A Game for Crowdsourcing Adversarial Examples for False Information Detection

Jan Cegin[1,2,*], Jakub Simko[2] and Peter Brusilovsky[3]

[1]Faculty of Information Technology, Brno University of Technology, Bozetechova 1/2, Brno, 61200, Czech Republic
[2]Kempelen Institute of Intelligent Technologies, Mlynske nivy 5, Bratislava, 81109, Slovakia
[3]University of Pittsburgh, 135 North Bellefield Avenue, 15260, USA

Abstract

False information detection models are susceptible to adversarial attacks. Such susceptibility is a critical weakness of detection models. Automated creation of adversarial samples can ultimately help to augment training sets and create more robust detection models. However, automatically generated adversarial samples often do not preserve the information contained in the original text, leading to information loss. There is a need for adversarial sample generators that can preserve the original information. To explore the properties such generators should have and to inform their future design, we conducted a study to collect adversarial samples from human agents using a *Game with a purpose* (GWAP). Player's goal is to modify a given tweet until a detection model is tricked thus creating an adversarial sample. We qualitatively analysed the collected adversarial samples and identified desired properties/strategies that an adversarial information-preserving generator should exhibit. These strategies are validated on detection models based on a transformer and LSTM models to confirm their applicability on different models. Based on these findings, we propose a novel generator approach that will exhibit the desired properties in order to generate high-quality information-preserving adversarial samples.

Keywords

adversarial data generation, machine learning, false information detection, game with a purpose, human interaction task

1. Introduction

False information is used to manipulate opinions about a certain topic using inaccurate or invented plots. The proliferation of false information gave rise to *false information detection models* to help mitigate its spread. The state-of-the-art detectors often combine natural language processing techniques (NLP) and machine learning methods. However, they are not without their drawbacks as they can be tricked into misclassifying false information as real information [1, 2, 3, 4, 5].

Adversarial machine learning is used to understand how machine learning classifier models classify various samples and use this knowledge to exploit weaknesses of these models. This can be done maliciously, but sometimes there is a positive intent: to augment the training data and fine-tune the model, making it more robust. Recent works [1, 2] suggest that false information detection models are susceptible to adversarial attacks, which underlines the need for more robust models.

Current automatic adversarial text-augmenting techniques may lead to loss or change of the samples information. Multiple techniques like those standardized under the approach of TextAttack [6] can generate additional samples with deletion of least important words, synonym swaps, etc. However, such approaches may lead to the information of the original sample being modified. Changing names, locations, inserting or substituting words may also tamper with the original information the author wanted to convey.

Human-in-the-loop methods have been demonstrated as a promising approach for creating adversarial samples [3, 4, 7]. Human workers can be tasked with modifying the sample texts in return for a reward (e.g., monetary). Crowdsourcing platforms are often employed for this process [3, 4]. Created adversarial samples are then used for further training of a model, increasing robustness with each data collection and fine-tuning iteration. Human-in-the-loop methods are, however, expensive in terms of resources (time, money, etc.) to produce a meaningful amount of augmented data.

Given the existing issues of automatic text-augmenting adversarial generation techniques (information loss between the original and adversarial samples) and high costs of human-in-the-loop methods, *there is a need for automatic adversarial text-augmenting information-preserving methods.*

> *Our goal is to explore desired properties of information-preserving adversarial example generation methods to inform their future design (in the domain of false information detection).*

AIofAI '22: 2nd Workshop on Adverse Impacts and Collateral Effects of Artificial Intelligence Technologies, Vienna, Austria
*Corresponding author.
jan.cegin@kinit.sk (J. Cegin); jakub.simko@kinit.sk (J. Simko); peterb@pitt.edu (P. Brusilovsky)
0000-0003-2692-9320 (J. Cegin); 0000-0003-0239-4237 (J. Simko)

To reach this goal we conducted a study to collect adversarial samples from human workers using a *Game with a purpose* (GWAP) [8]. The players were tasked with tricking a false information detection model without modifying the information of the original sample. The game used a selection of tweets that the players modified. After the live study, we qualitatively analysed the collected adversarial samples and identified desired modification strategies that players used. Based on these strategies, we constructed the desired properties of information-preserving adversarial generators. Furthermore, the strategies were validated by an expert player on our GWAP with different underlying detection models based on a transformer and LSTM architectures. This validated the ability of these strategies to trick different detection models.

We seek answers to following research questions:

1. *RQ1: What player text modification strategies lead to information-preserving adversarial samples?*
2. *RQ2: Are the identified strategies of text modification able to trick more complex models?*

The main contributions of this paper are:

- A GWAP-based method for creation of information-preserving adversarial samples for false information detection.
- A qualitative analysis of the collected adversarial samples with identified possible strategies and behaviours for future automated generator.
- Verification of the identified strategies against a transformer and LSTM models using an expert in false information detection.

As a subject of our future work, we also propose a novel automated model-free information-preserving adversarial generator. This generator, built on reinforcement learning methods, will exhibit the desired identified strategies in order to produce high-quality adversarial samples.

The rest of this paper is organized as follows: related work is summarized in Section 2, the proposed GWAP is explained in detail in Section 3, the data collection study is described in Section 4 with its limitations in Section 5. The future work with the proposed model-free information-preserving adversarial generator is described in Section 6.

2. Related work: Text-based adversarial attack generation

Adversarial attacks have been employed in recent years to identify potential vulnerabilities of neural networks. This is mostly dominant in tasks such as autonomous driving, computer vision and natural language understanding tasks where the ability to discover potential issues is becoming more and more relevant [6, 9, 10, 1].

In most cases, the process of adversarial data generation consists of manipulating the given input intentionally to test the robustness of the model under test. These samples can be then used in further training of the model, increasing its robustness.

We have divided the relevant text-based adversarial attacks approaches into two categories: *automatic adversarial attacks* in Section 2.1 and *adversarial attacks with human input* in Section 2.2.

2.1. Automatic adversarial attacks

Over the years multiple methods have risen in the process of exploring robustness of NLP models. One of the automatic methods is *TextAttack* [6]. Its authors identified a difficulty in comparing various attacks and their effectiveness against models due to high variance of models under test and datasets. *TextAttack* was introduced in order to unify existing text-based adversarial attacks. It contains 16 types (character deletion, character insertion, word swap by various criteria, etc.) of adversarial attacks from various sources in literature. This enables easier benchmarking and comparison of models and datasets in terms of robustness.

