Cascade-LSTM: A Tree-Structured Neural Classifier for Detecting Misinformation Cascades

Francesco Ducci
ETH Zürich
8092 Zürich, Switzerland
fducci@student.ethz.ch

Mathias Kraus
ETH Zürich
8092 Zürich, Switzerland
mathiaskraus@ethz.ch

Stefan Feuerriegel
ETH Zürich
8092 Zürich, Switzerland
sfeuerriegel@ethz.ch

ABSTRACT

Misinformation in social media – such as fake news, rumors, or other forms of deceptive content – poses a significant threat to society and, hence, scalable strategies for an early detection of online cascades with misinformation are in dire need. The prominent approach in detecting online cascades with misinformation builds upon neural networks based on sequences of simple structural features of the propagation dynamics (e. g., cascade size, average retweeting time). However, these structural features neglect large parts of the information in the cascade. As a remedy, we propose a novel tree-structured neural network named *Cascade-LSTM.*

Our Cascade-LSTM draws upon a tree-structured long short-term memory network that is carefully engineered to the structure of online information cascades. Specifically, we suggest a novel bi-directional encoding similar to the information flow, extend inner nodes with further covariates from retweets, and fuse the network with global information from the root. As a result, our Cascade-LSTM overcomes inherent limitations from feature engineering, since it learns propagation features along the *complete* cascade. The effectiveness of our Cascade-LSTM is demonstrated based on a classification task to predict the veracity of 2,156 Twitter cascades. We improve the detection if misinformation in terms of AUC over the status quo with cascade features by 2.8 %.

Altogether, our Cascade-LSTM entails important implications: (1) it presents the first neural classifier that learns the complete cascade. (2) It demonstrates a promising approach to practitioners for detecting misinformation through mining retweet behavior. (3) The model is fairly general, which ensures widespread applicability for inferences from online cascades.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; • **Information systems** → *Social networks*; • **Computer systems organization** → *Neural networks.*

KEYWORDS

Information cascades; propagation dynamics; veracity detection; deep learning; tree-structured LSTM

ACM Reference Format:
Francesco Ducci, Mathias Kraus, and Stefan Feuerriegel. 2020. Cascade-LSTM: A Tree-Structured Neural Classifier for Detecting Misinformation Cascades. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403317

1 INTRODUCTION

Social media such as Twitter or Facebook is frequently undermined by the spread of misinformation. As a result, a vast number of social media users are exposed to false stories, which was empirically confirmed in the case of, e. g., electoral debates [1, 2], medical discussions [47], and scientific conspiracy theories [5]. Furthermore, users even interact with misinformation by sharing the content [46] and, on top of that, the repeated exposure to misinformation has led many users to erroneously believe that it was true [33]. Given the imminent threat of misinformation [22], social media platforms and policy stakeholders alike are seeking ways to detect and eventually prevent the dissemination of misinformation [22].

The spread of misinformation – such as rumors or intentional "fake news" – in online social media has been investigated from different angles. First, computational models have helped in theorizing the underlying diffusion of misinformation as a contagion process [38]. Second, descriptive studies have shed light on the structural properties of such information cascades [10, 45, 46]. For instance, rumors as a specific type of misinformation are characterized by wider, deeper, and faster cascades [46]. Third, structural characteristics of information cascades have been used for predicting the underlying veracity [44, 45].

The conventional approach to predicting the veracity of cascades draws upon neural networks with feature engineering [44, 45]: the shape of online cascades is translated into a sequence of structural features and, subsequently, these are fed into a neural network classifier. The potential set of features is diverse and includes, e. g., the size of the cascade, in- and out-degrees of nodes, depth-to-breadth ratio, and retweeting time. However, feature engineering encompasses descriptive measures of the cascade and does not account for the individual behavior of retweeting. Thus, the actual propagation dynamics from each individual retweet are lost.

An appropriate prediction mechanism must react in time to provide early warnings [41], which rules out time-consuming manual fact-checking by humans which often exceeds 24 hours [34]. As a remedy, the use of statistical classifiers to pre-filter the tweets sent for manual verification of the facts seems beneficial. Further, it has been argued before that in practice a prediction should not rely on individual linguistic clues [26], but should only consider the general topic of a tweet in addition to other user variables and the dynamics of retweets. Therefore, the data structure of retweet cascades must be modeled, that is, the branching that causes the cascade LSTM [9].

Recently, works have been proposed that utilize sequential neural networks to process the sequences of information (e. g. tweets), sorted in time. In previous research long short-term memory networks (LSTMs) [28], gated recurrent units (GRUs) [26, 28] or convolutional neural networks (CNNs) [26] have been applied for this task. Additionally, hidden Markov models (HMMs) have been utilized to model the sequence of information spreading [45]. Although these approaches are capable of processing information in time, they neglect the propagation dynamics.

Proposed Cascade-LSTM:[1] We overcome the shortcomings of the former classifiers by learning the propagation dynamics along the complete cascade. That is, we propose a novel tree-structured neural network, which we call *Cascade-LSTM*. This classifier is carefully tailored to the tree-based structure of information cascades for the purpose of making inferences, thus enabling our Cascade-LSTM to encode each individual retweet with additional covariates. Methodologically, we draw upon a tree-structured long short-term memory (Tree-LSTM) [37, 40, 53]; however, we propose a variant that models the structure of information cascades by introducing the following extensions: (1) we suggest a novel bi-directional encoding that reflects the direction of information flow in cascades. (2) The inferences inside the Tree-LSTM (or, more precisely, its hidden states) are altered, so that each node is explicitly fed with propagation information (e. g., retweet time, user characteristics). (3) The inference at the root node is extended by an additional "fused" network in order to accommodate global features from the root message (e. g., its affective dimensions).

Findings: Our Cascade-LSTM was evaluated based on a set of 2,156 cascades with potential misinformation from Twitter with 1,888,758 million individual (re-)tweets. For each cascade, the veracity – i. e., whether the content is fake or real – had to be predicted. Our Cascade-LSTM attains an out-of-sample AUC of 0.741, which demonstrates that veracity detection based on propagation dynamics is highly effective. More importantly, our Cacade-LSTM outperforms previous classifiers by 2.77 percent in terms of AUC. Finally, this high predictive power is already available at early stages of a cascade (e. g., after 2 hour or 200 retweets).

Contributions: This work extends the body of research on online information cascades in the following directions:

(1) We present the first end-to-end classifier that learns the *complete* cascade. This is achieved by deriving a tailored neural network that adapts to the tree structure of information cascades. As a result, our Cascade-LSTM considers each individual retweet together with user characteristics (rather than only aggregated, high-level features).
(2) Our computational evaluations demonstrate that our Cascade-LSTM is highly effective. Particularly, it outperforms sequence learning and classifiers where cascades have been subject to feature engineering. Evidently, retweet behavior is a strong predictor and could thus function as an implicit form of crowd intelligence.
(3) With few exceptions, predictions from information cascades are largely performed by taking the fully-unfolded cascade as input. In contrast, we specifically adapt to the needs of practitioners: we study whether harmful cascades can be detected early, i. e., shortly after dissemination.

Beyond the use case in this paper, our Cascade-LSTM should be seen as a general-purpose classifier when making inferences from cascades, such as for determining factual from emotional content or for predicting the reach of a cascade.

1.1 Information Spreading in Online Social Networks

Extensive literature has studied information spreading in online social networks. We provide a summary according to different objectives in the following.

