Knowledge-Based Systems 267 (2023) 110422

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Deception detection in conversations using the proximity of linguistic markers

Nikesh Bajaj [a,*], Marvin Rajwadi [a], Tracy Goodluck Constance [a], Julie Wall [a], Mansour Moniri [a], Thea Laird [b], Chris Woodruff [b], James Laird [b], Cornelius Glackin [b], Nigel Cannings [b]

[a] School of Architecture, Computing and Engineering, University of East London, UK
[b] Intelligent Voice Ltd., London, UK

ARTICLE INFO

Article history:
Received 13 May 2022
Received in revised form 21 February 2023
Accepted 23 February 2023
Available online 3 March 2023

Keywords:
Deception
Fraud
Conversational speech
Linguistic markers
Proximity features
Proximity model
Decision engine
BERT
Linguistic analysis

ABSTRACT

Detecting the elements of deception in a conversation takes years of study and experience, and it is a skill set primarily used in law-enforcement agencies. In ever-growing business opportunities, organisations employ teleoperators to provide support and services to their large customer base, which is a potential platform for fraud. With technological advancements, it is desirable to have an automated system that spots the deceptive elements in the conversation, and provides this information to the teleoperators to better support them in their interactions. We propose the Decision Engine to detect deceptive conversation based on the proximity of linguistic markers present, which produces a deception score for a conversation and highlights the potential deceptive elements of the conversation. In collaboration with behavioural experts, we have selected ten linguistic markers that potentially indicate deception. We have built a variety of models to detect the trigger terms for selected linguistic markers without ambiguity, using either regular expressions or the BERT model. The BERT model has been trained on a conversational dataset that we collated and was labelled by our behavioural experts. The proposed Decision Engine employs the BERT model and regular expressions to detect the linguistic markers and compute the proximity features to further estimate the deception score. We evaluated the proposed approach on the Columbia-SRI-Colorado (CSC) dataset and a real-world Financial Services dataset. In addition to accuracy, we have also employed the True Positive Rate metric, with a high enough threshold to avoid any false-positive cases, which we indicate as TPR_{F0}. The Decision Engine achieves 69% accuracy and 46% TPR_{F0} for the CSC dataset and 72% accuracy and 60% TPR_{F0} for the Financial Services dataset. In contrast, a baseline model, which uses non-proximity features achieves 67% accuracy and 32% TPR_{F0} for the CSC dataset and 67% accuracy and 10% TPR_{F0} for the Financial Services dataset. Furthermore, using the Decision Engine, the impact of the proximity of markers on the deception score has been analysed by our behavioural experts to provide insight into linguistic behaviour in relation to deception.

1. Introduction

Deception detection is a very challenging task. For an average person, the ability to detect a lie is only slightly greater than chance, with about 54% accuracy [1,2]. The successful detection of deception in human interactions has long been of interest across many domains and it has been studied in various sub-fields related to psychology and law enforcement. It takes years of study to master the skills. To date, this skill set is primarily required for interrogation in law enforcement. However, the skill of detecting suspicious behaviour or elements of deception is also highly useful in financial and legal services. These organisations generally employ teleoperators or customer service representatives to meet the demand of a large customer base. These teleoperators are not trained to spot the cues of deception, hence organisations lose a large amount of money every year as a result of fraud [3]. With recent advances in Natural Language Processing (NLP), it is desirable to have an automated system that flags the deceptive elements in a conversation, with at least the same performance as behavioural experts and experienced interrogators, if not better. A system with the capability to flag deceptive cues in real-time, can support teleoperators to navigate their enquiries to reach a better conclusion, which can save organisations from potential fraud.

Most studies for preventing financial fraud are based on the analysis of banking transaction patterns [4]. Irrespective of the

* Corresponding author.
E-mail addresses: n.bajaj@uel.ac.uk, n.bajaj@qmul.ac.uk (N. Bajaj).

https://doi.org/10.1016/j.knosys.2023.110422

field, several studies exploit the linguistic approach to detect deception in written opinions or transcribed conversations. There is a difference between a written and an oral conversation. While writing, a person can be careful with their choice of words to prevent the leakage of any deception cues. Whereas, while speaking, a person has an additional limitation to responding in a timely fashion, which impacts the freedom to choose words carefully. Historically, several linguistic features for communication have been explored, known as linguistic markers. Linguistic markers are features that focus on specific terms (words/phrases) in oral or written communication. A widely used linguistic approach is based on the Linguistic Inquiry and Word Count (LIWC) tool, using the Bag-of-Words (BoW) methodology, it counts the words of different categories such as Negations, Pronouns, Emotions, etc. [5]. To detect deception in written opinions, studies have explored datasets that contain deceptive reviews and essays, [6–8]. In addition to features from LIWC, lexical and context-free grammar features were also employed. Using the same approach, a study also explored cross-cultural deception in essay writing [9]. Fake news is another platform to spread deceptive opinions, and this has been explored with a variety of linguistic features [10]. A recent study used a total of 16 linguistic markers along with sentiment features to detect fraud in telephone conversations [11]. Some of these linguistic markers overlap with the linguistic features in LIWC.

Another mode for potential fraud is online chat rooms, a type of fraud categorised as cybercrime. A few studies have used similar linguistic approaches to detect deception in online messages [12], and extend it to an environment including non-native speakers [13]. MafiaScum [14] is a large dataset for deception detection, based on the messages exchanged in an online game. From this dataset, a few selected linguistic features, e.g. pronouns, sensory, and quantifiers, were demonstrated as the best features for indicating deception [14]. A study [15], conducted an experiment to collect examples of truth and lies online and this showed that age and gender influence the choice of words, which potentially indicate deception. A few studies have explored beyond linguistic features by including other modalities, such as audio, video [16,17], electroencephalogram and eye gaze [18]. These additional modalities allow the capture of detailed cues of deceptive behaviour. However, it is impractical for many situations to employ these sensors, due to various reasons such as privacy.

Conversation is the most common means in organisations for representatives and customers to communicate, and this is the modality we are focused on in this work. To investigate deception detection in speech, a widely used dataset is the Columbia-SRI-Colorado (CSC) dataset [19], containing the audio and transcribed conversations of 32 interviews. This dataset has been explored with linguistic features from the transcribed conversations and acoustic features from the audio recordings [20]. With additional datasets, it has been shown that deception is influenced by individual and cultural differences [21,22]. There are a few studies that have explored only the acoustic features of the CSC dataset [23], which demonstrated the significance of glottal waveform features [24] and pauses [25] in relation to deception. Most of the findings in the above-mentioned studies are consistent with widely accepted theories of deception [26].

To date, most of the work based on linguistic markers is dependent on the frequency of the terms (words/phrases), including the LIWC tool, which counts specific words from different categories. Although counting the terms of linguistic markers is effective [6,9,11,12], the temporal relationship and interaction between linguistic markers is lost, such as the order in which they appear and their proximity to each other in a conversation. The interaction of different markers in a conversation can inform about the cognitive state of the subject and potential cues of deception, such as two Negation terms near to each other.

In this paper, we propose the Decision Engine that exploits the temporal relationship of linguistic markers using the proximity model. We use a set of 10 linguistic markers, all of which are potential indicators of deception. These markers have been carefully validated by linguists and expert interrogators. We have used the Bidirectional Encoder Representations from Transformers (BERT) model [27] to detect and remove the ambiguity of trigger terms for a number of these selected linguistic markers. Based on the proximity of linguistic markers, the Decision Engine estimates the deception score for a given transcribed conversation. To evaluate the Decision Engine, we have used (1) the CSC dataset [19] and (2) a real-world Financial Services dataset [11]. It is critical to evaluate a system that detects deception or fraud. While expecting the system to have a high accuracy for detecting fraud, it is also very important to minimise the False Positive Rate to negligible for two reasons. Firstly, it is not desirable to have false alarms that incorrectly classify a truth as deception and flags the cases. Secondly, to avoid the Base Rate fallacy, also known as the False Positive paradox [28]. To accommodate this, in addition to accuracy as a performance metric, we use a metric TPR_{F0}, which is a true positive rate with a high enough threshold on the deception score to eliminate any false positive cases. In other words, TPR_{F0} reflects the fraction of deceptive cases that can be detected without raising any false alarms, the False Positive Rate becomes equal to zero.

The rest of the paper is organised as follows: Section 2 describes the linguistic markers and how they are detected. Section 3 explains the formulation of the proximity model to extract proximity-based features. The Decision Engine is explained in Section 4. In Section 5, we outline the datasets used to evaluate the Decision Engine, the experimental settings and discuss the results. Section 6 analyses the interaction between the markers from a linguistic perspective. Finally, we conclude the proposed approach and discuss future directions in Section 7.

2. Linguistic markers and detection

In this section, we explain the selected linguistic markers and their unambiguous detection for the Decision Engine.

2.1. Linguistic markers

In addition to physiological responses and body language, analysing linguistic cues in written or verbal communication is one of the techniques behavioural analysts use to spot deception in forensic analysis [29]. Over the last century, the choice of words and phrases have been identified as linguistic cues, to indicate the cognitive and emotional response of the subject. Several linguistic markers have been proposed in the literature [5,30], from which in consultation with behavioural experts, we adopted a subset of carefully selected markers that potentially indicate deception. These 10 markers are defined in Table 1, along with examples of their trigger terms.