Another approach, *DANCin SEQ2SEQ* [11], is based on Generative adversarial networks (GANs) to produce adversarial samples given a discriminator model. The GAN was tasked with tricking the discriminator while trying to be as similar as possible to the original sample. A REINFORCE framework [12] was used with cosine similarity to achieve this goal. *LexicalAT* [13] employed a similar method as *DANCin SEQ2SEQ* in producing adversarial samples targeting a sentiment classifier.

Adversarial attacks targeting machine translations were developed by [14] using an Actor-critic method. This was done using GANs and collecting K best candidates for each token in the model's vocabulary using Euclidean distance.

A reinforcement learning approach [15] utilized small perturbations in the form of misspellings or paraphrases in order to trick a given model. They targeted IMDB sentiment classification and news categorization tasks. The generated samples fool machine learning detection models while not confusing humans in the targeted classification task.

While these automated methods achieve state-of-art results in terms of adversarial samples generation, they might lose the original information in the process of generating these samples. As such, there is no guarantee that the information of the original sample from which the adversarial samples were generated is preserved. This implies the need for enhanced automatic adversarial generators so that their goal is not only to generate adversarial

samples, but also preserve the original information.

2.2. Adversarial attacks with human input

Adversarial NLI [16] employed human-in-the-loop methods in producing a large-scale NLI (natural language inference) benchmark dataset. This was done iteratively, with human workers devising samples that the model could not correctly label. These samples were then added to the training dataset and used for training a new iteration of the model.

Trick Me If You Can! [7] also used human-in-the-loop techniques to collect adversarial samples. Experts in a question answering game were tasked with tricking the model into choosing the wrong answer for the given question. However, the correct answer to the question was meant to be the same. Distractions and changes in reasoning were identified and employed by the human workers.

Another work [17] focused on adversarial samples created against various models by human workers. The goal of the workers was to create adversarial questions in reading comprehension tasks. Such adversarial samples were used to *(i)* first expose the weaknesses of given models and *(ii)* second to observe how additional training on different subsets of samples increased the model's robustness.

Human input has been successfully used in crodwsourcing to produce adversarial samples targeting various text-based models. However, these methods are often time- and resource-consuming, which implies a need for approaches that minimize the time and resources needed to produce these samples. These might be semi-automatic methods as well as automatic methods built on a qualitative analysis of the collected adversarial samples from human agents.

3. The Proposed method: GWAP

As a source for human-created adversarial samples of disinformative texts, we created a game with a purpose (GWAP), where players modify existing tweets (previously classified as disinformative) with a goal to trick the detection model yet still preserve the information in the tweet. The aim of the GWAP is to provide samples for further qualitative analysis to identify exploitation and exploration strategies of players that could inform future automated sample generators.

Our GWAP main scenario is as follows: A player is shown one of the possible samples, e.g. the tweet: *"The moon landing newer happened, it was a fake endeavour by the US to hide the reptilian takeover!"* Then, the goal of the player is to rewrite the tweet until the model changes its prediction from false information to true information. However, the information in the tweet (in this case the claim of the moon landing being faked in order to cover up another event) needs to be preserved.

For example, the player may change the tweet to *"The lunar landing didn't happen, complete cover up story by the US to hide the reptilian takeover!"* The model should indicate to the player the percentage of how much has its confidence[1] changed, leading the player in the right direction. In theory, this should lead the player in creating information-preserving adversarial samples.

The Trick Me If You Can study [7] served as an inspiration for our method. As the goal of the human workers in this study was to modify questions to trick the model while not changing the correct answer to the question, it is similar to our goal of producing information-preserving adversarial false information.

3.1. Player interface

The player interface is displayed in Figure 1 and is organized as follows: In the top section (1) under *Original tweet* is the original false information tweet. This section does not change through a game session so the player always has a point of reference to the original information of the tweet. Beneath (2) is the *Your last edit here* section with the latest modification of the tweet. This section is re-rendered with new text after every change made by the player. It also contains highlighted words as hints. The text area in *Edit Tweet Here* is where the player edits the tweet. The text in this area is identical to the last edit. The modified tweet is submitted to the server and evaluated, when the player hits the *submit* button.

The evaluated changes are displayed in the *Predictions* section (3) at the bottom. Here, the goal of the game session is displayed (e.g. "get the real news prediction to a given percentage"). To help with this goal, the table displays the model's confidence (in percentage) for the last edit of the tweet in terms of prediction probability towards false news and true news prediction. The *Change from last* column indicates the change in percentage between the latest's edits of the tweet. The text in cells of this column changes colour based on the positive or negative change for false/true news. If the change was positive, the text in the cell is coloured green and if negative, the colour is set to red. This is to indicate to the player if the changes that were made are producing desired results - tricking the model.

The *Skip* button is used to skip the current challenge in order to avoid frustration of the players. A modal window that needs to be confirmed is displayed in order for the challenge to be skipped.

The right side of the page (4) contains information about the current challenge. The number of tries left

[1] We refer to confidence of the model as the model's predicted probability for the given sample being real information.