Distinctive characteristics of online information cascades have been studied via descriptive statistics. Here the focus is on structural properties of cascades such as depth and size [6, 10, 23, 24, 39, 52], the propagation dynamics of retweeting [3, 11, 32], or topological properties of the network such as the diameter, reciprocity, and virality [11, 18]. There are also works that study the heterogeneity among users [12], thereby finding that propagation is described by the homophily among users [18, 31, 35, 49]. In designing our Cascade-LSTM, we acknowledge the importance of propagation dynamics and further adhere to earlier findings by including key variables concerning both retweet behavior and user heterogeneity.

Information cascades have also served as input to predictive classifiers. At the message level, these aim at inferring the expected probability of a retweet [35]. At the cascade level, the objective is to forecast the reach as given by the number of exposed users [32], virality in the form of relative growth [7, 36], size [13, 51], retweet probability [16], or even the diffusion path [14]. The classifier must be chosen in accordance to the predicted variable. When cascade-wide inferences from propagation dynamics are desired, the status quo is given by neural networks that process sequences of features engineered to include information about the cascade structure [26, 28]. The proposed neural network architectures include long short-term memory networks [28], gated recurrent units [26, 28] or convolutional neural networks [26] or a mixture of the above. Although these approaches are capable of processing information in time, they neglect the propagation dynamics.

1.2 Tree-Structured Neural Classifiers

Tree-structured neural networks have been utilized previously in natural language processing, where they aid in learning syntax trees [37, 40] or discourse structures [50]. Formally, the idea behind the Tree-LSTM is to extend a LSTM to a recursive structure, so that the complete tree can be eventually encoded. Two different architectures of the Tree-LSTM have been promoted: the n-ary Tree-LSTM assumes a tree with a fixed out-degree and is thus not applicable to our setting, where the number of retweets is arbitrary. In contrast, the child-sum Tree-LSTM can adapt to arbitrary branching levels by combining the hidden states of all sub-trees.

The Tree-LSTM has been subject to extensions. Bi-directional variants were developed [27, 30, 40], where upward and downward tree structures were trained simultaneously. However, bi-directional

[1]The code is available from https://github.com/MathiasKraus/CascadeLSTM.

processing from the literature is only suited to exchange information among leaves and not to aid predictions at the root. Hence, this approach is not applicable to our research. As a remedy, we later develop a customized Tree-LSTM based on a novel bi-directional processing scheme. Different from prior literature, it no longer combines the two estimated Tree-LSTMs ex post at sequence level; instead, it actually fuses the learning process of two individual Tree-LSTMs.

2 THE PROPOSED CASCADE-LSTM

2.1 Problem Statement

In the following, we develop the Cascade-LSTM which processes tree-shaped information cascades $T_1, T_2, \dots$ in order to predict associated labels $y_1, y_2, \dots \in Y$. This prediction task entails several unique characteristics that are owed to the specific structure of information cascades.

An online information cascade $T_i = (X_i, E_i)$ is given by a tree structure with nodes X_i and directed edges E_i between two nodes. Within each cascade, the root $x_1 \in X_i$ defines the initial source tweet, while the nodes $x_j, j = 2, \dots, |X_i|$, refer to individual retweets, with $|X_i|$ being the number of nodes in the cascade T_i.[2] A retweet can follow either the root or any previous retweet as specified by the edges E_i. Each cascade has further propagation dynamics as follows:

(1) *Response time.* Each retweet $j = 2, \dots |X_i|$ comes with a delay τ_j that denotes the time difference between the parent tweet and tweet j. By definition, we set the delay of the root node as $\tau_1 = 0$.
(2) *Covariates.* Each tweet, i. e., each node $x_j, j = 1, \dots, |X_i|$, is linked to additional covariates that either specify user-specific covariates v_j or tweet-specific information t_j. The former comprises the number of followers, the number of people the user is following, the account age, a binary flag denoting if the account is verified, and a relative activity score. Tweet-specific information refers to the retweet hour and weekday.
(3) *Retweet count.* Each tweet $j = 1, \dots, |X_i|$ counts a certain number of retweets that we denote by $|C(j)|$ where $C(j)$ denotes the set of children of a node inside the tree. For leaves, the number of retweets $|C(j)|$ is obviously zero.
(4) *Depth.* The depth d_j of a node $x_j, j = 1, \dots, |X_i|$, is defined as the overall number of retweets that happened between the root until reaching x_j.
(5) *Root message features.* Cascades entail global features m_i that stem from the initial message (e. g., the affective dimensions of the root tweet).

A schematic overview of the above cascade structure is shown in Figure 1.

For mathematical convenience, we refer to the node-level propagation features by introducing the notation $\psi_j = [\tau_j; v_j; t_j; |C(j)|; d_j]^T$ and $\hat{\psi}_j = [\tau_j; v_j; t_j]^T$ for all $j = 1, \dots, |X_i|$. The former is explicitly fed with both the outdegree and the depth of each node, which are otherwise only implicitly encoded in the tree structure.

[2]Throughout this work, the terms "node" and "retweet" are used interchangeably. However, in cases where we explicitly refer to inner nodes and not the root, the term "retweet" is preferred. Moreover, we use the index i when enumerating cascades and j when referring to nodes.

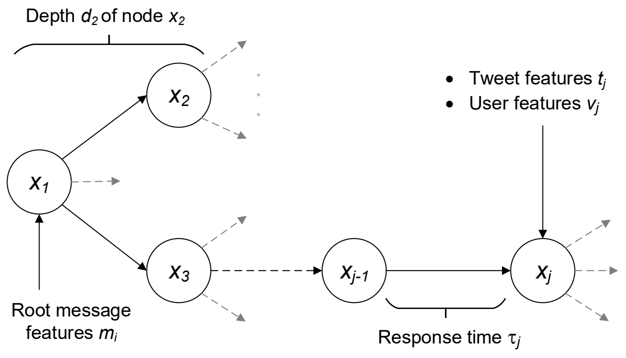

Figure 1: Exemplary cascade structure with propagation dynamics.

Notably, the number of nodes $|X_i|$ is large for online cascades, highly variable across cascades [24, 39, 48], and not limited in magnitude. This introduces the reasons why a traditional feature-based classifier fails when making inferences from an information cascade T_i. As a remedy, the following section develops a tree-structured neural classifier that encodes the complete information from T_i.

2.2 Cascade-LSTM vs. Tree-LSTM

Relationship to Tree-LSTM: In order make inferences from the complete cascade, we develop a novel tree-structured neural classifier called Cascade-LSTM. Such a classifier must learn over tree-structured data, where only one label is given per complete tree. Hence, it comes naturally to draw upon earlier concepts from Tree-LSTMs [25, 37, 53]. As the N-ary Tree-LSTM [30, 37] can only process tree-structures with fixed numbers of children, i. e., $|C(j)| = c^*$ for all j, it is not applicable to our case. In contrast, the child-sum Tree-LSTM [30, 37] is capable of learning over a variable number of children $|C(j)|$ in each node x_j, which is needed as tweets can have different numbers of retweets, i. e., their outdegree $|C(j)|$ is variable. Consequently, we build the Cascade-LSTM upon the idea of child-sum Tree-LSTMs.

Novelty: A simple application of the Tree-LSTM to information cascades is not sufficient, because valuable information from the propagation dynamics would be discarded. Hence, our Cascade-LSTM advances the current model by incorporating the following innovations:

(1) *Bi-directional processing.* The naïve Tree-LSTM traverses a tree from leaf to root, whereas the information flow in a cascade is root to leaf. Hence, we develop a bi-directional variant of the Tree-LSTM that should better learn the actual direction of the propagation dynamics. That is, we extend the input of the downward Tree-LSTM by the hidden states from an upward Tree-LSTM. Here the mechanism is designed, so that the hidden states from all nodes are eventually channeled to each other node. Put simply, each node obtains access to the full information from the cascade: from the upward structure, from its directly downward sub-tree, and implicitly from its siblings' sub-trees.
(2) *Node-level covariates.* Key characteristics of propagation dynamics, such as user/tweet covariates, retweet count, and

depth, are encoded in the nodes of the cascade and must be additionally learned. This is later achieved by incorporating node-level propagation features, i. e., ψ_j for $j = 1, \ldots, |X_i|$.

