as 1900 where the scientist Tyurin successfully implemented the idea and with the invention of modern computers, the research into this field has increased a lot. The OCR was initially developed to assist the visually challenged individuals to read the text, and then it was adopted into other application context such as banking, health and security in combination to Machine Learning (ML). A common OCR application is handwriting recognition, centred on smart devices that are capable of transforming handwritten text into its digital counterpart [41]. OCR is utilised to recognise languages of different origins not only in Roman script [42] but also in the case of degraded language documents. Another use of OCR to recognise handwriting is presented in [43], where the authors used the n-gram model, which produces groups of words that overlay on each other. The authors extended such model and utilised the likelihood score, from a statistical machine translation system, as main feature, so as to be able to capture the image of the words and to translate each image into other languages.

A further application of OCR is provided in [44], which describes a method to recognise Bangla handwritten numerical characters using local binary pattern. Whereas in [45], the authors used OCR technique to detect printed English and Fraktur text using Long Short Term Memory (LSTM). The authors described that the handwritten content may vary in the position and baseline of the text, as a consequence a text line normalisation approach has been used to uniformly position the text. Such normalisation approach was based on an dictionary, containing data generated on the basis of the connected component shapes and associated baseline. For recognition and classification, the authors implemented 1-D Bidirectional Long Short Term Memory model, in order to establish the ground-truth align-

ment by using a forward-backward algorithm, which also provides a decoding mechanism to map the frame network upon a sequence of symbols.

Further applications of OCR, for supporting the recognition of handwritten digits, are presented in literature. For example in [46], the use of Random Forest, Decision Tree and Hoeffding Tree as classifiers have been proposed to build a tool that can easily recognise the digits. For this, they utilised the handwritten digit dataset of MNIST5, that is preprocessed before being used, in order to extract potential characterising features. In such work, a comparative study among the different classifiers has been conducted by concluding that the Hoeffding tree was the most performing one by reaching 73% accuracy. Another approach to support character recognition was based on the use of the Tesseract OCR open source engine to train a Tamil OCR model [47]. In particular, a segmentation approach was adopted by using a box file system. Each character was numbered, so that the number of boxes was equal to the number of training characters. Starting from such boxed character a classifier has been trained by using various images with different font size type. The evaluation of the proposed OCR system was performed using 20 scanned images from 20 ancient Tamil books and over 14,000 characters, by obtaining 81% accuracy.

A method for the recognition of Roman Script and English language was proposed in [21], based on Artificial Neural Network (ANN) and Nearest Neighbour (NN) to detect and interpret scanned English documents in three different font types. Using such method, the authors achieved 98% accuracy, by experimenting it on a dataset consisting of English alphabets in different fonts created by themselves, which is not available. As a consequence, the database could have been biased. Unfortunately, this is not verifiable, as no information is available neither regarding such dataset nor on the adopted evaluation approach.

3.2. The Art of Writing

When we talk about *Word Art*, we typically refer to the graphic structural aspect with which words are written, commonly known as calligraphy. It is often considered to be the craft of writing text in an appealing form, and it is regularly used in font design, typography, logo and graphic design. Word Art can be depicted as a text modifier which includes visual enhancements to text like shadows, outlines, colors, skew and 3-D effects to make it more attractive to the user. *Text fonts* are also an integral part of the modern writing. Since there are multiple fonts to choose, a user can use a font to describe the purpose of the text and to emphasised the mood of the article. For example, a user might choose the calligraphy font for invitations and calibri font for formal documents or even create custom fonts to describe a unique style of presenting a textual content.

In [48], a distinction between the term of *Computer Art* and *Digital Art* is discussed. *Digital art*, which is considered more general, represents the use of the computers capacity to convert different media types (such as Music, Pictures, Movies, Story and Text) into a digital form and to process them for multiple purposes. For example, a digitally encoded video of an event can be integrated using music of another origin while displaying text of another. The integration is possible into one, as all of them are fundamentally a series of binary code. In contrast to this, *Computer Art* is the art created by exploiting any external methods, by only using the available tools. The initial stages were just art with characters available with the standard keyboard to create the images. These types of images are called ASCII art, which rely only on the use of standard characters to make images, due to the lack of other graphical resources and professional tools. The use of ASCII art is still prominent today in chats and forums like Reddit, Stackoverflow as well as by players in a multiplayer game communities. In addition, the symbols used to create the full picture are typically linked to the meaning and the context of the pictures itself.

As argued in [49], *ASCII art* is a more complex art form than the intended originally ones. It requires precision to provide the proper alignment of the text to avoid any misinterpretation, which in turn would give inconsistent result for recognition techniques like optical character recognition (OCR). Furthermore, with the help of the computers today, it is easier to provide text in multiple forms and fonts, as the operating systems offer built-in

writing systems, that support a multitude of fonts, design styles as well as design tools for creating the so popularly called "Word Art".

Another form of writing text that combines "Text Art" and "Font" is called *Leet Text*, typically of websites, which uses characters found on the keyboard to type out text, by playing with the similarities of the characters to be represented. Since the language used on the Internet is predominantly English and the keyboard of most users have Roman characters on it, it is generally used for Roman scripts. Other languages are also possible but the scope may be limited. The term "leet" comes from the word "Elite" and it would be used online as a symbol of proficiency in certain fields and especially in Gaming. The representation of the word "leet" itself in leet text is represented as "l337". Leet is considered similar to ASCII or Word art and it is also synonymous of emotions. Leet code is based on conventions but no standard rules are defined.

Since leet speak is dependent on a language base for communication, it does adhere to its grammatical constraints. However, leet speak has some of its own unique sets of texts which are generally known to its users. It uses misspellings and abbreviations to convey a word or message and it uses numbers as valid letters to write a word. Furthermore, it has an extensive use of special characters to depict certain characters. Leet speak is also used to provide censorship to certain text which can be provocative or hate inducing. These texts require precision and practise to be understood and interpreted. More insights about the complexities of leet speak have been provided in [33]. For example, it is mentioned that there are many varieties of leet speak and that many others can be developed on the basis of the specific context.

4. Propaganda Detection in Mixed-Code Text

In this section, the proposed method for supporting the detection of online propaganda hidden in mixed-code text is elaborated. In particular in Section 4.1, first different mixed-code types are presented, and then the adopted research process is briefly introduced. In Section 4.2, the proposed normalisation algorithm for transforming mixed-code into standard text is elaborated, whereas the approach for the analysis and classification of propaganda-related content is described in Section 4.3.

