algorithms

MDPI

Article

Fact-Checking Reasoning System for Fake Review Detection Using Answer Set Programming

Nour Jnoub *, Admir Brankovic and Wolfgang Klas

Faculty of Computer Science, University of Vienna, 1090 Vienna, Austria; a01063191@unet.univie.ac.at (A.B.); wolfgang.klas@univie.ac.at (W.K.)
* Correspondence: nour.jnoub@univie.ac.at

Abstract: A rising number of people use online reviews to choose if they want to use or buy a service or product. Therefore, approaches for identifying fake reviews are in high request. This paper proposes a hybrid rule-based fact-checking framework based on Answer Set Programming (ASP) and natural language processing. The paper incorporates the behavioral patterns of reviewers combined with the qualitative and quantitative properties/features extracted from the content of their reviews. As a case study, we evaluated the framework using a movie review dataset, consisting of user accounts with their associated reviews, including the review title, content, and the star rating of the movie, to identify reviews that are not trustworthy and labeled them accordingly in the output. This output is then used in the front end of a movie review platform to tag reviews as fake and show their sentiment. The evaluation of the proposed approach showed promising results and high flexibility.

Keywords: online fake review detection; answer set programming; fact checking

Citation: Jnoub, N.; Brankovic, A.; Klas, W. Fact-Checking Reasoning System for Fake Review Detection Using Answer Set Programming. *Algorithms* **2021**, *14*, 190. https://doi.org/10.3390/a14070190

Academic Editors: Frank Werner and Giovanni Amendola

Received: 31 May 2021
Accepted: 22 June 2021
Published: 24 June 2021

1. Introduction

Social media and e-commerce platforms have become a fundamental element of today's society [1]; as a result, the data on the Web is hugely growing, but the quality of this data requires more investigation [2]. Social media platforms have become where people share the urge to stay connected and express themselves by discussing the news, giving opinions about politics, watching movies or products, but, unluckily, this online data can be fast and quickly disseminated [3], and it is not so hard to manipulate with it. The fake news and misleading information have become apparent today, especially in moments of crisis, such as what we face today, the COVID-19 pandemic [4]. Different studies have proven that there is a need to flatten the curve of the rumors and misinformation about the virus with a view to flatten the curve of COVID-19 [5]. Besides, e-commerce websites are also an essential instance where a large number of fake reviews and fake profiles exist due to the rising number of new products and the intense competition between companies [6].

Nevertheless, online reviews can be fraudulent and are yet necessary for both consumers and firms. For the consumers, it is necessary to check them to decide for the quality of the products and whether to buy it or not. In contrast, firms need online reviews in order to improve their products and boost the sales [7].

Thus, developing fact-checking tools that can detect fake reviews or fake users will help improve the clarity and quality of online data; otherwise, the Web will be a sea of abnormal distribution of rumors and false information.

Obviously, this is not an easy task: researchers in this domain invested a lot of effort, but it still needs to be improved. To understand the problem well, we should first recognize the characteristics of such data and the behavior patterns of the fake users.

In Reference [8,9], sockpuppets are individuals with pretended profiles that incorporate wrong information and do not show their real identities; they have suspicious activities, like spamming or flooding, to mislead other users and hide the facts. Those

activities can be exposed by their behavioral patterns and the other accounts who cooperate with them; for instance, by checking the IP addresses, it is possible to obtain information indicating spamming behaviors, mainly when the same user with the same IP address uses multiple fake accounts. Accordingly, identifying those accounts is highly important and removing them requires well-designed systems, which can identify such strange behaviors and malicious patterns.

This work is an extensive study of this kind of user's behavior that writes fake reviews and has malicious patterns to detect it. We define a knowledge base (KB) and a reasoning rules-based approach, using Answer Set Programming (ASP), where the following study [10] showcases the established usage of ASP in industry fields.

Our approach works as follows: the input is the movie reviews. Then, each review is checked using natural language processing to determine the sentiment to distinguish positive and negative criticism; next, a Levenshtein-distance algorithm [2] identifies deviations between user review data. Afterwards, the results of these algorithms and user data, such as review rating and IP addresses data, are grounded and put in all in an ASP KB with predefined rules that fact check them. This way, the framework can identify reviews that are not trustworthy and label them accordingly in the output. The evaluation showed that this technique is able to detect fake reviews and works well with high flexibility since more algorithms can later be considered as extended rules can be for better performance.

The remainder of the paper is organized as follows: Sections 2 and 3 gives an overview of related work and highlight the importance of reasoning systems for fake review detection. Section 4 discusses the basic patterns of user behavior in the context of detecting fake reviews. Section 5.2 explains the original dataset used for this work. Section 5 presents our approach using an ASP technique to realize a fake review detection system. Sections 6 and 7 demonstrate the output results and the evaluation of it. Section 8 covers a discussion of the presented approach, and Section 9 concludes the paper.

2. Related Works

The fake reviews detection problem was introduced first by Jindal [11] and Liu [12] concerning product reviews in 2008. The problem has been tackled using several approaches, like analyzing the review content or the user behavioral characteristics, or applying sentiment analysis techniques, to classify the reviews as fake or genuine.

Many researchers tried to create state-of-the-art techniques for detecting reviews as spam or genuine reviews. Various detection levels should be taken into consideration as stated in Reference [13], they are mainly the following: (a) review content-based detection, (b) deviations among rating-based detection, and (c) review content along with user behavior-based detection. Supervised models use the first two categories, but it may not be easy to obtain precise results as training the models may require a big dataset to get accurate results. Thus, the authors recommended considering the third category (c) to enhance the overall performance.

Lately, intensive research has been done for developing spamming detection systems using machine learning techniques. In Reference [14], two different approaches for fake review recognition are presented, where supervised learning techniques have been used for extracting and classifying new semantic features from real Yelp review dataset. The paper proposes a set of behavioral features and proves that, by using such features, it helps to achieve better accuracy than using n-grams.

Support Vector Machine (SVM) has been used in Reference [14] for classification and the authors reported that they achieved high accuracy when including the following behavioral features that are: the length of the review, time intervals of each review, and the ratio of each positive/negative reviews.

In Reference [15], a feature framework for fake review detection has been introduced based on a classification approach. The work addresses the problem by scraping information from real Yelp dataset (https://www.kaggle.com/yelp-dataset/yelp-dataset (accessed on 31 May 2021)) in order to construct a new dataset concerning the customer electronics

domain. The framework did use two types of features for extracting features: review-centric features that are related to review itself as a text, and user-centric features, which concern the behavioral patterns of the reviewers, like personal, social, and review activity.

The approach presented in Reference [16] showed that non-verbal behavioral features, such as the number of likes/dislikes, average posting rate, review updates, etc., can be effective to detect fake reviews as much as verbal features, such as the review length or review content. The paper recommended non-machine learning techniques, such as graph-based or pattern matching methods, where these different techniques examine the relationship between the reviewers, reviews, and content similarities.

Furthermore, the work in Reference [17] demonstrates that using the reviewers behavioral features boosted the accuracy of approximately 20 percent in comparison to n-grams, but they also mentioned that more information could be used in order to improve the results, like IP addresses, user logs, or duration of sessions.