For each marker in Table 1, a list of trigger terms (keywords and phrases) has been created by our behavioural experts. The first approach to identify the markers in a subject's utterance is to use a list of trigger terms to spot the respective marker by employing Regular Expressions (RegEx). It is also known as a BoW or keyword spotting. For a few markers, such as Negation, the RegEx approach works quite well. However, for markers such as Hedging and Explainers, the RegEx produces a high False Positive Rate (FPR) due to the ambiguity of the trigger terms, i.e. the context of the marker can vary. To illustrate, consider two examples: (1) I have *about* 10 million objects (2) We need to talk

Table 1
Linguistic Markers.

k	Marker	Example
1.	**Disfluencies:** Interrupts the flow of conversation and can be used as a stalling tactic, detectable as hesitations or fillers, indicative of potential deception [31].	Um, Er, Ah
2.	**Explainers:** Statements that provide an explanation for former context in a conversation. A potential indicator of deception is when the explanation was not required or not relevant in the current context [32].	Because, Therefore, Since
3.	**Hedging:** Denotes a doubt or a lack of commitment to what is being said [33].	About, Maybe, Like
4.	**Implied Repetition:** Implies that the following information was already conveyed. The emphasis on repetition can indicate the psychological need to imply that something has been said without necessarily having said it. The speaker also shows a particular awareness of the audience at this point and seeks to convince of the former [34].	As I mentioned, Like I said, As I told you
5.	**Memory Loss:**[a] Feigning memory loss or having no recollection of events, is a potential sign of deception [33].	I forgot, I don't remember
6.	**Negation:**[a] Anything which is reported in the negative, especially when related to time [35].	Not, Can't, Didn't, Never
7.	**Temporal Lacunae:** Unexplained lapses of time, indicative of deception when not expected [33].	Afterwards, Later
8.	**Uncertainty:** Denotes imprecision, vagueness, or incompleteness in a conversation. A likely indication to hide or create false information. [36].	Something, Someone
9.	**Untruthful Words:** Expresses some level of uncertainty by overselling [33]	Honestly, Cross my heart, God knows
10.	**Withheld Information:** Omitting to disclose vital information intentionally [37].	Generally, Loosely, Largely speaking

[a]To avoid duplicates due to the inclusion of Negation terms in Memory Loss, such instances are considered as Memory Loss only.

about Mark. In both examples a trigger term *about* is used, which is indicative of Hedging, however, in example (2), the term *about* is not Hedging. To remove such ambiguities in trigger terms, more contextual information is needed. To achieve this, we created a conversational dataset labelled by behavioural experts, which is explained in the next subsection.

It is important to note that historically, linguistic markers are defined and analysed based on native speakers. Although the impact of most of the linguistic markers translates to non-native speakers, a marker such as *Disfluencies*, could be less sensitive to deception. A subject who is speaking a second language (non-native) might be using more frequent fillers as compared to a native speaker, which could indicate the cognitive load to form sentences as opposed to hesitation or stalling tactics. In the case of non-native speakers, it is suggested to consider the language fluency of the subject.

2.2. Dataset

To remove the ambiguities in detecting linguistic markers, we created a dataset, namely A Conversational Dataset to detect Linguistic Markers (CDLM). As our focus is on oral communication rather than written, the dataset was created from the Cornell movie dialogues corpus (600+ movie scripts) and an American sitcom (TV series) Seinfeld [38,39]. From the dialogue scripts, we extracted the utterances detected using the RegEx approach. Each utterance was then examined by behavioural experts and labelled to indicate whether the respective trigger term represents a marker, for example, Hedging or Non-Hedging labels for an utterance with a Hedging keyword. From the analysis performed on this expertly labelled data, it was observed that the RegEx approach produces a very high FPR of 59%, 30%, and 50% for Hedging, Explainers, and Memory Loss, respectively. In contrast, for the remaining seven markers, the FPR was quite low. Because of this, we decided to train a BERT model to achieve more accurate identification of these three markers. The final CDLM dataset consists of labelled utterances for these three linguistic markers, Hedging, Explainers, and Memory Loss. To improve the performance of the MTDNN-BERT, the CDLM dataset was extended and populated with utterances that do not include any trigger terms. The resulting CDLM dataset consists of 6,837, 4,151, and 604 labelled utterances for Hedging, Explainers, and Memory Loss, respectively.

Table 2
Testing results of MTDNN-BERT for correctly detection of marker, along with the respective FPR of RegEx.

Linguistic marker	RegEx	MTDNN-BERT	
	FPR	Error rate	F1
Hedging	0.59	0.11	0.87
Explainer	0.31	0.07	0.96
Memory Loss	0.50	0.07	0.93

2.3. Detection using BERT

BERT is a promising language model, which requires very little training for a new NLP task to achieve state-of-the-art performance [40]. To reduce the FPR for the Hedging, Explainers, and Memory Loss markers, a pre-trained BERT model was used. To avoid the use of multiple BERT models, i.e. one for each marker, and the computational cost this would entail, the Multi-Task Deep Neural Network (MTDNN) architecture is used. This uses BERT's shared transformer layers [41] followed by three output layers, one for each marker (MTDNN-BERT). The 12-layers of BERT transformer output 768 features which are fed to three linear layers in parallel to generate the probability score for the presence of each marker. As this is a multi-task problem, each probability score is independent of the others. The shared transformer layers allow the knowledge learned in one task to be used for the other tasks.

With a 70-30 split on the extended CDLM dataset, for training and testing, the MTDNN-BERT was trained (fine-tuned) using Adamax optimiser, with a very low learning rate of $5e-5$ and a batch size of 16. Table 2 shows the error rate and F1-score achieved on the test data for each class alongside the FPR of the RegEx for the respective markers. From Table 2, it can be seen that the MTDNN-BERT has reduced the FPR for each marker significantly, i.e. in comparison to RegEx, the MTDNN-BERT achieves a total error rate of 11% for Hedging, which also includes false negative cases, thus FPR < 0.11.

Using the CDLM dataset, the MTDNN-BERT has been trained to classify an utterance as containing a true marker or not, i.e. to detect if the utterance contains true Hedging, Explainers, and/or Memory Loss. However, the trigger terms within the utterance for the respective markers are not identified by MTDNN-BERT itself. One approach to identify the respective trigger terms is to apply the RegEx as a post-processing step. For example, if MTDNN-BERT classifies an utterance to be Hedging, the RegEx list for Hedging

$$PF_{i,n}(k) = \begin{cases} T_k(j) - T_i(n) & \text{where } j = \text{argmin}_j(|T_k(j) - T_i(n)|) \quad \text{if } k \neq i \text{ and } T_k \neq \emptyset \\ T'_{i,n}(j) - T_i(n) & \text{where } j = \text{argmin}_j(|T'_{i,n}(j) - T_i(n)|) \quad \text{if } k = i \text{ and } T'_{i,n} \neq \emptyset \\ \infty \quad (d_{max}) & \text{else} \end{cases} \qquad \text{for } k = 1, 2..K \tag{2}$$

can be applied to the processed utterance to identify the actual Hedging term or phrase within the utterance.

To identify the trigger terms in an utterance for the respective markers, instead of using the RegEx as a post-processing step, we exploited MTDNN-BERT's language model functionality. As BERT has been originally trained on a variety of datasets and as it is a language model, MTDNN-BERT can also be used to identify further potential and similar trigger terms which were not present in the original list of trigger terms. We used the Text Deconvolution by Occlusion (TDO) technique [42], which allows us to find the weight for each word in the utterance with respect to the overall prediction score, i.e. the probability score for a class. This is achieved by masking one word at a time and observing the fluctuation in the prediction score. The weight produced by TDO corresponds to the importance of each word in a sentence towards the MTDNN-BERT prediction. The weights of all words are then normalised and with empirical observations, a threshold of 0.85 was set to identify the relevant term or phrase in the utterance. Compare to RegEx step as post-processing, TDO was found to much more effective. For example, MTDNN-BERT was able to detect sentences as Hedging, where no trigger term from our list was present, however, with TDO, a word similar to Hedging was revealed in given context. By using TDO, instead of RegEx as post-processing, we were able to surface more trigger terms in complex context than we had originally identified. All new surfaced trigger terms for a particular marker were verified by our behavioural team.

3. Proximity model

The classical approach uses the frequency of all instances of linguistic markers in a given text or piece of transcribed speech to detect deception and fraud [6,11,14]. However, using the frequency of all instances, we lose the information of the order in which these markers appear in speech and their proximity to one another, i.e. how the markers interact. For example, two markers appearing in a different order will still have the same count. To overcome this, we introduce the Proximity Feature. For ease of explaination, all the notations used for proximity model formulation are tabulated in Table 3.

3.1. Proximity feature

To capture the patterns in which different markers appear in utterances, we designed a proximity model, which extracts the patterns of how different markers appear by computing the distance of closely appearing markers. An extracted pattern is considered as a Proximity Feature (PF), which is further used for predictive modelling by the Decision Engine. The PF is a vector and the formulation of the PF is explained as follows.

Consider there are K linguistic markers; M_1, M_2, ...M_K. There can be multiple instances of a given marker M_i, detected at different locations (word indices) in the utterance. For example, *I have* ***not*** *seen that object,* ***never*** *in my life*. There are two Negation markers (emboldened) present, one at word location 3, and another at 7. For each marker M_i, a set of locations of marker instances; T_i can be defined as:

$$M_i \rightarrow T_i = \{\forall t \in \mathbb{Z} \mid t \text{ location of marker } M_i\} \tag{1}$$

Table 3
Notations for proximity model.

For K linguistic markers, and $i = 1, 2, \ldots K$	
M_i	ith Linguistic Marker
T_i	Location Set for ith Marker, (also considered as an indexed list)
$T_i(n)$	nth instance of ith Marker
t_n^i	Location of nth instance of ith Marker, i.e. $T_i(n) = t_n^i$
$T'_{i,n} = T_i \setminus \{t_n^i\}$	Location Set for ith Marker, excluding nth element
N_{T_i}	Total number of instances of ith Marker, i.e. length of Location Set T_i
$PF_{i,n}$	Proximity Feature Vector for nth instance of ith Marker. The dimension of PF vector K, i.e. $PF_{i,n} \in \mathbb{Z}^K$
$PF_{i,n}^{\eta_g}$	Proximity Feature vector normalised by Gaussian kernel; $\eta_g(\cdot)$
$PF_{i,n}^{\eta_h}$	Proximity Feature vector normalised by Tanh kernel; $\eta_h(\cdot)$
PF_i	Proximity Feature matrix for ith Marker, that is created by concatenating PF vectors of all the instances. The dimension of PF matrix $PF_i \in \mathbb{Z}^{N_{T_i} \times K}$
d_{max}	Maximum proximity range

such that, each marker M_i has a location set T_i.