(3) *Root-level features.* The initial message that is retweeted entails further features m_i for cascade T_i. However, the Tree-LSTM does not allow to accommodate features at root level. Hence, we introduce an additional stacked neural layer that is responsible for making the final prediction. It fuses the root features m_i with the hidden cells from the tree-structured network.

2.3 Model Specification

The Cascade-LSTM comprises three different components: (C1) provides a Tree-LSTM with node-level features that processes the cascade tree via an upward traversal. (C2) provides another Tree-LSTM that is responsible for a downward traversal. Both Tree-LSTM are supposed to yield encodings $h^{\uparrow}_{\text{root}}$ and $\tilde{h}^{\downarrow}_{\text{leaf}}$, respectively. In (C3), a final "fused" network combines both encodings where, in addition, root message features are inserted.

In terms of notation, let σ denote the sigmoid function and $\odot$ the element-wise multiplication. Arrows $\uparrow$ and $\downarrow$ refer to the different computation schemes of the upward and downward Tree-LSTM, respectively.

(C1) Upward Tree-LSTM encoding $h^{\uparrow}_{\text{root}}$

This component is a tailored variant of the Tree-LSTM that traverses the cascade leaf to root.

In the upward Tree-LSTM, each node $x_j, j = 1, \ldots, |X_i|$, is passed through a LSTM unit. The idea is then to introduce a recursive computation, so that information between the LSTM unit of x_j and its children $C(j)$ is exchanged. Formally, each LSTM unit comprises an input gate i_j, an output gate o_j, an update gate u_j, a memory cell $c^{\uparrow}_j$, and a hidden state $h^{\uparrow}_j$. Intuitively, these gates control how much of the information to block or pass during the recursive computation. The relationship between the different gates inside a single LSTM unit is illustrated in Figure 2.

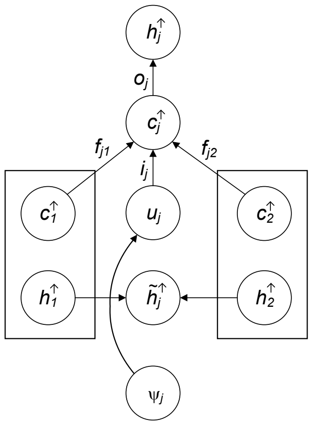

Figure 2: Presented is a single LSTM unit from an upward Tree-LSTM at node x_j with two child nodes 1 and 2.

The input to each LSTM unit at node x_j is three-fold: the hidden states $h^{\uparrow}_k$ for all children $k \in C(j)$, the memory cells $c^{\uparrow}_k$ for all children $k \in C(j)$, and additional node-level covariates ψ_j. There are further forget gates f_{jk}, one for each child $k \in C(j)$. The recursive architecture is shown in Figure 3 for the complete tree.

Given node x_j with input ψ_j, one computes the hidden state $h^{\uparrow}_j$ for each node from the children $C(j)$ via

$$\tilde{h}^{\uparrow}_j = \sum_{k \in C(j)} h^{\uparrow}_k, \tag{1}$$

$$i_j = \sigma\left(W^{\uparrow}_i \psi_j + U^{\uparrow}_i \tilde{h}^{\uparrow}_j + b^{\uparrow}_i\right), \tag{2}$$

$$f_{jk} = \sigma\left(W^{\uparrow}_f \psi_j + U^{\uparrow}_f h^{\uparrow}_k + b^{\uparrow}_f\right) \quad \forall k \in C(j), \tag{3}$$

$$o_j = \sigma\left(W^{\uparrow}_o \psi_j + U^{\uparrow}_o \tilde{h}^{\uparrow}_j + b^{\uparrow}_o\right), \tag{4}$$

$$u_j = \tanh\left(W^{\uparrow}_u \psi_j + U^{\uparrow}_u \tilde{h}^{\uparrow}_j + b^{\uparrow}_u\right), \tag{5}$$

$$c^{\uparrow}_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c^{\uparrow}_k, \tag{6}$$

$$h^{\uparrow}_j = o_j \odot \tanh(c^{\uparrow}_j) \tag{7}$$

with matrices $W^{\uparrow}_i, W^{\uparrow}_f, W^{\uparrow}_o, W^{\uparrow}_u$ for weighting the input, matrices $U^{\uparrow}_i, U^{\uparrow}_f, U^{\uparrow}_o, U^{\uparrow}_u$ for weighting the hidden states, and $b^{\uparrow}_i, b^{\uparrow}_f, b^{\uparrow}_o, b^{\uparrow}_u$ as bias vectors.

The information thus propagates upstream in such a way that the bottom-up hidden state of every node summarizes the knowledge from the complete downstream part of the tree. The computation terminates with the root node $h^{\uparrow}_{\text{root}}$, which combines the information from the complete cascade.

(C2) Downward Tree-LSTM encoding $h^{\downarrow}_{\text{leaf}}$

This component is similar to the previous one, yet with the exception that it traverses root to leaf. Accordingly, each additional retweet serves as an accumulated form of crowd intelligence that refines the encoding beyond the previous retweet. In that sense, it can reinforce previously-hold beliefs from the classifier or alter them. It also allows theoretically to control for the fact that a retweet might remain in a small echo chamber where, for some reasons, retweet dynamics occur that are different from the rest. However, simply passing observable variables from the complete prior cascade structure downward is not trivial and thus requires an appropriate encoding. In our case, we accomplish it by making the hidden states $h^{\uparrow}_j$ from the previous upward traversal of (C1) available to each LSTM unit of the downward pass.

Formally, we introduce the following changes to a default Tree-LSTM. We no longer draw upon input ψ_j, but replace it with $\psi'_j = [\psi_j; h^{\uparrow}_j]$. Given the traversal from root to leaf, each LSTM unit does not encounter a set of children but, instead, it must be based on a single parent $P(j)$. For the same reason, there is only a single forget gate f_j per node j. Hence, the downward Tree-LSTM is given by

$$i_j = \sigma\left(W^{\downarrow}_i \psi'_j + U^{\downarrow}_i h^{\downarrow}_{P(j)} + b^{\downarrow}_i\right), \tag{8}$$

$$f_j = \sigma\left(W^{\downarrow}_f \psi'_j + U^{\downarrow}_f h^{\downarrow}_{P(j)} + b^{\downarrow}_f\right) \tag{9}$$

$$o_j = \sigma\left(W^{\downarrow}_o \psi'_j + U^{\downarrow}_o h^{\downarrow}_{P(j)} + b^{\downarrow}_o\right), \tag{10}$$

$$u_j = \tanh\left(W_u^{\downarrow}\psi'_j + U_u^{\downarrow}h^{\downarrow}_{P(j)} + b_u^{\downarrow}\right), \tag{11}$$

$$c_j^{\downarrow} = i_j \odot u_j + f_j \odot c^{\downarrow}_{P(j)}, \tag{12}$$

$$h_j^{\downarrow} = o_j \odot \tanh(c_j^{\downarrow}) \tag{13}$$

with matrices $W_i^{\downarrow}, W_f^{\downarrow}, W_o^{\downarrow}, W_u^{\downarrow}$ for weighting the input, matrices $U_i^{\downarrow}, U_f^{\downarrow}, U_o^{\downarrow}, U_u^{\downarrow}$ for weighting the hidden states, and $b_i^{\downarrow}, b_f^{\downarrow}, b_o^{\downarrow}, b_u^{\downarrow}$ as bias vectors.