Other approaches, e.g., Reference [18], proposed a fake review recognition system by capturing doubtful time intervals of reviews. Both refs. [17,18] agree that using the private information, like IP addresses and MAC addresses, can boost the fake detection system performance. Both papers agree on the problem that there are no quality standards for datasets that can be used to achieve 100 percent of accuracy to tell if a review is fake or not.

Moreover, the approach presented in Reference [19] demonstrates diverse features that help to spot the fake reviews. Those features are mainly used to spot the review content similarities including the reviewers, their reviews, and the ability of repeating reviews. In addition, the items that have been reviewed, as well as the frequency of each review, are taken into consideration.

Studies in Reference [20] showed the direct impact of online reviews on the box office revenue of a given movie, implying that people are using these online services to sometimes decide if they are going to watch a movie or not.

Based on the previous papers, and as mentioned in Reference [21], one of the significant challenges when using machine learning approaches arises from the imbalance in the state of two classes, false and factual information from the previously mentioned approaches. Obtaining labels for fraudulent information is also a challenging task. In many cases, these are obtained manually by experts, trained volunteers, or Amazon Mechanical Turk workers. The process requires considerable manual effort, and the evaluators are possibly unable to classify all misinformation that they face. In contrast to those approaches, rule-based techniques are recognized as a white box, which provides traceability and transparency for significant decisions that require a higher degree of explanation than usually produced by machine learning approaches. Furthermore, the decision of whether to go for a rule-based system or machine learning system depends on the problem, and it is mainly a trade-off between effectiveness, training costs, and understanding.

Furthermore, the following papers are all describing and related to how ASP can be used. Authors in Reference [10] gave a good explanation and introduction on ASP use cases. It is more interesting to take a look at if and how ASP can be used to create recognition systems. According to the authors, ASP can solve classification problems and detect inconsistency of information. They have given an example of using ASP for an e-tourism portal. It can be similar to the proposed approach as a recognition system because both systems are knowledge-dependent and have particular rules/algorithms to follow. Furthermore, they hinted that ASP could be used to detect fake news and inconsistency in data.

However, ref. [22] emphasizes how straightforward the implementation of ASP can be and how easy difficult-looking problems can be solved. In this paper, the authors demonstrate a simple process of building artificial intelligence system for games, which is usually not an easy task for imperative programming. They reported that ASP is excellent in dealing with knowledge incomplete and intensive applications.

As mentioned in the literature, fake review detection systems would work more efficiently if the sentimental analysis is used, but natural language processing (NLP) is not an

easy task at all for the machines. Reference [23] describes the usage of ASP function elements together with some external NLP modules, which have textual entailment functions. Furthermore, the authors mentioned that similar approaches are essential because they provide the machines with the possibility to solve reasoning problems with background knowledge written in natural language.

In addition, Refs. [24,25] are both focusing on discovering inconsistencies by using ASP. Although they are different papers and have different application domains, it is possible to see similarities in using the approach and motivation behind choosing ASP. Both papers search for inconsistencies in a specific type of data by determining minimal representations of conflicts. Both papers claim that ASP provides a straightforward method to model and represent the problem.

3. Why Reasoning System for Fake Review Detection Is Important?

As mentioned in Reference [26], 90 percent of customers who remembered reading online reviews claimed that positive online reviews impacted their buying considerations. At the same time, 86 percent of them said negative online reviews affected their buying decisions. Additionally, around 40 percent of all online reviews on Amazon are fake (https://www.brightlocal.com/research/local-consumer-review-survey/ ((accessed on 31 May 2021). Thus, the need to have a reasoning system that is able to detect such reviews is in high demand specially when taking the following points in consideration:

1. Fake reviews deceive customers of being fully aware of all the information about the product they are buying.
2. Some fake reviews are written in order to distract customers from positive rating of the product.
3. A lot of fake reviews are written to promote or even to demote the product.
4. Pure supervised learning approaches could not be a proper choice due to the lack of training data that considers the behavioral patterns of users/reviewers. The provided approach uses supervised learning for sentiment analysis of the review, which is considered in one rule in the KB.

4. On the Characteristics of Fake Reviewers and Reviews

Behind every fake review, there is a user that normally has a well-defined malicious behavior. As described in Reference [14], certain patterns, such as posting only positive or negative reviews, or posting reviews in a specific time interval, for example, every day or every week at the same period of time, can be detected as a spamming behavior. Another interesting pattern for a fake review can be the content itself. For example, there are many redundant reviews with only minor changes or spam reviews with a bad quality texting and few counts of words [27].

Another well-described pattern of malicious behavior, described in both References [27,28], is the control of different accounts by a single spammer, who uses them for spamming and domination. These accounts are also known as bot accounts or sockpuppets and can be recognized by following their actions in connection with other similar users. Most likely, the grouping that will appear is going to be bot users. Another aspect for fake review recognition could be a single individual user coupling in a group, which describes the behavior of multiple users acting similarly relatively at the same time, which is often done by bot accounts [27]. Furthermore, accounts with some special "sequential" user naming are often associated with bot users. Another way to detect fake reviews can be achieved by checking the dislike counts. Many e-commerce websites offer some functionality that allows people to vote if a review was helpful or not. Sometimes the community is doing the job for us by disliking the suspicious reviews. This makes the dislike count be represented as a feature that needs to be taken into consideration.

The work in Reference [9] discusses the anonymity factor that may need further consideration, as well. For example, on Amazon, users can post a review under a nickname

or as an anonymous user. Anonymous users post many fake reviews because the user may want to avoid getting tracked. In our proposed system, the requirement to post a new review is that the user initially registers with the help of the registration step; the anonymous review feature is not feasible.

Another significant factor in the recognition process is the review feature similarities of a new or updated review with all other reviews in the system. It is possible to compute these similarities by using dynamic algorithmic approaches as discussed in Reference [2]. Given the deviation values results of the analyzed reviews, the three reviews with the highest similarity scores are considered, and the number of high and low deviating reviews is used in the ASP knowledge base (see Section 5.5).

5. Approach

ASP is a well-known technique in the domain of Knowledge Representation and Reasoning (KRR) [29]. It belongs to the group of so-called constraint programming languages. In ASP, the specification of the problem is given, and the specification model delivers the solution. It is also a form of declarative programming, which is mainly used to solve non-deterministic search problems. As described in Reference [24,25], it got prevalent *"because of its appealing combination of a rich yet simple modeling language with high-performance solving potentials"*. In ASP, search problems are reduced to compute solid models and answer sets. It utilizes solvers that are reasoners, which outcome stable model generation [24].

Figure 1 presents the ASP solving process. The aim is to deliver one or several solutions to the given problem. The primary step is to model the problem into a logic program with rules, first-order predicates, and variables. Then, the grounder will reduce the variables by substituting them with ground terms in the language. As a consequence, a propositional program will only contain propositional atoms. The program is then taken by the solver, which will calculate all the stable models and interpret the solution [30]. Therefore, it is essential to understand the data and the problem itself to start building our KB.