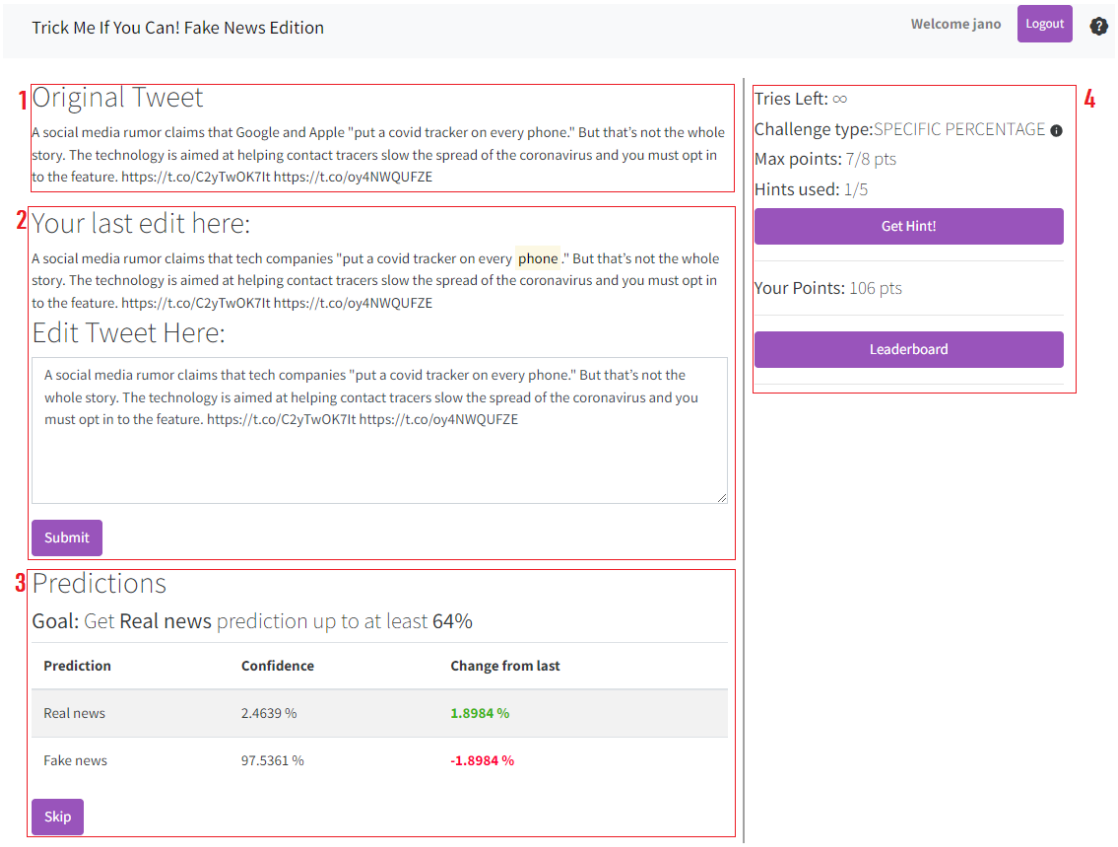

Figure 1: The interface of the GWAP with all its components: (1) contains the original tweet, (2) consists of the last tweet the player submitted with highlighted hint and the text area where the player can edit the tweet further, (3) displays the goal of the game, changes in prediction and the confidence of the detection model with the *Skip* button to skip the challenge and (4) contains challenge type, number of tries left, hints left, points, leaderboards and *Get Hint!* button.

on the current challenge is displayed with the challenge type. The information indicator displays a hover window with further information about the type of the challenge. Different challenges are explained in depth in Section 3.2.

Points that can be earned, if the challenge is successfully fulfilled, are displayed in the format of *points awarded/maximum points*, as players can get varying number of points for a successful challenge. *Get Hint!* button displays a hint in the *Your last edit here* by highlighting a hinted word. *Hints used* displays the remaining hints. Hints are further explained in Section 3.2.4. User points are below with a *Leaderboard* button that display the leaderboard based on points.

Players can always display the instructions by clicking on the question mark button in the top right. This displays the instructions in a modal window.

3.2. Game scenarios

3.2.1. Challenges

Tweets are assigned to players randomly, without filtering their contents or the confidence of the model for the tweet. Tweets are assigned in forms of challenges, which were introduced in order to incorporate gamification techniques. The different challenges are:

1. *CLASSIC*: In this challenge the player is tasked with changing the accuracy of the model for the given tweet to at least 70% in the *Real information* class. The player can use hints and is not limited by any number of tries. If successful, the player is rewarded a maximum of 10 points.
2. *TRIES*: In this challenge the player is tasked with changing the accuracy of the model for the given tweet to at least 70% in the *Real information* class. The player can use hints and is limited by a randomly set number of tries between 9 to 17. The

maximum number of points rewarded if successful is 1.1x to 1.3x more compared to the *CLASSIC* challenge, as the limited number of tries is more challenging than the *CLASSIC* challenge.

3. *SPECIFIC-PERC*: In this challenge the player is tasked with changing the accuracy of the model for the given tweet to at least a randomly generated percentage between 50 - 70% in the *Real information* class. The player can use hints and is not limited by any number of tries. The number of points rewarded if successful is 0.7x to 1x less compared to the *CLASSIC* challenge, as raising the confidence of the model to a smaller percentage is in principle easier than the *CLASSIC* challenge.
4. *NO-ATTENTION*: In this challenge the player is tasked with changing the accuracy of the model for the given tweet to at least 70% in the *Real information* class. The player can not use hints and is not limited by any number of tries. The maximum number of points rewarded if successful is 1.8x to 2x more compared to the *CLASSIC* challenge, as the ability to not use hints for this challenge makes it the most challenging of them all.

The idea behind the introduction of these challenges was to increase the game time of players, as they would be motivated to fulfil different challenges and increase the number of collected adversarial samples in the process. A player can work with the same tweet up to 4 times, each time as a different challenge.

A point system was introduced to further increase game engagement. Leaderboards display top 5 players based on points and players can be rewarded if they successfully complete their challenges. As we observed during our experiments, this motivated some players to keep playing in order to beat other players, increasing the number of collected adversarial samples even further.

3.2.2. Seed data

In order to pick the most fitting dataset for this study, multiple criteria were considered. As most of the participants of the study would be Slovak nationals, datasets that contained political false information from other nations were not considered. To reduce player cognitive load, only datasets consisting of tweets were considered, due to their short length compared to other forms. As such, a dataset of false information with a familiar topic to Slovak nationals was considered. Covid false information was chosen as it is a recent topic. We chose a dataset [18] of Covid related false information tweets consisting of 10,700 samples, with a 60:20:20 train:test:validation split. The dataset has two possible labels for its samples - *false* and *real*.

3.2.3. False information detection model

The baseline detection model used in our study was a SVM model based on TF-IDF features. This model was used as a baseline for a competition[2], where our chosen dataset was used and is considered the best performing baseline in the paper [18] associated with the dataset. For a follow-up experiment, we chose different models, namely a pretrained Distilbert [19] from the *Hugging Face*[3] library and a bidirectional LSTM with 300 units and one hidden layer of size 256 with dropout set at 0.5. The models were fine-tuned on our dataset for 5 epochs with *1e-5* learning rate.