The top-down hidden state of every node summarizes the knowledge of all the upward part of the tree. Since such knowledge also includes the bottom-up information stored in all $h_j^{\uparrow}$, we synthesize the full information of the cascade, where every node is seen in relationship with the others – downward, upward, and in the other branches. Given that the number of leaf nodes is variable, we later take the average over the hidden states of all leaves, which is computed via

$$\tilde{h}^{\downarrow}_{\text{leaf}} = \frac{1}{\#\text{leaves}} \sum h^{\downarrow}_{\text{leaf}}. \tag{14}$$

(C3) Fused network

The above Tree-LSTMs are now combined for making the final prediction. Formally, this step leverages the hidden states of C1 and C2, which are given by $h^{\uparrow}_{\text{root}}$ and $\tilde{h}^{\downarrow}_{\text{leaf}}$, respectively. The feed-forward neural network additionally draws upon the root message features m_i. Hence, the overall input to it for cascade i is

$$z_i = [h^{\uparrow}_{\text{root}}; \tilde{h}^{\downarrow}_{\text{leaf}}; m_i]. \tag{15}$$

The input is then plugged into a deep feed-forward neural network that generates a prediction $\tilde{y}_i \in [0, 1]$ belonging to cascade T_i. It refers to the probability of the cascade to be true.

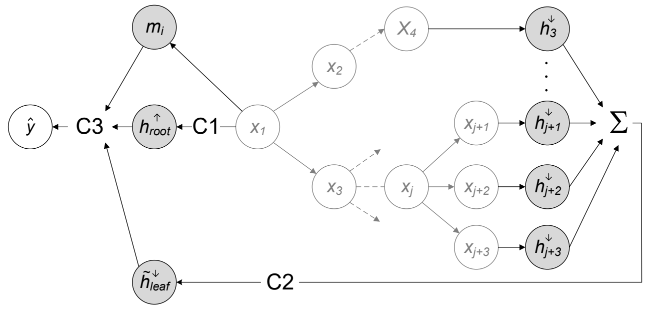

Figure 3: Schematic diagram of the Cascade-LSTM that is applied to a cascade structure (illustrated by nodes x_* and root features m_i) as follows: the Cascade-LSTM combines an upward Tree-LSTM encoding (C1) and a downward encoding (C2) in order to make inferences at the root-level from the cascade structure (C3).

2.4 Model Variations

The previous Cascade-LSTM is subject to several adaptations in order to study the relevance of individual model components to the overall performance. Hence, we later experiment with the following adaptations:

(1) **Uni-directional Cascade-LSTM.** Our original Cascade-LSTM was deliberately based upon a novel bi-directional processing, so that the tree is traversed both leaf to root and root to leaf. The intention was that this better reflects the information flow inside the cascade. Different from that, we now obtain a uni-directional Cascade-LSTM by combining the upward encoding of the tree structure from a standard Tree-LSTM with the root message features (i. e., only components C1 and C3 of the Cascade-LSTM). The input z to the final prediction layer of the uni-directional Cascade-LSTM is thus

$$z = [h^{\uparrow}_{\text{root}}; m_i]. \tag{16}$$

(2) **Root-less Cascade-LSTM.** The predictive power of the root message features is quantified by changing the final prediction layer so that it no longer depends on root message features m_i. That is, this model variation is only based on the upward and downward Tree-LSTMs (i. e., components C1 and C2 of the Cascade-LSTM) with input

$$z = [h^{\uparrow}_{\text{root}}; \tilde{h}^{\downarrow}_{\text{leaf}}]. \tag{17}$$

(3) **Shallow Cascade-LSTM.** We incorporated both depth and outdegree inside ψ_j in order to help the Cascade-LSTM in learning such relationship, even though this could be learned from data. Hence, we replace ψ_j with $\hat{\psi}_j$, i. e., the response time, user-specific covariates and tweet-specific covariates.

(4) **Flat LSTM.** We further investigate a sequential bidirectional LSTM, where cascades are flattened into a sequential structure. For this purpose, we sort the retweets by their timestamp, generating a sequence of retweets by time. Thereby, the features representing a retweet are similar to the features representing retweets for the Cascade-LSTM. Thus, this quantifies the added value of having a tree-structure.

2.5 Data Augmentation

The Cascade-LSTM is – analogous to the Tree-LSTM – characterized by a high-dimensional parameter space and thus prone to overfitting. Beyond regularization, this risk is further addressed by data augmentation: here the objective is to enlarge the original training set by creating additional samples with slight structural modifications. By adapting ideas from [17], we develop three strategies that are tailored to cascade structures.

Node reordering: Node reordering rearranges inner nodes as follows (Figure 4):

(1) Choose a random inner node $j \in \{2, \dots, |X_i|\}$, i. e., not the root.
(2) Choose a random sibling $k \in C(P(j)) \setminus \{j\}$ of j, i. e., a child from the same parent.
(3) Insert j between k and its former parent node. Accordingly, the parent of j remains unchanged, but j appears in a different path.
(4) Remove all children $C(j)$ of j and add them to the parent $P(j)$ of j.

Artificial leaf insertion: Artificial leaf insertion generates samples with larger trees by expanding leaves with subtrees (Figure 5):

(1) Choose a random leaf node j with covariates ψ_j.
(2) Choose a random number $w \in [0, 1]$.

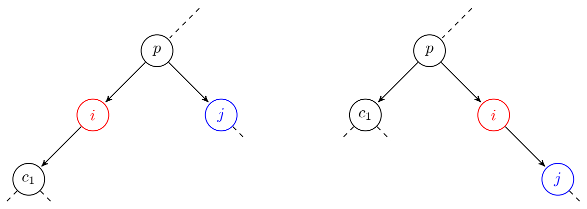

Figure 4: Node reordering where node j is moved to a different subtree of its former parent $P(j)$.

(3) Append to j two children $x_j^{(1)}$ and $x_j^{(2)}$ with corresponding covariates $\psi_j^{(1)} = w \odot \psi_j$ and $\psi_j^{(2)} = (1-w) \odot \psi_j$, respectively.

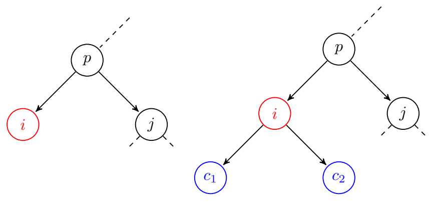

Figure 5: Artificial leaf insertion where deeper trees are obtained by replacing a leave with a subtree consisting of two children.

Root feature perturbation. We expect the root message features m_i to be highly predictive and, hence, perform data augmentation via perturbations:

(1) Fit two multivariate Gaussian distributions to both the training samples labeled as "fake" and "real", respectively. Each distribution is then specified by a mean μ_y and variance Σ_y with $y \in \{\text{"fake"}, \text{"real"}\}$.
(2) Given root features m_i for a cascade with label y_i, we create a new training observation with a perturbed $\hat{m}_i = (1-\lambda)m_i + \lambda p$ where p is randomly sampled via $p \sim \mathcal{N}(\mu_{y_i}, \Sigma_{y_i})$, i. e., it adds noise according to training samples with the same label. In our experiments, we set $\lambda = 0.1$.

We detail the learning process of the Cascade-LSTM in Appendix C.