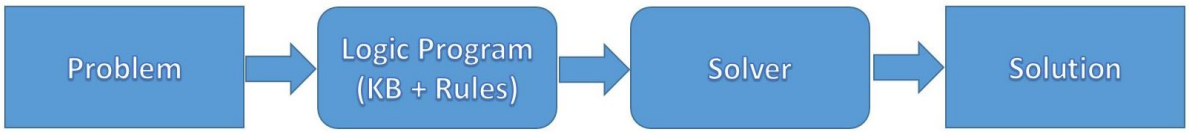

Figure 1. Answer Set Programming solving process.

This will allow for creating a formal model of the problem and the input data by means of a KB, constraints, and rules that have to be satisfied (see Sections 5.5 and 5.5.2). In the upcoming illustrating example, we show the approach considering only three movies just for simplicity. This simplification does not cause any limitation of the approach.

While developing the KB, with all the constraints and facts, we have used the online tool Potassco (The Potsdam Answer Set Solving Collection, https://potassco.org/clingo/run/ (accessed on 31 May 2021)) to test the resulting labels and analyze the running time of the solver. Solving a KB with four reviews, including their facts, such as IP address, sentiment, star rating, and cosine similarity, took a total of 0.013 s to solve, showcasing the efficiency of ASP solvers.

5.1. ASP Fundamentals

As previously mentioned, the KB consists of various facts and rules which are used to generate an answer set, which includes the possible solutions of the given problem. ASP programs are based on the AnsProlog (also called A-Prolog [31]) programming language and use the predefined rules and facts, instead of a classical algorithmic approach, to find a solution to a given problem. The following shows examples of defining a rule and constraint.

$$a_0 \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m. \quad (1)$$

The example rule (1), shows a list of atoms in first order logic $b_1, \ldots, b_n, c_1, \ldots, c_m$, which is called the body of the rule, whereas a_0 is called the head of the rule. It is also important to note that rules without a body are called facts [32].

$$\leftarrow b_1, \neg c_1. \tag{2}$$

The example constraint (2), shows a list of atoms in first order logic $b_1, \neg c_1$, which is different from a rule in such a way that the head is empty. A constraint is used to remove some elements from the answer set, as in the given example constraint, and the answer set will not include b_1 if c_1 is not generated.

5.2. Dataset

To achieve accurate results that mimic real-world scenarios, we implemented a website for evaluating movie reviews. It allowed us to create a more extensive dataset that can help us explain the reviewers' behavior. We wanted to have complete control over the dataset so that we can get all the required attributes. In order to achieve this, the website uses a local database instance for storing all related data, including user data, review data, and movie-related data. The dataset can be used later to inspect the best or worst-rated movies, movies with many fake reviews, and similar measurements.

5.3. Analysis of Structured and Unstructured Data

The approaches of analyzing fake data on the web can be categorized in two segments. The first one is based on structured data, where the content can be directly analyzed as such. As structured data is well organized, depending on the type of the data, it is much easier to generate a well defined KB, which will yield clear answer sets, given the rules and constraints defined in the KB.

On the other side, with unstructured data, the approach includes different steps of organizing the unstructured raw data, to result in processed data that can be used in the KB. These steps include behavioral based analysis steps, that can be used to analyze predefined classifications of behaviors or to generate new behavioral patterns from the given data. Another way of interpreting the data is using natural language processing steps, to analyze, i.e., the sentiment of the data. The main idea behind these approaches is to transform the unstructured data into well structured data, which can be used as a basis for the KB generation. These steps are crucial to understand, implement and evaluate in the early steps of generating the KB, meaning that the rules and constraints within the KB need to comply with the processed unstructured data, to establish a well defined answer set.

The final step of the process is to evaluate the inputs to the KB, and generate the answer set that recognizes the given data as, in our case, fake or genuine. These classifications can be extended and adapted to other requirements, giving the opportunity to evaluate different data inputs for other, possibly more domain-specific, datasets, based on an existing KB that can be easily extended to meet the new requirements. This is one of the aspects that illustrate the advantage of ASP approaches for problem solving tasks.

Figure 2 shows the given steps of the analysis of structured and unstructured data, regarding the generation of a KB.

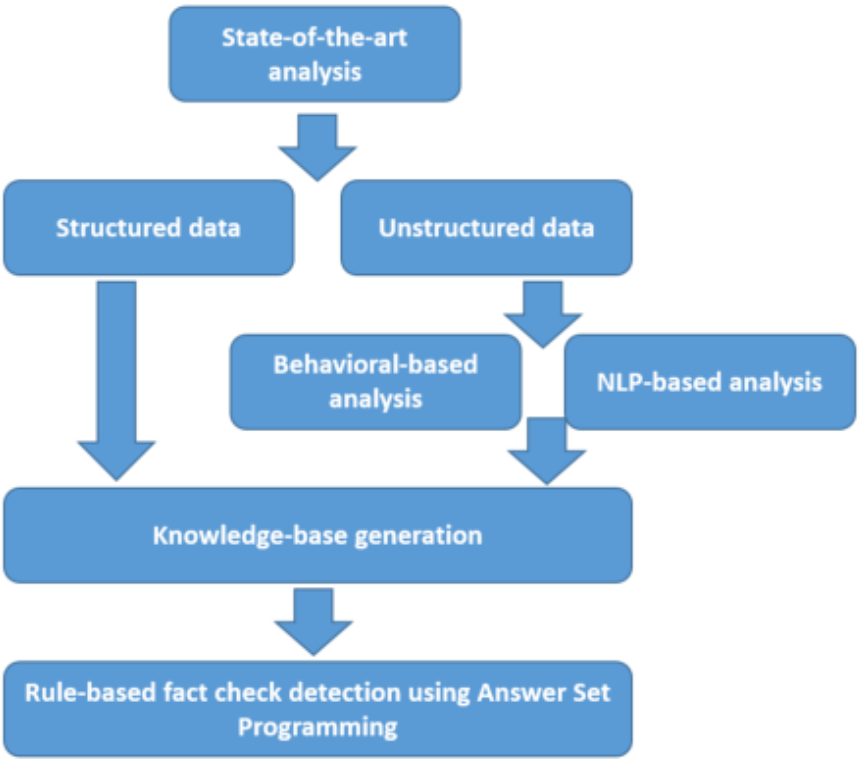

Figure 2. Process for structured and unstructured data.