4.1. Writing Styles and Mixed-Code Categorisation

The way of expressing thoughts in written form can take place differently. Indeed, not only a text is a composition of symbols and characters that are used to build a word in order to convey a message, but it can also be used to depict some phonetic structures or art forms. The basic way of using writing is called *Text on Document (ToD)*, which is characterised by the use of standard alphabetic characters, that is, the symbols belong to a specific language, whose structure is governed by a set of grammar rules, semantics and vocabulary. *ToD* is supposed to contain an explicit message easy to read and to understand. *Text in Visual Media (TiVM)*, instead, represents the use of the text, in graphical representations, to improve the visual information by imitating sounds or emotions. It might not necessarily follow grammar rules or having any semantic identity. Whereas, a *Text as Art Form (TaAF)* aims to emphasise the artistry of the writer, by using forms or symbols that are not part of the alphabet, in order to represent a code or a coded message in clear.

Mixed-code text belongs to *TaAF*, and according to the conduced research, we classified it into three main types (i.e., *Single row—multiple language*, *Single row—single language*, and *Multiple rows*) by considering two main parameters, as depicted in Figure 1: (i) the *language*, that is, whether the mixed-code text contains multilanguage factors or writing styles linked to one or more languages and (ii) the *graphical writing style*, that is, whether a standard row-based writing style is followed or multiple rows are used in order to graphically represent the alphabetic characters.

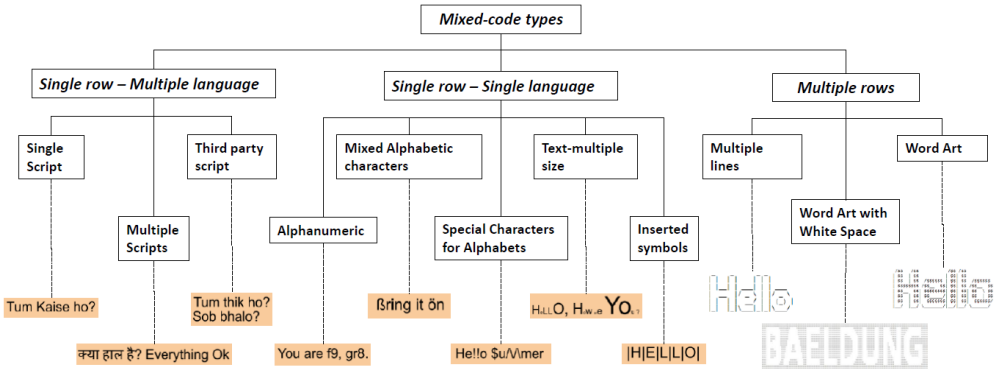


Figure 1. Mixed-code categories.

Moreover, for each of them further subcategories have been identified, which are described in the following:

- *Single Script*: the text is written in a particular script, but the basic language might be different. For example, when Hindi is written in Roman script, that means that the script used for Hindi is Devnagri but the usage of Roman script is adopted to write Hindi.
- *Multiple Scripts*: the words retain their original forms and scripts but different languages are adopted. The grammar is difficult to grasp; however, the greater number of words used with a particular language generally dominates the grammar rules.
- *Third party script*: the text is based on two different languages, but the script utilises a third party script whose grammar fluctuates between the two used languages.
- *Alphanumeric*: the mixed code language utilises alphabets and numbers to develop the scripts. It is important that the phonetics of the numbers are in accordance with the reference language.
- *Different Alphabetic/Language characters*: the text uses the intended script of the language, but there is also sporadic use of symbols from different languages. The symbols are not part of the language characters but do resemble some of the characters to provide the suitable replacement.
- *Special Characters for Alphabets*: the text uses the intended script of the language but it also utilises the special characters in the system to create the alphabet required to sell out the word. The use of special characters may depend on the characters it is required to replace. The characters can be depicted using a single special characters or multiple special characters. This also depends on the code that is being used, i.e., if there is a predetermined set of rules for such replacement, then the special character selection depends on that.
- *Multiple size*: the text is written over different height and width. The text may have some characters that vary in size, so as to make the text inconsistent with the standard practise of writing This is generally done to add more effect to the text.
- *Inserted symbols*: in this writing form the text is written normally, but it contains symbols inserted among the basic characters to produce a special effect or to decorate a word.
- *Multiple lines*: this is the way of using special characters to build a word or a letter that spans over multiple lines of the text. In this case, different symbols are used individually over multiple lines in such a way that they give the visual effect that a text is represented.
- *Word Art*: this is a variant of the previous one, by using letters or special characters not only to provide a textual information but rather to see the picture formed by carefully aligning texts over multiple lines. This type of arrangement is also called ASCII art.
- *Word Art with white space*: the text is written over a large area is such a way that the white area, left intentionally empty, gives the appearance of a text. It means that the text or characters are used to enclose the white space in such a manner to represent the word.

Due to the high diversity of such subcategories, their analysis requires in turn an individual and specific study. That is why the rest of this research work focused on one of them and in particular on the *Text as Art Form* called "Special Characters for Alphabets", which has not been investigated yet.

Figure 2 depicts the adopted analysis process. In particular, given a textual Input, a first check verifies whether the text, which is from now on referred to as *TaAF*, contains special characters. If so, the mixed-code text *Normalization Algorithm* is applied, so as to transform the *TaAF* into a standard computer readable text. At this point, the transformed text can be further analysed. In particular, the normalised *TaAF* is given in input to the "Propaganda detection classifier", trained on a specific dataset, which is able to discriminate whether it is Propaganda or Nonpropaganda related. If the textual Input is not a *TaAF*, the propaganda detection step can be directly applied. The next sections elaborate the proposed "Normalization Algorithm" as well as the training of the "Propaganda detection classifier".

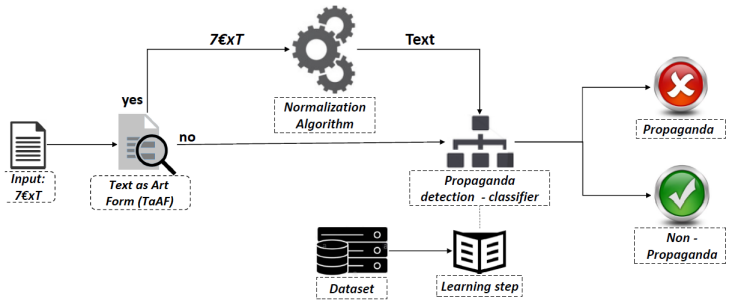


Figure 2. Research approach: data management, phases and work-products.

4.2. A Mixed Code Text Normalisation Algorithm

This section presents the proposed algorithm to normalise mixed-code text related to "Special Characters for Alphabets", whose pseudocode is reported in Algorithm 1. It consists of four main steps so labelled: (i) *Text Segregation*, which splits words by generating different subset of symbols, (ii) *Character Transformation*, that aims to derive, from each subset of symbols, letters of the alphabet, (iii) *Word Selection*, which deals with the generation and selection of existing words according to a dictionary on the basis of the derived letters, (iv) *Sentence Reconstruction*, which aims to replace the initially mixed-code text with the selected words in order to obtain a meaningful standard textual sentence.