By the definition of linguistic markers (see Table 1) and their exclusive nature, the location sets of markers are such that: $T_i \cap T_j = \emptyset$ for $i \neq j$ (non-overlapping). A location set T_i can be an Empty set if no instance of marker M_i appeared in the given utterance. Thus, the length of location set $N_{T_i} = |T_i|$ is the total number of instances of marker M_i in the utterance. The Proximity Model is designed to compute a pattern for each instance of the markers, such that for a given utterance, the number of PF vectors will be equal to $\sum_{i=1}^{K} N_{T_i}$, i.e. sum of the number of instances of all the markers. By abuse of notation, for simplicity, we consider a set T_i also as a sequence, to index an element $t \in T_i$. From given location sets T_1, T_2... T_K, for K markers, consider that the location of nth instance of marker M_i is t_n^i, e.g. $T_i(n) = t_n^i$. Then the PF vector, $PF_{i,n}$ for t_n^i can be computed as defined by Eq. (2). The details of computation from Eq. (2) are explained below with a help of Fig. 1. To compute $PF_{i,n}$ for t_n^i, Eq. (2), iterates over K markers ($k = 1, 2 .. K$), and computes the distance of nearest marker instance to t_n^i (i.e. $T_i(n)$) from each marker, thus $PF_{i,n} \in \mathbb{Z}^K$, a K-dimensional vector of integers. Eq. (2) has three cases to compute $PF_{i,n}$, which are explained as follows;

(1) While iterating over other markers T_k, ($k \neq i$), if the location set of another marker is not empty, i.e. $T_k \neq \emptyset$ (the first case of Eq. (2)), the computation is straightforward. Finding a marker instance $T_k(j)$, which has minimum absolute distance to $T_i(n)$ (i.e. t_n^i), and assign kth element of $PF_{i,n}(k) = T_k(j) - T_i(n)$, which is a signed distance to closest instance of the kth marker. To keep the direction of marker instance (before or after), we use distance with sign $T_k(j) - T_i(n)$, instead of absolute distance $|T_k(j) - T_i(n)|$, where a positive value means, the nearest marker instance appears after t_j^k, and vice-versa.

(2) Since $PF_{i,n}$ is computed for the nth element of the ith marker, when ($k = i$) (e.g. computing over the same marker), the first thing is to recreate the location set T_i

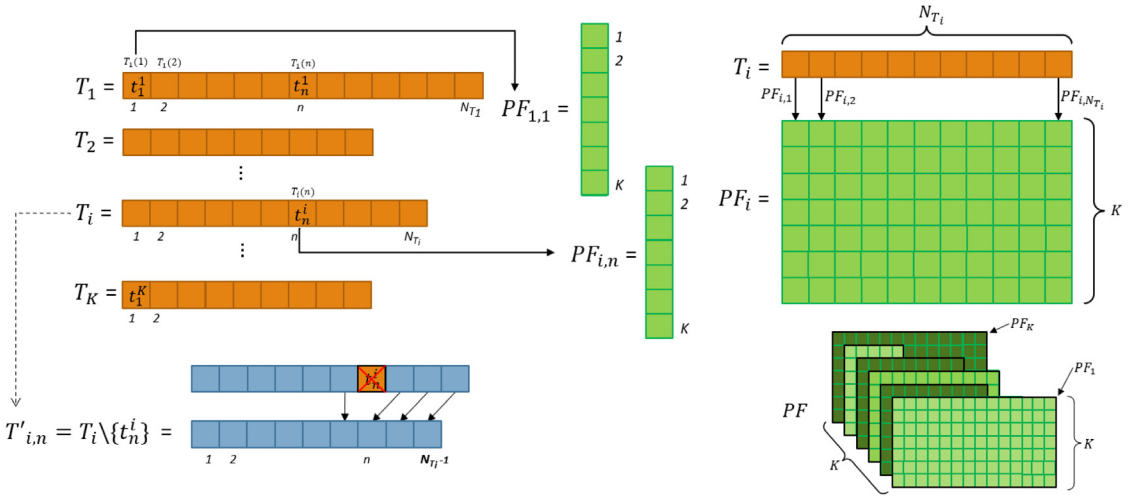


Fig. 1. Illustration of Proximity Feature vector $PF_{i,n}$ from location sets T_i and construction of Proximity Feature matrix PF_i.

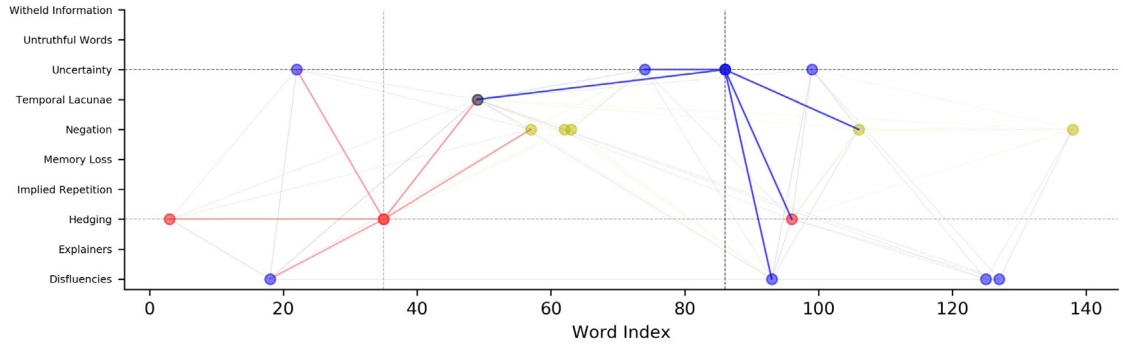


Fig. 2. Proximity Feature patterns for all the markers in the given utterance. For explanation, two instances of Hedging and Uncertainty are highlighted in red and blue, respectively.

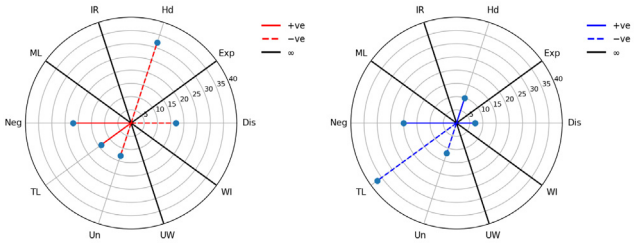
as the one to exclude the nth element from it, which is $T'_{i,n} = T_i \setminus \{t^i_n\}$, as illustrated in Fig. 1. If a recreated location set is not empty, $T'_{i,n} \neq \emptyset$ then we compute the minimum distance for $PF_{i,n}(i)$ in the same way as we did in the first case above.

(3) In the case of an empty location set, e.g. $T_k = \emptyset$, the distance of the nearest marker is set to $PF_{i,n}(k) = \infty$, which indicates that the marker is present somewhere far away. However for computational purposes, this far away value is set to a predefined higher value (d_{max}), such as the total length of the given utterance. The d_{max} can also be set to a smaller value to restrict the proximity range to a shorter temporal region.

Using Eq. (2) the $PF_{i,n}$ vector is computed for each marker instance in the ith-marker, which further is concatenated and represented as matrix PF_i, PF for ith marker, as illustrated in Fig. 1. Since there are N_{T_i} elements in T_i location set, PF_i will have N_{T_i}, K-dimensional vectors, $PF_i \in \mathbb{Z}^{N_{T_i} \times K}$. Finally, for K markers, we obtain a set of K PF matrices, i.e. PF_1, PF_2, …, PF_K. However, due to the different number of instances in each marker, they can not be combined as a 3D Tensor, which is shown in Fig. 1.

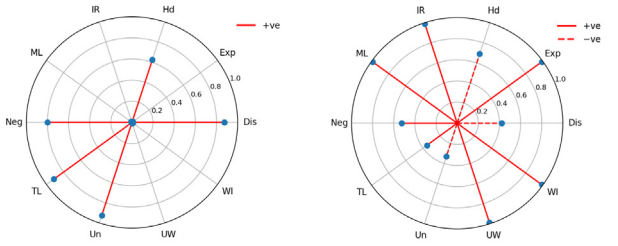
To further support the explanation of PF and its computation from Eq. (2), we present an example of a 140 word long utterance from a conversation. The linguistic markers for the utterance are shown in Fig. 2. To keep the focus on computation, we omit the actual conversation and only consider the location sets extracted for 10 linguistic markers, which are as follows; $T_1 = \{18, 93, 125, 127\}$, $T_2 = \{\}$, $T_3 = \{3, 35, 96\}$, $T_4 = \{\}$, $T_5 = \{\}$, $T_6 = \{57, 62, 63, 106, 138\}$, $T_7 = \{49\}$, $T_8 = \{22, 74, 86, 99\}$, $T_9 = \{\}$, $T_{10} = \{\}$. The order of the location sets for linguistic markers is same as in Table 1 i.e. T_1 is for $k = 1$, Disfluencies. The instances of each linguistic marker are shown as dots in Fig. 2, where, the x-axis is the location (word index) of marker instances and the y-axis is the marker. Markers are identified and plotted for only one speaker (customer) in a conversation. Note that for 5 markers there are no instances, thus the respective location sets are empty. As shown in Fig. 2, each marker instance is connected to the nearest marker instance of every other category, including itself, as computed by Eq. (2). Two instances are highlighted in particular, the second instance of Hedging ($T_3(2) = t^3_2 = 35$) and the third instance of Uncertainty ($T_8(3) = t^8_3 = 86$), with solid red and blue lines connecting these instances to the nearest instance of every marker, respectively.