3.2.4. Ingame hints

To compute a hint for the baseline model used in the GWAP and LSTM model used during evaluation, we simulated how the model predictions change when a particular embedding of a word is set to zero. As such, word removal was approximated for each word and the most impactful word was highlighted as the hint [20].

For the transformer model, the library *transformers-interpret*[4] was used to retrieve words that the transformer model assigns the highest attention to.

Usage of hints was, however, penalised. For each hint used, the player lost one point from the maximum points that could be awarded for successfully completing the challenge. This was introduced to reduce the usage of hints, as exploitation of them could lead to high bias in producing adversarial samples.

3.2.5. Data used in the GWAP

To choose fitting tweets for the GWAP various factors were considered. First, the validation split of the dataset mentioned in Section 3.2.2 was utilized. Second, only tweets which the model correctly classified as false information were chosen. Third, tweets shorter than 90 characters were filtered out. Finally, the remaining tweets were filtered manually. This was to remove ambiguous statements and tweets consisting mostly of names and places (more than 50% of the tweets length). As such, the final number of tweets used in the GWAP was 41.

For the model confidence calculation of the submitted edited tweet, a few limitations were put in place. The maximum length of the tweet was limited to a total of 280 characters, limiting the addition of new words or phrases. Tags of other twitter users and links were removed before the calculation of the model confidence, as those were of no concern for this study. Diacritic was also not considered, as the focus was set on word-based approaches. Finally, in order to prevent excessive cheating, a measure

[2]https://competitions.codalab.org/competitions/26655
[3]https://huggingface.co/docs/transformers/model_doc/distilbert
[4]https://github.com/cdpierse/transformers-interpret

of Levenshtein distance between the original and edited tweet was introduced. The maximum threshold for the Levenshtein distance is $max(lev) = 0.7 * l(t_o)$, where $l(t_o)$ denotes the length of the original tweet.

If the length of the tweet or the maximum Levenshtein threshold were surpassed, a notification was displayed to the player informing them of an invalid submission and prompting the player to change the edit.

3.2.6. Technical details

The backend of the GWAP was written in Django Rest Framework with the user interface written in React JS in a client-server architecture. For the highlighting of tweets, the *react-highlight-words*[5] library was used. Every submition of the edited tweet was saved into a PostgreSQL database on the server. The GWAP was dockerized for easier deployment.

4. Study: Data collection and evaluation

To collect adversarial samples, we deployed our GWAP in a time window of two weeks. Our players were Slovak nationals familiar with false information detection, aged from 20 to 50 (13 males, 4 females), and with background in IT. The instructions for the GWAP were given in the form of a 7 minute video playthrough with commentary that the players were asked to watch before beginning the study.

In total, 130 adversarial samples were collected from 17 players during roughly 17 hours of player activity. We manually controlled the samples and removed 2 of them. The removed samples had modified information which was different compared to the seed data. All other samples satisfied the information equality criterion (which was due to a disciplined player base). Three samples that were skipped instantly were also not included for the qualitative analysis. 125 samples were left, out of which 80 samples (64 %) were successful in tricking the model.

4.1. Qualitative analysis of collected adversarial samples

We performed a qualitative analysis of the collected adversarial samples, addressing first of our research questions to detect possible game text modification strategies and other behaviours.

> *RQ1: What player text modification strategies lead to information-preserving adversarial samples?*

In order to answer this question, reoccurring steps in the process of fulfilling a challenge were identified.

We identified three basic steps: *addition of a word*, *deletion of a word*, and *substitution of a word*. Of these, there were 187 *additions*, 188 *deletions* and 1530 *substitutions*.

66.8% of all of these basic steps have had a positive impact; the model confidence towards real information increased after their usage. The number of cases of positive and negative impact of the basic steps had was different for successful and failed challenges. 69.72% of the basic steps had a positive impact in the successful challenges, while only 58.4% of the basic steps had a positive impact in the failed challenges.

These basic steps have been then used to identify more complex types of strategies which are listed below. We differentiate between *strategy steps*, which consist of complex modifications of the original tweet with some intent, and *behavioural strategies*, which affect the players exploitation/exploration behaviour.

In total, 7 *strategy actions* were identified that the players used during their gameplay. The number of *strategy steps* in adversarial samples can be found in Table 2. The identified *strategy steps* are:

- *Usage of abbreviations* - substituting abbreviated forms of words for their full forms and vice versa.
- *Changes in spelling* - misspelling or correcting the spelling of a word.
- *Change of tense* - changing the tense of the sentence/tweet.
- *Phrase manipulation* - manipulating a phrase from the tweet by adding/deleting/substituting one or more words.
- *Singular/Plural of nouns* - changing a noun from singular to plural and vice versa.
- *Obfuscation of names* - obfuscating names and places by substituting them for their synonyms.
- *Change of units* - change of units and values associated with them like weight, distance, etc.

Examples of usage for each of the strategy steps can be found in Table 1.

Players have on average spent 8m 16s on a challenge. For successful challenges, the average time spent was 9m 46s, with shortest at 24s and longest at 1h 17m 56s. For failed challenges, the average time spent was 3m 4s, with shortest at 44s and longest at 56m 15s.

Strategy steps have been used in various combinations multiple times during challenges. It has been observed that the most successful players used all of these steps at some point, while less successful players often used only one or two steps during their playtime - mostly *Phrase manipulation* and *Abbreviations*, as noted in Table 2. This led less successful players to frustration, as some samples required multiple different steps to be successfully transformed into an adversarial sample.

[5] https://www.npmjs.com/package/react-highlight-words

Table 1
Examples of *strategy steps* that were identified from the data of the study with the original sample and modified sample.

Strategy step	Original text	Modified text
Misspellings	The hospital beds apperaing on the scene during the opening show...	The hospital beds **appearing** on the scene during the opening show...
Abbreviations	...is 100 times more “toxic” 72 hours after death...	...is 100 times more “toxic” 72**h** ~~hours~~ after death...
Tense change	Facebook post that compares deaths in Italy...	Facebook post **is comparing** deaths in Italy...
Phrase manipulation	Cure for Corona Virus.	Treatment for #covid19!
Singular/Plural	...therefore people should "avoid salads" over fears of contracting COVID-19.	...therefore people should "avoid salad~~s~~" over fears of contracting COVID-19.
Obfuscation of name	... claimed that a vaccine for coronavirus has been discovered.	...which claimed that a vaccine for **covid19** has been discovered.
Unit change	...milk sweets older then 48 hours.	...milk sweets older then **2 days**.