3 EXPERIMENTAL SETUP

3.1 Dataset

For this study, we utilized 2,156 Twitter cascades with at least 100 retweets belonging to different (mis-)information from [46]. That is, each cascade received a binary label based on the veracity of the underlying claim according to six different fact-checking organizations. The resulting dataset is highly imbalanced, as only 310 cascades were labeled as "Veracity = real", while the majority, i. e., 1,845, were classified as "Veracity = fake". These imbalances are later addressed during training based on combined over-/under-sampling, as well as by reporting the AUC. The mean cascade size amounts to 876.5 retweets, while we observe smaller cascades labeled as "real" (on average, 408.1 retweets with standard deviation of 345.5) as compared to "fake" ones (955.1 retweets with standard deviation of 1497.7). In total, our dataset comprises 1,888,758 tweets. Further descriptive statistics are presented in Appendix B.

Several variables were collected at the cascade, tweet, and user level. The complete list is given in Table 1. Accordingly, the dataset comprises various information that are conventionally displayed on Twitter, e. g., the account age and follower counts. Additional variables that specify the propagation dynamics were derived from the cascade structures, i. e., the response time, the number of retweets (i. e., children), and the node depth. The original tweet at the root was subject to affective computing in order to extract its emotions according to Plutchik's with 8 different emotion categories [4]. If necessary, conventional transformations were applied in order to address skewed distributions and different ranges (see Table 1).

Variable	Transformations
User covariates v_j	
# Followers	log, standardization
# Accounts followed	log, standardization
Account age (in days)	log, standardization
Verified account (binary)	—
Relative activity[†]	log, standardization
Tweet covariates t_j	
Retweet hour	cyclic encoding[‡] standardization
Retweet weekday (0–6)	cyclic encoding[‡], standardization
Propagation dynamics	
Response time τ_j (in seconds)	log, standardization
# Children $\lvert C(j)\rvert$	standardization
Node depth d_j	standardization
Root features m_i	
anger, anticipation, disgust, fear, joy, sadness, surprise, trust $\in [0, 1]$	standardization

[†] Number of user responses relative to the account age.
[‡] An hour h is encoded as a tuple $(\sin h/23 \times 2\pi, \cos h/23 \times 2\pi)$ as this preserves the distances before/after midnight (e. g., 11 pm is closer to 1 am than 7 pm). Analogously, the weekday wd is encoded as $(\sin wd/6 \times 2\pi, \cos wd/6 \times 2\pi)$.

Table 1: Variables from the Twitter cascades, grouped according to Cascade-LSTM.

The dataset is randomly split in a training and and test set with ratios of 85 % and 15 %, respectively.

3.2 Baseline Models

Our proposed Cascade-LSTM is compared against a series of sequential models. During their design, we closely followed previously proposed architectures [cf. 26, 28, 45], but we incorporated adaptations due to the nature of our dataset where necessary. For instance, we additionally included root message features (i. e., the affective dimensions).

Architectures: These include gated recurrent units [28], long short-term memory networks [28], and a combination of a gated recurrent unit and a convolutional neural network [26]. Further, we include a Hidden Markov Model (HMM) with a multivariate Gaussian emission probability [45].

4 RESULTS

4.1 Classification Performance for Complete Cascades

We now compare our proposed Cascade-LSTM against a series of neural networks. Our hypothesis is that the Cascade-LSTM benefits from a carefully-designed neural network architecture, which allows it to encode the propagation dynamics along the complete cascade and it should thus outperform all baselines. Again, we remind that the baselines are designed according to earlier literature [26, 29]. However, note that their methodology is designed for the detection of rumors (including multiple cascades), whereas our prediction is on cascade-level. This yields different predictive performances.

Table 2 lists the results when the complete cascades (i. e., as observed ex post) are classified. Given that our dataset is highly imbalanced, we report the area under the receiver operating characteristic curve (AUC). Consistent with needs in practice, we further provide the precision, i. e., the ratio between cascades with "fake" labels and "fake" prediction.[3] Thereby, higher precision means that falsely classified real cascades (i. e., candidates of fake messages that turn out to be real) occur less frequently, thus reducing the need for potential follow-up costs resulting from manual verification of the facts.

In terms of area under the curve, the Cascade-LSTM is the best classifier with an AUC of 0.741. It yields an improvement of 2.8 % over the best baseline classifier, the CNN LSTM [26]. Similarly, in terms of precision, the Cascade-LSTM outperforms all the other models, with a precision of 0.848. This is an improvement of 2.05 percent with respect to the Tree-LSTM and an improvement of 5.08 percent over the Flat GRU [28]. We note that the standard Tree-LSTM is only ranked fifth with regard to AUC. Altogether, our findings confirm that the Cascade-LSTM is consistently superior.

Approach	AUC	Precision
HMM [45]	0.602	0.694
Flat GRU [28]	0.709	0.807
Flat LSTM [28]	0.714	0.739
CNN LSTM [26]	0.721	0.798
Tree-LSTM	0.703	0.831
Cascade-LSTM	**0.741**	**0.848**

Table 2: Performance of Cascade-LSTM in comparison to feature-based classifiers during veracity detection of complete cascades. The best result is highlighted in bold.

4.2 Classification Performance of Partial Cascades

The previous section evaluated our Cascade-LSTM based on complete cascades, whereas an imminent challenge is to identify misinformation early [34]. In practice, this can theoretically introduce the possibility for interventions that limit the reach of misinformation. Hence, we perform a sensitivity analysis:

[3]In all experiments, we adapt to the severe class imbalances by reporting a weighted precision where the precision is weighted by the support of the class, so that it is scaled by class frequency.

(1) *Lifetime.* We study the prediction performance based on cascades after a predefined lifetime. Formally, the cascades are cropped, so that they only include retweets up a certain time $t_1 + \sigma$ after the original tweet has been published at time t_1.
(2) *Retweet count.* An alternative threshold is given by a maximum number of retweets, i. e., the cascade is cropped to time t_γ, so that only γ retweets are included.

In the following, we experimented with different ranges $\{2\,\text{h}, 3\,\text{h}, 6\,\text{h}, 12\,\text{h}, 24\,\text{h}\}$ and $\gamma \in \{200, 500, 1000, 2000\}$. For fair comparisons, the classifiers were re-estimated for all experiments where σ or γ were varied and the same cropping was also applied to the training set.[4]

Figure 6 compares the AUC. Evidently, the Cascade-LSTM consistently proves to be the best classifier. It is particularly superior at early stages of cascades. Here a performance gain of our Cascade-LSTM over other models is observed. For instance, after 2 hours, the second-best classifier (i. e., Flat LSTM) merely reaches a AUC of 0.674, while the Cascade-LSTM attains an AUC of 0.694. This is an increase by 2 percentage points in terms of AUC.

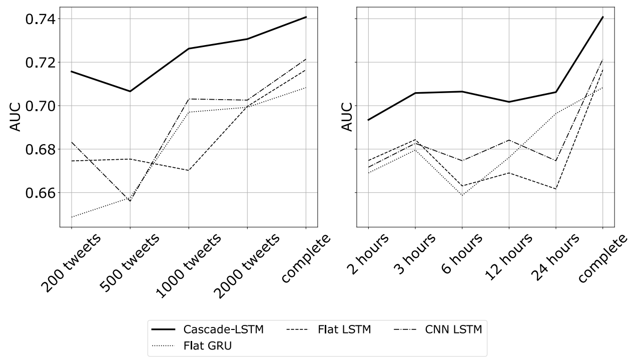

Figure 6: Comparison of AUC of partially-observed cascade as demanded for early interventions, i. e., after certain retweet counts (left) or lifetimes (right). Again, the Cascade-LSTM yields the best overall best performance.