A more detailed description of each step is exemplified through a simple reference example. In particular, given the sentence *S* in input, as it is represented through the equation (1), which contains a mixed-code text called from now on *Art Form Word—AFW* (e.g., "F{}{}T"), each step of the proposed normalisation algorithm is elaborated in the following.

$$S = My\ \text{F\{\}\{\}T}\ is\ cold \tag{1}$$

Text Segregation step: this step has a syntactic function, and it works on the single words. It aims to create groups of symbols starting form the *AFW* in consideration. In particular, after splitting the sentence into subwords and identify those containing nonalphabetic characters, for each *AFW* the algorithm reworks its sequence of characters by grouping them differently, and generating several combinations, as it is shown in Table 3. In this approach a recursive operation is used to make combinations. It iterates through all the characters to make all the possible combinations. The process keeps one character constant and makes combinations with the rest of the remaining characters. Then it takes the first and second characters together and then makes combinations with the rest of the remaining characters. This operation is carried out exhaustively. In is assumed that, the order of the symbols through which the mixed-code text is built, and as a consequence the order of the combinations, reflects the order of the letters in the standard word. It means that the word has been written in the order it is meant to be read and it is not an anagram.

Algorithm 1: Preudocode for text normalisation

```
Data: String S={W_1, W_2, ..., W_{k-1}, W_k };
Placeholder={P_1, P_2, ..., P_{k-1}, P_k };
CandidateWords cw[];
Integer i=1; j=1; z=1;
Word w; CharactersGroup cg; String S_Norm;
SegregationMatrix SM [K][N][M];
TransformedCharacter TC [M];
Text Segregation step: while i<= S.size() do
    // - as long as there are words
    w=S.getWord(i); //take the i-th word
    if w.isNotMixedCode() then
        //if the i-th word is a regular word
        S_Norm.append_i(w); //it does not have to be normalised
    else
        S_Norm.append_i(P_i); // create a place holder in the i-th position
        SM_{k=i}.segregate(w); // segregate the i-th word
    end
    i++;
end
Character Transformation step: while j<= SM.GroupSize() do
    // for each segregated word
    cg=SM.getGroup(j); // select the j-th group of characters
    if cg.isAnAlphabeticCharacter() then
        //if it is one of the standard alphabetic letters
        TC[j]=cg; //apply the rule-1 by considering it as it is
    else
        TC[j]=convertIntoSingleCharacter(cg); //apply the rule-2 by using OCR to transform the j-th cg group of
          symbols into one alphabetic character.
    end
    j++;
end
k=1;
Word Selection step: while k<=S.size() do
    z=1; while z<=SM(k).CombinationNumber() do
        //as long as there is a combination for each word
        c_z=SM(k).getCombination(z);
        c_z.replaceGroupsWithTheTransformedCharacter(SM,TC);
        if c_z.string().isAnExistingEnglishWord() then
            //if the generated word is not part of the English vocabulary, it is not considered
            SM(k).removeCombination(c_z)
        else
            // otherwise it become a candidate placeholder replacement
            cw.addCandidateWord(k,c_z)
        end
        z++;
    end
    k++;
end
k=1; Sentence Reconstruction step: while k<=S.size() do
    S_Norm.replacePlaceholder(k,cw(k).candidateWord());
    k++;
end
Result: S_Norm;
```

Table 3. An example of segregation with Art Form Word (AFW) = "**F{}{}T**".

Combination	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	...	Group M
1			F	{}	{}	T			
2			F	{}			{}T		
3		F{						}{}	T
...									
N	F{}{}								T

Character Transformation step: in this step, for each combination generated in the previous step the following transformation rules have been defined:

- *rule-1*: if a set of grouped characters coincides exactly with one standard alphabetic character (e.g., a, b, c, ..., Z), then it is considered like it is. For example, in all generated combinations when F and T are not grouped with any other symbol, they are interpreted as F and T.
- *rule-2*: if a set of characters consists of one not standard alphabetic character, including numbers (such as !, ?, @, 0, 1, 2 and so on), or it consists of a combination of one or more characters (e.g., F{, {}, {}T, }{, and so on) then an "optical character recognition—(OCR)" technique is applied. The idea is to emulate the human behaviour, from an optical point of view, so as to automatically recognise individual characters and letters. In particular, first each group of characters/symbols is transformed into images. Then machine learning techniques are exploited to recognise and associate each group of characters/symbols to a standard letter of the alphabet by trying to identify a match within the letters of the standard (English) alphabet. As it is shown in Figure 3, for each group of symbols different letters of the alphabet can be generated; however, the one with the highest similarity score is then chosen.

A classifier for character recognition has been trained by using the dataset provided in [50], which consists of a collection of character images of the alphabet belonging to the 26 English characters both handwritten and typed through a computer. Furthermore, other additional characters of other languages (such as *ü, Ü, ä, Ä,ö, Ö, ß, Ø, ø* and so on) have been further created in order to extend and enrich the initial dataset.

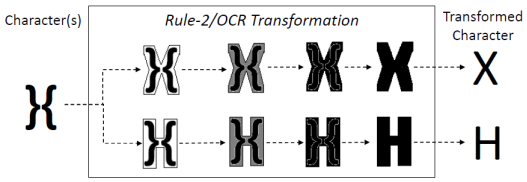
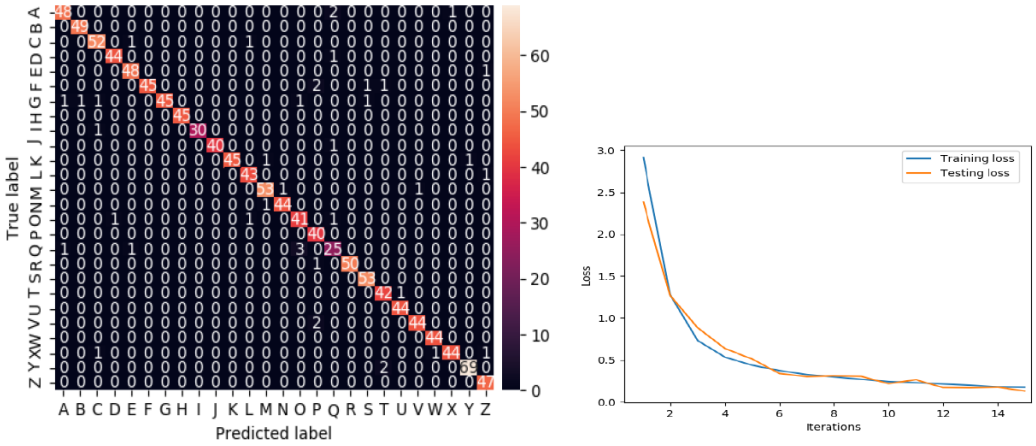


Figure 3. An example of a recognition rule.