Different usage of hints and hinted-at-words was also noted. Less successful players often relied too much on hints, changing only hinted words. However, such behaviour led players into a very greedy strategy - always changing only the current highlighted-hinted word. Hints change after every submit of the edited tweet, displaying possibly different hints after each edit. As such, less successful player, employing only some of the strategy steps and changing only hinted words, had harder and harder time increasing the confidence of the model toward real information the longer the challenge went on. This was due to the fact that the limited number of used strategies limited their actions and changing hinted at words confused them even further, ultimately leading to frustration and failure.

On the other hand, the most successful players did not focus on hinted at words too much. Rather, when using hints, they focused on words around these highlighted, hinted words - changing phrases with the hint, changing tense of the sentence, etc. These changes led to a steadier increase of the model's confidence toward real information after each submit. This underlines the fact that all of the identified *strategy steps* are needed for successful creation of adversarial samples as consistent usage of all of these steps proved important (although each of these steps was used with different frequencies).

In terms of *behavioural strategies*, several differences were noted between successful and less successful players. Successful players learned to reuse certain *strategy steps* and words associated with them in order to get a head start, exploiting the model's weaknesses. For example, the words *coronavirus* or *covid-19* could be changed with (in most cases) great increase towards the *real information* label to *covid19*. Steps like *Changes of units* and *Obfuscation of names* could be often reused on certain recognized words by players. As such, players learned and successfully exploited some of the observed patterns.

This could have also, however, led to a greedy approach. In these cases, the players edited only those parts of the tweet that, given their previous experience, should have yielded the highest increase in the models confidence. Such a greedy strategy then led to exploitation of learned options without any exploration of new options. As more and more of the original tweet was then changed, the players had less and less options left in their greedy strategy as the challenge progressed. Without any desire to explore different options, the players then gave up. This finding underlines the need for a balanced exploration/exploitation approach.

Players who successfully finished multiple tweets have employed local exploration in the challenges they received - searching for the best *strategy step* given the current edit of the tweet. This was often done in the form of very small perturbations to observe their effect and, if an impactful change was found, players continued in replicating this wherever possible. This led to a balanced strategy, where exploration strategies were used to learn new strategies.

Finally, players with failed challenges or multiple unsuccessful edits of a tweet tended to get frustrated and a human factor must be taken into consideration. As noted above, the higher average time spent on successful attempts is an indication that players needed to be patient with their changes in order to successfully pass the challenge.

As such, the identified *behavioural strategies* can be summarized as follows:

- *Exploration/exploitation balance* - the ideal strategy consists of reusing learned exploits with exploring local best perturbations to learn new strategies. Usage of hints is also included here, as

Table 2
Number of occurrences of strategy steps in all adversarial samples. Value in brackets denotes the relative percentage of the particular strategy step out of all steps in that category.

Strategy step	All occ.	Successful occ. (% of suc. steps)	Failure occ. (% of fail. steps)
Misspellings	48	37 (12.09%)	11 (9.56%)
Abbreviations	90	58 (18.95%)	32 (27.82%)
Tense change	47	33 (10.78%)	14 (12.17%)
Phrase manipulation	71	58 (18.95%)	13 (11.3%)
Singular/Plural	29	24 (7.84%)	5 (4.34%)
Obfuscation of name	112	77 (25.16%)	35 (30.43%)
Unit change	24	19 (6.21%)	5 (4.34%)

changing of words surrounding the hinted word yielded the best results compared to changing hinted words directly.

- *A diverse set of strategy steps* - a diverse set of *strategy steps* leads to less frustration (using something different when current steps fail) and a higher chance of success.
- *Patience* - a patient approach to the given challenge yields success, as creating an adversarial sample takes considerable time.

In conclusion, complex *strategy steps* and *behavioural strategies* that were used in creating successful adversarial samples were identified. The usage of all of these is needed in order for the players to not get stuck during their trials and to successfully finish a challenge. A good balance between exploitation (reusing learned behaviours and strategies) and exploration (using strategy steps on new phrases, words, etc.) is also required for consistent performance. Finally, players have to be patient in order to create successful adversarial samples, as their creation took a considerable amount of time.

4.2. Evaluation of adversarial samples and strategies on a transformer and LSTM model

RQ2: Are the identified strategies of text modification able to trick more complex models?

Our RQ2 aims at testing the generality of our findings. As more complex models we chose the ones built on a transformer and LSTM architecture (a *DistilBert* pretrained model fine-tuned on our study dataset and a bidirectional LSTM as described in Section 3.2.3). Against these models, we compared the collected adversarial samples. Then, we let an expert use the identified strategies against the complex model (in a follow-up gameplay study).

In order to test if the identified *strategy steps* and *behavioural strategies* are able to trick a more complex models we conducted a follow-up gameplay study with the complex models. An expert that participated in the study and played the game has been employed. The expert was one of the best players in terms of accumulated points and the player has exhibited the identified *strategy steps* and *behavioural strategies*. The player was also acquainted with the qualitative analysis that was conducted and its findings.

The follow-up study setup was the same as for the main study (with underlying detection model different). The expert was playing with a new instance of the GWAP with the same data and interface. We expected that the expert might encounter the same tweets again, however this was not a confound as player's learned strategies may not work with the new model. The same expert was employed to play the GWAP with the transformer and LSTM models respectively.

The expert was playing the GWAP with a transformer detection model first followed by the LSTM detection model next. In total, the expert played 25 challenges with the transformer detection model with 18 (72%) of them successful in tricking the model. All unsuccessful challenges occurred in a row in the beginning of the session. This happened for multiple reasons - the average starting confidence of the model toward real information was 0.096% for the transformer model compared to the SVM model where the average value was 6.35%. It was also noted that compared to the original setup, the edits made to the tweet did not mostly result in steep changes in confidence of the model. Most of the strategy steps conducted on the tweet resulted only in a very small (smaller than 0.001%) change in confidence which the expert had harder time accommodating to. Such difference in game's behaviour confused the player at the start. This led to frustration in the first tries and skipped challenges.