Levenshtein/Cosine Similarity

Presents a technique that helps in pinpointing contradicting and deviating pieces of information in structured data sources. It incorporates two algorithms, Levenshtein Distance (LD) and Cosine Similarity (CS). LD measures the similarity of two input strings [33]. It is also known as "edit distance", which helps in many applications, like text analyzing and correcting errors [34]. In this case, the distance is the number of substitutions, deletions, and insertions required to create the target string from the source string. A high distance would indicate that the two inputs are notable and deviate from each other. Moreover, a low distance indicates higher similarity and that the strings do not deviate as much. According to Reference [2], obtaining these results involves three succeeding steps: The first step is to edit input data to compare them. This step includes extracting unnecessary characters from strings with regular expressions and parsing dates to a standard format. The second step is to calculate pairwise deviations between the reference object and each item in a dataset, where every reference object or a single object in the compared dataset consists of a movie title, a movie date, and the movie cast names. After normalizing the computed values, the distance matrix is then calculated. The next step of the algorithm is to specify further the Levenshtein distance analysis matrix, which was created in the second part of the algorithm and comprises ordering the results in descending or ascending order, which is done using what so-called sensitivity vectors. It allows investigating which sort of deviation to examine out of all movie features. For example, a sensitivity vector of (1, 1, 1) would look for significantly deviating data and creates a list as output, which shows movies that have a high deviation across all categories. In contrast, a vector (0, 0, 0), on the other hand, does the opposite: It orders the created list by non or most minor deviating data, which results in a list that holds low deviating movies in the front. The chosen sensitivity vector contributes to a cosine similarity function with the deviation matrix created in step 2. The output is a list that orders the deviating data by the preferences earlier defined in the sensitivity vector and later used to analyze the algorithms' calculations by initially ordering the feature list received by the function.

This algorithm is a robust tool in identifying deviating data in structured data elements. The most significant part of this algorithm is that it does not require a training phase, like machine learning approaches. Another advantage is that there is no need to use sample data to determine a specific models' parameters [2]. Consolidating this algorithm into a given project is very straightforward because, as discussed earlier, it does not need sample data or training phase.

5.4. Demo Application Scenario

In our demo application, we assume that there are movies, each of them having assigned some reviews written by some users. Each review has assigned a number of (dis)likes reflecting the opinion of other users, number of stars reflecting the author opinion of the movie, a sentiment score of the review (ranging from 1 as strongly negative to 5 as strongly positive), a timestamp encoding the time the review was posted, and an IP address encoding the users' machine as the source of the review.

All information about reviews and authors, such as (dis)likes, star ratings, sentiment scores, timestamps, and IP addresses, are encoded in a KB which we briefly discuss in Section 5.5.

In our application, it is assumed that reviews may be considered *fake*, *possiblyFake*, *possiblyGenuine*, *genuine*, or *contradicted*. We include many rules that will lead us to detect the correct classification of the reviews, helping the users to decide whether a review on a certain movie can be trusted or not.

All these rules determine the results by labeling a review as *fake*, *possiblyFake*, *possiblyGenuine*, *genuine*, or *contradicted*, determining which reviews where posted from the same IP address, and labeling reviews as candidates for spam in case they have been written within a short time interval and stemming from the same IP address (e.g., a pattern often observed in the case of sockpuppetry [8]), determining the polarity scores of the reviews between the review star rating and the review sentiment analysis score, finally analyzing the review similarity with all other reviews in the system. In Section 5.5.2, we illustrate some sample rules.

5.5. Knowledge Base

The system uses two knowledge bases, one for the review classification and another for the author classification.

5.5.1. Review Knowledge Base

In the review KB (see Listing 1), all constants should be declared first. The only constant value in the KB is 30 s refers to the time interval between the posting time of two reviews.

Next, we declare some timestamps t_i and IP addresses ip_i, followed by the declaration of movies m_i and reviews r_i. Subsequently, we declare which reviews are assigned to a movie (see *hasRev*).

Review features are being set, such as the review stars (see *stars*), sentiment score (see *sentScore*), and review Levenshtein/Cosine similarity scores (see *revLCS*).

Then, the timestamps and the source IP addresses for all reviews are modeled (see *timestamp_of_Review* and *hasIp*). Finally, we declare the labels used for the classification of reviews.

Listing 1. Abstracted samples of facts from the review KB.

```
#const time_interval = 30.

#const t1 = 2. t2 = 4. t3 = 6. t4 = 7. t5 = 8. t6 = 13. ...
#const ip1 = 858513516. ip2 = 199188126120. ip3 = 960233229. ...

movie(m1;m2;m3; ...;mn).
review(r1;r2;r3;...;rn).
hasRev(m1,(r1;r2;r3)). hasRev(m2,(r4;r5;r6)). ...

stars(r1, 5). stars(r2, 4). stars(r3, 1). stars(r4, 1). ...
sentScore(r1, 4). sentScore(r2, 4). sentScore(r3, 5). sentScore(r4, 2). ...
revLCS(r1, 0, 3). revLCS(r2, 0, 3). revLCS(r3, 3, 0). revLCS(r4, 2, 1). ...

timestamp_of_Review(r1, t1). timestamp_of_Review(r2, t2).
timestamp_of_Review(r3 ,t3). timestamp_of_Review(r4, t4). ...

hasIp(r1, 1010). hasIp(r2, 2020). hasIp(r3, 1010). hasIp(r4, 2220). ...

reviewLabel(fakeReview). reviewLabel(possiblyFakeReview).
reviewLabel(possiblyGenuineReview). reviewLabel(genuineReview).
reviewLabel(contradictedReview).
```

5.5.2. Review Constraints and Rules

Based on the facts from the KB, constraints and rules satisfying conditions on the facts are declared such that they formally encode the problem that needs to be solved. In our scenario, the problem we want to solve can be stated like this: *Given some facts about reviews on movies, we want to detect fake and genuine reviews and identify candidates for spam reviews*.

In our case, a set of illustrating constraints and rules (see Listing 2) has been declared, which state the following:

1. Each review can be assigned as highly, moderately, or normal regarding the review polarity. A review will be of *highPolarityDiff* if the difference between the sentiment score and the star rating of the review is greater or equal to 3.
2. Analogously to the previous rule, *moderatePolarityDiff* checks if the difference between the sentiment score and the star rating of the review is greater or equal to 1 and is not believed to be of *highPolarityDiff*.
3. Analogously to the previous two rules, *normalPolarityDiff* checks if the difference between the sentiment score and the star rating of the review is not believed to be of *highPolarityDiff* and *moderatePolarityDiff*.
4. A review is considered as a unique review (*spamByLCS*) if the Levenshtein/Cosine similarity scores are higher in the number of lower deviating reviews than high deviating reviews.
5. The given rule states that, if any two different reviews of a certain movie have an identical source, IP address, and both reviews were written within a short time interval (e.g., 30 s, i.e., $|z_2 - z_1| < 30$ s), it has to be derived and concluded that both reviews are spam candidates.
6. The given rule checks if the review can be concluded to be a genuine review if the review is not a spam candidate by the *spamByCS* rule, not spam by timestamp, and neither high nor moderate polar (see *genuineReview*).
7. The given rule checks if the review can be concluded to be a possibly genuine review if all previous rules match. However, the review is considered to be of moderate polarity (see *possiblyGenuineReview*).
8. The given rule checks if the review can be concluded to be a possibly fake review if the review is considered to be spam by the Levenshtein Cosine similarity measure, spam by timestamp, and a moderately polar review (see *possiblyFakeReview*).

9. The given rule checks if the review can be concluded to be a fake review if the review is considered to be spam by the Levenshtein Cosine similarity measure, spam by timestamp, and a highly polar review. (see *fakeReview*).
10. If the review is not believed to be of any of the previous classifications, it is considered to be a contradicted review (see *contradictedReview*).