The computation of the Proximity Feature for the second instance of Hedging ($PF_{3,2}$), using Eq. (2) is as follows. For $k = 1$, the nearest marker instance for second instance of Hedging from T_1 is 18, so $PF_{3,2}(1) = 18 - 35 = -17$ (case 1 in Eq. (2)). For $k = 2$, since T_2 is the empty set, $PF_{3,2}(2) = \infty$ (case 3 in Eq. (2)). Similarly, for $k = 4, 5, 9, 10$, respective location sets are empty, so $PF_{3,2}(4) = PF_{3,2}(5) = PF_{3,2}(9) = PF_{3,2}(10) = \infty$. For $k = 3$, since it is the same marker (i.e. Hedging, T_3), we get a closet marker instance of Hedging, excluding the second element 35, ($T'_{3,2} = T_3 \setminus \{35\} = \{3, 96\}$), which is 3, so $PF_{3,2}(3) = 3 - 35 = -32$. (case 2 in Eq. (2)). Similarly, for $k = 6$, $PF_{3,2}(6) = 57 - 35 = 22$, $k = 7$, $PF_{3,2}(7) = 49 - 35 = 14$, $k = 8$, $PF_{3,2}(8) = 22 - 35 = -13$. Considering $d_{max} = 100$, the computation of $PF_{3,2}$ can be written



(a) Hedging instance: $PF_{3,2}$ (b) Uncertainty instance: $PF_{8,3}$

Fig. 3. Proximity Features extracted from location sets highlighted in Fig. 2 for instances of Hedging and Uncertainty from all the markers; Hedging (Hd), Explainers (Exp), Disfluency (Dis), Withheld Information (WI), Untruthful Words (UW), Uncertainty (Un), Temporal Lacunae (TL), Negation (Neg), Memory Loss (ML) and Implied Repetition (IR).



(a) Using Gaussian kernel: $PF^{\eta_g}_{3,2}$ (b) Using Tanh kernel: $PF^{\eta_h}_{3,2}$

Fig. 4. Scaled Proximity Features of a Hedging instance (as in Fig. 3(a)), highlighted in Fig. 2, scaled with (a) Gaussian kernel (b) Tanh kernel. Markers: Hedging (Hd), Explainers (Exp), Disfluency (Dis), Withheld Information (WI), Untruthful Words (UW), Uncertainty (Un), Temporal Lacunae (TL), Negation (Neg), Memory Loss (ML) and Implied Repetition (IR).

as;

$$PF_{3,2} = \begin{bmatrix} 18-35 \\ \infty \\ 3-35 \\ \infty \\ \infty \\ 57-35 \\ 49-35 \\ 22-35 \\ \infty \\ \infty \end{bmatrix} = \begin{bmatrix} -17 \\ d_{max} \\ -32 \\ d_{max} \\ d_{max} \\ 22 \\ 14 \\ -13 \\ d_{max} \\ d_{max} \end{bmatrix} = \begin{bmatrix} -17 \\ 100 \\ -32 \\ 100 \\ 100 \\ 22 \\ 14 \\ -13 \\ 100 \\ 100 \end{bmatrix}$$

The PF vectors for the above-mentioned two highlighted instances and vectors $PF_{3,2}$ and $PF_{8,3}$ are shown as radar charts in Figs. 3(a) and 3(b) respectively. In Figs. 3(a) and 3(b), the positive values of PF are shown as solid lines, the negative values as dashed lines and ∞ as solid black lines.

3.2. Scaling of proximity feature

Furthermore, we scale the PF using a kernel function $\eta : \mathbb{Z} \rightarrow \mathbb{R}$, with two options as defined by Eqs. (3) and (4), denoting $\eta_g(\cdot)$ and $\eta_h(\cdot)$ for Gaussian and Tanh kernel, respectively.

$$\eta_g(d) = exp(-(d/\alpha)^2) \tag{3}$$

$$\eta_h(d) = tanh(\alpha d) \tag{4}$$

For the Gaussian kernel, Eq. (3), $\alpha = (-d^2_{max}/log(\varepsilon))^{\frac{1}{2}}$, satisfying $\eta_g(d_{max}) = \varepsilon$ and for the Tanh kernel, Eq. (4), $\alpha = tanh^{-1}(1-\varepsilon)/d_{max}$, satisfying $\eta_h(d_{max}) = 1-\varepsilon$; for a small positive value of $\varepsilon > 0 \sim 0.01$ and a large d_{max} value.

The scaling characteristics of Gaussian and Tanh are shown in Fig. 5, for $d_{max} = 200$ and $\varepsilon = 0.01$. The scaling function η is an overloaded function and can be used on multi-dimensional vectors or tensors, i.e. $\boldsymbol{\eta} : \mathbb{Z}^{N\times K} \rightarrow \mathbb{R}^{N\times K}$, which allows us to apply a selected scaling function directly on PF_i, i.e. $PF^{\eta}_i = \eta(PF_i)$. The objective of the scaling function is to accommodate two important characteristics of the PFs:

(1) **Exponential Decay:** From a linguistic point of view, two markers close to each other have a significant relationship and this relationship fades away exponentially with distance, due to the change in the context of the conversation. For example, a Negation and Hedging marker appearing close to each other in the same utterance has a significant relationship. However, if they are 100 words apart, they have very little impact on each other. The distance between two markers is therefore mapped using a non-linear (Gaussian or Tanh) scaling function.

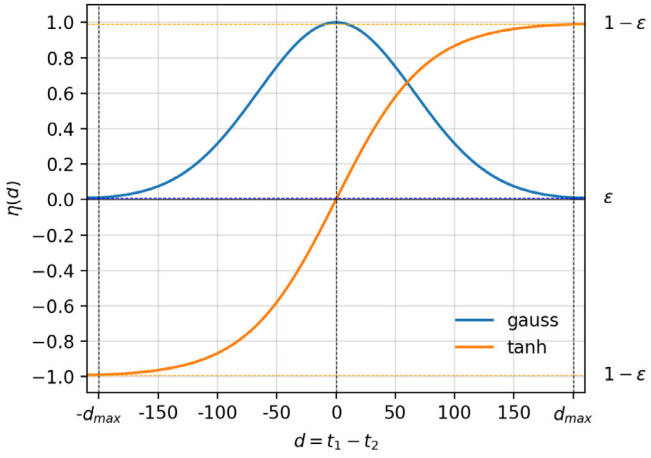


Fig. 5. Kernel functions for scaling. $d_{max} = 200$, $\epsilon = 0.01$.

(2) **Normalisation:** The kernel function η, allows a mapping of PF values between 0 to 1 (using a Gaussian kernel) or -1 to 1 (using a Tanh kernel), a well-defined range considered as normalisation of PF for any utterances. Note that by using the Gaussian kernel, we only preserve distance between markers and lose the direction information (+ve/−ve). However, the direction of markers in respect to each other, is preserved if the Tanh kernel is used.

The scaled PF for the highlighted example of a Hedging instance from Fig. 2 is shown in Figs. 4(a) and 4(b), with Gaussian and Tanh kernel function respectively. The choice of the two kernel functions, Gaussian and Tanh, is carefully selected to exploit the important characteristics of kernel functions and to impose them on the PFs. As shown in Fig. 5, the Tanh kernel preserves the direction of markers whereas the Gaussian does not. This allows us to control, whether we like to retain the directionality. Gaussian and Tanh also map the values in opposite direction of magnitude. For Gaussian, the proximity of two very close marker instances will be mapped to a higher value (close to 1), while for Tanh, this will be mapped close to zero.

Another distinction between the two kernel functions is the slope. Unlike the Gaussian, for the Tanh the slope for smaller magnitudes is higher (almost linear). These characteristics play an important role when choosing one kernel over another to satisfy the assumptions. For example, if we assume that the effect of markers changes linearly with the distance between them when they are in close proximity, and it changes very slowly when they are far away, then the Tanh kernel is more suitable.

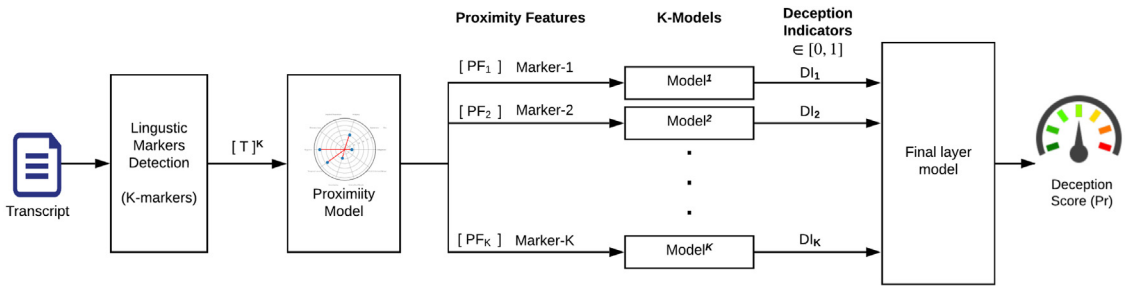


Fig. 6. The Decision Engine, based on the proximity of detected linguistic markers.

3.3. Proximity range

While computing PF using Eq. (2), a predefined value d_{max} can be set. Adjusting the value of d_{max} and enforcing $max(T_i) \leq d_{max}$ (by clipping), allows us to control the proximity range for the markers or restrict the range of proximity. In other words, how far we want a Proximity Model to look for other markers. Setting a low value of d_{max} will result in a small proximity range, which will exploit all the relationships of markers under close proximity only. For any marker above, the range will be considered as the same as one at d_{max} or having no marker. These characteristics can be exploited in conversations, where we know the rate of change of context. If the context of conversation changes quickly, then setting a small proximity is suitable, however for a full conversation with the same context, the proximity range can be set to the length of the conversation.

Considering $d_{max} = 1$ will be a special case, where an instance of a marker carries only the presence of itself and no information of other markers in its proximity. In this case, the PF vector will be a constant for all the instances and the proximity model turns to simply counting the trigger terms. This ensures that the Proximity Model is a generalised model, where counting trigger terms is a special case with $d_{max} = 1$.

4. Decision engine

The Decision Engine is designed to produce the final deception score for a conversation. The schematic of the Decision Engine is shown in Fig. 6. It consists of four processes: (a) Linguistic Marker Detection (b) Proximity Model to extract features (c) K-Models (d) and a Final Layer. The first two processes are explained in Sections 2.3 and 3, respectively, considering the list of linguistic markers in use (see Table 1). In this section, we will explain the last two processes of the Decision Engine.