4.3 Sensitivity to Model Variations

Network architecture: We now experiment with different variations of our Cascade-LSTM (see Section 2.4). This should confirm the effectiveness of our proposed network architecture. Table 3 lists the results. Evidently, our novel bi-directional encoding of the cascade that reflects the direction of the information flow is highly important: replacing the Cascade-LSTM with a uni-directional processing (i. e., as in a simple Tree-LSTM) reduces the AUC by 0.025. Utilizing a sequential LSTM on the flatted cascades reduces the AUC by 0.027. The additional inclusion of root message features affects the AUC by 0.016, confirming that they entail considerable predictive power. Finally, the shallow variation without the two propagation variables revealed only a negligible effect; hence, the Cascade-LSTM can successfully infer such structural information implicitly during learning.

[4]We neglect the HMM for this task due to high computational costs.

Model variation	Components	Variables	AUC	Prec.
Flat LSTM [cf. 28]	(C3)	(ψ, m_j)	0.714	0.739
Uni-directional architecture	(C1, C3)	(ψ, m_j)	0.716	0.818
Root-less architecture	(C1, C2)	(ψ)	0.725	0.824
Shallow architecture	(C1, C2, C3)	$(\hat{\psi}, m_j)$	0.734	0.840
Cascade-LSTM	(C1, C2, C3)	(ψ, m_j)	**0.741**	**0.848**

Table 3: Sensitivity of Cascade-LSTM to variations in the network architecture. Best performance in bold.

Data augmentation: Our earlier intuition behind developing different techniques for data augmentation was that they were supposed to reduce the risk of overfitting.[5] We now confirm this computationally in Table 4. In short, the combined application of different techniques for data augmentation helps in increasing the AUC by 5.4 %. As a comparison, this change is larger than any of the previous changes when we varied the network architecture (e. g., replacing bi-directional processing by a uni-directional Cascade-LSTM). Perturbations for root features appear especially effective. This coincides with the observations that the root message features entail considerable predictive power. Furthermore, reordering the nodes has a bigger impact than leaf insertion. We attribute the latter to the size of the cascades before/after applying it: leaf insertion can only impact a single leaf node, whereas node reordering can significantly change the structure of the tree by swapping two full sub-trees.

Leaf insertion	Node reordering	Root feature perturbation	AUC	Precision
✗	✗	✗	0.703	0.838
✓	✗	✗	0.720	0.829
✗	✓	✗	0.732	0.840
✗	✗	✓	0.736	0.847
✓	✓	✓	**0.741**	**0.848**

Table 4: Data augmentation appears important when training our Cascade-LSTM. Best performance in bold.

5 DISCUSSION

The public, as well as social media platforms, call for novel strategies that effectively prevent the spread of online misinformation such as “fake news”. A potential remedy originates from an early detection based on the online spreading behavior. These propagation dynamics have also the benefit of (theoretically) being robust against manipulation [8]. The reason is that each individual retweet acts as a form of crowd intelligence. However, incorporating propagation dynamics requires a classifier that, rather than leveling out heterogeneity in a cascade-wide aggregation, actually takes each single retweet into consideration. This motivated the design of our novel Cascade-LSTM, which operates along the *complete* cascade, thereby overcoming limitations of conventional feature engineering [8]. The latter was confirmed numerically in our experiments as our Cascade-LSTM yields a superior detection accuracy.

Differences to previous work: There are several differences beyond the body of research that we find important: (1) we model the cascade as a tree, whereas others [8, 19] consider it as an undirected graph and thus miss the actual direction of the propagation. (2) We specifically measure precision in addition to AUC [8, 43], since we are primarily concerned with detecting fake information. (3) We evaluate the ability of early detection as demanded in practice. In contrast to that, the majority of prior works [e. g., 8, 43, 45] report the classification accuracy only after the complete cascade was observed.

Why this model: Our model shows multiple extensions over previous research to allow processing of cascade structures (cf. Section 2.2): (1) In contrast to the majority of previous research, our model processes the actual tree-structure instead of a simplified 1-dimensional representation. (2) The Cascade-LSTM specifically follows the root-to-leaves paths which represents the spreading of twitter cascades. (3) Node-level covariates (e. g. user/tweet covariates) are included at each node of the tree. (4) Global features, such as information about the initial message are concatenated to the root node. Altogether this model is specifically carved to process cascade structures in a natural way, wherein each information being added at the appropriate node.

Generalizability: This work presents the first end-to-end classifier tailored to the tree structures of online information cascades. Needless to say, the use of our Cascade-LSTM is not limited to veracity detection. It rather presents a general-purpose classifier for propagation dynamics with a wide range of applications. Beyond detecting misinformation, the Cascade-LSTM could be used for discriminating fact against emotional content or predict characteristics of the root (e. g., personal traits of the sender or whether it is a bot). It could also be applied to forecasting the reach or the lifetime of a cascade. Analogous to the outcome variable, the input could also be subject to customization in order to meet the format of other online platforms (e. g., such as Facebook). For this reason, we have specifically decided upon a general formulation, so that a variety of additional covariates from users, tweets, or root messages could be integrated.

ACKNOWLEDGMENTS

We are indebted to Twitter for collaborating in this research project and providing us access to their data. Financial supported by ETH Zurich Research Grant ETH-16 18-2 is acknowledged.

REFERENCES

[1] Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of Economic Perspectives* 31, 2 (2017), 211–236.
[2] Eytan Bakshy, Solomon Messing, and Lada A. Adamic. 2015. Exposure to ideologically diverse news and opinion on Facebook. *Science* 348, 6239 (2015), 1130–1132.
[3] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. 2012. The role of social networks in information diffusion. In *WWW*.
[4] Bernhard Kratzwald, Suzana Ilić, Mathias Kraus, Stefan Feuerriegel, and Helmut Prendinger. 2018. Deep learning for affective computing: Text-based emotion recognition in decision support. *Decision Support Systems* 115 (2018), 24–35.
[5] Alessandro Bessi, Mauro Coletto, George A. Davidescu, Antonio Scala, Guido Caldarelli, and Walter Quattrociocchi. 2015. Science vs conspiracy: Collective narratives in the age of misinformation. *PLOS ONE* 10, 2 (2015), e0118093.

[5] For instance, the classical Cascade-LSTM counts 339 parameters inside the tree-structured networks from both C1 and C2, and 529 parameters in the fused network of C3. This amounts to a total of 868 parameters, which approaches the size of our dataset.

[6] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. 2009. A measurement-driven analysis of information propagation in the Flickr social network. In *WWW*. 721–730.

[7] Justin Cheng, Lada A. Adamic, Alex P. Dow, Jon M. Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted?. In *WWW*.

[8] Mauro Conti, Daniele Lain, Riccardo Lazzeretti, Giulio Lovisotto, and Walter Quattrociocchi. 2017. It's always April fools' day!: On the difficulty of social network misinformation classification via propagation features. In *IEEE Workshop on Information Forensics and Security (WIFS)*.

[9] Riley Crane and Didier Sornette. 2008. Robust dynamic classes revealed by measuring the response function of a social system. *PNAS* 105, 41 (2008), 15649–15653.

[10] Adrien Friggeri, Lada A. Adamic, Dean Eckles, and Justin Cheng. 2014. Rumor cascades. In *ICWSM*.

[11] Sharad Goel, Ashton Anderson, Jake Hofman, and Duncan J. Watts. 2015. The structural virality of online diffusion. *Management Science* 62, 1 (2015), 180–196.

[12] Caitlin Gray, Lewis Mitchell, and Matthew Roughan. 2018. Super-blockers and the effect of network structure on information cascades. In *WWW*. 1435–1441.

[13] Liangjie Hong, Ovidiu Dan, and Brian D. Davison. 2011. Predicting popular messages in Twitter. In *WWW*.

[14] Wenjian Hu, Krishna Kumar Singh, Fanyi Xiao, Jinyoung Han, Chen-Nee Chuah, and Yong Jae Lee. 2018. Who Will Share My Image?. In *WSDM*.