Listing 2. Sample review constraints and rules.

```
#1:
highPolarityDiff(X) :- review(X), stars(X, S1), sentScore(X, S2),
                       |S1 - S2| >= 3.

#2:
moderatePolarityDiff(X) :- review(X), stars(X, S1), sentScore(X, S2),
                       |S1 - S2| > 1, not highPolarityDiff(X).

#3:
normalPolarity(X) :- review(X), not highPolarityDiff(X), not moderatePolarityDiff(X).

#4:
spamByLCS(X) :- review(X), revCS(X, Y, K), Y > K.

#5:
spamByTimestamp(X;Y) :- hasRev(M1,X), hasRev(M2,Y), M1=M2, X!=Y,
                        hasIp(X,I1), hasIp(Y,I2), I1=I2, review(X),
                        timestamp_of_Review(X,Z1), review(Y),
                        timestamp_of_Review(Y,Z2), |Z2-Z1|<time_interval.

#6:
genuineRev(X) :- review(X),  not spamByLCS(X),
                 not spamByTimestamp(X),
                 not highPolarityDiff(X), not moderatePolarityDiff(X).

#7:
possiblyGenuineRev(X) :- review(X), not spamByCS(X), not spamByTimestamp(X),
                         not highPolarityDiff(X).

#8:
possiblyFakeRev(X) :- review(X), spamByCS(X), spamByTimestamp(X),
                      moderatePolarityDiff(X).

#9:
fakeRev(X) :- review(X), spamByLCS(X), spamByTimestamp(X),
              highPolarityDiff(X).

#10:
contradicatedRev(X) :- review(X), not fakeRev(X), not possiblyFakeRev(X),
                       not genuineRev(X), not possiblyGenuineRev(X).
```

5.5.3. Author Knowledge Base

In the author KB (see Listing 3), the main focus lies on the number of reviews posted by the author for a given movie and how these reviews are rated by other users of the system.

The facts include the author's object, movies and reviews, the number of likes and dislikes of a review, and what review is assigned to which movie.

Listing 3. Abstracted samples of facts from the author KB.

```
author(a).
movie(m1). movie(m2). movie(m3). movie(m4).

review(r1). review(r2). review(r3). review(r4). review(r5). review(r6).
reviewLikes(r1, 10). reviewLikes(r2, 0). reviewLikes(r3, 16).
reviewLikes(r4, 79). reviewLikes(r5, 2). reviewLikes(r6, 30).

reviewDislikes(r1, 2). reviewDislikes(r2, 15). reviewDislikes(r3, 3).
reviewDislikes(r4, 5). reviewDislikes(r5, 2). reviewDislikes(r6, 7).

forMovie(r1, m1). forMovie(r2, m1). forMovie(r3, m1).
forMovie(r4, m2). forMovie(r5, m3). forMovie(r6, m4).
```

5.5.4. Author Constraints and Rules

Based on the facts from the KB, constraints and rules satisfying conditions on the facts are declared such that they formally encode the problem that needs to be solved. In our scenario, the problem we want to solve can be declared like this: *Given some facts about reviews on movies, we want to detect fake and genuine reviews and identify candidates for spam reviews*.

In our case, a set of illustrating constraints and rules (see Listing 4) has been declared, which state the following:

1. Using the *sum* aggregate, the number of likes overall reviews is being calculated.
2. Using the *sum* aggregate, the number of dislikes overall reviews are being calculated.
3. The *highDiff* rule checks how many reviews have a higher number of likes compared to the number of dislikes, and returns the positive number of likes of a review. For example, if review *r1* has 15 likes and 5 dislikes, the rule will return *highDiff(r1,10)* as an answer set.
4. To get the best-rated review, the rule *bestRev*, uses the *max* aggregate and the *highDiff* rule, calculates which review has the highest number of likes compared to the number of dislikes, over all reviews.
5. Analogously to the *highDiff* rule *lowDiff*, this rule calculates the reviews that have a higher dislike count than like count. For example, if review *r1* has 3 likes and 20 dislikes, the answer set of this rule will be *lowDiff(r1,−17)*.
6. Analogously to the *bestRev* rule, *worstRev* uses the *min* aggregate and the *lowDiff* rule to calculate which review has the highest number of dislikes compared to the number of likes, over all reviews.
7. The *highCount* rule gets the number of liked reviews.
8. The *lowCount* rule gets the number of disliked reviews.
9. Using the *liked* rule, it returns if the author has more liked than disliked reviews.
10. Using the *neutral* rule, it returns if the author has the same number of liked and disliked reviews.
11. Using the *disliked* rule, the author is classified as disliked if the author is not believed to be liked or neutral.

5.6. Computed Results

The results computed by an ASP solver always satisfy all the constraints and rules. The computed results are considered as an optimal solution with the highest accuracy and precision compared to other work we previously mentioned in Section 2, where most of them apply machine learning techniques and natural language processing for fake reviews detection.

The set of constraints and rules can be easily extended to cover much more complicated scenarios built upon a variety of behavior patterns, and not just a few characteristics the spammers or the spams may have, which are well understood and studied in the literature [35].

Listing 4. Sample author constraints and rules.

```
#1:
likes(N) :- N = #sum{S, R : reviewLikes(R,S)}.

#2:
dislikes(N) :- N = #sum{S, R : reviewDislikes(R,S)}.

#3:
highDiff(R, X) :- reviewLikes(R, L), reviewDislikes(R1, D), R = R1, L > D, X == L - D.

#4:
bestRev(R, X) :- review(R), highDiff(R, X), #max {XX,1:highDiff(Z, XX)} = X.

#5:
lowDiff(R, X) :- reviewLikes(R, L), reviewDislikes(R1, D), R = R1, L < D, X == L - D.

#6:
worstRev(R, X) :- review(R), lowDiff(R, X), #min {XX,1:lowDiff(Z, XX)} = X.

#7:
highCount(S) :- S = #count{ R : highDiff(R,X)}.

#8:
lowCount(S) :- S = #count{ R : lowDiff(R,X)}.

#9:
liked(X) :- highCount(S), lowCount(S2), S > S2, person(X).

#10:
neutral(X) :- person(X), highCount(S), lowCount(S2), S == S2.

#11:
disliked(X) :- person(X), not liked(X), not neutral(X).
```

6. The Demo Application

The major components of the proposed system have been implemented as a website. The system allows users to register for an account to be able to write reviews for movies obtained from the IMDB API (https://developer.imdb.com/ (accessed on 31 May 2021)). When the user is logged in, it is possible to add reviews, such as review title, review content, and review star rating, insert reviews, delete their reviews, update their reviews, and downvote other reviews. All the inserted data will be investigated by the proposed rule-based reasoning approach to detect the review classification.

We strongly believe that such a tool is one of the first rule-based website systems that offer the possibility to collect datasets, annotate reviews, and use ASP as a powerful reasoning paradigm.