4.1. K-Models

A set of K-Models is designed, with respect to K linguistic markers. The objective of $Model^i$ in a set, is to estimate the likelihood of an instance of a marker $m_i \in M_i$ being part of a deceptive conversation, D, given its PF $PF^{\eta}_{i,n}$, i.e. $Pr(y = D|x = PF^{\eta}_{i,n})$. Conventionally, this deception likelihood can be simply estimated by counting the frequency of detected markers in deceptive and non-deceptive conversations. However, such an approach ignores the order of the markers present, and it is this information that the proximity model is based on. For marker M_i, $Model^i$ generates the deception likelihood score for each instance m_i, and these are aggregated to produce a single score for each marker M_i, denoted as Dl_i. The aggregated score Dl_i indicates the overall deceptiveness of the marker M_i. For example, a high value of $Dl_i \sim 1.0$ indicates that all the marker instances of M_i are highly deceptive. For the absence of a marker M_i, i.e. $T_i = \emptyset$, the score is set to zero, i.e. $Dl_i = 0$. Similarly, when aggregate Dl_i is zero for n marker instances, this indicates that none of the n marker instances are part of a deceptive conversation. A set of K-Models generates K aggregated scores with respect to the K linguistic markers for a given conversation, where each score indicates the overall deceptiveness of each respective marker.

4.2. Final layer model

The Final layer model is designed to estimate the final Deception Score Pr of a conversation, for a given K deceptive scores Dl estimated by the K-Models. By design, the Final layer focuses on only the deceptiveness of each marker in a conversation. Each of the K-Models and the Final layer model is a pipeline of polynomial feature extraction, followed by a classifier, such as a Support Vector Machine (SVM), Decision Trees, Ensemble models, etc. The choice of polynomial degree and classifier are the hyperparameters optimised by the Decision Engine.

5. Experiment and results

In this section, we provide information on the datasets used to evaluate the Decision Engine, the evaluation metrics used, the experimental settings, and finish with a discussion of the results.

5.1. Datasets

For testing the proposed approach, we used two datasets; (1) the CSC Deceptive Speech dataset, which consists 32 h of interviews from 32 subjects (16 male, 16 female) in a lab setting. Each interview includes questions based on performance of interviewee in tests on different topics, a complete session based on a topic is labelled as either Truth, Lie-Down or Lie-Up. Interviewees were instructed to lie about their performance for a few topics, either convincing as to have better (Lie-Up) performance or worst (Lie-Down) compared to their original performance. When combined, the Lie-Up and Lie-Down data are labelled as Global Lie [19]. A total of 315 sessions were extracted. We have used the two classes: Truth and Lie (Global Lie). To balance the Truth and Lie classes, while training, we augmented the location sets of the True class by varying the locations of markers with a small random value, which raises the chance level performance to 50%. (2) A Financial Services dataset, consisting of real-world fraudulent/non-fraudulent phone calls, collected from two different financial services institutions. This dataset contains a total of 56 transcribed calls, 24 Non-Fraud, and 32 Fraud calls (chance level 57%). The calls in this dataset were concernedw with accessing bank account details. Fraud and Non-fraud calls were labelled by respective institutions upon follow-up investigation to identify the legitimacy of a caller. The average number of responses were 19. For both datasets, the conversation transcripts were divided into training and testing sets using a 70-30 split.

5.2. Evaluation metrics and base rate fallacy

In the field of deception/fraud detection, in addition to accuracy, false positives are critical to measure for two reasons. (1) A false positive is a very sensitive issue, as it corresponds to wrongly accusing someone of being deceptive or fraudulent. The desired system, therefore, should be able to surface the deceptive cases without any false positives, even if some of the deceptive cases are classified as non-deceptive. (2) To mitigate the Base rate fallacy, also known as the False rate paradox [28]. According to Base rate fallacy, the situation is misleading when the FPR of a system trained on experimental data is higher than the TPR of a real world scenario.

To illustrate the Base Rate fallacy for a deception detection system, consider a dataset of a 1000 examples with 500 as true positive cases (deceptive cases) resulting in a 50% True Positive Rate. A system evaluated on this dataset produces a 5% FPR and 0% FNR, which means that for this dataset, the system will detect 545 cases as deceptive, of which 500 are true, and 25 are false positives, which means a case that is flagged by the system is deceptive with 95% (500/525) confidence. However, if the same system is used in a real-world scenario, where 100 out of 10,000 cases are truly deceptive (1% is a most likely scenario), the same system with a 5% FPR will identify $100 + 10{,}000 \times 0.99 \times 0.05 = 595$ cases as deceptive. In this scenario, a case is flagged by the systems as deceptive with only 17% (100/595) confidence.

A simple approach to avoid this situation is to increase the threshold on the probability score, which reduces the FPR of the system. However, it also reduces the TPR of the system. Thus, higher accuracy for a system does not translate to the real-world scenario. This is the one reason why a higher AUC (area under the curve) value for a system is preferred. Although this approach does not guarantee that a threshold exists for which all the examples (at least 1, > 0) above the threshold are true positives. This depends on the distribution of the probability score generated by the classifier.

Therefore, in addition to accuracy and the F1-score, we also compute a score TPR_{F0}, which is the TPR with a threshold high enough to eliminate all the false positive cases, defined as:

$$TPR_{F0} = \frac{1}{|C_+|} \sum_{x \in C_+} \big(Pr(x) > thr\big)?1 \tag{5}$$

For threshold thr, such that

$$\sum_{x \in C_-} \big(Pr(x) > thr\big)?1 = 0 \tag{6}$$

which can be computed as

$$thr = max\{Pr(x) : x \in C_-\} \tag{7}$$

where $Pr(x)$ is a probability score estimated by the Decision Engine for an example x. C_+ is a set of all positive examples, i.e. the deceptive conversation (Fraud/Lie) class and C_- is a set of all negative examples, i.e. the Non-Fraud/Truth class. $|C_+|$ is the length of set C_+, a total number of examples in the positive class (Fraud/Lie). The score TPR_{F0} is assumed to be a very important criterion for the deception detection system. It allows us to observe if there exists a threshold on the deception score of a trained model that produces no false alarms. It is considered that a system with a high TPR_{F0} is better than a system with a low TPR_{F0}, regardless of poorer accuracy.

5.3. Training and results

As explained in Section 5.1, two datasets (1) CSC and (2) Financial Services, are used to evaluate the proposed approach.

Table 4
Testing results of the Decision Engine for deception detection using the Financial Services dataset.

Model for	RegEx			+BERT		
	Acc.	F1	TPR_{F0}	Acc.	F1	TPR_{F0}
Disfluencies	0.75	0.84	0.36	0.68	0.78	0.06
Explainers	0.58	0.38	0.00	0.79	0.22	0.00
Hedging	0.71	0.46	0.00	0.14	0.00	0.00
Memory Loss	0.95	0.00	0.00	0.85	0.80	0.13
Negation	0.58	0.43	0.07	0.56	0.33	0.14
Temporal Lacunae	0.00	0.00	0.00	0.00	0.00	0.00
Uncertainty	0.61	0.00	0.00	0.78	0.50	0.00
Final Layer	**0.67**	0.77	**0.50**	**0.72**	0.76	**0.60**

Table 5
Testing results of the Decision Engine for deception detection using the CSC Dataset.

Model for	RegEx			+BERT		
	Acc.	F1	TPR_{F0}	Acc.	F1	TPR_{F0}
Disfluencies	0.67	0.67	0.20	0.62	0.63	0.03
Explainers	0.65	0.67	0.25	0.63	0.67	0.06
Hedging	0.74	0.74	0.20	0.65	0.71	0.11
Memory Loss	0.70	0.80	0.13	0.68	0.72	0.02
Negation	0.67	0.66	0.18	0.64	0.63	0.05
Temporal Lacunae	0.33	0.50	1.00	0.33	0.50	1.00
Uncertainty	0.58	0.54	0.14	0.52	0.51	0.00
Untruthful Words	0.53	0.63	0.42	0.93	0.96	1.00
Withheld Infor.	0.67	0.73	1.00	0.67	0.77	0.57
Final Layer	**0.72**	0.72	**0.43**	**0.69**	0.68	**0.46**

First, the location sets for the linguistic markers, listed in Table 1, are extracted from the interviewee and customer responses in the conversational sessions. It is worth mentioning again, that for each conversation, $K = 10$ location sets are extracted, one for each marker, as described in Section 3. The location sets are extracted using the RegEx and MTDNN-BERT model, as described in Section 2.3. The experiment is conducted in two settings: (a) RegEx, where all the location sets are extracted using only the RegEx; and (b) +BERT, where the location sets of Hedging, Explainers, and Memory Loss are extracted using MTDNN-BERT and the remaining markers are extracted using RegEx. Once the location sets are extracted for all the sessions and conversations, the data is divided into training and testing sets, with a 70-30 split.

While training the K-Models and the Final Layer, a pipeline of polynomial feature extractors with degree 2 and four different classifiers, namely Logistic Regression, Naïve Bayes, SVM and XGBoost, with their respective hyperparameters were explored to achieve the maximum TPR_{F0}. The testing results of the Decision Engine, including the performance of each model (K-Models and Final Layer), are shown in Tables 5 and 4 for the CSC and the Financial Services dataset, respectively. The classifiers and respective hyperparameters for the Decision Engine to achieve the above-mentioned results are tabulated in Table 6. For PF extraction the Tanh kernel with $\epsilon = 0.01$ and $d_{max} = 500$ is used.

For comparative analysis with a baseline, we also trained the same four classifiers using only the frequency of linguistic markers, the results are tabulated in Table 7.

5.4. Discussion

It is worth reiterating here that the objective of a model in a set of K-Models is to estimate the deceptiveness of an individual marker instance, and the objective of the Final Layer is to estimate

Table 6
Classifiers and respective hyperparameters of the Decision Engine used to obtain the results shown in Table 4 and 5. Here, Poly (2) is a polynomial feature extractor with degree 2; for XGBoost, n is the number of estimators (trees) and d is the maximum depth of a tree; SVM (rbf) is SVM with rbf kernel.

Model	Financial services		CSC dataset	
	RegEx	+BERT	RegEx	+BERT
	Poly (2)			
K-Models	XGBoost ($n = 10, d = 5$)	XGBoost ($n = 10, d = 5$)	XGBoost ($n = 25, d = 10$)	XGBoost ($n = 15, d = 7$)
		Poly (2)	Poly (2)	
Final layer	Naïve Bayes	Logistic Regression	SVM (rbf)	SVM (rbf)

Table 7
Baseline testing results of deception detection for CSC and Financial Services (FS) dataset.