[15] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Ryota Kobayashi and Renaud Lambiotte. 2016. Tideh: Time-dependent Hawkes process for predicting retweet dynamics. In *ICWSM*.

[17] Mathias Kraus and Stefan Feuerriegel. 2019. Sentiment analysis based on rhetorical structure theory: Learning deep neural networks from discourse trees. *Expert Systems with Applications* 118 (2019), 65–79.

[18] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media?. In *WWW*.

[19] Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *ICDM*.

[20] Daniel J. Lasser. 1961. Topological Ordering of a list of randomly-numbered elements of a network. *Commun. ACM* 4, 4 (1961), 167–168.

[21] Jorma Laurikkala. 2001. Improving identification of difficult small classes by balancing class distribution. In *Conference on Artificial Intelligence in Medicine in Europe (AIME)*.

[22] David M. J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. 2018. The science of fake news. *Science* 359, 6380 (2018), 1094–1096.

[23] Kristina Lerman and Rumi Ghosh. 2010. Information contagion: An empirical study of the spread of news on Digg and Twitter social networks. In *ICWSM*.

[24] Jure Leskovec, Mary McGlohon, Christos Faloutsos, Natalie Glance, and Matthew Hurst. 2007. Patterns of cascading behavior in large blog graphs. In *SDM*.

[25] Xiangang Li and Xihong Wu. 2015. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[26] Yang Liu and Yi-Fang Brook Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *AAAI*.

[27] Huaishao Luo, Tianrui Li, Bing Liu, Bin Wang, and Herwig Unger. 2018. Improving aspect term extraction with bidirectional dependency tree representation. *arXiv preprint arXiv:1805.07889* (2018).

[28] Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*.

[29] Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on Twitter with tree-structured recursive neural networks. In *ACL*.

[30] Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *ACL*.

[31] Seth A. Myers and Jure Leskovec. 2014. The bursty dynamics of the Twitter information network. In *WWW*.

[32] Seth A. Myers, Chenguang Zhu, and Jure Leskovec. 2012. Information diffusion and external influence in networks. In *KDD*. 33–41.

[33] Gordon Pennycook, Tyrone D. Cannon, and David G. Rand. 2018. Prior exposure increases perceived accuracy of fake news. *Journal of Experimental Psychology. General* 147, 12 (2018), 1865–1880.

[34] Chengcheng Shao, Giovanni Luca Ciampaglia, Alessandro Flammini, and Filippo Menczer. 2016. Hoaxy: A platform for tracking online misinformation. In *WWW*.

[35] Sofus A. Macskassy and Matthew Michelson. 2011. Why do people retweet? Anti-homophily wins the day!. In *ICWSM*.

[36] Karthik Subbian, Aditya B. Prakash, and Lada A. Adamic. 2017. Detecting large reshare cascades in social networks. In *WWW*.

[37] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from Tree-structured long short-term memory networks. In *ACL*.

[38] Marcella Tambuscio, Giancarlo Ruffo, Alessandro Flammini, and Filippo Menczer. 2015. Fact-checking effect on viral hoaxes: A model of misinformation spread in social networks. In *WWW*.

[39] Io Taxidou and Peter M. Fischer. 2014. Online analysis of information diffusion in Twitter. In *WWW*.

[40] Zhiyang Teng and Yue Zhang. 2017. Head-lexicalized bidirectional Tree LSTMs. *Transactions of the Association for Computational Linguistics* 5, 2 (2017), 163–177.

[41] The Economist. 2017. How the world was trolled. *9065* 425 (2017), 21–24.

[42] Ivan Tomek. 1976. Two modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics* 6, 11 (1976), 769–772.

[43] Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on Twitter. In *ACL*.

[44] Soroush Vosoughi. 2015. *Automatic detection and verification of rumors on Twitter*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[45] Soroush Vosoughi, Mostafa 'Neo' Mohsenvand, and Deb Roy. 2017. Rumor Gauge: Predicting the veracity of rumors on Twitter. *ACM Transactions on Knowledge Discovery from Data* 11, 4 (2017), 1–36.

[46] Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science* 359, 6380 (2018), 1146–1151.

[47] Przemyslaw M. Waszak, Wioleta Kasprzycka-Waszak, and Alicja Kubanek. 2018. The spread of medical fake news in social media : The pilot quantitative study. *Health Policy and Technology* 7, 2 (2018), 115–118.

[48] Karol Wegrzycki, Piotr Sankowski, Andrzej Pacuk, and Piotr Wygocki. 2017. Why do cascade sizes follow a power-law?. In *WWW*.

[49] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. 2013. Virality prediction and community structure in social networks. *Scientific Reports* 3 (2013), 2522.

[50] Xianghua Fu, Wangwang Liu, Yingying Xu, Chong Yu, and Ting Wang. 2016. Long short-term memory network over rhetorical structure theory for sentence-level sentiment analysis. In *Asian Conference on Machine Learning (ACML)*.

[51] Jaewon Yang and Jure Leskovec. 2010. Modeling information diffusion in implicit networks. In *ICDM*.

[52] Chengxi Zang, Peng Cui, Chaoming Song, Christos Faloutsos, and Wenwu Zhu. 2017. Quantifying structural patterns of information cascades. In *WWW*.

[53] Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*.

A RELATIONSHIP OF OUR CASCADE-LSTM TO BI-DIRECTIONAL TREE-LSTMS

While there have been works on bi-directional Tree-LSTMs [27, 40], these are concerned with a different objective. These works process sequential data where trees only provide an intermediate representation. Further, they make predictions at node level and not at cascade level. Hence, they are in stark contrast to our cascade classification.

The work in Teng and Zhang [40] gives a tool for handling sequential data under syntax dependencies but not for learning the actual tree; hence, prohibiting adaptations to our research. Specifically, it simply estimates an upward and a downward Tree-LSTM in isolation without an intelligent combination, while the average hidden states are later used to make predictions along a sequence (and not a tree). The work in [30] also targets sequential data. Here the hidden states of both the upward and downward LSTM-RNN are combined as a basis for making prediction at each element of sequence. Similarly, the authors of [27] apply an upward and a downward Tree-LSTM, yet again concatenating their hidden states for predicting sequential output. Because of the previous arguments, the existing approaches are not readily applicable, and, therefore, we suggest a different approach to bi-directional processing.

B DESCRIPTIVE STATISTICS

Table 5 present several descriptive statistics of our dataset.

Feature	y = "real"		y = "fake"	
	Mean	**SD**	**Mean**	**SD**
Cascade structure				
Size	408	346	955	1498
Depth	4.33	1.56	4.84	2.42
Breadth	291	280	642	992
Root features				
Anger	0.131	0.044	0.133	0.051
Anticipation	0.151	0.053	0.142	0.056
Disgust	0.219	0.070	0.225	0.084
Fear	0.100	0.047	0.098	0.056
Joy	0.091	0.052	0.083	0.005
Sadness	0.073	0.039	0.065	0.034
Surprise	0.132	0.045	0.147	0.070
Trust	0.102	0.044	0.107	0.047
Propagation dynamics				
Response time (in hours)	114	887	197	1001

Table 5: Descriptive statistics of features computed at cascade and tweet level.

C LEARNING DETAILS

Our Cascade-LSTM is prone to overfitting due to several reasons, e. g., small dataset, high-dimensional parameter space, severe class imbalance. Hence, in addition to data augmentation (cf. Section 2.5), we adopted several remedies as detailed in the following.