However, after the expert successfully completed one of the challenges, a streak of 18 successful attempts followed. As the expert noted, the very small positive changes to confidence were used as a guide to create a successful adversarial sample. Further, it was observed that while most of the changes to confidence were very

Table 3
Number of occurrences of strategy steps in successful adversarial samples of LSTM and DistilBert detection models setups during expert evaluation. Value in brackets denotes the relative percentage of the particular strategy step out of all steps in that category.

Strategy step	Successful occ. LSTM (% of steps)	Successful occ. DistilBert (% of steps)
Misspellings	7 (5.65%)	10 (6.45%)
Abbreviations	17 (13.71%)	23 (14.84%)
Tense change	11 (8.87%)	18 (11.61%)
Phrase manipulation	30 (24.2%)	45 (29.03%)
Singular/Plural	6 (4.84%)	10 (6.45%)
Obfuscation of name	42 (33.87%)	40 (25.81%)
Unit change	11 (8.86%)	9 (5.81%)

small, the last applied strategy step led to a huge (higher than 70%) leap in terms of confidence, successfully completing the challenge.

The expert employed all of the identified *strategy steps* with the *behavioural strategies* that were identified above. The *strategy steps* were used interchangeably where applicable with the expert exploiting learned words, while also exploring new possible edits on every new challenge. After the first few successful attempts, the expert patiently found the best perturbations for a given *strategy step*, which the expert then applied to the current tweet. The positive or negative confidence changes of the transformer model towards *real information* label were used as a guide for this process. If the expert used hints, mostly words close to the hinted word were changed, leading to successful changes.

The expert played in total 29 challenges with the LSTM detection model out of which 23 (79,31%) were successful in tricking the model. The unsuccessful challenges again occurred near the beginning when the expert adapted to the new detection model. The expert again employed all of the *strategy steps* with the *behavioural strategies* that were identified above and used them interchangeably. The behaviour of the expert and the perturbations produced by the expert were similar to the ones from the transformer detection model setup.

The comparison of the number of *strategy steps* for the LSTM and transformer setups can be found in Table 3. As can be observed in comparison with Table 2, the relative amount of occurrences of *Misspellings*, *Abbreviations* and *Singular/Plural* have decreased both in the LSTM and DistilBert GWAP setup. In contrast, *Obfuscation of names* and *Phrase manipulation* relative occurrences have increased. As this is a comparison of one participant (namely the expert) on the LSTM and DistilBert GWAP setups and multiple participants on the original SVM setup no clear conclusions can be drawn from this comparison. However, it might still indicate that different types of strategy steps might be preferred and viable for different types of detection models.

Table 4
Time needed for the expert player to complete challenges successfully given different underlying detection models. The expert needed less time for the LSTM detection model that was used in the GWAP as the last detection model. This indicates a successful application of the learned *strategy steps*.

Detection model	Avg. time	Min. time	Max. time
SVM	3m 4s	24s	7m 8s
DistilBert	3m 55s	23s	11m 30s
LSTM	1m 25s	6s	5m 45s

The expert needed on average more tries (10 tries vs. 16 tries) and time (3m 4s vs. 3m 55s) to finish the challenge on the evaluation transformer setup. However, the expert needed on average fewer tries (7 tries) and time (1m 25s) to finish the challenges on the LSTM evaluation setup. These comparisons can be found in Table 4 and Table 5. This might have been due to the fact that the expert was accustomed to the possible strategies that were needed to trick the models and thus had no difficulty in producing adversarial sample.

In summary, the successful application of identified *strategy steps* together with the *behavioural strategies* led to a successful streak of challenges with a rough start, where the expert needed to accommodate to the transformer and LSTM models. We conclude that the identified strategies are able to trick more complex models, namely a transformer and LSTM model.

4.3. Gameplay evaluation

We collected feedback while conducting this study to identify possible improvements. To gather feedback, players were asked to provide a satisfaction rating from 1 (worst) to 5 (best) after each skipped or successful challenge. They could also provide text feedback.

The average rating obtained was 3.45 out of 5 which signifies a generally pleasant experience for the players. The average rating for successful challenges was 4.21 and

Table 5
Tries needed for the expert player to complete challenges successfully given different underlying detection models. The expert needed fewer tries for the LSTM detection model that was used in the GWAP as the last detection model. This indicates a successful application of the learned *strategy steps*.

Detection model	Avg. tries	Min. tries	Max. tries
SVM	10	2	37
DistilBert	16	2	42
LSTM	7	1	24

for unsuccessful challenges 2.43, showing that accumulated frustration with unsuccessful challenges led to a generally bad experience.

From the collected feedback, we identified possible improvements to the prototype:

- *Task assignment* - The random assignment of challenges is not ideal. It leads players to duplicated samples in a row and sometimes assigns harder challenges too early. This leads to a steep learning curve. A greedy approach could be implemented to mitigate these problems.
- *Tutorial level* - A tutorial level would serve as a more interactive starting point for the players rather than an instruction video.
- *Revert back button* - A button that would revert the player's last changes and display the best edit so far (with the highest confidence) was identified by the players as a needed addition to the interface. Players tended to forget what their best edit was, leading to more edits than necessary.
- *More challenging levels* - The most successful players had harder time engaging with the game after a period of successfully finishing their challenges, deeming them too easy. This could be redeemed by a human-in-the-loop approach – the created adversaries would be used in further training of the model and the best players would have to trick this new iteration of the model. Such approach could, in theory, create more robust models while also providing more challenging levels for the players.

5. Study limitations

In this work, the focus was only on the false information detection domain with only one dataset used. This may lead to strategies that could be potentially applicable only in one domain or only on one particular dataset. The applicability of the identified *strategy steps* in different domains is as such an open problem that can be explored further in future work.

Another limitation is the fact that manual checks were needed after the study was done to assert that all collected adversarial tweets contained the information of the original tweet. This is a clear bottleneck of the GWAP for the moment as it limits its capabilities for a fast and effective data collection tool. This could be mitigated by implementing checks for pairs of tweets. These pairs would be then evaluated to determine if the given pair of tweets has the same information. This could be achieved by machine learning methods or by an ensemble of metrics from the domain of natural language processing.