	Model	RegEx			+BERT		
		Acc.	F1	TPR_{F0}	Acc.	F1	TPR_{F0}
CSC	LR	0.57	0.41	0.11	0.56	0.33	0.08
	SVM	0.55	0.26	0.11	0.54	0.21	0.06
	NB	0.54	0.21	0.06	0.52	0.19	0.04
	XGB	**0.67**	**0.61**	**0.03**	**0.67**	**0.65**	**0.32**
FS	LR	0.56	0.64	0.10	0.56	0.64	0.10
	SVM	0.66	0.75	0.00	0.66	0.75	0.00
	NB	**0.67**	**0.75**	**0.10**	**0.67**	**0.75**	**0.10**
	XGB	0.56	0.64	0.40	0.56	0.64	0.40

the deceptiveness of the entire conversation given all the marker instances. It can be observed that the performance of not all 10 markers (from Table 1) appeared in Tables 4 and 5. Specifically, Withheld Information and Untruthful Words are missing in Table 4 and Implied Repetition is missing in both tables. Due to very few instances detected in the training set, the model for such markers were not trained, which, in effect, reduced the set of K-Models to 9 and 7 for CSC and the Financial Services dataset, respectively.

For the Financial Services dataset, the performance of the Final Layer from Table 4 shows that when using the RegEx alone, the Decision Engine is 67% accurate at predicting the fraud calls and the TPR_{F0} score shows that 50% of the fraud calls can be detected without any false alarms. The performance of the Decision Engine improves when using MTDNN-BERT for the Hedging, Explainers and Memory Loss markers. The accuracy increases to 72% and the TPR_{F0} to 60%. For the CSC dataset (See Table 5), when using only the RegEx, the accuracy of the Decision Engine is 72%, which goes down to 69% using BERT, however, the TPR_{F0} score improves from 43% to 46%. Comparing the results of the Decision Engine with the baseline models from Table 7, it can be observed that the Decision Engine with RegEx achieves the same accuracy (67%) as Naïve Bayes, and +BERT is better. On the other hand, the TPR_{F0} of the Decision Engine is higher in both cases compared to all the models in Table 7.

For the CSC dataset, the performance of the Final Layer from Table 5 shows that when using the RegEx alone, the Decision Engine is 72% accurate with 43% of TPR_{F0}, which goes up to 46% by using BERT for three markers. However, accuracy dropped to 69%. Compared to the baseline models from Table 7, it can be observed that the Decision Engine achieves better accuracy and performs much better at TPR_{F0} for both cases RegEx and +BERT.

Although CSC is a widely used dataset for deception detection, there is a lack of a well-defined benchmark to compare the results among the studies due to different experimental settings. The studies use different approaches, specifically different training and testing strategies, to evaluate their performance. Using a combination of lexical and prosodic features with a chance level of 60.2%, a study [19] achieved 62.8% accuracy, which was further improved to 66.4%, using speaker-dependent features with 5-fold cross-validation over 90-10 split. With similar features, another study [20] with a chance level of 60.4%, obtained 64.4% accuracy by averaging over 10-fold cross-validation. Using lexical and acoustic features from critical segments of speech with global lie, accuracy of 61.9% was achieved with a chance level of 50% [43]. Recent studies employing Deep Neural Networks have shown similar results. One study used lexical features and a Recurrent Neural Network (RNN) and achieved 61% accuracy [44]; another used acoustic features and an auto-encoder and achieved 62.78% accuracy [45]. In our study we employed 10 selected linguistic markers from the transcription and with the proposed Decision Engine, we achieved a testing accuracy of 72% and 69% with RegEx and +BERT, respectively. It is worth mentioning that we tuned the Decision Engine to maximise TPR_{F0}, not the accuracy. Although, our proposed approach cannot be directly compare to above mentioned studies, due to different training and testing strategies, compared to baseline models using only frequency of selected 10 linguistic markers from our study (See Table 7), proposed Decision Engine based on proximity of linguistic markers performed better. In terms of TPR_{F0}, proposed approached outperforms the baseline models with greater improvements.

Although the MTDNN-BERT model can detect three markers with better accuracy than the RegEx, using the MTDNN-BERT model does not always improve the performance of the respective models in the set of K-Models, i.e. comparing the performance of Hedging, Explainers and Memory Loss models in the RegEx and in +BERT (See Table 5). This is because the correct identification of a marker contributes to the PFs of all the other markers. Thus, using MTDNN-BERT might not contribute to an improved performance in the respective model, however, an improvement in the other models can be attributed to MTDNN-BERT.

From Table 6, it can be observed that for the K-Models, an ensemble approach based on Trees worked well, e.g. XGBoost. However, for the Final Layer, Naïve Bayes, Logistic Regression and SVM turned out to be a good choice. These classifiers, compared to the Tree-based XGBoost, produce a better probability distribution of the deception score, i.e. the distribution of samples around the decision boundary. This allows the Decision Engine to achieve the improved TPR_{F0} score.

6. Linguistic analysis of marker interaction

Together with our behavioural experts, we undertook a linguistic analysis of the trained Decision Engine to understand how the proximity of different markers affects the deception score, which has revealed several interesting observations. We only considered the interaction between two marker instances, to limit the complexity of interpretation, and this has been analysed for a Decision Engine trained on the CSC dataset.

It is important to note that the following analysis is on the pair of markers and the presence of 3 or more (multi-marker interaction) could be extrapolated from this, however, that increases the complexity of the analysis. Also, as mentioned, the following analysis is based on the Decision Engine trained on the CSC dataset, thus this is not representative of the population or other settings. The objective of this analysis is to demonstrate the richness and capacity of the proposed approach exploiting the Proximity Model to take the interaction markers into account.

The impact of the proximity of two marker instances on the deception score is computed by varying the distance of one

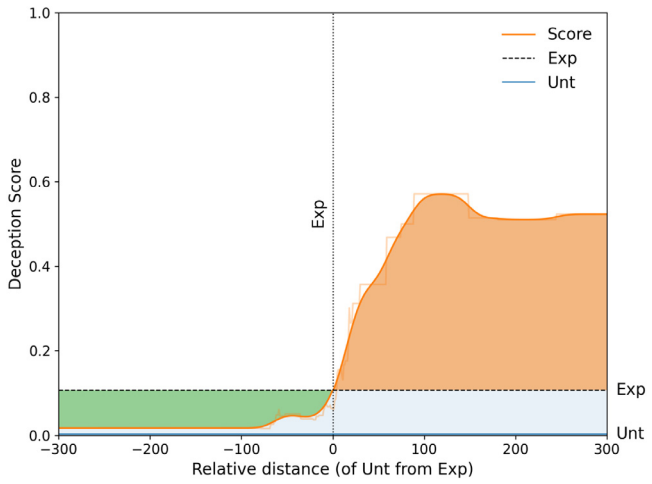


Fig. 7. Interaction between an Explainer (Exp) and an Uncertainty (Unt) marker instance.

marker instance with respect to another and keeping all the other markers absent. As we used the Tanh kernel for training, which preserves the direction of markers, the relative distance of one marker was varied to cover both directions to another. Fig. 7 shows the interaction between an instance of Explainers (Exp) and an instance of the Uncertainty (Unt) marker, by varying the relative distance of Unt with respect to Exp. Since the objective of the analysis is to observe the trend of impact, varying with respect to proximity, smoothing is applied to the raw deception score generated by the Decision Engine and plotted in Fig. 7. A baseline deception score due to the presence of an individual marker instance is shown by the horizontal line, i.e. only an instance of a respective marker is present and all the other markers are absent, which is 0.1 and 0.003 for Exp and Unt, respectively, shown in Fig. 7. The area with the smooth score is shaded to represent the impact of interaction on the deception score, i.e. the orange area represents increased deceptiveness and green represents decreased deceptiveness or increased truthfulness in comparison to the maximum of the baseline scores of the individual markers. We analysed the interaction between all the pairs, however, we will be discussing a few selected pairs in this paper.

Fig. 7 suggests that in a conversation, the presence of an Explainer followed by an Uncertainty marker is more likely to be deceptive. However, an Uncertainty marker followed by an Explainer is more likely to be truthful. This is consistent with our linguistic understanding of the markers, which suggests that an Explainer speaks to a perceived level of confidence and thereafter we expect the conversation to flow. An Uncertainty following an Explainer is more unexpected, and it lacks commitment to that which had already begun to be conveyed. In contrast, an Explainer following an Uncertainty is more expected, as to convey the explanation of uncertain information. The following examples of utterances, extracted from both CSC and CDLM datasets, demonstrate the above two cases with an Explainer because and an Uncertainty something. The first two examples are considered to be more suspicious, and the last two examples are more expected in a conversation. It is worth mentioning that these examples are isolated utterances from conversations and the Decision Engine was trained on samples of the entire conversation. Thus, such an utterance is not enough to categorise a conversation as deceptive or truthful. However, such events in the conversation do lead to the overall score for deception.

- *Well, because I'm interested in local politics and I'm interested in politics, my family was involved in politics since I was a little girl, so I've always, it's been something you know I've been exposed to so I've kept that up.*
- *She said she killed him because he was in love with my Mom or something.*
- *I think it was John or Jake or something. I remember, because I would always joke and call him Lentil Bean.*
- *I had to do something, because it's in violation of the building code.*

The interaction plot shown in Fig. 8(a) suggests that Disfluency and Explainers do not have much impact on the deception score individually. However, together they contribute to a higher deceptive score, indicating a more sensitive conversation towards deception or fraud. Linguistically, a Disfluency interrupts the flow of the conversation. It is used when the speaker needs time to think or articulate their response, which is a potential indicator of deception. Thus, Disfluency around an Explainer can denote greater sensitivity as it interrupts the flow of the explanation being conveyed. Disfluency following an Explainer is more sensitive than an Explainer following Disfluency since prior to a pause, reflected by the Disfluency, the subject knew the explanation and the Disfluency further slows the pace of the conversation. It is worth reiterating that the meaning of Disfluency could vary for a non-native speaker. In that case, Disfluency still indicates that subject is seeking time to articulate the response, however, the purpose of it might not be deceitful but rather an articulation of thoughts in a second language.