7. Data Analysis and Evaluation

Regarding the sentiment analysis of the review content, a trained logistic regression model is used as an adapted version of the Stanford Treebank dataset(https://nlp.stanford.edu/sentiment/treebank.html (accessed on 31 May 2021)). The adapted version uses the review content from the dataset. The dataset is preprocessed, and the sentiment score in the training and test datasets vary from 1 to 5, strongly negative to strongly positive, respectively. The trained sentiment analysis model, which was used as a tool in the review analysis system, has accomplished an accuracy of 41.22%, which is closely related to the accuracy of 40.7%, Ref. [36] and the authors of the used dataset have achieved that. In the analysis of the model, many of the neutral sentiment reviews were assigned either a weakly positive or weakly negative sentiment, as the trained model was kept very simplistic for this approach. In the future, the aim is to use a more sophisticated model for sentiment analysis to enhance the accuracy of the whole system, such as in Reference [3], where Convolutional Neural Network (CNN) and Shallow Neural Network (SNN) reached 81.0% and 87.0% accuracies, respectively, using the IMDB dataset. For this paper, a simple trained model was sufficient, as the main idea was to show how ASP can be used in combination with different tools to classify movie reviews considering the review content and its sentiment.

The following figures show examples of classified reviews on the website. They show the functionality of how a review is classified based on the review itself and the author, additionally including any reviews that have been flagged as highly similar with the current review by the Levenshtein/Cosine Similarity value. Figure 3 shows a genuine review with a neutral sentiment, where the author has no ratings yet. In contrast, Figure 4 shows a weakly positive genuine review where the author has reviews that other users have liked.

Figure 5 shows a spam review with a strongly negative sentiment, that has been posted from the same IP address. Figure 6 shows a contradicted review, as it has a positive sentiment but a low star rating; additionally, it is very similar to another review in the system, which has been flagged accordingly. It is also important to note that the review shown in Figure 6 has a label showing that the author posts multiple reviews for a single movie, which is considered to be negative.

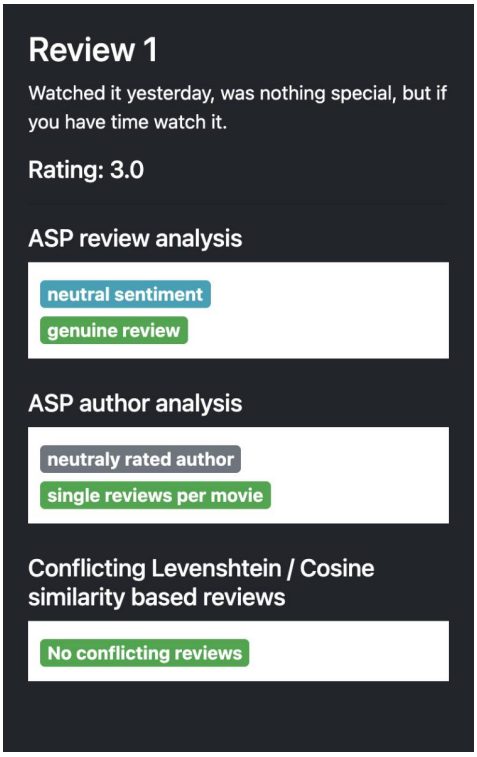

Figure 3. Neutral review example.

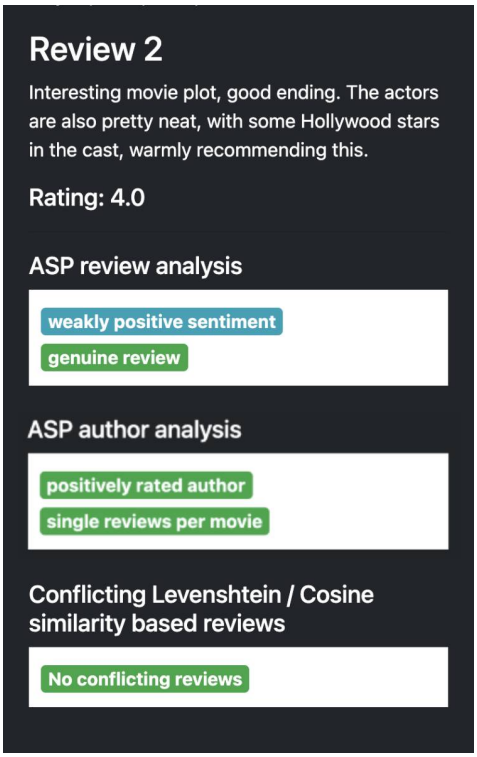

Figure 4. Genuine review example.

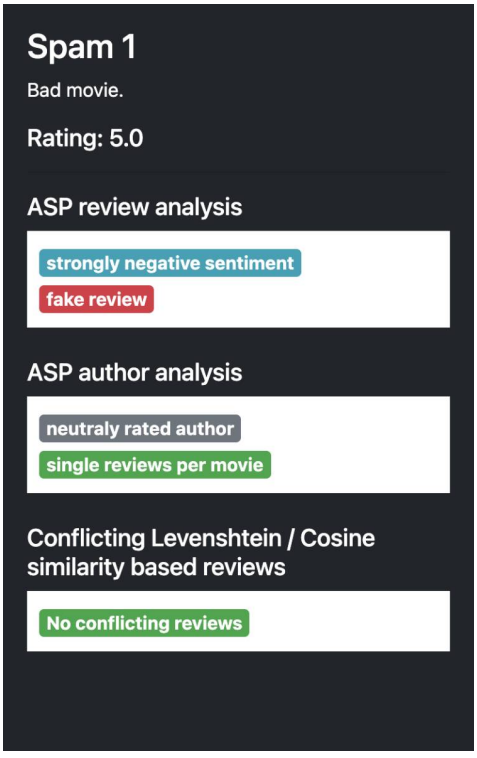

Figure 5. Spam review example.

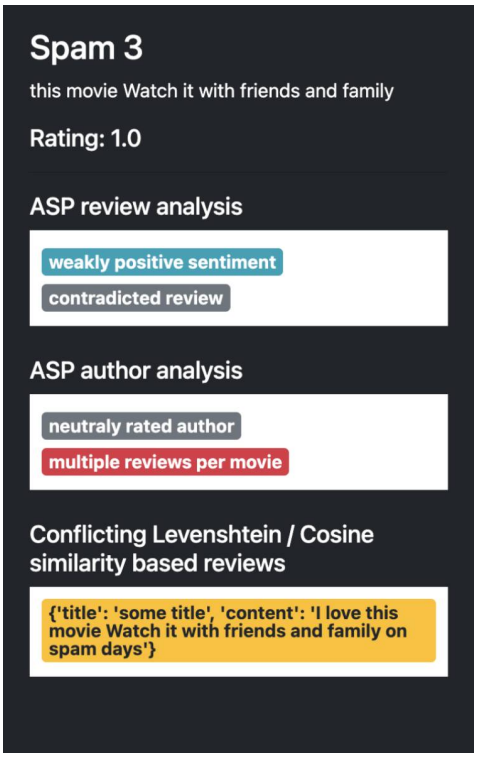

Figure 6. Contradicted review example.