Regularization loss: The parameters θ inside the Cascade-LSTM are estimated by proposing the following loss function $\mathcal{L}_\theta$. It combines a binary cross-entropy loss between the true labels $\boldsymbol{y}$ and the predicted values $\hat{\boldsymbol{y}}$ with an additional regularization. Let θ_T denote the parameters of the bi-directional tree structure (components C1 and C2) and let θ_NN denote the final prediction layer (component C3) from the Cascade-LSTM. Our proposed loss is then given by

$$\mathcal{L}_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_i l(y_i, \hat{y}_i) + \lambda_\text{T} \, \|\theta_\text{T}\|_2^2 + \lambda_\text{NN} \, \|\theta_\text{NN}\|_2^2 \qquad (18)$$

with hyperparameters λ_T and λ_NN and where i refers to the i-th sample in the batch. The function $l(\cdot,\cdot)$ gives the plain binary cross-entropy loss between a "real" label y_i and the corresponding prediction $\hat{y}_i$, i. e.,

$$l(y, \hat{y}) = \rho \, y \, \log(\hat{y}) + (1-y)\log(1-\hat{y}) \qquad (19)$$

with a constant $\rho = \frac{|\{y \in Y : y=\text{"fake"}\}|}{|\{y \in Y : y=\text{"real"}\}|}$ that denotes the ratio of "fake"-to-"real" labels. The constant is added, as it further helps in discriminating our minority class (i. e., "Veracity = real").

Implementation: All models are implemented with PyTorch and the Deep Graph Library.[6] The latter allows us to perform batch learning on different cascades in parallel. That is, the cascade structure is traversed according to the topological node ordering [20] and, based on this, the computations behind LSTM units are parallelized for all nodes where the hidden states are available (i. e., the children in the upward Tree-LSTM and the parent in downward Tree-LSTM). As a result, we observed a significant speedup.

Training process: We decided upon a "ceteris paribus" strategy for hyperparameter tuning rather than a grid search due to large number of hyperparameters inside the Cascade-LSTM. Formally, each parameter was tuned individually, while keeping the others constant. All tested hyperparameters are listed in Appendix D. We summarize a few choices. A dropout layer with a dropout rate $p_\text{drop} = 0.1$ was placed after each hidden layer in the fused network in order to reduce overfitting. In our experiments, we found it sufficient to set the regularization parameter λ_NN for the bi-directional Tree-LSTM to 0.003 and λ_T for the final prediction layer to 0.006. Our intuition is that a too large λ_NN would prohibit learning very deep or wide cascades, whereas a small λ_NN reduces the risk of overfitting but also the flexibility of the network.

The Cascade-LSTM was trained using mini-batches of size 25 and early stopping with patience set to ten epochs. We further used Adam [15] as our optimizer with a learning rate of 0.01 and a decay of 0.9 every 5 epochs. All computations were performed on Nvidia Titan V GPUs.

Over-/under-sampling: The labels in our dataset are highly imbalanced. In fact, 85.6 % of the cascades in our dataset are classified as "fake". In order address this class imbalance, we perform the following three-stage process of repeated over- and under-sampling that we found highly suited (especially for a dataset as ours that is fairly small with respect to the parameter space).[7]

First, we apply random over-sampling to the minority label, i. e., the cascades labeled as "real" until they are 30.0 % of the "fake" ones (or, equivalently, 23.1 % of the total samples). Second, we under-sample the majority class, i. e., the cascades labeled as "fake", via Tomek links [42]. This should theoretically reduce the overlap among similar samples (as per projections in Euclidean space).

[6]Deep Graph Library: https://www.dgl.ai

[7]We experimented with similarity metrics based on graph kernels, yet they revealed to be intractable due to the computational complexity. Therefore, we solely performed resampling based on aggregated statistics.

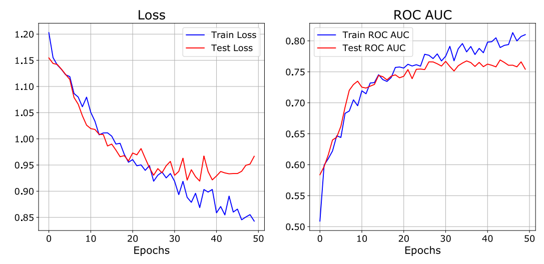

Figure 7: Loss (left) and AUC (right) of the Cascade-LSTM on the training (blue) and test (red) datasets.

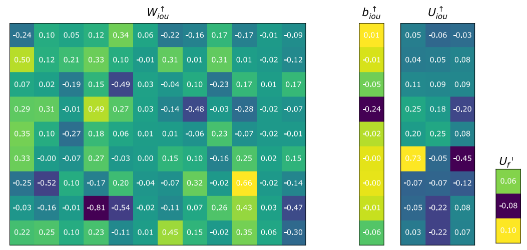

Figure 8: Weights of the upward LSTM units. Darker colors refer to negative values, bright colors to positive.

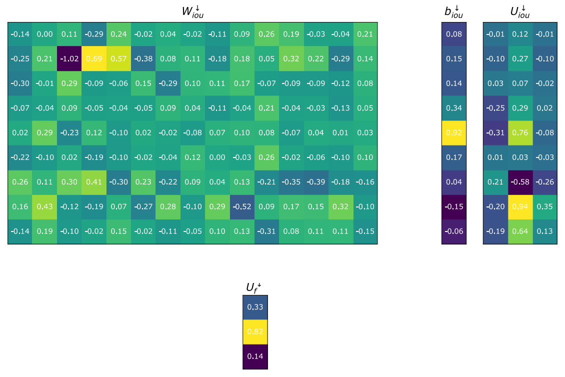

Figure 9: Weights of downward LSTM units. Darker colors refer to negative values, bright colors to positive.

Third, we under-sample the majority class via a neighborhood cleaning rule [21]. This process attains a distribution where the minority class represents 30 % of all samples.

Data augmentation: During all epochs, each sample from the original training set was subject to (i) $\xi = 5$ random applications of node reordering, (ii) $\xi = 5$ random applications of leaf insertion, and (iii) a single perturbation of the root feature.

D HYPERPARAMETER TUNING

We perform hyperparameter tuning in order to find the best parameter configuration for each model. The tuning parameters for the Cascade-LSTM with the corresponding tuning ranges are listed in Table 6.

Parameter	Tuning Range
Size of hidden cell	2, 3, 4
Batch size	20, 25, 30, 40
Dropout rate (fused network)	0.0, 0.1, 0.17, 0.4
Learning rate (upward/downward Tree-LSTM)	0.1, 0.05, 0.02, 0.01 0.005
Learning rate (upward/downward Tree-LSTM)	0.1, 0.05, 0.02, 0.01 0.005
Regularization strength (fused network)	0.006, 0.003, 0.001, 0.0005
Regularization strength (upward/downward Tree-LSTM)	0.006, 0.003, 0.001, 0.0005
Top network hidden layers	(16×16), $(16 \times 16 \times 4)$, (24×24)

Table 6: Hyperparameters used for tuning the Cascade-LSTM.

Parameter	Tuning Range
Size of hidden cell	2, 3, 4, 16, 32
Batch size	20, 25, 30, 40
Learning rate	0.1, 0.05, 0.02, 0.01 0.005
Regularization strength	0.006, 0.003, 0.001, 0.0005
Top network hidden layers	(16×16), $(16 \times 16 \times 4)$, (24×24)
States in HMM	$1, 2, \ldots, 10$

Table 7: Hyperparameters used for tuning the baselines.

E LOSS CURVE

Figure 7(left) shows the loss curves of our Cascade-LSTM during training and testing. Figure 7(right) additionally reports the corresponding results for the AUC on the train and test set, respectively.

E.1 Weight Matrices

The Cascade-LSTM yields different weight matrices $W_i^{\star}$, $W_i^{\star}$, $W_o^{\star}$, ..., $\star \in \{\uparrow, \downarrow\}$ that we now visualize as part of simple diagnostics. These are plotted for the upward part (Figure 8) and the downward component (Figure 9).