To create such images, the use of python and the drawing library of Pillow [51] was used for creating the images. It aimed to incorporate certain non-English characters belonging to other languages. The images are made by using a dimension of 200 × 200 px for the width and height. They were generated in a gray scale, because for the character identification purpose RGB, the colour does not provide any additional information. The best performance for the character recognition have been achieved by using a Convolutional Neural Network (CNN). A training and testing set with a ratio of 80:20 and number of epochs equal to 15, were used to configure the CNN. Its performance, in terms of Evaluation Metrics, Confusion Matrix as well as training vs. testing loss, is depicted in Figure 4.

Metric	Measurement
Accuracy	94.29%
Precision	94.31%
Recall	94.30%
F1 Score	94.22%

(a) - Evaluation Metrics



(b) - Confusion Matrix (c) - Traning loss vs. Testing loss

Figure 4. Performances in the recognition of single characters.

Table 4 shows an example of character transformation by applying the above mentioned rules. For example, only the letters of the groups 3, 6 and M have been used as they are, whereas the rest of the groups generated from the segregation process have been first transformed into images and then mapped with single letters.

Table 4. An example of Character Transformation.

Group	Character(s)	Recognition Rule	Transformed Character
1	F{}{}	Rule-2/OCR	X
2	F{	Rule-2/OCR	A
3	F	Rule-1	F
4	{}	Rule-2/OCR	O
5	{}	Rule-2/OCR	O
6	T	Rule-1	T
7	{}T	Rule-2/OCR	E
...	}{}	Rule-2/OCR	L
M	T	Rule-1	T

Word Selection step: this part of the algorithm aims to derive candidate words that can replace the related *AFW*. First, for each *Combination* generated during the *Text Segregation step*, each group of symbols is replaced with its related *Transformed Character* obtained within the *Character Transformation step*. As it is shown in Table 5, a set of words which are syntactically written only with letters of the alphabet are built.

Table 5. An example on derived words.

Combination	Group	Transformed Character	Derived Word
1	F, {}, {}, T	F, O, O, T	FOOT
2	F, {}, {}T	F, O, E	FOE
3	F{, }{}, T	A, L, T	ALT
...	...	...	...
N	F{}{}, T	X, T	XT

The existence of such derived words is then checked against an English dictionary. Through this step, only the words that are part of the standard English language are selected and, as a consequence, used for further analysis. Figure 5 shows graphically an example of word selection, on the basis of their existence in the English dictionary.

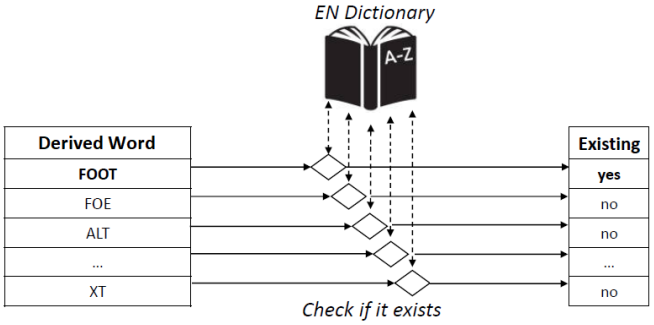


Figure 5. Selection of the derived words based on a dictionary.

Sentence Reconstruction step: this is the last part of the algorithm which is centred on two main input parameters: (i) all existing words that have been generated and selected, that means those that are part of the English dictionary and (ii) the initial sentence *S* containing a *placeholder* for each identified *AFW*. In particular, from the previous steps for each *AFW* a set of potential words have been derived and selected. As it is represented in Figure 6, at this point, the algorithm provides in output different versions of the normalised sentence $S_{Normalized}$={$S_1, S_2, ..., S_m$}, this means, all sentences which are possible to reconstruct by replacing the *placeholders* with the generated words in all possible combinations.

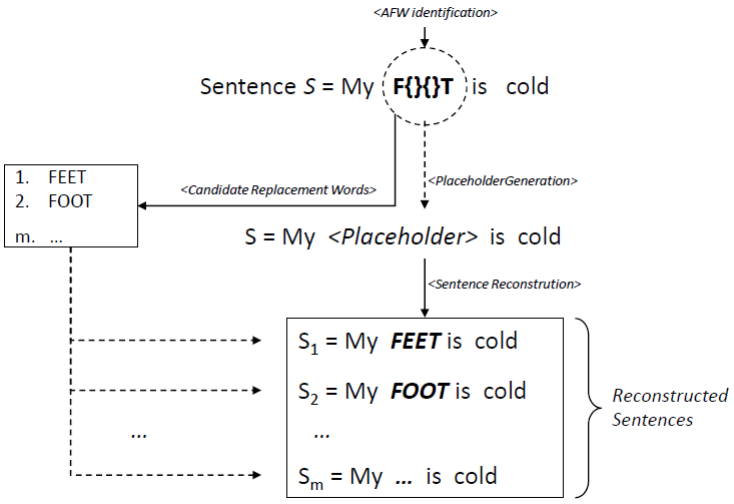


Figure 6. An example of the sentence reconstruction step.

In case of multiple sentences, all of them are evaluated. In particular, in the presence of an odd number of sentences, the final evaluation is based on the obtained majority value; in case of an even number of sentences, if the majority criterion is not applicable, then the evaluation is delegated to the human by requiring a manual intervention.

4.3. Dealing with the Propaganda Detection

In the previous section, the identification and transformation of a particular type of mixed-code text, based on "Special character for alphabets", has been faced by proposing an algorithm which is able to derive potential sentences in natural language. This section, instead, focuses on the description of the propaganda detection approach of Figure 2,

which aims to automatically analyse the reconstructed sentences $S_{Normalized}$={$S_1, S_2, ..., S_m$}, in order to discriminate whether they are related to propaganda or not and which one.

4.3.1. Machine Learning Approach

Among the available analysis techniques, machine learning (ML) is one of the most popular ones. It is adopted in different research fields by facing with (i) computational finance for the evaluation of credit risk and algorithmic trading; (ii) image processing and artificial vision for facial recognition, motion detection and object identification; (iii) computational biology for the diagnosis of tumors, pharmaceutical research and DNA sequencing; (iv) energy production for price and load forecasts; (v) automotive, aerospace and manufacturing sectors, for predictive maintenance; (vi) natural language processing for speech recognition applications and so on.