The following two examples demonstrate the case of an Explainer followed by a Disfluency, which is less likely to be truthful.

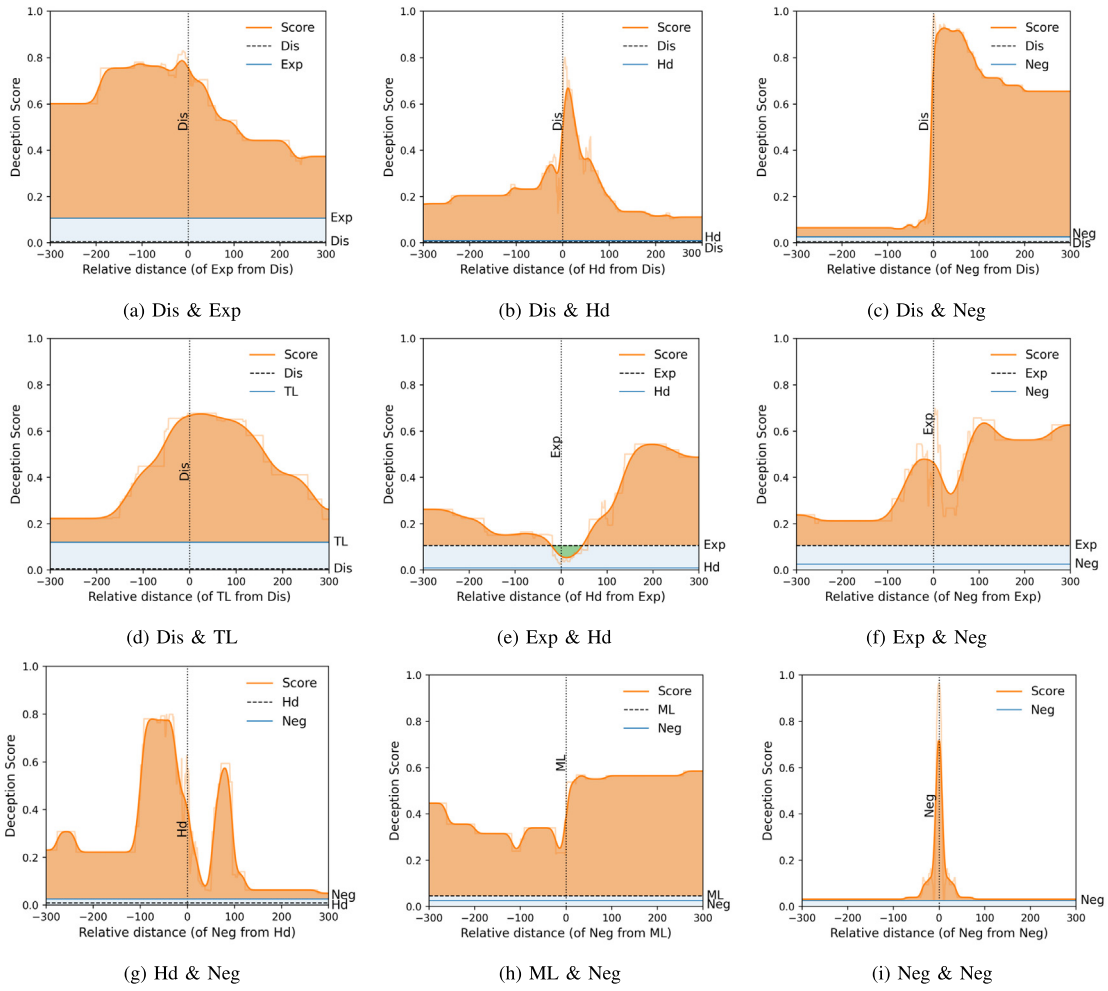
- *I do know because my brother um is also a chef.*
- *I've always had a problem with that because speaking Italian and having been uh a classical singer I want to say citarell.*

Fig. 8(b) suggests that Disfluency closely preceding Hedging is more likely to be deceptive and when it appears further away on either side, it is not as sensitive. From an analytical perspective, this impact depends on the type of question being asked. In response to an open question, the closeness of these markers indicates greater sensitivity towards deception. If both of these markers appear close together, this can be indicative of either the doubt or lack of commitment shown by the subject. However, Hedging preceding Disfluency is more likely, depending on the topic. If a subject is responding to a question where they go into memory of an event then this is expected, thus indicating a reduced level of sensitivity towards deception. The following examples demonstrate a Disfluency preceding Hedging (*guess*, *actually*), which is likely to be deceptive.

- *Um I pay a lot of attention to politics so it was excellent I guess.*
- *Um actually it was last night my boyfriend asked me.*

Fig. 8(c) shows the very interesting characteristics of Disfluency and Negation interaction. It shows that Negation following Disfluency is very likely to be very deceptive. However, Negation followed by Disfluency is not very sensitive. Anything reported in the Negative and preceded by a Disfluency weakens the assertion which the Negation may convey. Negation is itself a sensitive term and a weak assertion of Negation increases its sensitivity greatly. In the examples below, the first example is not likely to be deceptive and the second example is very likely to be deceptive.

- *Um I don't go regularly and I go once in a while.*
- *Not anymore but in this weather um back in Illinois I used to go camping.*



(a) Dis & Exp (b) Dis & Hd (c) Dis & Neg

(d) Dis & TL (e) Exp & Hd (f) Exp & Neg

(g) Hd & Neg (h) ML & Neg (i) Neg & Neg

Fig. 8. Interaction between two marker instances from the following markers; Hedging (Hd), Explainers (Exp), Disfluency (Dis), Negation (Neg), Memory Loss (ML) and Temporal Lacunae (TL).

Fig. 8(d) shows that Disfluency appearing close to a Temporal Lacunae in either direction is highly sensitive. This combination seeks to slow down the pace of conversation and provide thinking time before skipping over a specific piece of information, which could have been deliberately withheld.

The interaction of Explainers and Hedging is shown in Fig. 8(e) and suggests that Hedging appearing before an Explainer is less likely to be deceptive and when very close can negate deception and be more truthful. The order of markers speaks to a priority. An Explainer is the most sensitive indicator. Another sensitivity indicator following it will further increase the sensitivity. A subject asserts a reason for something (Explainer) and then expresses doubt (Hedging), whilst this is not a definitive indicator of deception, but it is unexpected. The following two examples demonstrate an Explainer and Hedging (*rather than, really*) close together, which are more likely to be truthful.

- *because I like to talk rather than listen.*
- *oh I did really good on that because I'm a tour guide.*

Fig. 8(f) shows that a Negation close to an Explainer or following further away is more indicative of deception. Negation is a sensitive indicator for several reasons, which could be towards truthful or deceptive. Negation with an Explainer reveals that the speaker deems what they said to be very sensitive information in their mind, often without knowing and they start explaining, for example, someone reports why something did not happen. This is specifically more sensitive if a question with *why* wasn't asked. An example of such a case is as follows:

- *I didn't do that because ...*

In Fig. 8(g), the interaction of Hedging and Negation shows that Hedging appearing closely before Negation is less likely to be deceptive. Negation before Hedging in context shows an increased likelihood of deception being present in the utterance. Hedging as the priority shows an element of uncertainty about what is to follow with the negative term, which can be expected in a conversation. A Negation preceding a Hedge term reflects increased sensitivity as the speaker's priority is to report in the negative, which is then followed by doubt or lack of commitment, weakening the assertion. Consider the following examples;

- *I have not spent much time there*
- *I don't know if I would say that*
- *I just moved back to new york so I was registered in texas and I have not yet registered here in new york.*
- *I don't think it was working.*
- *I think it was not working*

The last two examples in the above list, essentially convey the same information, but with a different order of Negation and Hedging. In this case, the first example (*I don't think it was working*) is commonly used and flows well in a conversation, whereas the second example (*I think it was not working*) takes more cognitive load, and thus is less expected.

The interaction between Memory Loss and Negation is shown in Fig. 8(h), which primarily shows that Memory Loss in presence of Negation is sensitive to deception, however, individually they are not. Specifically; Memory Loss followed by Negation is more likely to be deceptive than Negation following Memory Loss. However, both cases are sensitive and they contribute to an increased likelihood of deception. Even though some terms of Memory Loss include the Negation itself, such as *don't remember* and *don't recall*, our computation attributes these terms solely to Memory Loss. Linguistically, a Memory Loss with a Negation reflects a form of double negation. While recalling any past event, when a subject is saying *I don't remember*, this reflects that the subject is in effect in the present tense saying, I remember I have forgotten. It is unnecessary language.

- *No ... I don't remember that*
- *I don't remember what you talking about, I didn't do it.*

Fig. 8(i) demonstrates the interaction of Negation with another Negation, showing some interesting characteristics. The proximity of two Negation terms spikes the likelihood of deception. Linguistically, a Negation term is sensitive itself. Though, Decision Engine doesn't show a very high deception score for a single instance of Negation. However, a double Negation is extremely sensitive. It is an overselling statement of Negation. It can be seen in both truthful and deceptive conversations. It depends on the question asked and a psychological need for the subject to either persuade someone or a reliable denial such as:

- *No, I did not do that ...*
- *I wouldn't do it, never ever in my life*

It may not necessarily be nefarious but is worthy of further investigation.

From a linguistic perspective, the above analysis drawn from a trained Decision Engine is expected in a conversation with respect to deception and truth. Due to a small number of hits (counts) for Untruthful Words, Withheld Information, and Implied Repetition in the CSC dataset, the impact of proximity relating to these markers cannot be used for linguistic analysis, i.e. a small number of hits of any marker is not sufficient to generalise the impact of its proximity with other markers on the deception score. Thus, the analysis relating to these markers is not reported in this paper.