8. Discussion

The problem of fake review detection is complicated since fake reviews are linked to at least two sources: the review content and the reviewers' behavior. Commonly, the content is analyzed and classified by adopting natural language processing techniques. However, determining the behavior of a fake reviewer is a complex problem. Various spatial-temporal patterns of behavior can be modeled as part of an optimization problem and because of the complexity, these problems are time-consuming and require a lot of computational resources to be solved. ASP is an exciting and powerful tool to address such problems, mainly offering a declarative way to specify the problem and properties of the desired solution(s). ASP inherits the advantage of rule-based approaches, (a) flexibility which means that adding new rules or updating the system requires only a few changes compared to other paradigms, (b) it does not require a vast amount of training data, (c) rule-based techniques are considered a white box; thus, it provides traceability and transparency for critical decisions that demand a higher degree of explanation usually provided by machine learning approaches, (d) same rules can be used for another data type, like tweets or Facebook posts, for fact-checking purposes, and (e) clear and understandable specification for users. Regarding threshold values used in ASP approaches, they can be calculated regularly and automatically based on newly inserted data. Additionally, adding new threshold values can help in restricting and limiting the spamming behavior.

Table 1 shows a list of different reviews, representing examples of labeled reviews by the ASP solver, using the proposed review-centric KB. The table consists of the review content in the first column, following the review rating, sentiment score using the trained model, the Levenshtein/Cosine review similarity score, the triggered rules, and the actual label has given review. The last two reviews show an example of two reviews posted shortly after each other, thus resulting in the second review being labeled as a contradicted review, according to the given input parameters for the ASP solver.

Table 1. Example reviews alongside their triggered ASP rules.

Review List					
Review Content	**Review Rating**	**Review Sentiment**	**Similar Reviews (LCS)**	**Triggered Rules**	**Label**
"bad movie"	5	1	3	highPolarityDiff, spamByLCS	fake review
"Watched it yesterday, pretty good movie, would recommend to watch!"	4	4	0	normalPolarity	genuine review
"Would say it was worth watching it, but in the end the story was too vague and we were not happy with the ending."	2	4	0	moderatePolarityDiff	possibly genuine review
"Great movie and awesome actors, one of our favorites!"	2	5	0	highPolarityDiff	contradicted review
"Would say it was worth watching it, but in the end the story was too vague and we were not happy with the ending."	2	4	0	moderatePolarityDiff	possibly genuine review
"This movie is just not for me." *(Posted shortly after the previous review from the same IP address)*	2	2	2	moderatePolarityDiff, spamByTimestamp, spamByLCS	possibly fake review

Figure 7 shows a confusion matrix of 50 movie reviews, obtained from the Rotten Tomatoes dataset, including their sentiment score from 1 to 5, that were processed by the website to analyze the accuracy of the sentiment analysis. The matrix shows that the prediction of the sentiment analysis is correct for most of the reviews, while only few reviews had a difference higher than 1, as, for example, the one review that had a true sentiment score of 1 and the system predicted a score of 3. The highest accuracy was for reviews with the sentiment score of 4.

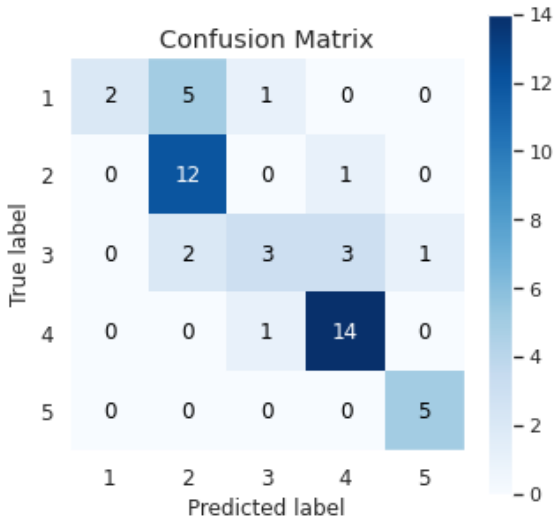

Figure 7. Confusion matrix of 50 movie reviews for sentiment analysis.

9. Conclusions

This paper proposes a dynamic recognition system for detecting fake reviews, spam, and spammers on the Web introducing fake reviews. After looking in-depth at the behavioral patterns of users known from the state-of-the-art literature, a set of illustrating

constraints and rules has been designed. The problem could be transformed into an optimization problem that can be solved using ASP. As proven in Reference [10], ASP is already established in the industrial segment as a state-of-the-art tool for different approaches, such as search or scheduling problems and usages for engineering tasks and optimizing systems, such as caching strategies for optimizing performance in network systems.

Different components used in the reasoning system, which process movie reviews and user data, were implemented and combined under a fact-checking framework. Such a system is necessary because the problem of opinion spam has increased dramatically in the past few years. The output of the implemented system and other user and review data is fact-checked with declarative rules coded in ASP. The constraints and behavioral rules can be altered, edited, and deleted easily due to the flexibility of this system. Behavioral rules are used on user data to examine the trustworthiness of said reviews and users according to their overall behavior. Sentiments of user reviews are determined, which analyzes the raw review content.

Deviations in movie features are detected with an algorithm based on Levenshtein Distance that calculates a deviation score and then provides a list that is sorted by high or low deviation.

Such rule-based systems and the KB are in high demand to be well-designed. It allows for achieving a higher degree of accuracy and a properly working model. A proper design also leads to an extensible model, which will enable us to process much more complex scenarios at the same level of high accuracy. One of our next steps is to consider the social and personal features of the users; the reason is, based on their activities and profiles, it is likely that fake users can be identified [15].

This project shows the high capability of ASP to solve such problems and we believe that much scientific work can be done by further developing our proposed approach for fake review detection, a simple yet powerful and highly flexible tool for detecting opinion spam in movie reviews, although we believe that the same approach can work for different domains. Moreover, many more behavioral rules can be added to point out suspicious users or spamming patterns; it might require additional input data to work correctly.

For future work, we want to implement a more sophisticated machine learning model for the sentiment analysis of the reviews, using deep learning approaches.

Author Contributions: Conceptualization and Methodology: N.J.; Formal analysis: N.J., A.B.; Supervision: N.J. and W.K. All authors have read and agreed to the published version of the manuscript.

Funding: Open Access Funding by the University of Vienna.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The authors ensure that there are no personal circumstances, interest, or sponsors that may be perceived as inappropriately influencing the representation or interpretation of reported research results.