Furthermore, from the literature review, which is described in the related work section, it emerged that four main ML techniques represent the most popular algorithms in the automatic detection field, that is: (i) *Multinomial Naïve Bayes (MNB)*: which is a variant of Naive Bayes classifiers which uses a multinomial distribution for each of the features, (ii) *Support Vector Machine (SVM)*: that is a linear model for classification and regression problems which is used to solve linear and nonlinear problems, (iii) *Logistic Regression (LR)*: which is a linear predictive analysis algorithm based on the concept of probability to face with classification problems (iv) *Convolutional Neural Network (CNN)*: that is a type of artificial neural network, which is used in image recognition and processing, centred on deep learning to perform both generative and descriptive tasks.

As a consequence, given a sentence S, to be able to assess whether it is Propaganda or Nonpropaganda related, four different classifiers have been trained and evaluated (one for each of the above mentioned ML techniques respectively) regarding propaganda detection. The best one has been then selected.

4.3.2. Dataset Description and Evaluation Metrics

The four classifiers have been trained using an available and free downloadable dataset [52], which represents a research results achieved by a collaboration among the MIT Computer Science and Artificial Intelligence Laboratory of Cambridge (USA), the Qatar Computing Research Institute (Qatar) and University of Bologna (Italy). The dataset details are fully reported in [37]. It contains a collection of 15,847 textual items labelled with "0" and "1" to indicate Propaganda and Nonpropaganda. In particular, 4270 items are Propaganda related, and 11,577 are Nonpropaganda related. As the dataset was unbalanced, in order to reduce the bias of the classifier, two sampling techniques have been applied and evaluated for dealing with it: undersampling and oversampling approach. In particular, by applying the under sampling approach, we used the same number of textual items both for Propaganda and Nonpropaganda for a total of 8540, whereas by applying the oversampling technique we used all the 11,577 Nonpropaganda related items and we increased the number of Propaganda related item to 9440. In both cases, the generated datasets have been divided into two subsets: 80% of the items have been used to train the four models, whereas 20% of the items have been used to test the classifiers. To allow the comparison with previous research contributions, the trained classifiers have been evaluated through Accuracy, Precision, Recall and F1-Score, which are the most popular metrics used in ML.

4.3.3. Classifier Assessment

Different experiments have been conducted to determine the best classifier among the selected ones. More specifically, in order to reduce possible bias, that could have been arised from several factors, during the training and classification phase, MNB, SVM and LR classifiers have been assessed by using different vectorization approaches. In particular, Count Vectorizer, Term frequency–inverse document frequency, Term Frequency inverse document frequency with word n-gram, Term frequency inverse document frequency

with character n-gram, were used. The overall performance reached from all classifiers, including the CNN, are reported in Table 6 .

In particular, it resulted that the Convolutional Neural Network performed better than the other classifiers. By using an undersampled dataset, it reached the following results: 73% accuracy, 74% precision, 73% recall and 73% F1-Score. Whereas, as it is reported in Table 6, by oversampling the same dataset, better performances have been reached after retraining and reassessing the CNN-based classifier. In particular, it showed the following performances: 94% accuracy, 92% precision, 92% recall and 91% F1-Score, which have shown not only better performance than the other ones tested within this research work, but it is also inline in comparison to the performances of the other related works, that are reported in Table 2.

Table 6. Comparison of the performance achieved from each classifier with oversampling.

Trained Classifier	Accuracy	Precision	Recall	F1-Score
Multinomial Naive Bayes count_vectorizer	0.74	0	0.74	0.74
Multinomial Naive Bayes tfidf_vectorizer_word	0.71	0.70	0.70	0.70
Multinomial Naive Bayes tfidf_vectorizer_word_ngram	0.68	0.68	0.66	0.65
Multinomial Naive Bayes tfidf_vectorizer_ngram_chars	0.68	0.68	0.67	0.68
Support Vector Machine count_vectorizer	0.80	0.80	0.80	0.80
Support Vector Machine tfidf_vectorizer_word	0.82	0.82	0.82	0.82
Support Vector Machine tfidf_vectorizer_word_ngram	0.72	0.72	0.70	0.71
Support Vector Machine tfidf_vectorizer_ngram_chars	0.82	0.82	0.82	0.82
Logistic Regression count_vectorizer	0.79	0.78	0.79	0.78
Logistic Regression tfidf_vectorizer_word	0.73	0.72	0.72	0.72
Logistic Regression tfidf_vectorizer_word_ngram	0.68	0.68	0.66	0.66
Logistic Regression tfidf_vectorizer_ngram_chars	0.71	0.71	0.71	0.71
Convolutional Neural Network	0.94	0.92	0.92	0.91

5. Evaluation and Results Discussion

In the previous section, the proposed algorithm for supporting the analysis of mixed-code text has been described, as well as the approach for assessing and selecting the most promising machine learning classifier for the propaganda detection step has been contextually presented.

This section, instead, aims to discuss and show how the evaluation of the overall workflow has been conducted by explaining the encountered problem, by clarifying how the experiment has been setup, as well as by discussing the achieved results.

One of the problems encountered during the evaluation part concerned the lack of available dataset containing word art mixed code text and, especially, related to propaganda. To deal with it, the dataset described in Section 4.3 [37] has been taken in consideration as a starting point. In particular, a subset, called *SS*, of its instances has been selected by creating a smaller balanced dataset, that means by taking 50% of sentences labelled as propaganda and 50% of them labelled as nonpropaganda related.

After that, an online tool, called Universal Leet [53] has been used. Given in input to such tool a word W, it is able to automatically generate a possible related *Art Form Word* (W_{AFW}). As a consequence, the experiment dataset has been built by replacing in each sentence of the called SS subset, at least one word or even the full text with its related art form word, automatically generated from the above mentioned tool, so as to obtain a *Art Form Word* dataset (SS_{AFW}) with labels. In particular, at first we used alternatively three standard available modalities: *basic*, *advanced*, and *ultimate* leet to obtain the first version, and then in order to make such versions nonstandard but more human-like in terms of their variety, we have further updated part of them manually, so as to make them more difficult and less machine-related. All this encoding step has been done manually, because based on our knowledge we have not found any APIs, which allowed us to automate this process. An excerpt is shown in Table 7.

Table 7. An excerpt of the propaganda related dataset.