During our evaluation we employed only one expert in the game. This expert has first played the GWAP with a transformer detection model setup and then with an LSTM detection model setup. The expert had far better performance on the second, LSTM, GWAP setup. It is not clear if the order in which the expert played the game with different GWAP setups affected the resulting performance, but we will explore this possibility in the future to determine if a certain order of different GWAP setups could result in a better or worse performance given by the player.

6. Future work: Proposed adversarial generator

Based on the findings of our studies, we propose a model-free, information-preserving adversarial sample generator for short false information texts. This design will be elaborated and evaluated as part of our future work.

The *strategy steps* and *behavioural strategies* identified in Section 4.1 serve as the desired properties for a future automated approach. The future approach should be able to replicate the identified *strategy steps* while focusing on a balance between exploration and exploitation. It should be a model-free and dataset independent solution to serve as an adversarial generator for future classification models, regardless of their classification task, where information preservation of the original samples is important.

To the ends mentioned above, the reinforcement learning (RL) agent approaches [21] are the most fitting. Their ability to perform a given action in the current state observed in an environment and receive a reward for their actions is fitting for the problem of creating adversarial samples to trick a given model. The ability to learn policies enables these approaches to exploit learned actions while also exploring other options.

Q-learning and policy gradient methods are some of the most popular methods for training an agent. As per [5], deep Q-learning is a good fit for this proposed approach. In deep Q-learning, the agent uses a neural network as an approximator to find the best action regarding the current state.

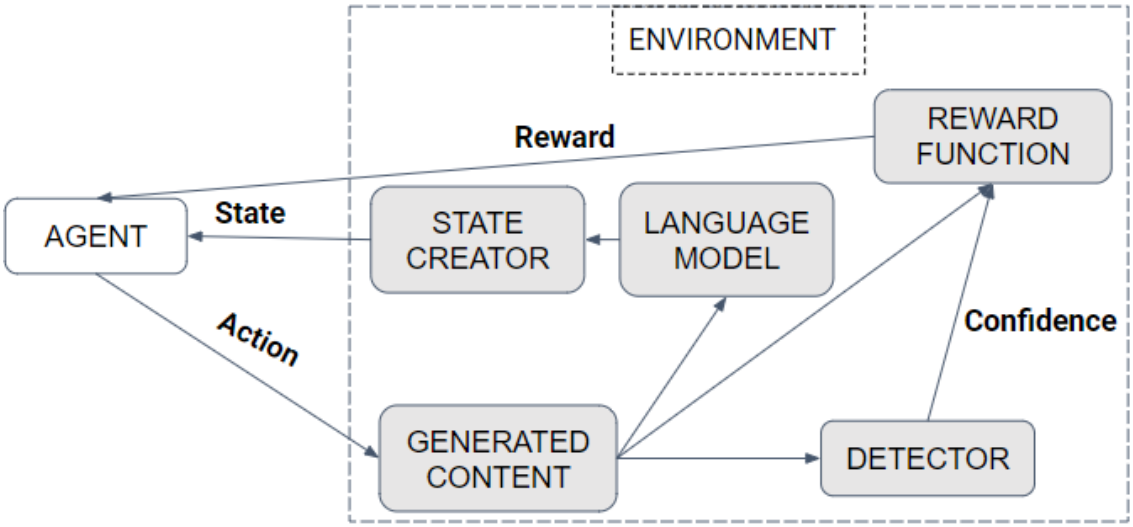

Figure 2: An illustration of the proposed reinforcement learning approach for generating adversarial samples from original samples. The agent interacts with the environment to generate a new adversarial sample for which it is rewarded using a reward function. A new state is returned from the state creator and the training process continues. Figure inspired by [5].

In order to use this approach, different types of actions, state and the reward functions have to be considered. An illustration of this approach can be found in Figure 2.

Actions For the actions that the agent can take, two possibilities will be considered. First, the approach for generating synthetic news [5] will serve as an inspiration. In most cases of natural language generation, language models are utilized to, given the rest of the sentence, generate the next word using a softmax function. This can be replaced with a RL agent, where the chosen next word is not based on a probability distribution, but on a task-specific value based on the agent's action (in our case - adversarial generation). The action would consist of picking the next word from the most probable K words given by a language model.

The second approach to actions is built directly on the findings of this study. The strategy steps as they are displayed in Table 1 will serve as inspiration in designing the actions of the agent. The agent will pick an action and based on the chosen action one corresponding *strategy step* will be used.

These two approaches will be compared in order to determine the quality and information preservation of the generated adversarial samples. This will enable us to determine if an agent that uses the identified strategy steps from this study performs better or worse compared to a different approach.

It should be noted that to create more robust models, the datasets used in fine tuning the models should contain as many different perturbations as possible. This is due to an observed phenomenon, where model results show that the lack of perturbation diversity limits their effectiveness in out-of-distribution generalization [22]. Given these findings, a future approach should be diverse in the actions it uses and not overfitted on a subset of them.

State The state should, in general, contain the information of the current generated adversarial sample and information about next possible actions. Similarly to the topic preserving synthetic news generator [5], autoencoders [23] will be trained and utilized. The autoencoder's hidden state will be used as a representation of the current state.

Reward function Rewards determine the value of a given action for the agent - the higher, the better. As such, we want to reward the agent for each successful action that brings it closer to its goal of tricking the detector model and preserving the original information. The model's confidence if a given sample is real news is the first part of the reward function. The second part of the reward consists of an ensemble of natural language evaluation metrics that will be used to determine if a given adversarial sample has kept its original information. Multiple different ensembles will be evaluated to determine the best performing one with natural language metrics [24] such as *BLEU*, *METEOR*, *ROUGE*, etc. These two parts of the reward function will be then combined to produce the final reward for an agents action.

Human input As was demonstrated in the field of Imitation learning [25], human input can serve as a guiding process for training a reinforcement learning agent. We will explore human input in training this agent in

the form of inverse reinforcement learning, where the idea is to learn the reward function of the environment based on the expert's demonstrations, and find the optimal policy using reinforcement learning. The collected samples from our study will be preprocessed to this goal and we will compare inverse reinforcement learning with traditional reinforcement learning to evaluate their performance.