7. Conclusion

For deception detection, a widely used approach is based on counting the terms (words/phrases) of different linguistic features (markers). This approach misses the interaction between the markers. In this paper, we propose an approach for deception detection based on the proximity of linguistic markers, which captures the interaction of the markers. For linguistic marker detection, in addition to RegEx, we use the MTDNN-BERT model to minimise the ambiguity of markers. MTDNN-BERT was trained on our CDLM dataset labelled by behavioural experts for three markers. Our proposed approach extracts the proximity features from the location set of linguistic markers and trains the Decision Engine to estimate the deception score for a given transcribed conversation. We evaluate the Decision Engine on two datasets: the CSC dataset and a real-world Financial Services dataset of fraudulent/non-fraudulent phone calls. We evaluated the performance of the Decision Engine with an additional metric TPR_{F0}, which is a True Positive Rate with a high enough threshold to eliminate any false-positive cases. With an accuracy of 69% and 72%, we achieve higher TPR_{F0}, 46% and 60% for the CSC and Financial Services datasets, respectively. These results show that a trained Decision Engine can identify 46% of (60%) deceptive (fraudulent) conversations for the CSC dataset (Financial Services) without raising any false alarms. The results of the Decision Engine are compared with a baseline model, which uses the frequency of the markers. For both datasets, the proposed Decision Engine performs better than the respective baseline models. Although there is a lack of a uniform test set for a benchmark, when compared to other studies on the CSC dataset with various models, including recent studies using a Deep Neural Network, our testing accuracy is higher. Furthermore, we conducted a linguistic analysis on the interaction of the markers, which provides a deeper insight into the nature of deceptive behaviour in language, exposed by the interaction of linguistic markers. Since the analysis is conducted using the Decision Engine trained on the CSC dataset, the insights into deception are not representative of the population, which might vary with non-native speakers or in real-world scenarios. As our linguistic analysis suggests, for some cases the sensitivity of a response can be dependent on the question being asked, which we plan to further explore in the future to improve the Decision Engine, along with acoustic cues of deception.

CRediT authorship contribution statement

Nikesh Bajaj: Methodology, Writing – original draft, Conceptualization, Software. **Marvin Rajwadi:** Software, Visualization. **Tracy Goodluck Constance:** Software, Visualization, Validation. **Julie Wall:** Writing – review & editing, Supervision. **Mansour Moniri:** Writing – review & editing, Supervision. **Thea Laird:** Formal analysis, Validation. **Chris Woodruff:** Formal analysis, Validation. **James Laird:** Conceptualization, Data curation, Funding acquisition, Project administration. **Cornelius Glackin:** Project administration, Software. **Nigel Cannings:** Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: James Laird has patent #US 2021/0407514 A1 issued to United States.

Data availability

The authors do not have permission to share data.

Acknowledgments

This work was supported by Innovate UK National Project (Grant no. 104817): Automation and Transparency across Financial and Legal services: Mitigating Risk, Enhancing Efficiency and Promoting Customer Retention through the Application of Voice and Emotional AI.

References

[1] C.F. Bond Jr., B.M. DePaulo, Accuracy of deception judgments, Pers. Soc. Psychol. Rev. 10 (3) (2006) 214–234.
[2] M.G. Aamodt, H. Custer, Who can best catch a liar? Forensic Exam. 15 (1) (2006).
[3] ONS, Nature of fraud and computer misuse in England and Wales, 2019, URL https://www.ons.gov.uk/aboutus/transparencyandgovernance/freedomofinformationfoi/fraud.
[4] A. Bănărescu, Detecting and preventing fraud with data analytics, Proc. Econ. Finance 32 (2015) 1827–1836.
[5] J.W. Pennebaker, M.E. Francis, R.J. Booth, Linguistic inquiry and word count: LIWC 2001, Mahway: Lawrence Erlbaum Assoc. 71 (2001) (2001) 2001.

[6] R. Mihalcea, C. Strapparava, The lie detector: Explorations in the automatic recognition of deceptive language, in: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, Association for Computational Linguistics, 2009, pp. 309–312.
[7] M. Ott, Y. Choi, C. Cardie, J.T. Hancock, Finding deceptive opinion spam by any stretch of the imagination, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Association for Computational Linguistics, 2011, pp. 309–319.
[8] S. Feng, R. Banerjee, Y. Choi, Syntactic stylometry for deception detection, in: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, Association for Computational Linguistics, 2012, pp. 171–175.
[9] V. Pérez-Rosas, R. Mihalcea, Cross-cultural deception detection, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2014, pp. 440–445.
[10] N.K. Conroy, V.L. Rubin, Y. Chen, Automatic deception detection: Methods for finding fake news, Proc. Assoc. Inf. Sci. Technol. 52 (1) (2015) 1–4.
[11] N. Bajaj, T.G. Constance, M. Rajwadi, J.A. Wall, M. Moniri, C. Glackin, N. Cannings, C. Woodruff, J. Laird, Fraud detection in telephone conversations for financial services using linguistic features, in: AI for Social Good Workshop At 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), vol. http://arxiv.org/abs/1912.04748, 2019, arXiv:1912.04748. [Online] Available http://arxiv.org/abs/1912.04748.
[12] A. Mbaziira, J. Jones, A text-based deception detection model for cybercrime, in: Int. Conf. Technol. Manag, 2016.
[13] A.V. Mbaziira, J.H. Jones, Hybrid text-based deception models for native and non-native english cybercriminal networks, in: Proceedings of the International Conference on Compute and Data Analysis, 2017, pp. 23–27.
[14] B. de Ruiter, G. Kachergis, The mafiascum dataset: A large text corpus for deception detection, 2018, arXiv preprint arXiv:1811.07851.
[15] V. Pérez-Rosas, R. Mihalcea, Experiments in open domain deception detection, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1120–1125.
[16] Z. Wu, B. Singh, L.S. Davis, V. Subrahmanian, Deception detection in videos, 2017, arXiv preprint arXiv:1712.04415.
[17] G. Krishnamurthy, N. Majumder, S. Poria, E. Cambria, A deep learning approach for multimodal deception detection, 2018, arXiv preprint arXiv:1803.00344.
[18] V. Gupta, M. Agarwal, M. Arora, T. Chakraborty, R. Singh, M. Vatsa, Bag-of-lies: A multimodal dataset for deception detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019.
[19] J.B. Hirschberg, S. Benus, J.M. Brenier, F. Enos, S. Friedman, S. Gilman, C. Girand, M. Graciarena, A. Kathol, L. Michaelis, et al., Distinguishing deceptive from non-deceptive speech, 2005.
[20] M. Graciarena, E. Shriberg, A. Stolcke, F. Enos, J. Hirschberg, S. Kajarekar, Combining prosodic lexical and cepstral systems for deceptive speech detection, in: 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 1, IEEE, 2006, p. I.
[21] F. Enos, S. Benus, R.L. Cautin, M. Graciarena, J. Hirschberg, E. Shriberg, Personality factors in human deception detection: Comparing human to machine performance, in: Ninth International Conference on Spoken Language Processing, 2006.
[22] S.I. Levitan, A. Maredia, J. Hirschberg, Linguistic cues to deception and perceived deception in interview dialogues, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 1941–1950.
[23] F. Enos, Detecting Deception in Speech (Ph.D. thesis), Citeseer, 2009.
[24] J.F. Torres, E. Moore, E. Bryant, A study of glottal waveform features for deceptive speech classification, in: 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2008, pp. 4489–4492.
[25] S. Benus, F. Enos, J.B. Hirschberg, E. Shriberg, Pauses in deceptive speech, 2006.
[26] P. Ekman, M. O'Sullivan, W.V. Friesen, K.R. Scherer, Invited article: Face, voice, and body in detecting deceit, J. Nonverbal Behav. 15 (2) (1991) 125–135.
[27] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[28] M. Bar-Hillel, The base-rate fallacy in probability judgments, Acta Psychol. 44 (3) (1980) 211–233.
[29] B.M. DePaulo, J.J. Lindsay, B.E. Malone, L. Muhlenbruck, K. Charlton, H. Cooper, Cues to deception, Psychol. Bull. 129 (1) (2003) 74–118.
[30] S.L. Humpherys, A system of deception and fraud detection using reliable linguistic cues including hedging, disfluencies, and repeated phrases, 2010.
[31] F. Choudhury, Can language be useful in detecting deception? The linguistic markers of deception in the jodi arias interview, Diffusion-the UCLan J. Undergrad. Res. 7 (2) (2014).
[32] N. Smith, Reading Between the Lines: An Evaluation of the Scientific Content Analysis Technique (SCAN), Home Office, Policing and Reducing Crime Unit, Research, Development and …, 2001.
[33] J. Bachenko, E. Fitzpatrick, M. Schonwetter, Verification and implementation of language-based deception indicators in civil and criminal narratives, in: Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008), 2008, pp. 41–48.
[34] R.C. Jackson, et al., The pragmatics of repetition, emphasis and intensification (Ph.D. dissertation), Salford, 2016.
[35] S.H. Adams, Communication under stress: indicators of veracity and deception in written narratives (Ph.D. dissertation), Virginia Tech, 2002.
[36] L. Zhou, A. Zenebe, Representation and reasoning under uncertainty in deception detection: A neuro-fuzzy approach, IEEE Trans. Fuzzy Syst. 16 (2) (2008) 442–454.
[37] M. Hartwig, P.A. Granhag, L.A. Strömwall, A. Vrij, Detecting deception via strategic disclosure of evidence, Law Hum. Behav. 29 (4) (2005) 469–484.
[38] C. Danescu-Niculescu-Mizil, L. Lee, Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs, in: Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011, 2011.
[39] Seinfeld, Seinfeld TV series, scripts of 180 episodes, 1989-1998, URL https://www.seinfeldscripts.com/seinfeld-scripts.html. (Last accessed on 2019-12-03).
[40] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[41] X. Liu, P. He, W. Chen, J. Gao, Multi-task deep neural networks for natural language understanding, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 4487–4496, [Online] Available https://www.aclweb.org/anthology/P19-1441.
[42] M. Rajwadi, C. Glackin, J. Wall, G. Chollet, N. Cannings, Explaining sentiment classification, in: Interspeech 2019, 2019, pp. 56–60.
[43] F. Enos, E. Shriberg, M. Graciarena, J.B. Hirschberg, A. Stolcke, Detecting deception using critical segments, 2007.
[44] A. Chow, J. Louie, Detecting Lies via speech patterns, 2017.
[45] H. Fu, P. Lei, H. Tao, L. Zhao, J. Yang, Improved semi-supervised autoencoder for deception detection, PLoS One 14 (10) (2019) e0223361.