Id	Original Sentence$W \in SS$	Label	Sentence with $W_{AFW} \in SS_{AFW}$
01	Build the Wall	Propaganda	ßU!LD T|=|E \/\/ALL
02	Purge every homosexual	Propaganda	Pu|2ge every |=|omosexual
03	Abolish Wage System	Propaganda	Äbolish \/\/age $ystem
04	Whites should rule	Propaganda	Whit €s should |2ule
05	Right around 3 billion miles	Nonpropaganda	|2!g|-|t around 3 |3illion miles
06	America First	Propaganda	ÄmericÄ |=!rst
07	Here is the dirty secret	Propaganda	Here is the |)irtŸ $e(ret
...	...	...	...

Instead, the WordNet package [54] has been used as reference English dictionary, in order to implement the *Word Selection step* of the proposed normalisation algorithm, described in Section 4.2. It is centred on the Python's Natural Language Toolkit (NLTK) module and it consists of a database where the collected nouns, adjectives, adverbs and verbs are grouped into a set of cognitive synonyms, which are called synsets. As it has been already mentioned, on the basis on the results gathered from the classifier assessment described in Section 4, a CNN has been chosen for supporting the propaganda detection part of the process, as the trained classifier performed the best in comparison to the others. The configuration parameters, that have been used to setup the CNN classifier, are reported in Table 8.

Table 8. Configuration parameters of the CNN classification model.

Layer	Convolution
01	Conv2d(1, 100, kernel_size=(3, 300), stride=(1, 1))
02	Conv2d(1, 100, kernel_size=(4, 300), stride=(1, 1))
03	Conv2d(1, 100, kernel_size=(5, 300), stride=(1, 1))
04 (Fully Connected)	(fc1):Linear(in_features=300,out_features=2, bias=True

An example of the results, gathered by experimenting the method presented in Section 4, are reported in Table 9.

As it is possible to see in Table 9, for each sentence of Table 7, at least one reconstructed sentence is obtained. Indeed, according to the *Word Selection step* of the method, different "normal" words, and as a consequence multiple reconstructed sentences, can be generated from one single Art Form Word. Consequently, different evaluations are possible as for the sentence with Id = "1" or like the sentence with Id = "6" which is also reconstructed in different way but with the same result evaluation. Whereas, the sentence with Id = "7" is not properly reconstructed and then misclassified. To overcome the classification problem in case of discordant multiple classifications, the human intervention is required, in order to select one of the possible available alternatives.

Table 9. Reconstructed sentences and related classification results.

Id	Reconstructed Sentence(s)	Predicted Label(s)	Results
01	Build the Wall/Bag the wall	Propaganda/Nonpropaganda	True/False
02	Purge every homosexual	Propaganda	True
03	Abolish Wage System	Propaganda	True
04	Whites should rule	Propaganda	True
05	Right around 3 billion miles	Nonpropaganda	True
06	America First/America Fist	Propaganda/Propaganda	True/True
07	Here is the dirty met	Propaganda	False
...	...	...	...

Figure 7 summarises, instead, the confusion matrix at the end of the overall evaluation process based on the selected Convolutional Neural Network (CNN), which shows that only 9% of the instances are wrong classified and in particular only 5% of those related to propaganda are missclassified as nonpropaganda related.

True label	**Propaganda**	46%	5%
	Non-Propaganda	4%	45%
		Propaganda	**Non-Propaganda**
		Predicted label	

Figure 7. Confusion matrix of propaganda detection in hidden mixed-code text.

Whereas, Figure 8 shows the classification performances, in terms of accuracy, precision, recall and f1-score, by comparing the detection of propaganda in a standard text (i.e., with mixed code and as a consequence without applying the normalisation algorithm) with the detection of propaganda hidden mixed-code text. In particular, not only the diagram shows very similar performances, meaning that in presence of mixed-code, the normalisation algorithm is able to reconstruct and analyse appropriately the sentences; but it also highlights that the proposed approach is in average inline with the performances of the related works presented in Section 2, in terms of propaganda detection, by ranging from 90% to 92% in terms of accuracy, precision, recall and f1-score. Moreover, the correctness of the heuristic to normalise a text can be expressed as the number of correctly reconstructed sentences on the basis of the total original ones. Intuitively, a correct classification of a sentence/text is directly related to its correct normalisation process. This means that the classification values presented in Figure 8 represent, as a consequence, the lower bound of the heuristic to be able to correctly retrieve the original text starting from its TaAF representation.

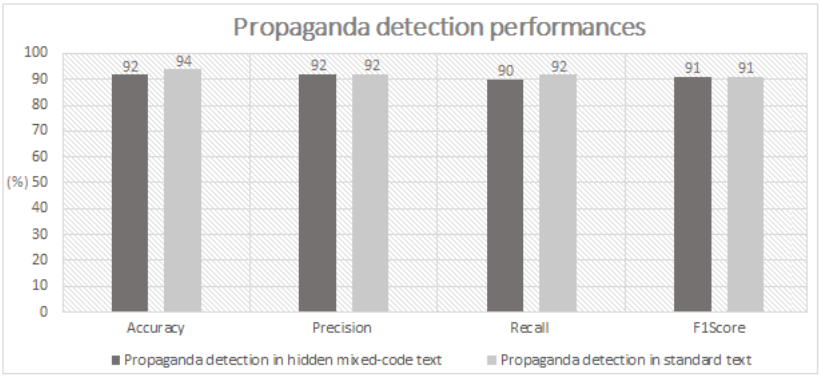


Figure 8. Results evaluation using CNN.

It is worth noting that the current results are improved, in comparison to those presented in the previous version of this work [20]. This result is not due to a new version of the presented heuristics, as its logic has not been changed, but it derived from a better training of the CNN algorithm, which makes less misclassification, by positively impacting on the overall process.

6. Conclusions

The paper dealt with the identification of mixed-code text for the detection of hidden propaganda. First a possible categorisation of different types of existing mixed-code on the basis of two parameters, related to the *language* and the *graphical writing style* has been provided. The study has been focused on the analysis of one type of *Text as Art Form* written on a single row (called "Special Characters for Alphabets"), by adopting a methodological approach centred on two main aspects: (i) mixed code text analysis and (ii) hidden propaganda detection. In particular, regarding the mixed code text analysis, a four-step algorithm (called *Text Segregation*, *Character Transformation*, *Word Selection* and *Sentence Reconstruction*) that supports both the identification of mixed code in text along with its normalisation into natural language has been proposed.

Whereas, regarding the hidden propaganda detection, a Convolutional Neural Network classifier has been chosen among other ML algorithms, as it provided the best performance in the detection of propaganda. The overall performance of the method has been experimented on a public available dataset containing a collection of 15,847 textual propaganda and nonpropaganda related items. The results showed good performances, by achieving 92% Accuracy and Precision, whereas 91% F1-Score and 90% Recall, which are on average better in comparison to the related work.