References

1. Mourão, R.R.; Robertson, C.T. Fake News as Discursive Integration: An Analysis of Sites That Publish False, Misleading, Hyperpartisan and Sensational Information. *J. Stud.* **2019**, *20*, 2077–2095. [CrossRef]
2. Jnoub, N.; Klas, W.; Kalchgruber, P.; Momeni, E. A Flexible Algorithmic Approach for Identifying Conflicting/Deviating Data on the Web. In Proceedings of the 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), Colmar, France, 11–13 July 2018; pp. 1–5.
3. Jnoub, N.; Al Machot, F.; Klas, W. A Domain-Independent Classification Model for Sentiment Analysis Using Neural Models. *Appl. Sci.* **2020**, *10*, 6221. [CrossRef]
4. Ceron, W.; de Lima-Santos, M.F.; Quiles, M.G. Fake news agenda in the era of COVID-19: Identifying trends through fact-checking content. *Online Soc. Netw. Media* **2021**, *21*, 100116. [CrossRef]

5. Akhtar, M.S.; Chakraborty, T. Overview of Constraint 2021 Shared Tasks: Detecting English Covid-19 Fake News and Hindi Hostile Posts. In *Combating Online Hostile Posts in Regional Languages during Emergency Situation: First International Workshop, CONSTRAINT 2021, Collocated with AAAI 2021, Virtual Event, 8 February 2021*; Revised Selected Papers; Springer Nature: Berlin/Heidelberg, Germany, 2021; p. 42. Available online: https://www.springer.com/gp/book/9783030736958 (accessed on 31 May 2021).
6. Lappas, T.; Sabnis, G.; Valkanas, G. The impact of fake reviews on online visibility: A vulnerability assessment of the hotel industry. *Inf. Syst. Res.* **2016**, *27*, 940–961. [CrossRef]
7. Jnoub, N.; Klas, W. Declarative Programming Approach for Fake Review Detection. In Proceedings of the 2020 15th International Workshop on Semantic and Social Media Adaptation and Personalization, Zakynthos, Greece, 29–30 October 2020; pp. 1–7.
8. Kumar, S.; Cheng, J.; Leskovec, J.; Subrahmanian, V. An army of me: Sockpuppets in online discussion communities. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3 April 2017; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2017; pp. 857–866.
9. Fornaciari, T.; Poesio, M. Identifying fake Amazon reviews as learning from crowds. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, Gothenburg, Sweden, 26–30 April 2014; pp. 279–287.
10. Falkner, A.; Friedrich, G.; Schekotihin, K.; Taupe, R.; Teppan, E.C. Industrial applications of answer set programming. *KI Künstliche Intell.* **2018**, *32*, 165–176. [CrossRef]
11. Dewang, R.K.; Singh, A.K. State-of-art approaches for review spammer detection: A survey. *J. Intell. Inf. Syst.* **2018**, *50*, 231–264. [CrossRef]
12. Ahmed, H.; Traore, I.; Saad, S. Detecting opinion spams and fake news using text classification. *Secur. Priv.* **2018**, *1*, e9. [CrossRef]
13. Chowdhary, N.S.; Pandit, A.A. Fake Review Detection using Classification. *Int. J. Comput. Appl.* **2018**, *180*, 16–21.
14. Wang, X.; Zhang, X.; Jiang, C.; Liu, H. Identification of fake reviews using semantic and behavioral features. In Proceedings of the 2018 4th International Conference on Information Management (ICIM), Oxford, UK, 25–27 May 2018; pp. 92–97.
15. Barbado, R.; Araque, O.; Iglesias, C.A. A framework for fake review detection in online consumer electronics retailers. *Inf. Process. Manag.* **2019**, *56*, 1234–1244. [CrossRef]
16. Zhang, D.; Zhou, L.; Kehoe, J.L.; Kilic, I.Y. What online reviewer behaviors really matter? Effects of verbal and nonverbal behaviors on detection of fake online reviews. *J. Manag. Inf. Syst.* **2016**, *33*, 456–481. [CrossRef]
17. Mukherjee, A.; Venkataraman, V.; Liu, B.; Glance, N. Fake Review Detection: Classification and Analysis of Real and Pseudo Reviews. UIC-CS-03-2013. Technical Report. 2013. Available online: Availableonline:https://www.semanticscholar.org/paper/Fake-Review-Detection-%3A-Classification-and-Analysis-Mukherjee-Venkataraman/4c521025566e6afceb9adcf27105cd33e4022fb6?p2df (accessed on 31 May 2021).
18. Heydari, A.; Tavakoli, M.; Salim, N. Detection of fake opinions using time series. *Expert Syst. Appl.* **2016**, *58*, 83–92. [CrossRef]
19. Lin, Y.; Zhu, T.; Wu, H.; Zhang, J.; Wang, X.; Zhou, A. Towards online anti-opinion spam: Spotting fake reviews from the review sequence. In Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014), Beijing, China, 17–20 August 2014; pp. 261–264.
20. Wang, H.; Guo, K. The impact of online reviews on exhibitor behaviour: Evidence from movie industry. *Enterp. Inf. Syst.* **2016**, *11*, 1–17. [CrossRef]
21. Johnson, J.M.; Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J. Big Data* **2019**, *6*. [CrossRef]
22. Fuscà, D.; Germano, S.; Zangari, J.; Calimeri, F.; Perri, S. Answer set programming and declarative problem solving in game AIs. *CEUR Workshop Proc.* **2013**, *1107*, 81–88.
23. Mitra, A.; Clark, P.; Tafjord, O.; Baral, C. Declarative Question Answering over Knowledge Bases containing Natural Language Text with Answer Set Programming. *arXiv* **2019**, arXiv:1905.00198.
24. Gebser, M.; Schaub, T.; Thiele, S.; Veber, P. Detecting Inconsistencies in Large Biological Networks with Answer Set Programming. *arXiv* **2010**, arXiv:1007.0134.
25. Ramli, C.D. Detecting incompleteness, conflicting and unreachability XACML policies using answer set programming. *arXiv* **2015**, arXiv:1503.02732.
26. Gerhards, E.V. Your Store Is Gross-How Recent Cases, the FTC, and State Consumer Protection Laws Can Impact a Franchise System's Response to Negative, Defamatory, or Fake Online Reviews. *Franch. LJ* **2014**, *34*, 503.
27. Mukherjee, A.; Liu, B.; Glance, N. Spotting fake reviewer groups in consumer reviews. In Proceedings of the 21st international conference on World Wide Web, New York, NY, USA, 16 April 2012; pp. 191–200.
28. Patil, M.S.; Bagade, A. Online review spam detection using language model and feature selection. *Int. J. Comput. Appl.* **2012**, *59*, 7.
29. Gelfond, M.; Kahl, Y. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*; Cambridge University Press: Cambridge, UK, 2014.
30. Gebser, M.; Kaufmann, B.; Kaminski, R.; Ostrowski, M.; Schaub, T.; Schneider, M. Potassco: The Potsdam answer set solving collection. *AI Commun.* **2011**, *24*, 107–124. [CrossRef]
31. Baral, C. *Knowledge Representation, Reasoning and Declarative Problem Solving*; Cambridge University Press: Cambridge, UK, 2003. [CrossRef]
32. Al Machot, F.; Mayr, H.; Ranasinghe, S. *A Hybrid Reasoning Approach for Activity Recognition Based on Answer Set Programming and Dempster—Shafer Theory*; Springer: Cham, Switzerland, 2018; pp. 303–318. [CrossRef]

33. Haldar, R.; Mukhopadhyay, D. Levenshtein distance technique in dictionary lookup methods: An improved approach. *arXiv* **2011**, arXiv:1101.1232.
34. Yujian, L.; Bo, L. A normalized Levenshtein distance metric. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1091–1095. [CrossRef] [PubMed]
35. Tan, E.; Guo, L.; Chen, S.; Zhang, X.; Zhao, Y. Spammer behavior analysis and detection in user generated content on social networks. In Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems, Macau, China, 18–21 June 2012; pp. 305–314.
36. Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C.D.; Ng, A.; Potts, C. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 8–21 October 2013; pp. 1631–1642.