In conclusion, the proposed adversarial generator will be based on reinforcement learning agent methods, most notably on the deep Q-learning method. Multiple approaches to the actions, states and possible reward functions of such an agent as have been listed above will serve as basis for an adversarial model-free information-preserving generator. Imitation learning will be also considered to determine the usefulness of human input for the adversarial generator.

7. Conclusion

In this paper a game with a purpose (GWAP) was used for collecting adversarial information-preserving samples from human players (for tricking false information detection models). In a qualitative analysis of the collected data we identified player text-modification strategies. From these, we derived properties of a future automated adversarial sample generator. To confirm their wider usability, the identified strategies were verified by an expert against more detection models (transformer and LSTM model). Based on these findings, we propose a new information-preserving adversarial sample generator which employs reinforcement learning methods for a model-free approach. The evaluation of this generator is a subject of future work.

Acknowledgments

This work was partially supported by The Ministry of Education, Science, Research and Sport of the Slovak Republic under the Contract No. 0827/2021; and by the Central European Digital Media Observatory (CEDMO), a project funded by the European Union under the Contract No. 2020-EU-IA-0267

References

[1] L. J. Y. Flores, Y. Hao, An Adversarial Benchmark for Fake News Detection Models (2022). URL: http://arxiv.org/abs/2201.00912. arXiv:2201.00912.

[2] Z. Zhou, H. Guan, M. M. Bhat, J. Hsu, Fake news detection via NLP is vulnerable to adversarial attacks, ICAART 2019 - Proceedings of the 11th International Conference on Agents and Artificial Intelligence 2 (2019) 794–800. doi:10.5220/0007566307940800.

[3] M. Soprano, K. Roitero, D. La Barbera, D. Ceolin, D. Spina, S. Mizzaro, G. Demartini, The many dimensions of truthfulness: Crowdsourcing misinformation assessments on a multidimensional scale, Information Processing and Management 58 (2021) 1–33. doi:10.1016/j.ipm.2021.102710. arXiv:2108.01222.

[4] R. Meng, Y. Tong, L. Chen, C. C. Cao, Crowdtc: Crowdsourced taxonomy construction (2015) 913–918. doi:10.1109/ICDM.2015.77.

[5] A. Mosallanezhad, K. Shu, H. Liu, Topic-Preserving Synthetic News Generation: An Adversarial Deep Reinforcement Learning Approach (2020). URL: http://arxiv.org/abs/2010.16324. arXiv:2010.16324.

[6] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, Y. Qi, TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP (2020) 119–126. doi:10.18653/v1/2020.emnlp-demos.16. arXiv:2005.05909.

[7] E. Wallace, P. Rodriguez, S. Feng, I. Yamada, J. Boyd-Graber, Trick Me If You Can: Human-in-the-Loop Generation of Adversarial Examples for Question Answering, Transactions of the Association for Computational Linguistics 7 (2019) 387–401. doi:10.1162/tacl_a_00279. arXiv:1809.02701.

[8] L. Ahn, Games with a purpose, Computer 39 (2006) 92 – 94. doi:10.1109/MC.2006.196.

[9] J. Wang, G. Dong, J. Sun, X. Wang, P. Zhang, Adversarial sample detection for deep neural network through model mutation testing, in: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019, pp. 1245–1256. doi:10.1109/ICSE.2019.00126.

[10] R. Feinman, R. R. Curtin, S. Shintre, A. B. Gardner, Detecting adversarial samples from artifacts, 2017. URL: https://arxiv.org/abs/1703.00410. doi:10.48550/ARXIV.1703.00410.

[11] C. Wong, DANCin SEQ2SEQ: Fooling Text Classifiers with Adversarial Text Example Generation (2017). URL: http://arxiv.org/abs/1712.05419. arXiv:1712.05419.

[12] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, D. Jurafsky, Adversarial learning for neural dialogue generation, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 2157–2169.

[13] J. Xu, L. Zhao, H. Yan, Q. Zeng, Y. Liang, X. Sun, LexicalAT: Lexical-based adversarial reinforcement training for robust sentiment classification, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th In-

ternational Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 5518–5527. doi:10.18653/v1/D19-1554.

[14] W. Zou, S. Huang, J. Xie, X. Dai, J. Chen, A reinforced generation of adversarial examples for neural machine translation, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 3486–3497.

[15] P. Vijayaraghavan, D. Roy, Generating black-box adversarial examples for text classifiers using a deep reinforced model, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2019, pp. 711–726.

[16] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, D. Kiela, Adversarial nli: A new benchmark for natural language understanding, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 4885–4901.

[17] M. Bartolo, A. Roberts, J. Welbl, S. Riedel, P. Stenetorp, Beat the ai: Investigating adversarial human annotation for reading comprehension, Transactions of the Association for Computational Linguistics 8 (2020) 662–678.

[18] P. Patwa, S. Sharma, S. Pykl, V. Guptha, G. Kumari, M. Akhtar, A. Ekbal, A. Das, T. Chakraborty, Fighting an infodemic: Covid-19 fake news dataset, Commun. Comput. Info. Sci. (2021) 21–29.

[19] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. arXiv:1910.01108.

[20] E. Wallace, S. Feng, J. Boyd-Graber, Interpreting neural networks with nearest neighbors, EMNLP 2018 (2018) 136.

[21] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A survey, Journal of Artificial Intelligence Research 4 (1996) 237–285.

[22] N. Joshi, H. He, An investigation of the (in) effectiveness of counterfactually augmented data, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2022, pp. 3668–3681.

[23] A. S. Chandar, S. Lauly, H. Larochelle, M. M. Khapra, B. Ravindran, V. Raykar, A. Saha, An autoencoder approach to learning bilingual word representations, in: Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2, 2014, pp. 1853–1861.

[24] A. B. Sai, A. K. Mohankumar, M. M. Khapra, A survey of evaluation metrics used for nlg systems, ACM Computing Surveys (CSUR) 55 (2022) 1–39.

[25] A. Hussein, M. M. Gaber, E. Elyan, C. Jayne, Imitation learning: A survey of learning methods, ACM Computing Surveys (CSUR) 50 (2017) 1–35.