The impact of this solution lies in the ability to automate and consequently speed up the identification of sources and individuals, who use mixed-code to dissemination propaganda content, linked to extremist behaviour, so as to support the Law Enforcement Agencies (LEAs) and Police Forces (PFs) the fight against this phenomenon. Future works will focus on: (i) improving the performance of the current algorithm by defining a more efficient heuristic related to the Text Segregation step, in term of computational time. The current version generates all possible combination of symbols that works fine as long as the input is "reasonably contained". A smarter way to segregate the text needs be investigated to make it work also in larger context; (i) extending and experimenting the proposed method for supporting the detection of hidden propaganda by considering other types of mixed-code.

Author Contributions: Software, G.M.; Supervision, M.M.; Research, Writing—original draft, A.T. The authors contributed equally in all parts of the article in terms of literature review, adopted methodology, feature identification, model definition, experimentation and results analysis. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially performed in the context of the CHAMPIONs research project, which receives funding from the European Union's Internal Security Fund—Police, grant agreement no. 823705.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Acknowledgments: The authors want to thanks José L. Diego, Project Manager at Valencia Local Police (Spain), for supporting this research activity.

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

Abbreviations

The following abbreviations are used in this manuscript:

AFW	Art Form Word
ML	Machine Learning
OCR	Optical Character Recognition
NLP	Natural Language Processing
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
LR	Logistic Regression
SVM	Support Vector Machine
MNB	Multinomial Naive Bayes
RF	Random Forest
LSTM	Long Short Term Memory
BoW	Bag of Word
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
ToD	Text on Document
TiVM	Text in Visual Media
TaAF	Text as Art Form
NLTK	Natural Language Toolkit
LEAs	Law Enforcement Agencies
PFs	Police Forces

References

1. Richards, J.C.; Schmidt, R.W. *Language and Communication*; Routledge: London, UK, 2014.
2. Slobin, D.I. The many ways to search for a frog. *Relat. Events Narrat.* **2004**, *2*, 219–257.
3. Corea, S. Cultivating technological innovation for development. *Electron. J. Inf. Syst. Dev. Cities* **2000**, *2*, 1–15. [CrossRef]
4. Metropolitan Police 1. Available online: https://www.met.police.uk/advice/advice-and-information/t/ter-rorism-in-the-uk/signs-of-possible-terrorist-activity/ (accessed on 29 December 2020).
5. Nekovee, M.; Moreno, Y.; Bianconi, G.; Marsili, M. Theory of rumour spreading in complex social networks. *Phys. Stat. Mech. Appl.* **2007**, *374*, 457–470. [CrossRef]
6. Metropolitan Police 2. Available online: https://www.met.police.uk/advice/advice-and-information/t/terrorism-in-the-uk/ (accessed on 29 December 2020).
7. Bates, R.A. Dancing with wolves: Today's lone wolf terrorists. *J. Public Prof. Sociol.* **2012**, *4*, 1.
8. Tundis, A.; Huber, F.; Jäger, B.; Daubert, J.; Vasilomanolakis, E.; Mühlhäuser, M. Challenges and available solutions against organized cyber-crime and terrorist networks. In *WIT Transactions on the Built Environment*; WIT Press: Southampton, UK, 2018; Volume 174, pp. 429–441.
9. Jirovský, V.; Pastorek, A.; Mühlhäuser, M.; Tundis, A. Cybercrime and organized crime. In Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, 25–28 August 2018.
10. Tundis, A.; Jain, A.; Bhatia, G.; Muhlhauser, M. Similarity Analysis of Criminals on Social Networks: An Example on Twitter. In Proceedings of the 28th International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, 29 July–1 August 2019; pp. 1–9.
11. Falcone, A.; Garro, A.; Tundis, A. Modeling and Simulation for the performance evaluation of the on-board communication system of a metro train. In In Proceedings of the 13th International Conference on Modeling and Applied Simulation (MAS '14), Bordeaux, France, 10–12 September 2014; pp. 20–29.
12. Tundis, A.; Mazurczyk, W.; Mühlhäuser, M. A review of network vulnerabilities scanning tools: Types, capabilities and functioning. In Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018), Hamburg, Germany, 27–30 August 2018.
13. Chatfield, A.T.; Reddick, C.G.; Brajawidagda, U. Tweeting propaganda, radicalization and recruitment: Islamic state supporters multisided twitter networks. In Proceedings of the 16th Annual International Conference on Digital Government Research, Phoenix, AZ, USA, 27–30 May 2015; pp. 239–249.
14. Berzinji, A.; Abdullah, F.S.; Kakei, A.H. Analysis of Terrorist Groups on Facebook. In Proceedings of the European Intelligence and Security Informatics Conference, Uppsala, Sweden, 12–14 August 2013; p. 221.
15. Tundis, A.; Bhatia, G.; Jain, A.; Mühlhäuser, M. Supporting the Identification and the Assessment of Suspicious Users on Twitter Social Media. In Proceedings of the IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 1–3 November 2018; pp. 1–10.
16. Tundis, A.; Mühlhäuser, M. A multi-language approach towards the identification of suspicious users on social networks. In Proceedings the International Carnahan Conference on Security Technology (ICCST), Madrid, Spain, 23–26 October 2017; pp. 1–6.

17. Tundis, A.; Böck, L.; Stanilescu, V.; Mühlhäuser, M. Limits in the data for detecting criminals on social media. In Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES '19), Kent, Canterbury, UK, 26–29 August 2019; pp. 1–8.
18. Tundis, A.; Böck, L.; Stanilescu, V.; Mühlhäuser, M. Experiencing the detection of radicalized criminals on facebook social network and data-related issues. *J. Cyber Secur. Mob.* **2020**, *9*, 203–236. [CrossRef]
19. Cox, I.; Miller, M.; Bloom, J.; Fridrich, J.; Kalker, T. *Digital Watermarking and Steganography*; Morgan Kaufmann: Burlington, MA, USA, 2007.
20. Tundis, A.; Mukherjee, G.; Mühlhäuser, M. Mixed-code text analysis for the detection of online hidden propaganda. In Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES 2020), Dublin, Ireland, 25–28 August 2020.
21. Mehta, H.; Singla, S.; Mahajan, A. Optical character recognition (ocr) system for roman script english language using artificial neural network (ann) classifier. In Proceedings of the the International Conference on Research Advances in Integrated Navigation Systems (RAINS), Bangalore, India, 6–7 May 2016; pp 1-5.
22. Auer, P. *Bilingual Conversation*; John Benjamins Publishing: Amsterdam, The Netherlands, 1984.
23. Auer, P. *Code-Switching in Conversation: Language, Interaction and Identity*; Routledge: London, UK, 2013.
24. Mahesh, R.; Sinha, K.; Thakur, A. Machine translation of bi-lingual hindi-english (hinglish) text. In Proceedings of the 10th Machine Translation Summit (MT Summit X), Phuket, Thailand, 12–16 September 2005; pp. 149–156.
25. Bohra, A.; Vijay, D.; Singh, V.; Akhtar, S.S.; Shrivastava, M. A dataset of hindi-english code-mixed social media text for hate speech detection. In Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media, New Orleans, LA, USA, 6 June 2018.
26. Barman, U.; Das, A.; Wagner, J.; Foster, J. Code mixing: A challenge for language identification in the language of social media. In Proceedings of the First Workshop on Computational Approaches to Code Switching, Doha, Qatar, 25 October 2014; pp. 13–23.
27. Mathur, P.; Shah, R.; Sawhney, R.; Mahata, D. Detecting offensive tweets in hindi-english code-switched language. In Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media, Melbourne, Australia, 20 July 2018; pp. 18–26.
28. Santosh, T.Y.S.S.; Aravind, K.V.S. Hate speech detection in hindi-english code-mixed social media text. In Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, CoDS-COMAD '19, Kolkata, India, 3–5 January 2019; pp. 310–313.
29. Sharma, S.; Srinivas, P.; Balabantaray, R.C. Sentiment analysis of code-mix script. In Proceedings of the 2015 International Conference on Computing and Network Communications (CoCoNet), Trivandrum, India, 16–19 December 2015; pp. 530-534.
30. Jhanwar, M.G.; Das, A. An ensemble model for sentiment analysis of hindienglish code-mixed data. *arXiv* **2018**, arXiv:1806.04450.
31. Khandelwal, A.; Swami, S.; Akhtar, S.S.; Shrivastava, M. Humor detection in english-hindi code-mixed social media content: Corpus and baseline system. *arXiv* **2018**, arXiv:1806.05513.
32. Singh, R.; Choudhary, N.; Shrivastava, M. Automatic normalization of word variations in code-mixed social media text. *arXiv* **2018**, arXiv:1804.00804.
33. Ferrante, T.; Ferrante, C.M. *E-Leetspeak: All New! The Most Challenging Puzzles Since Sudoku!* Author House: Bloomington, IN, USA, , 2008; p. 136, ISBN-10: 1438923554.
34. Zhou, Y. Pro-ISIS fanboys network analysis and attack detection through twitter data. In Proceedings of the 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, China, 10–12 March 2017; pp. 386–390.
35. Nouh, M.; Nurse, R.C.J.; Goldsmith, M. Understanding the radical mind: Identifying signals to detect extremist content on twitter. In Proceedings of the 2019 IEEE International Conference on Intelligence and Security Informatics (ISI), Shenzhen, China, 1–3 July 2019. pp.1–7.
36. Johnston, A.H.; Weiss, G.M. Identifying sunni extremist propaganda with deep learning. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 27 November–1 December 2017; pp. 1–6.
37. Martino, G.D.S.; Yu, S.; Barrón-Cedeño, A.; Petrov, R.; Nakov, P. Fine-grained analysis of propaganda in news articles. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019.
38. Aggarwal, K.; Sadana, A. NSIT@NLP4IF-2019: Propaganda detection from news articles using transfer learning. In Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom, Censorship, Disinformation, and Propaganda, Hong Kong, China, 4 November 2019; pp. 143–147.
39. Tundis, A; Shams A.A; Mühlhäuser, M. Concepts of a Pyramidal Model for Assessing Terrorist Propaganda. In Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 24–27 November 2020; pp. 1–4.
40. Govindan, V.K.; Shivaprasad, A.P. Character recognition—A review. *Pattern Recognit.* **1990**, *23*, 671–683. [CrossRef]
41. Vamvakas, G.; Gatos, B.; Perantonis, S.J. Handwritten character recognition through two-stage foreground sub-sampling. *Pattern Recognit.* **2010**, *43*, 2807–2816. [CrossRef]
42. Dutta, S.; Sankaran, N.; Sankar, K.P.; Jawahar, C.V. Robust recognition of degraded documents using character n-grams. In Proceedings of the IEEE 10th IAPR International Workshop on Document Analysis Systems, Washington, DC, USA, 27–29 March 2012; pp. 130–134.

43. Devlin, J.; Kamali, M.; Subramanian, K.; Prasad, R.; Natarajan, P. Statistical machine translation as a language model for handwriting recognition. In Proceedings of the IEEE International Conference on Frontiers in Handwriting Recognition, Bari, Italy, 18–20 September 2012; pp. 291–296.
44. Hassan, T.; Khan, H.A. Handwritten bangla numeral recognition using local binary pattern. In Proceedings of the IEEE International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), Savar, Bangladesh, 21–23 May 2015; pp. 1–4.
45. Breuel, T.M.; Ul-Hasan, A.; Al-Azawi, M.A.; Shafait, F. Highperformance ocr for printed english and fraktur using lstm networks. In Proceedings of the 12th IEEE International Conference on Document Analysis and Recognition, Washington, DC, USA, 25–28 August 2013; pp. 683–687.
46. Lavanya, K.; Bajaj, S.; Tank, P.; Jain, S. Handwritten digit recognition using hoeffding tree, decision tree and random forests - A comparative approach. In Proceedings of the 2017 International Conference on Computational Intelligence in Data Science (ICCIDS), Chennai, India, 2–3 June 2017; pp. 1–6.
47. Liyanage, C.; Nadungodage, T.; Weerasinghe, R. Developing a commercial grade tamil ocr for recognizing font and size independent text. In Proceedings of the 15th IEEE International Conference on Advances in ICT for Emerging Regions (ICTer), Colombo, Sri Lanka, 24–26 August 2015; pp. 130–134.
48. Lopes, D. *A Philosophy of Computer Art*; Routledge: London, UK, 2009.
49. Xu, X.; Zhang, L.; Wong, T. Structure-based ascii art. *ACM Trans. Graph.* **2010**, *29*, 1–9.
50. De Campos, T.E. The Chars74k Dataset: Character Recognition in Natural Images. Available online: http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/ (accessed on 29 December 2020).
51. Pillow. Available online: https://pillow.readthedocs.io/en/stable/ (accessed on 29 December 2020).
52. Shared Task on Fine-Grained Propaganda Detection NLP4IF, 2019. Available online: https://propaganda.qcri.org/nlp4if-shared-task/data/ (accessed on 29 December 2020).
53. Universal Leet Converter. Available online: http://www.robertecker.com/hp/research/leet-converter.php?lang=en (accessed on 29 December 2020).
54. WordNet Interface—NLTK Corpus Reader. Available online: https://www.nltk.org/howto/wordnet.html (accessed on 29 December 2020).